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

Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R23-11

Document Change History

Date	Release	Changed by	Description
2023-11-23	R23-11	AUTOSAR Release Management	• Improve diagnostic configuration • Improve firewall configuration • minor corrections / clarifications / editorial changes
2022-11-24	R22-11	AUTOSAR Release Management	• Support for MACsec • Support for firewall • Modeled State Management • Support for Service-oriented Vehicle Diagnostic • minor corrections / clarifications / editorial changes
2021-11-25	R21-11	AUTOSAR Release Management	• Removal of REST support • Rework of signal-to-service conversion • Rework of raw data stream configuration • Rework of diagnostic mapping • minor corrections / clarifications / editorial changes

△

2020-11-30	R20-11	AUTOSAR Release Management	• Remodeling of Phm contribution • Reporting of Security Events • Support for cryptographic Operations • Remodeling of Diagnostic Mapping • minor corrections / clarifications / editorial changes
2019-11-28	R19-11	AUTOSAR Release Management	• Overhaul of Signal-to-Service Translation • Support for Raw Data Streams • Support for Vehicle Package • Support for Service Versioning • Changed Document Status from Final to published
2019-03-29	19-03	AUTOSAR Release Management	• Introduction of Diagnostic Port Interfaces • Overhaul of Software Cluster and introduction of Software Package • Support for Identity and Access Management • Network Management Configuration
2018-10-31	18-10	AUTOSAR Release Management	• Finish introduction of CppImplementationDataType • Support for optional elements in structures • Rework configuration of adaptive platform modules
2018-03-29	18-03	AUTOSAR Release Management	• Time Synchronization • DDS Deployment

▽

△

2017-10-27	17-10	AUTOSAR Release Management	• Optional elements in Service Interfaces • Interaction with web services • Secure Communication • Support for interaction with crypto and persistency • Signal-to-Service translation • Support for E2E communication • Platform Health Management • Uploadable Software Package
2017-03-31	17-03	AUTOSAR Release Management	• Initial release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Contents

References

[1] Software Component Template
AUTOSAR_CP_TPS_SoftwareComponentTemplate

[2] Diagnostic Extract Template
AUTOSAR_CP_TPS_DiagnosticExtractTemplate

[3] Layered Software Architecture
AUTOSAR_CP_EXP_LayeredSoftwareArchitecture

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

[5] Standardization Template
AUTOSAR_FO_TPS_StandardizationTemplate

[6] Generic Structure Template
AUTOSAR_FO_TPS_GenericStructureTemplate

[7] Demonstrator Design of Functional Cluster Identity and Access Management
AUTOSAR_AP_EXP_FCDesignIdentityAndAccessManagement

[8] Explanation of ara::com API
AUTOSAR_AP_EXP_ARAComAPI

[9] SOME/IP Protocol Specification
AUTOSAR_FO_PRS_SOMEIPProtocol

[10] Specification of Communication Management
AUTOSAR_AP_SWS_CommunicationManagement

[11] Specification of Execution Management
AUTOSAR_AP_SWS_ExecutionManagement

[12] Specification of Platform Health Management
AUTOSAR_AP_SWS_PlatformHealthManagement

[13] Specification of Persistency
AUTOSAR_AP_SWS_Persistency

[14] 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

[15] Specification of Time Synchronization
AUTOSAR_AP_SWS_TimeSynchronization

[16] Specification of Cryptography
AUTOSAR_AP_SWS_Cryptography

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

[18] System Template
AUTOSAR_CP_TPS_SystemTemplate

[19] Specification of Diagnostics
AUTOSAR_AP_SWS_Diagnostics

[20] Security Extract Template
AUTOSAR_FO_TPS_SecurityExtractTemplate

[21] Specification of Operating System Interface
AUTOSAR_AP_SWS_OperatingSystemInterface

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

[23] Log and Trace Protocol Specification
AUTOSAR_FO_PRS_LogAndTraceProtocol

[24] ASAM Service Oriented Vehicle Diagnostics V1.0.0
http://www.asam.net
ASAM_SOVD_BS_V1-0-0_PR.pdf

[25] Multicast DNS
https://www.ietf.org/rfc/rfc6762.txt

[26] DNS-Based Service Discovery
https://www.ietf.org/rfc/rfc6763.txt

[27] mDNS IPv4 Parameters
https://www.iana.org/assignments/multicast-addresses/multicast-addresses.txt

[28] mDNS Port Parameters
https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt

[29] mDNS IPv6Parameters
https://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.txt

[30] SOME/IP Service Discovery Protocol Specification
AUTOSAR_FO_PRS_SOMEIPServiceDiscoveryProtocol

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

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

[33] DDS Consolidated XML Syntax, Version 1.0
https://www.omg.org/spec/DDS-XML

[34] Specification of SW-C End-to-End Communication Protection Library
AUTOSAR_CP_SWS_E2ELibrary

[35] DDS Security, Version 1.1
https://www.omg.org/spec/DDS-SECURITY/1.1

[36] Specification of Secure Onboard Communication
AUTOSAR_CP_SWS_SecureOnboardCommunication

[37] Glossary
AUTOSAR_FO_TR_Glossary

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

Section 9 describes the options for configuring a machine by means of a *manifest*.

Section 8 represents that counterpart to section 3 on deployment level, it describes the content of the so-called *execution manifest*.

Section 10 contains a string of sub-sections that explain the manifest content of platform module functionality.

Section 11 provides a detailed description of how service-oriented communication shall be configured on *manifest* level.

Section 12 describes the deployment modeling for raw data stream communication.

Section 13 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 14 explains the modeling of interactions among functional clusters in the AUTOSAR stack.

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

Finally, section 16 makes some remarks about the interaction of the *AUTOSAR adaptive platform* with software on the *AUTOSAR adaptive platform*.

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

1.1 Modeling Approach

1.1.1 General Considerations

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.

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*.

1.1.2 Modeling of Diagnostic Configuration

The configuration of the “external behavior” of the AUTOSAR diagnostic stack is done by means of the so-called `Diagnostic Extract`. The `Diagnostic Extract` in general extends to both the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform*.

In particular, some parts of the `Diagnostic Extract` apply only for the *AUTOSAR classic platform*, some parts are shared between the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform*, and some parts apply only for the *AUTOSAR adaptive platform*.

The parts that are specific to the *AUTOSAR classic platform*, and the parts that apply to both the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform* are documented in the deliverable “TPS Diagnostic Extract Template” [2].

For technical reasons, the parts that apply to the *AUTOSAR adaptive platform* cannot be included in the document “TPS Diagnostic Extract Template” and are consequently documented in this document, “TPS Manifest Specification”.

The parts about the configuration of Diagnostic Management, as described in this document, fit seamlessly into the same modeling framework that is used on the *AUTOSAR classic platform*. This means that the specific parts for the *AUTOSAR adaptive platform* utilize the shared parts that are described in the “TPS Diagnostic Extract Template”.

In other words, readers who want to focus on `Diagnostic Extract` specifically for application on the *AUTOSAR classic platform* may concentrate on reading the “TPS Diagnostic Extract Template”.

On the other hand, readers who’d like to understand how the `Diagnostic Extract` works on the *AUTOSAR adaptive platform* will have to read the diagnostics-related chapters of the “TPS Manifest Specification”, plus the relevant parts of the “TPS Diagnostic Extract Template”.

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 [3] 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) [4]. This meaning has the strongest relation to the usage of the term *service* on the *AUTOSAR adaptive platform*.

1.3 Document Conventions

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

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

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

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

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

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

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

The points in the workflow where constraints shall be enforced, sometimes also known as the "binding time" of the constraint, are different for each model category, e.g. on the

classic platform, the constraints defined for software-components are typically enforced prior to the generation of the RTE while the constraints against the definition of an Ecu extract shall be applied when the Ecu configuration for the Com stack is created.

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

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

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

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

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

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

The headers in the table have the following meaning:

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

Type: The type of an attribute of the class.

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

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

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

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

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see Standardization Template, chapter Support for Traceability ([5]).

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template, chapter Support for Traceability ([5]).

1.4 About Tables

There is a number of tables in this document that describe the applicability or multiplicity of model elements. These tables contain empty cells, which represent a “not applicable” meaning. Model elements that correspond to an empty cell shall not exist in the context described by the respective table cell.

If the value of a table cell shall indicate that that either the existence or multiplicity of the represented model element shall be ignored anyway, the table cell gets the content “d/c” for “don’t care”. This means that if the respective model element exists, it shall be ignored in the downstream processing of the model.

2 Big Picture of Manifest Definition

2.1 Design vs. Deployment

2.1.1 Overview

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

Model elements discussed in this document are either related to *design* or *deployment*, there is no overlap between the two groups, but it could happen that *design* model elements are required in a concrete model for *deployment* mode elements to make sense.

As mentioned before, some *deployment* model element may need one or more references to *design* model elements and the *design* model elements need to be uploaded to the target platform for the sake of completing the semantics of the *deployment* model.

Please note that a *design* model element does not automatically become a *deployment* model element just because it is required to be uploaded to a target platform.

2.1.2 Relation between Design and Deployment Models

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

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

Example 2.1

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

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

Example 2.2

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

The definition of AdaptivePlatformServiceInstances may then have to respect the existing modeling on the software side.

Example 2.3

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

Some model elements described in this document that are part of the content that gets uploaded to the target platform have direct counterparts on the design level.

One example for such a relation is the definition of ProcessDesign and Process. In these cases it would be easier to draw the line between design and deployment aspects of the model.

There are cases where the mere existence of a reference into the design model is already representing valuable information. References that expose this capability are decorated with the stereotype ≪atpUriDef≫.

Specifically, it is possible for the software on the platform to derive the value of an InstanceSpecifier from the content of an instanceRef decorated with ≪atpUriDef≫. This is a very important mechanism for the platform software to interact with the application layer.

Another example is the definition of the checkpointId in the context of the configuration of the Platform Health Manager.

The modeling of the PHM interaction on the application layer involves the definition of the value of the PhmCheckpoint.checkpointId and the PortPrototype where a specific PhmCheckpoint.checkpointId is used is identified by means of a reference stereotyped as ≪atpUriDef≫.

And because the target of the reference is not necessarily existing on the platform, the SupervisionCheckpoint.checkpointId is replicated in the manifest model so that the platform software has access to this important piece of information (this aspect is also explained in Figure 10.9).

2.1.3 Structure of the document

The structure of the document maps to the division between *design* and *deployment* such that the *design* aspect is mostly described in sections 3, 4, 6, 7, and most of 13.

In contrast, chapters 9, 8, 10, 11, 13.3, 14, and 15 focus on *deployment*-related content.

2.2 About Manifest

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

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

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

2.3 Serialization Format

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

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

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

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

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

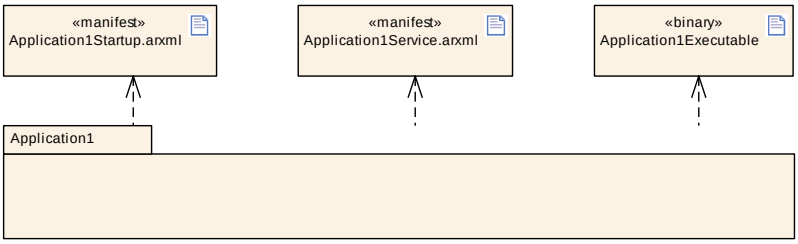


Figure 2.1: Example usage of several manifest files within one software delivery

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

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

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

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

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

2.4 Scope

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

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

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

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

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

⌋*(RS_MANI_00015)*

2.5 Manifests described in this Document

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

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

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

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

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

Please find more information regarding this topic in section 8.

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 11.

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 sections 9 and 10.

Raw Data Stream Manifest This kind of `Manifest` describes in section 12 the configuration of client and server for the purpose of communicating via raw data streams.

Software Distribution This kind of `Manifest` describes the packaging and logistics aspects of software on the *AUTOSAR adaptive platform*.

Please find more information regarding this topic in section 15.

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 different 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*.

There are also some extensions of existing modeling used on the *AUTOSAR classic platform* specifically for the usage on the *AUTOSAR adaptive 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 application structure, consisting of `SwComponentType` for the *AUTOSAR adaptive platform* (section 3.2.1), `Executable` as the design-level representation of the smallest executable unit (section 3.2.2), as well as Description of how to front-load the configuration of a `Process` by means of `ProcessDesign`, see section 3.2.4
- Definition of relevant `PortInterface`s, among them `ServiceInterface` as the pivotal element for service-oriented communication (section 3.3.1), `PortInterface`s for diagnostics (section 3.3.8), PHM (section 3.3.7), persistency (section 3.3.5), and crypto (section 3.3.9)
- Definition of data types specifically for the *AUTOSAR adaptive platform* (section 3.4), including the handling of optional elements in data structures, see section 3.4.5
- Configuration of transformation properties (section 3.5)
- Description of the design-level IAM configuration by means of the `GrantDesign`, see section 3.6

3.2 Application Structure

3.2.1 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 3.4) that are exclusive to the *AUTOSAR adaptive platform*.

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

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

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

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

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

Later versions of this specification may fix the missing regulation.

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

Table 3.1: AdaptiveApplicationSwComponentType

Class	**AdaptiveSwcInternalBehavior**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::AdaptiveInternalBehavior			
Note	This meta-class represents the ability to define an internal behavior of an AtomicSwComponentType used on the AUTOSAR adaptive platform. Please note that the model of internal behavior in this case, in stark contrast to the situation of the AUTOSAR classic platform, is very minimal.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	AdaptiveApplicationSwComponentType.internalBehavior			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
service Dependency	SwcService Dependency	*	aggr	This represents the collection of SwcService Dependencys owned by AdaptiveInternalBehavior.

Table 3.2: AdaptiveSwcInternalBehavior

3.2.2 Executable

3.2.2.1 Overview

An `Executable` represents an executable program in the AUTOSAR meta-model. An explanation of the semantics of the attributes of `Executable` can be found in the following sub-sections.

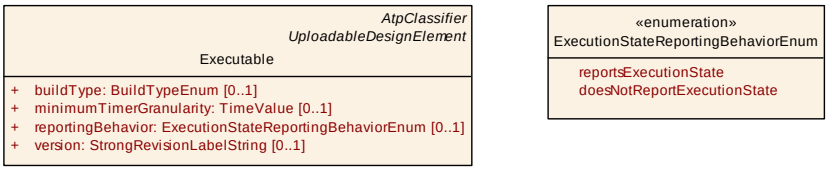


Figure 3.1: Modeling of the `Executable`

Class	**Executable**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This meta-class represents an executable program. **Tags:** atp.recommendedPackage=Executables			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
buildType	BuildTypeEnum	0..1	attr	This attribute describes the buildType of a module and/or platform implementation.
implementation Props	Executable ImplementationProps	*	aggr	This aggregation contains the collection of implementation-specific properties necessary to properly build the enclosing Executable.
minimumTimer Granularity	TimeValue	0..1	attr	This attribute describes the minimum timer resolution (TimeValue of one tick) that is required by the Executable.
reporting Behavior	ExecutionState ReportingBehavior Enum	0..1	attr	this attribute controls the execution state reporting behavior of the enclosing Executable.
rootSw Component Prototype	RootSwComponent Prototype	0..1	aggr	This represents the root SwCompositionPrototype of the Executable. This aggregation is required (in contrast to a direct reference of a SwComponentType) in order to support the definition of instanceRefs in Executable context.
traceSwitch Configuration	TraceSwitch Configuration	*	aggr	Configuration of the MsgId based trace switch **Tags:** atp.Status=draft
version	StrongRevisionLabel String	0..1	attr	Version of the executable.

Table 3.3: Executable

Enumeration	**BuildTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation
Note	This enumeration defines the possible buildTypes a software module may be implemented.
Aggregated by	Executable.buildType

▽

△

Enumeration	**BuildTypeEnum**
Literal	***Description***
buildTypeDebug	Used for debugging. **Tags:** atp.EnumerationLiteralIndex=1
buildTypeRelease	Used for releasing. **Tags:** atp.EnumerationLiteralIndex=0

Table 3.4: BuildTypeEnum

Class	**TraceSwitchConfiguration**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This class maps a DltMessage to a trace switch type. **Tags:** atp.Status=draft			
Base	*ARObject*			
Aggregated by	Executable.traceSwitchConfiguration			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
traceMessage	DltMessage	0..1	ref	Reference to the DltMessage that has to be routed in the trace switch **Tags:** atp.Status=draft
traceSwitch	TraceSwitchEnum	0..1	attr	Defines how the message is routed, either to • ARTI trace, • Logger (default), • ARTI trace and logger • None **Tags:** atp.Status=draft

Table 3.5: TraceSwitchConfiguration

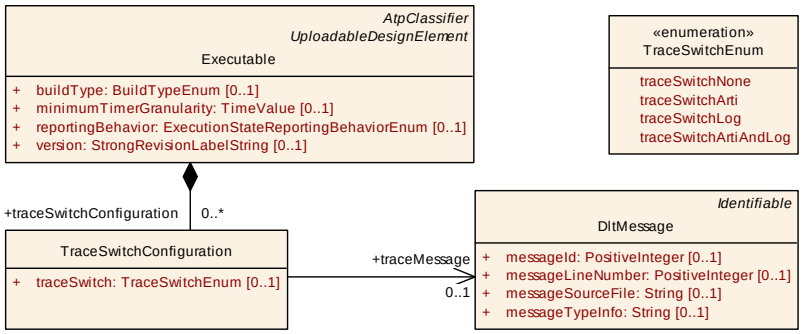


Figure 3.2: Modeling of the `TraceSwitchConfiguration`

[constr_5386]{DRAFT} **`Executable.traceSwitchConfiguration`** ⌈The aggregation in the role `Executable.traceSwitchConfiguration` shall exist **`at the time before the generation of the ara API starts`**.⌋*()*

[constr_5387]{DRAFT} **Existence of `Executable.traceSwitchConfiguration.traceMessage`** ⌈The aggregation in the role `Executable.traceSwitchConfiguration.traceMessage` shall exist **`at the time before the generation of the ara API starts`**.⌋*()*

[constr_5388]{DRAFT} **Existence of `Executable.traceSwitchConfiguration.traceSwitch`** ⌈The aggregation in the role `Executable.traceSwitchConfiguration.traceSwitch` shall exist **`at the time before the generation of the ara API starts`**.⌋*()*

Enumeration	**TraceSwitchEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure
Note	This enumeration provides possible trace switches for DltMessages. **Tags:** atp.Status=draft
Aggregated by	TraceSwitchConfiguration.traceSwitch
Literal	***Description***
traceSwitchArti	The message will be routed to ARTI-trace with minimal runtime influence. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=draft
traceSwitchArtiAnd Log	The message will be routed to ARTI-trace and to the logger. **Tags:** atp.EnumerationLiteralIndex=3 atp.Status=draft
traceSwitchLog	The message will be routed to the logger that further processes the message to different sinks (see LogMode) with different verbosities at runtime. **Tags:** atp.EnumerationLiteralIndex=2 atp.Status=draft
traceSwitchNone	The message will not be routed. No runtime resources will be used. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=draft

Table 3.6: TraceSwitchEnum

3.2.2.2 Category

[constr_1605]{DRAFT} **Standardized values of attribute `Executable.category`** ⌈The following values for attribute `Executable.category` are standardized by AUTOSAR:

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

This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

3.2.2.3 Root Software Component Prototype

[TPS_MANI_01010]{DRAFT} **Root element for a hierarchical software-component** ⌈`Executable` aggregates meta-class `RootSwComponentPrototype` in the role `rootSwComponentPrototype` to provide a root element for an arbitrarily nested hierarchy of software-components represented by the reference `RootSwComponentPrototype.applicationType`.⌋*(RS_MANI_00001, RS_MANI_00004)*

Please note that the aggregation of `RootSwComponentPrototype` by `Executable` is the basis for the applicability of an ≪`instanceRef`≫ reference into the hierarchy of software-components that represent the functionality of the `Executable`.

This modeling approach is similar to the modeling of a `System` on the *AUTOSAR classic platform*.

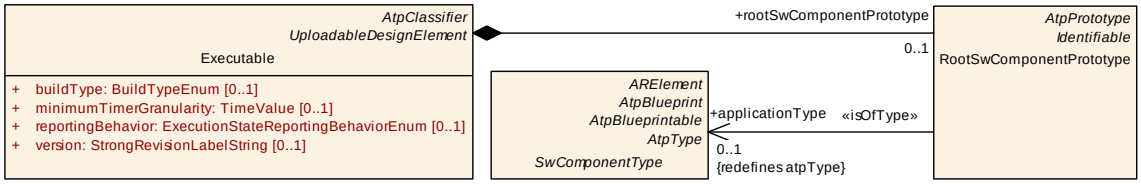


Figure 3.3: Modeling of the `RootSwComponentPrototype` in the context of `Executable`

[TPS_MANI_03056]{DRAFT} **Optionality of `Executable.rootSwComponentPrototype`** ⌈The aggregation `Executable.rootSwComponentPrototype` has been made optional in order to support the implementation of *platform modules* that do not utilize any service oriented communication and don't require any further formalization.⌋ *(RS_MANI_00023)*

[constr_1492]{DRAFT} **`SwComponentType` referenced in the role `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` **at the time before the generation of the ara API starts**.⌋*()*

Class	**RootSwComponentPrototype**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	The RootSwCompositionPrototype represents the top-level-composition of software components within an Executable. The contained SwComponentPrototypes are fully specified by their SwComponentTypes (including Port Prototypes, PortInterfaces, VariableDataPrototypes, etc.).			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Executable.rootSwComponentPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	RootSwComponentPrototype			
applicationType	SwComponentType	0..1	tref	This SwComponentType acts as the Type of the RootSw ComponentPrototype. **Stereotypes:** isOfType

Table 3.7: RootSwComponentPrototype

The example depicted in Figure 3.4 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`.

Class	*SwComponentType* (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for AUTOSAR software components.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	AdaptiveApplicationSwComponentType, *AtomicSwComponentType*, CompositionSwComponentType, ParameterSwComponentType			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
port	PortPrototype	*	aggr	The PortPrototypes through which this SwComponent Type can communicate. The aggregation of PortPrototype is subject to variability with the purpose to support the conditional existence of PortPrototypes. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=port.shortName, port.variationPoint.short Label vh.latestBindingTime=preCompileTime
portGroup	PortGroup	*	aggr	A port group being part of this component. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=portGroup.shortName, portGroup.variation Point.shortLabel vh.latestBindingTime=preCompileTime
swComponent Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=swComponentDocumentation, sw ComponentDocumentation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=-10

Table 3.8: SwComponentType

Class	**CompositionSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	A `CompositionSwComponentType` aggregates `SwComponentPrototype`s (that in turn are typed by `SwComponentTypes`) as well as `SwConnector`s for primarily connecting `SwComponentPrototype`s among each others and towards the surface of the `CompositionSwComponentType`. By this means, a 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*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
component	SwComponent Prototype	*	aggr	The instantiated components that are part of this composition. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=component.shortName, component.variation Point.shortLabel vh.latestBindingTime=postBuild
connector	SwConnector	*	aggr	`SwConnector`s have the principal ability to establish a connection among `PortPrototype`s. They can have many roles in the context of a `CompositionSwComponentType`. Details are refined by subclasses. The aggregation of `SwConnector`s 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 `AssemblySwConnector`s between `ApplicationSwComponentType`s and `ServiceSwComponentType`s during the ECU integration. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=connector.shortName, connector.variation Point.shortLabel vh.latestBindingTime=postBuild
constantValue Mapping	ConstantSpecification MappingSet	*	ref	Reference to the `ConstantSpecificationMapping` to be applied for initValues of `PPortComSpec`s and `RPortComSpec`. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=constantValueMapping
dataType Mapping	DataTypeMappingSet	*	ref	Reference to the `DataTypeMapping` to be applied for the used `ApplicationDataType`s in `ServiceInterface`s. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=dataTypeMapping
physical Dimension Mapping	PhysicalDimension MappingSet	0..1	ref	This reference identifies the `PhysicalDimensionMappingSet` that is applicable in the context of the enclosing `CompositionSwComponentType`. The `PhysicalDimensionMapping`s contained in the `PhysicalDimensionMappingSet` shall be taken into account for the assessment of the compatibility of `PhysicalDimension`s in the context of creation of a `PortInterfaceMapping` in the scope of the `CompositionSwComponentType`.

Table 3.9: CompositionSwComponentType

While the left part of Figure 3.4 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.

An obvious consequence of [constr_1492] is that no software-component that could be used on the *AUTOSAR classic platform* is allowed on the *AUTOSAR adaptive platform*, i.e. in the context of an `Executable`.`rootSwComponentPrototype`.`applicationType`.

Software-components on the *AUTOSAR adaptive platform* are mainly defined by their interaction with the outside world by means of `PortPrototype`s typed by `ServiceInterface`s. The definition of an internal behavior, with a minor exception, is not foreseen.

This lack of internal structure, in combination with decisions made regarding the scope of the generation of header files, leads to a situation where the implementation of a software component in source code is (in comparison to the situation on the *AUTOSAR classic platform*) way less subject to a strict separation.

In other words, there is no real motivation to implement software-components separately from each other. It would be possible, although not encouraged, to implement all software-components of a given executable program directly within the `Main()` function of the program.

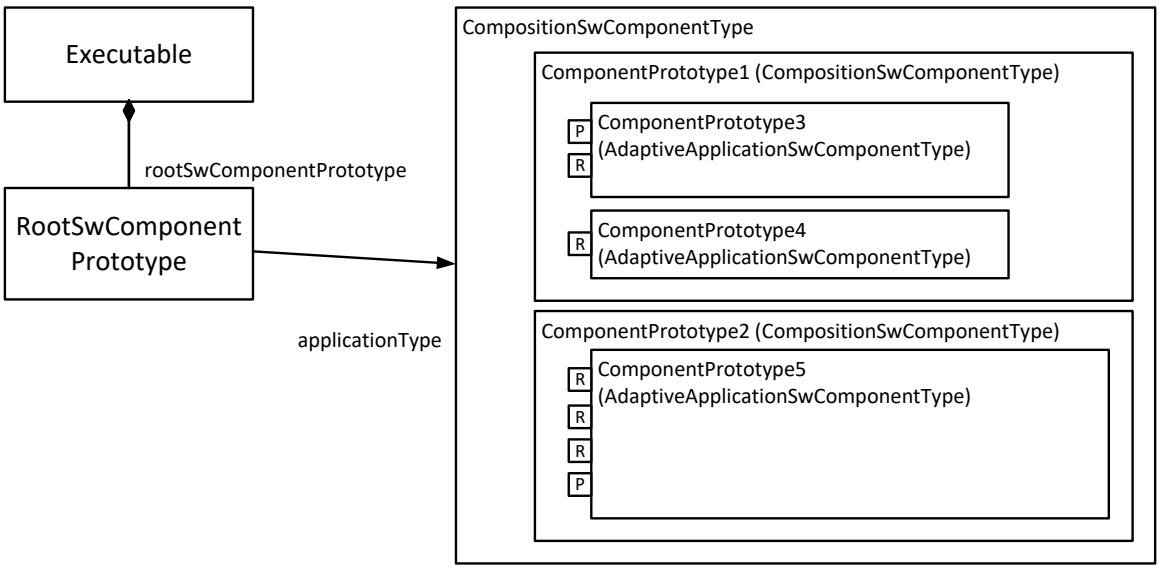


Figure 3.4: Example of the possible structure of an `Executable`

3.2.2.4 Reporting Behavior

[TPS_MANI_01279]{DRAFT} **Semantics of `Executable.reportingBehavior`** ⌈Attribute `Executable.reportingBehavior` shall be used to control the reporting

of the execution state of the enclosing `Executable` to the Execution Management. If the attribute does not exist, the `Executable` shall report its execution state to the Execution Management.⌋*(RS_MANI_00023)*

Enumeration	**ExecutionStateReportingBehaviorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure
Note	This enumeration provides options for controlling of how an Executable reports its execution state to the Execution Management
Aggregated by	Executable.reportingBehavior
Literal	***Description***
doesNotReport ExecutionState	The Executable shall not report its execution state to the Execution Management. **Tags:** atp.EnumerationLiteralIndex=1
reportsExecution State	The Executable shall report its execution state to the Execution Management. **Tags:** atp.EnumerationLiteralIndex=0

Table 3.10: ExecutionStateReportingBehaviorEnum

3.2.2.5 Implementation Props

[TPS_MANI_01370]{DRAFT} **Semantics of `ExecutableImplementationProps`** ⌈If an `Executable` is built for a specific purpose inside the *AUTOSAR adaptive platform*, i.e. the implementation of a functional cluster, then it may be necessary to add configurations that need to be considered at compile time, e.g. whether the implementation of the logging functional cluster needs access to resources provided by the synchronized time base functional cluster.

The configuration of such implementation-specific properties can be done by means of the `Executable.implementationProps`, specifically by a sub-class of `ExecutableImplementationProps`.⌋*(RS_MANI_00001, RS_MANI_00023)*

As depicted in Figure 3.5, one example for the utilization of `ExecutableImplementationProps` is the definition of `ExecutableLoggingImplementationProps` (see section 3.2.2.6).

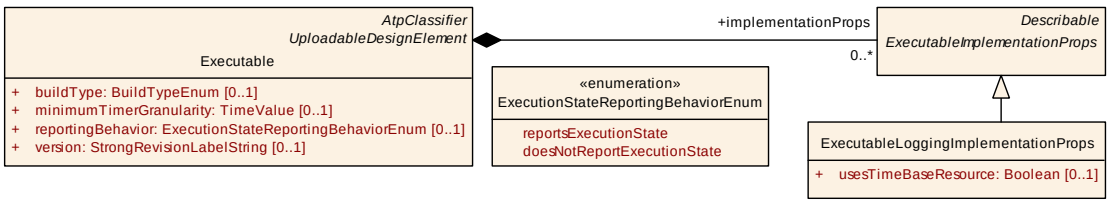


Figure 3.5: Modeling of the `ExecutableImplementationProps`

Class	**ExecutableImplementationProps** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This abstract class has the ability to act as a base class for classes that detail out the implementation-specific properties of an Executable.			
Base	*ARObject*, *Describable*			
Subclasses	ExecutableLoggingImplementationProps			
Aggregated by	Executable.implementationProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.11: ExecutableImplementationProps

3.2.2.6 Logging

[TPS_MANI_01383]{DRAFT} **Semantics of `ExecutableLoggingImplementationProps`** ⌈The implementation properties of an `Executable` that is implementing (a part of) the logging functional cluster can be configured using the `ExecutableLoggingImplementationProps`.⌋*(RS_MANI_00001, RS_MANI_00023)*

[constr_10113]{DRAFT} **Restriction for the existence of `ExecutableLoggingImplementationProps`** ⌈The aggregation of `ExecutableLoggingImplementationProps` in the role `Executable.implementationProps` is only allowed for an `Executable` where attribute `category` is set to the value `PLATFORM_LEVEL`.

This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

Class	**ExecutableLoggingImplementationProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This meta-class contains configuration relevant for the implementation of an Executable used in the context of the LogAndTraceInstantiation.			
Base	*ARObject*, *Describable*, *ExecutableImplementationProps*			
Aggregated by	Executable.implementationProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
usesTimeBase Resource	Boolean	0..1	attr	This attribute indicates that the implementation of the enclosing Executable is required to access resources provided by the synchronized time base functional cluster.

Table 3.12: ExecutableLoggingImplementationProps

3.2.3 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 PortPrototypes 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.

3.2.3.1 Service-oriented Communication

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

[constr_1473]{DRAFT} **No support for PRPortPrototype** ⌈A ServiceInterface shall not be referenced by a PRPortPrototype in the role providedRequiredInterface **at the time before the generation of the ara API starts**.⌋*()*

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

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

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

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

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

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

- The PortPrototype itself (i.e. as a whole) may need to be customized, independently of the kind or number of elements aggregated by the corresponding ServiceInterface. This aspect is discussed in section 3.2.3.4.
- The usage of elements of the corresponding ServiceInterface may need to be configured for a given PortPrototype. This aspect is discussed in section 3.2.3.5.

3.2.3.2 Interaction with Persistent Key-Value Storage

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

[TPS_MANI_01073]{DRAFT} **Semantics of PortPrototype typed by PersistencyKeyValueStorageInterface** ⌈The usage of a specific sub-class of PortPrototype typed by PersistencyKeyValueStorageInterface 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)*

Please note that the PersistencyKeyValueStorageInterface is described in chapter 3.3.5.2.

3.2.3.3 Interaction with Persistent File Storage

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

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

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

⌋*(RS_MANI_00027)*

Please note that the PersistencyFileStorageInterface is described in chapter 3.3.5.3.

3.2.3.4 Port Prototype Props

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

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

Class	***PortPrototypeProps*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This meta-class represents the ability to define a further qualification of semantics of sub-classes of Port Prototype.			
Base	*ARObject*			
Subclasses	RPortPrototypeProps			
Aggregated by	*PortPrototype*.portPrototypeProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.13: PortPrototypeProps

[constr_3642]{DRAFT} **Restriction of aggregation of PortPrototypeProps to the Adaptive Platform** ⌈The aggregation of PortPrototypeProps is only supported in the context of a SwComponentType that is (transitively) referenced by rootSwComponentPrototype.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

One example for this approach is the definition of the meta-class RPortPrototypeProps, sketched in Figure 3.6.

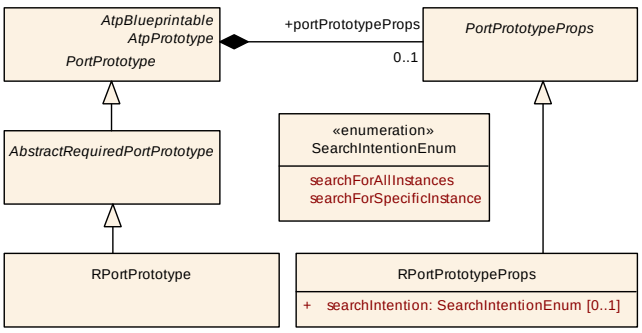


Figure 3.6: Modeling of the RPortPrototypeProps for RPortPrototype

[constr_3359]{DRAFT} **RPortPrototypeProps are related only to RPortPrototypes** ⌈The RPortPrototypeProps shall be aggregated only by a RPortPrototype in the role portPrototypeProps**at the time before the generation of the ara API starts**.⌋*()*

Class	**RPortPrototypeProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	PortPrototypeProps for a RPort.			
Base	*ARObject*, *PortPrototypeProps*			
Aggregated by	*PortPrototype*.portPrototypeProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
searchIntention	SearchIntentionEnum	0..1	attr	This attribute is used to specify the intention of the developer of the enclosing software-component in terms of whether the respective PortPrototype shall be use to search for a specific service instance or all instances of the given service. Please note that the value of this attribute does not create a binding contract. The actual search behavior is defined as part of the service instance manifest.

Table 3.14: RPortPrototypeProps

[TPS_MANI_01057]{DRAFT} **Semantics of `RPortPrototypeProps.searchIntention`** ⌈The value of the attribute `RPortPrototypeProps.searchIntention` clarifies whether the search for a corresponding offer shall be done as a search for all 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)*

Enumeration	**SearchIntentionEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign
Note	This meta-class allows for the definition of a dedicated search intention from the application's point of view.
Aggregated by	RPortPrototypeProps.searchIntention
Literal	***Description***
searchForAll Instances	This value represents the intention to search for all instances of the given service. **Tags:** atp.EnumerationLiteralIndex=0
searchForSpecific Instance	This value represents the intention to search for a specific instance of the given service. **Tags:** atp.EnumerationLiteralIndex=1

Table 3.15: SearchIntentionEnum

3.2.3.5 Port Prototype ComSpec

[TPS_MANI_01053]{DRAFT} **Usage of `ComSpec`s on the *AUTOSAR adaptive platform*** ⌈The aspect of further qualification of elements of the `ServiceInterface` used to type given `PortPrototype` is implemented by means of `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.

The configuration of transformation capabilities in the context of a ComSpec is possible by means of subclasses of meta-class TransformationComSpecProps.

Class	***TransformationComSpecProps*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	TransformationComSpecProps holds all the attributes for transformers that are port specific.			
Base	*ARObject*, *Describable*			
Subclasses	EndToEndTransformationComSpecProps, UserDefinedTransformationComSpecProps			
Aggregated by	ClientComSpec.transformationComSpecProps, *ReceiverComSpec*.transformationComSpecProps, ServerComSpec.transformationComSpecProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.16: TransformationComSpecProps

Class	**UserDefinedTransformationComSpecProps**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	The UserDefinedTransformationComSpecProps is used to specify port specific configuration properties for custom transformers.			
Base	*ARObject*, *Describable*, *TransformationComSpecProps*			
Aggregated by	ClientComSpec.transformationComSpecProps, *ReceiverComSpec*.transformationComSpecProps, ServerComSpec.transformationComSpecProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.17: UserDefinedTransformationComSpecProps

Class	**EndToEndTransformationComSpecProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	The class EndToEndTransformationIComSpecProps specifies port specific configuration properties for EndToEnd transformer attributes.			
Base	*ARObject*, *Describable*, *TransformationComSpecProps*			
Aggregated by	ClientComSpec.transformationComSpecProps, *ReceiverComSpec*.transformationComSpecProps, ServerComSpec.transformationComSpecProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clearFromValid ToInvalid	Boolean	0..1	attr	Clear monitoring window on transition from state Valid to state Invalid.
disableEndTo EndCheck	Boolean	0..1	attr	Disables/Enables the E2E check. The E2Eheader is removed from the payload independent from the setting of this attribute.
disableEndTo EndState Machine	Boolean	0..1	attr	Disables the E2EStateMachine (only E2E check functionality is performed)
e2eProfile Compatibility Props	E2EProfileCompatibility Props	0..1	ref	Reference to additional settings for the E2E state machine.
maxDelta Counter	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 Max DeltaCounter is 3, then at the next reception the receiver can accept Counters with values 2, 3 or 4.

▽

△

Class	**EndToEndTransformationComSpecProps**			
maxErrorState Init	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_INIT. The minimum value is 0.
maxErrorState Invalid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_INVALID. The minimum value is 0.
maxErrorState Valid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_VALID. 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.
minOkState Invalid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INVALID. The minimum value is 1.
minOkState Valid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_VALID. The minimum value is 1.
windowSizeInit	PositiveInteger	0..1	attr	Size of the monitoring window of state Init for the E2E state machine.
windowSize Invalid	PositiveInteger	0..1	attr	Size of the monitoring window of state Invalid for the E2E state machine.
windowSize Valid	PositiveInteger	0..1	attr	Size of the monitoring window of state Valid for the E2E state machine.

Table 3.18: EndToEndTransformationComSpecProps

[TPS_MANI_01327]{DRAFT} **Value of `EndToEndTransformationComSpecProps.disableEndToEndCheck` vs. value of `EndToEndTransformationComSpecProps.disableEndToEndStateMachine`** ⌈If the value of attribute `EndToEndTransformationComSpecProps.disableEndToEndCheck` is set to `true`, then the value of attribute `EndToEndTransformationComSpecProps.disableEndToEndStateMachine` shall be ignored.⌋ *(RS_MANI_00028)*

3.2.3.5.1 Port Prototypes typed by Service Interfaces

3.2.3.5.2 Receiver ComSpec

The `ReceiverComSpec` needs an attribute that indicates whether the enclosing `AdaptiveApplicationSwComponentType` has an intention to actually access the referenced `dataElement`. This attribute represents a security feature related to identity and access management [7].

Specifically, this aspect is typically summarized as a capability of the software, i.e. the AdaptiveApplicationSwComponentType expresses expresses its capability with respect to the specific dataElement. The term "capability" is an integral part of the jargon in the domain of identity and access management.

However, outside the identity and access management domain, this terminology is sometimes hard to motivate. What could be motivated is that the AdaptiveApplicationSwComponentType expresses its *intent* to actually access the dataElement.

From that perspective, the process of adding an event to a ServiceInterface adds the capability to use the dataElement. But whether the software that uses the ServiceInterface actually intends to access the dataElement can be expressed by an attribute in the ReceiverComSpec named receiverIntent.

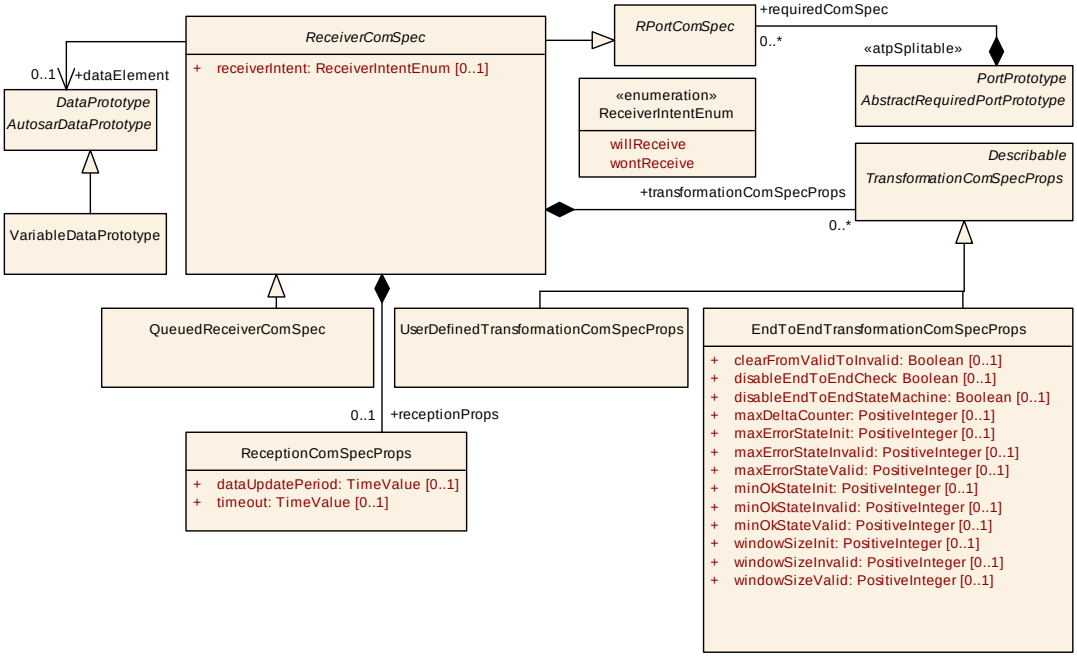


Figure 3.7: Modeling of the ReceiverComSpec on the *AUTOSAR adaptive platform*

Class	***ReceiverComSpec*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Receiver-specific communication attributes (RPortPrototype typed by ServiceInterface) that are relevant for events and field notifiers.			
Base	*ARObject*, *RPortComSpec*			
Subclasses	NonqueuedReceiverComSpec, QueuedReceiverComSpec			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	AutosarDataPrototype	0..1	ref	Data element these attributes belong to.

△

Class	***ReceiverComSpec*** (abstract)			
receiverIntent	ReceiverIntentEnum	0..1	attr	This attribute represents the expressed intent 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=candidate
receptionProps	ReceptionComSpec Props	0..1	aggr	"This aggregation represents the definition transmission props in the context of the enclosing ReceiverComSpec.
transformation ComSpecProps	TransformationCom SpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

Table 3.19: ReceiverComSpec

Class	**QueuedReceiverComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes specific to queued receiving.			
Base	*ARObject*, *RPortComSpec*, *ReceiverComSpec*			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.20: QueuedReceiverComSpec

[TPS_MANI_01106]{DRAFT} **Specification of intentions for the receiver of `event`s or `field` notifiers** ⌈The attribute `ReceiverComSpec.receiverIntent` 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	**ReceiverIntentEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec
Note	This meta-class represents the intent to specify how a given ServiceInterface is used from the perspective of a given event receiver. **Tags:** atp.Status=candidate
Aggregated by	*ReceiverComSpec*.receiverIntent
Literal	***Description***
willReceive	The receiver will receive the event or field notifier. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=candidate
wontReceive	The receiver won't receive the event or field notifier. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=candidate

Table 3.21: ReceiverIntentEnum

[TPS_MANI_03132]{DRAFT} **Semantics of E2E attributes in `ReceiverComSpec`** ⌈The `EndToEndTransformationComSpecProps` shall be used for the specification

of RPortPrototype-specific configuration options related to end-to-end protection of events or field notifiers.⌋(*RS_MANI_00028*)

3.2.3.5.3 Sender ComSpec

The SenderComSpec is modeled in the same way as described in the Software Component Template [1].

[TPS_MANI_03210]{DRAFT} **Specification of event specific communication attributes** ⌈The meta-class QueuedSenderComSpec can be used to specify communication attributes that are relevant for an event on the sender side.⌋(*RS_MANI_00002*)

[TPS_MANI_03211]{DRAFT} **Specification of field specific communication attributes** ⌈The meta-class FieldSenderComSpec can be used to specify communication attributes that are relevant for a field on the sender side.⌋(*RS_MANI_00002*)

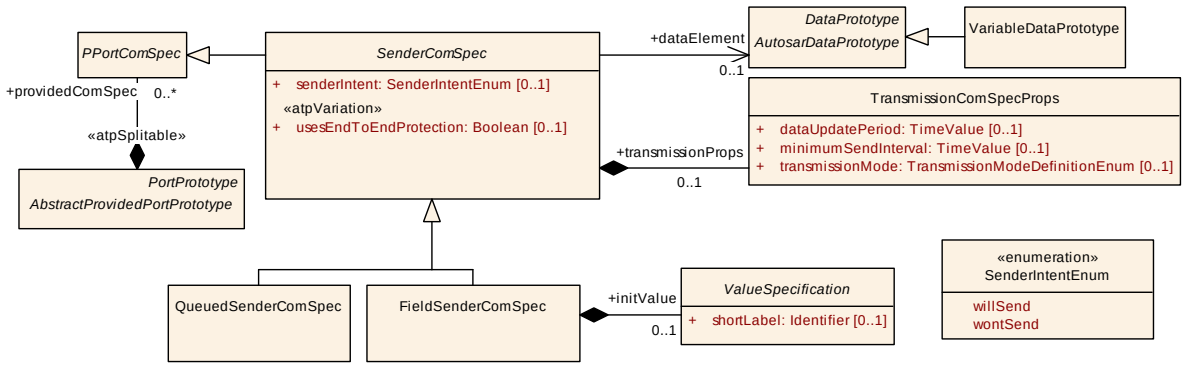


Figure 3.8: Modeling of the SenderComSpec on the *AUTOSAR adaptive platform*

Class	***SenderComSpec*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes for a sender port (PPortPrototype typed by ServiceInterface) that are relevant for events and field notifiers.			
Base	*ARObject*, *PPortComSpec*			
Subclasses	FieldSenderComSpec, NonqueuedSenderComSpec, QueuedSenderComSpec			
Aggregated by	*AbstractProvidedPortPrototype*.providedComSpec, PortPrototypeBlueprint.providedComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	AutosarDataPrototype	0..1	ref	Data element these quality of service attributes apply to.
senderIntent	SenderIntentEnum	0..1	attr	This attribute represents the expressed intent 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=candidate
transmission Props	TransmissionComSpec Props	0..1	aggr	This aggregation represents the definition transmission props in the context of the enclosing SenderComSpec.

△

Class	***SenderComSpec*** (abstract)			
usesEndToEnd Protection	Boolean	0..1	attr	This indicates whether the corresponding dataElement shall be transmitted using end-to-end protection. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table 3.22: SenderComSpec

Class	**QueuedSenderComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes specific to distribution of events (PPortPrototype, SenderReceiverInterface and dataElement carries an "event").			
Base	*ARObject*, *PPortComSpec*, *SenderComSpec*			
Aggregated by	*AbstractProvidedPortPrototype*.providedComSpec, PortPrototypeBlueprint.providedComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.23: QueuedSenderComSpec

Class	**FieldSenderComSpec**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec			
Note	Port specific communication attributes for a Field that is defined in a ServiceInterface.			
Base	*ARObject*, *PPortComSpec*, *SenderComSpec*			
Aggregated by	*AbstractProvidedPortPrototype*.providedComSpec, PortPrototypeBlueprint.providedComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initValue	ValueSpecification	0..1	aggr	Initial value for a Field that is set before the Service Interface is offered.

Table 3.24: FieldSenderComSpec

[constr_10152]{DRAFT} **Multiplicity of reference in the role `FieldSenderComSpec.initValue`** ⌈For each `FieldSenderComSpec`, the reference in the role `initValue` shall exist **`at the time before the generation of the ara API starts`**.⌋*()*

[TPS_MANI_03212]{DRAFT} **Specification of initial value for a `field`** ⌈The attribute `FieldSenderComSpec.initValue` can be used to specify an initial Value for a `field`.⌋*(RS_MANI_00002)*

A `field` has a valid value at any time as described in subsubsection 3.3.1.4. ara::com ensures that a service implementation providing a field has a field value before the field becomes visible to potential consumers.

This is explained in more detail in [8] where it is defined that the initial field value shall be set at least once via Update() by the application code before `OfferService()` gets called.

Custom-code (e.g. component model above ara::com) may use the defined `initValue` to call `Field.Update(initValue)`.

[TPS_MANI_01107]{DRAFT} **Specification of intentions for the sender of `event`s or `field` notifiers** ⌈The attribute `SenderComSpec.senderIntent` can be used to specify whether the software actually intends to send the referenced `event`s or `field` notifier.⌋ *(RS_MANI_00034)*

Enumeration	**SenderIntentEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec
Note	This meta-class represents the intent to specify how a given ServiceInterface is used from the perspective of a given event sender. **Tags:** atp.Status=candidate
Aggregated by	*SenderComSpec*.senderIntent
Literal	***Description***
willSend	The sender will send the event or field notifier. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=candidate
wontSend	The sender won't send the event or field notifier. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=candidate

Table 3.25: SenderIntentEnum

3.2.3.5.4 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 intentions.

[TPS_MANI_01108]{DRAFT} **Specification of intentions for the caller of a `method`s or `field` setter/getter** ⌈The attribute `ClientComSpec.clientIntent` can be used to specify whether the software actually intends to call the referenced `method`s or getter/setter of a referenced `field`.⌋ *(RS_MANI_00034)*

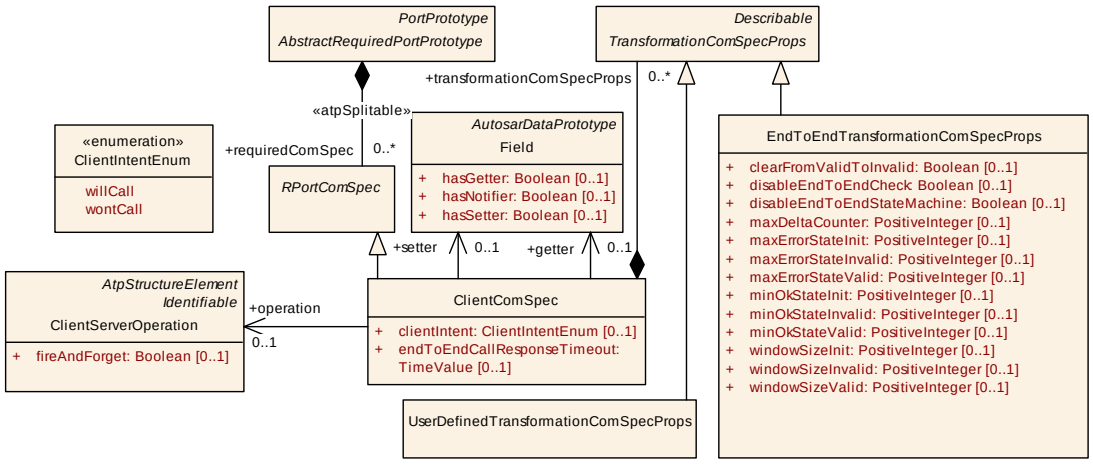


Figure 3.9: Modeling of the `ClientComSpec` on the *AUTOSAR adaptive platform*

[TPS_MANI_01324]{DRAFT} **Semantics of E2E attributes in `ClientComSpec`** ⌈The `EndToEndTransformationComSpecProps` shall be used for the specification of `RPortPrototype`-specific configuration options related to end-to-end protection of `method`s.⌋ *(RS_MANI_00028)*

Class	**ClientComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Client-specific communication attributes (RPortPrototype typed by ServiceInterface) that are relevant for methods and field getters and setters.			
Base	*ARObject*, *RPortComSpec*			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientIntent	ClientIntentEnum	0..1	attr	This attribute represents the expressed intent 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=candidate
endToEndCall Response Timeout	TimeValue	0..1	attr	This attribute defines the maximum time interval in which the application shall expect the servers's response (time between the sending of the call invocation until the arrival of the server's response).
getter	Field	0..1	ref	The existence of this reference indicates that the Client ComSpec refers to the getter of a Field.
operation	ClientServerOperation	0..1	ref	This represents the corresponding ClientServerOperation.
setter	Field	0..1	ref	The existence of this reference indicates that the Client ComSpec refers to the setter of a Field.
transformation ComSpecProps	TransformationCom SpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

Table 3.26: ClientComSpec

Enumeration	**ClientIntentEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec
Note	This meta-class represents the intent to specify how a given ServiceInterface is used from the perspective of a given client. **Tags:** atp.Status=candidate
Aggregated by	ClientComSpec.clientIntent
Literal	***Description***
willCall	The client will call this method. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=candidate
wontCall	The client won't call this method. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=candidate

Table 3.27: ClientIntentEnum

3.2.3.5.5 Server ComSpec

The ServerComSpec 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 intentions.

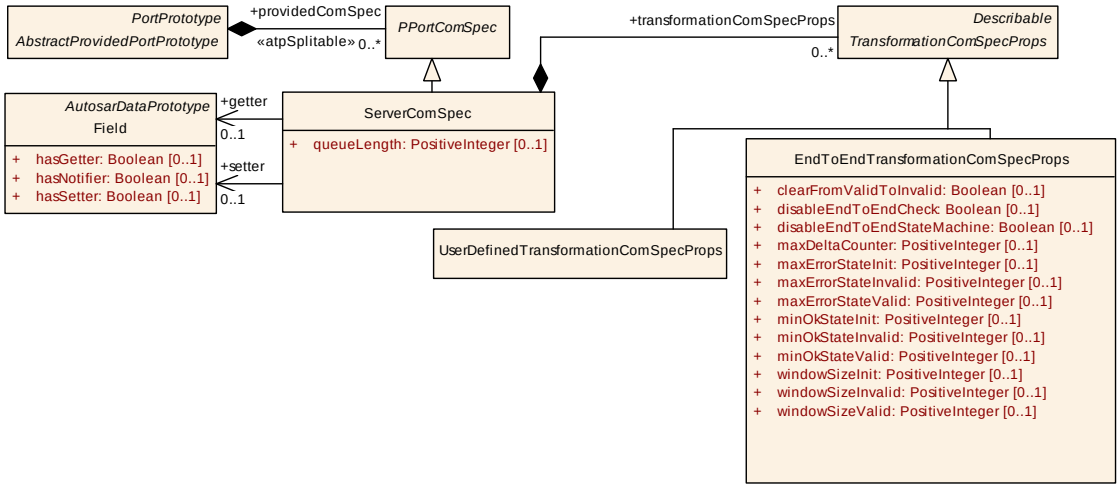


Figure 3.10: Modeling of the ServerComSpec on the *AUTOSAR adaptive platform*

[TPS_MANI_01325]{DRAFT} **Semantics of E2E attributes in ServerComSpec** ⌈The EndToEndTransformationComSpecProps shall be used for the specification of PPortPrototype-specific configuration options related to end-to-end protection of methods.⌋ *(RS_MANI_00028)*

Class	**ServerComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Server-specific communication attributes (PPortPrototype typed by ServiceInterface) that are relevant for methods and field getters and setters.			
Base	*ARObject*, *PPortComSpec*			
Aggregated by	*AbstractProvidedPortPrototype*.providedComSpec, PortPrototypeBlueprint.providedComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
getter	Field	0..1	ref	The existence of this reference indicates that the Server ComSpec refers to the getter of a Field.
operation	ClientServerOperation	0..1	ref	Operation these communication attributes apply to.
queueLength	PositiveInteger	0..1	attr	Length of call queue on the server side.
setter	Field	0..1	ref	The existence of this reference indicates that the Server ComSpec refers to the setter of a Field.
transformation ComSpecProps	TransformationCom SpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

Table 3.28: ServerComSpec

3.2.3.5.5.1 Communication behavior to be implemented by the Application Software

In order to support periodic data transmission and reception (the main reason for this is to be able to perform periodic invocation of the EndToEnd transformation), the application software needs to implement the period in its program flow. But there may also be other reasons why the application shall take care of the periodicity.

The period of the invocation of transmission and reception APIs can be defined using the `TransmissionComSpecProps` and `ReceptionComSpecProps` available at the `SenderComSpec` and `ReceiverComSpec`.

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

The attribute `ReceptionComSpecProps.dataUpdatePeriod` defines the time period in which the receiving application shall call the reception API to check for new data.

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

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

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

The attribute `End2EndEventProtectionProps.dataUpdatePeriod` also defines an expected period to be implemented by the application software for EndToEnd protection.

More specifically, the attribute `End2EndEventProtectionProps.dataUpdatePeriod` represents a network perspective, especially when no application software is defined yet and thus no `SenderComSpec` and `ReceiverComSpec` are available.

3.2.3.5.6 Port Prototypes typed by Persistency Data Interfaces

[TPS_MANI_01314]{DRAFT} **Further qualification of properties of `PortPrototypes` typed by `PersistencyKeyValueStorageInterfaces`** ⌈For `PortPrototypes` typed by `PersistencyKeyValueStorageInterfaces` it is possible to define further qualifying attributes for the required side.

For this purpose meta-class `PersistencyDataRequiredComSpec` is provided.⌋ *(RS_MANI_00027)*

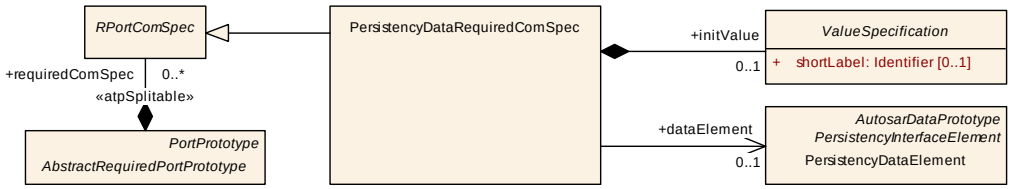


Figure 3.11: Modeling of ComSpec for persistency

[TPS_MANI_01160]{DRAFT} **Definition of initial value for PersistencyDataElement** ⌈The definition of an initial value for a PersistencyDataElement can be done on the level of a PortPrototype by means of PersistencyDataRequiredComSpec.initValue⌋*(RS_MANI_00027)*

Class	**PersistencyDataRequiredComSpec**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec			
Note	This meta-class represents the ability to define port-specific attributes for supporting use cases of data persistency on the required side.			
Base	*ARObject*, *RPortComSpec*			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	PersistencyData Element	0..1	ref	This refrence represents the PersistencyDataElement for which the PersistencyDataRequiredComSpec applies.
initValue	ValueSpecification	0..1	aggr	This aggregation represents the definition of an initial value for the PersistencyDataElement referenced by the enclosing PersistencyDataRequiredComSpec

Table 3.29: PersistencyDataRequiredComSpec

[constr_10153]{DRAFT} **Multiplicity of reference in the role PersistencyDataRequiredComSpec.dataElement** ⌈For each PersistencyDataRequiredComSpec, the reference in the role dataElement shall exist **at the time before the generation of the ara API starts**.⌋*()*

[constr_10081]{DRAFT} **Existence of initial values in the definition of PersistencyDataRequiredComSpec** ⌈For each PersistencyDataRequiredComSpec, if the value of attribute dataElement.updateStrategy is set to the value delete, then attribute PersistencyDataRequiredComSpec.initValue shall not exist **at the time before the generation of the ara API starts**.⌋*()*

3.2.4 Process Design

Within the definition of e.g. a diagnostic mapping, the assignment to the Process is typically done in a methodological step[1] that happens when all the diagnostic mapping[2] is already complete.

[1] i.e. during the creation of the execution manifest

[2] From the methodological point of view, the creation of the diagnostic mapping is typically considered a design-time activity.

Therefore, it would be good to implement a proxy for an actual Process that can stand in as the target of the relation to a Process at design time. This semantics is realized by meta-class ProcessDesign.

[TPS_MANI_01228]{DRAFT} **Semantics of meta-class ProcessDesign** ⌈Meta-class ProcessDesign shall be used whenever a design-time representation is required for a Process that is designed in a **later** step in the workflow as part of the deployment specification.⌋*(RS_MANI_00004)*

The integrator would have to take care that an actual Process refers to the corresponding ProcessDesign such that by means of this reference an AUTOSAR software tool is able to figure out the relation between a diagnostic mapping and a process, provided that each ProcessDesign is **only** referenced by a single Process.

[constr_1550]{DRAFT} **Reference from Process to ProcessDesign** ⌈Each ProcessDesign shall only be referenced from a single Process **at the time when the creation of the manifest is finished**⌋*()*

Note that the reference from the Process to the ProcessDesign acknowledges the fact that the Process is typically created later in time[3].

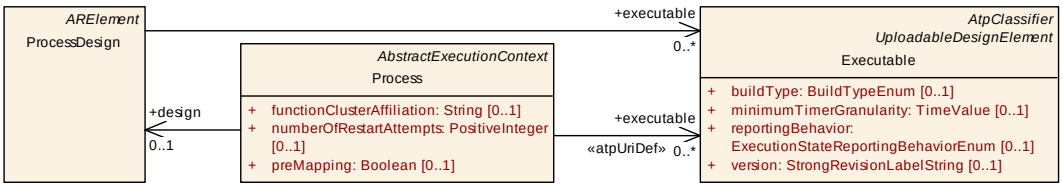


Figure 3.12: Modeling of the ProcessDesign

Class	**ProcessDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ProcessDesign			
Note	This meta-class has the ability to stand in for a Process at the time when the Process does not yet exist. But its future existence already needs to be considered during design phase and for that a dedicated model element is required.. **Tags:** atp.recommendedPackage=ProcessDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
executable	Executable	*	ref	Reference to executable that is executed in the process.

Table 3.30: ProcessDesign

[constr_3731]{DRAFT} **Upper multiplicity of reference in the role ProcessDesign.executable** ⌈In the context of ProcessDesign, the reference in the role executable shall exist at most once **at the time when the sub-system design is complete**.⌋*()*

[3]In other words, if references are needed between design-related and deployment-related meta-classes then the direction of these references shall always point from deployment to design.

Conceivably, the association of diagnostic mappings with Meta-class ProcessDesign may still happen as a finalizing last step of the activity to create the diagnostic mappings. To accommodate for this potential modeling, the reference from a diagnostic mapping to ProcessDesign has been decorated by stereotype ≪atpSplitable≫.

For more information concerning the semantics of this stereotype please refer to the specification of the AUTOSAR Generic Structure Template [6].

[constr_1693]{DRAFT} **Relation of Executable, ProcessDesign, and Process** ⌈Any Executable that is referenced by a ProcessDesign shall also be referenced by every Process that references the ProcessDesign **at the time when the creation of the manifest is finished**.⌋*()*

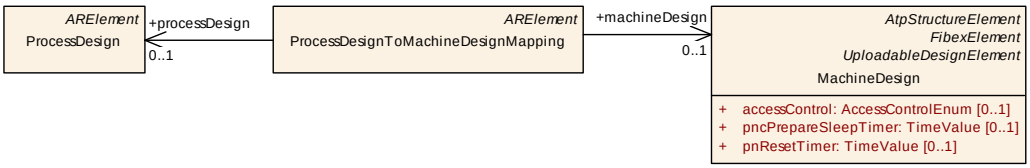


Figure 3.13: Modeling of the ProcessDesignToMachineDesignMapping

[TPS_MANI_01229]{DRAFT} **Pre-allocation of a given ProcessDesign on a specific MachineDesign** ⌈It is also possible to pre-allocate a given ProcessDesign on a specific MachineDesign. For this purpose meta-class ProcessDesignToMachineDesignMapping exists.⌋*(RS_MANI_00004)*

The semantics of meta-class MachineDesign is explained in section 6.

Class	**ProcessDesignToMachineDesignMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This element is used in the design phase to predefine a mapping of a process to a machine. Such a mapping may be overruled in the deployment phase. **Tags:** atp.recommendedPackage=ProcessDesignToMachineDesignMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
machineDesign	MachineDesign	0..1	ref	This reference identifies the MachineDesign in the context of the ProcessDesignToMachineDesignMapping.
processDesign	ProcessDesign	0..1	ref	This reference identifies the ProcessDesign in the context of the ProcessDesignToMachineDesignMapping.

Table 3.31: ProcessDesignToMachineDesignMapping

[constr_10154]{DRAFT} **Multiplicity of reference in the role ProcessDesignToMachineDesignMapping.processDesign** ⌈For each ProcessDesignToMachineDesignMapping, the reference in the role processDesign shall exist **at the time when the ProcessDesign is complete**.⌋*()*

Please note that an intended ProcessDesignToMachineDesignMapping may not be possible for utilization of the target machine and therefore a different ProcessToMachineMapping may be created in the deployment phase.

3.3 Port Interface

This section contains a documentation of all relevant PortInterfaces.

3.3.1 Service Interface

3.3.1.1 Overview

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

- aggregation of VariableDataPrototype in the role event with
- aggregation of meta-class Field in the role field with
- aggregation of ClientServerOperation in the role method
- aggregation of Trigger in the role trigger

within the same ServiceInterface.⌋*(RS_MANI_00001, RS_MANI_00003)*

The purpose of this modeling is to embrace the concept of service-oriented communication [4] and better support this paradigm for communication on the *AUTOSAR adaptive platform*.

Please note that, in terms of semantics, the ApApplicationError represents a sort of 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 3.3.1.8.

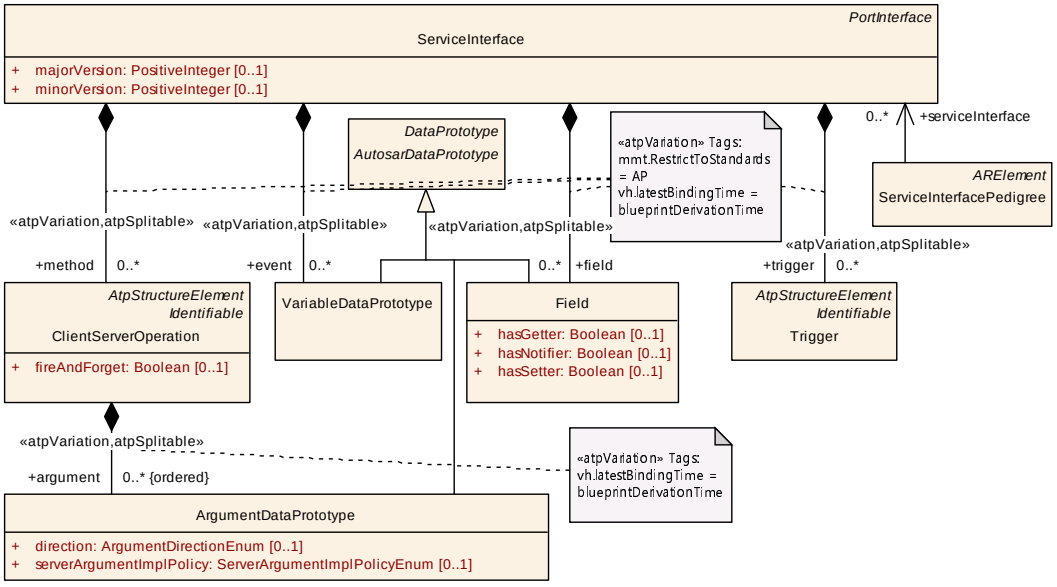


Figure 3.14: Modeling of the ServiceInterface

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

This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

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.recommendedPackage=ServiceInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
event	VariableDataPrototype	*	aggr	This represents the collection of events defined in the context of a ServiceInterface. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=event.shortName, event.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
field	Field	*	aggr	This represents the collection of fields defined in the context of a ServiceInterface. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=field.shortName, field.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=40
majorVersion	PositiveInteger	0..1	attr	Major version of the service contract. **Tags:** xml.sequenceOffset=10
method	ClientServerOperation	*	aggr	This represents the collection of methods defined in the context of a ServiceInterface. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=method.shortName, method.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=50
minorVersion	PositiveInteger	0..1	attr	Minor version of the service contract. **Tags:** xml.sequenceOffset=20

▽

△

Class	ServiceInterface			
trigger	Trigger	*	aggr	This represents the collection of triggers defined in the context of a ServiceInterface. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=trigger.shortName, trigger.variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=60

Table 3.32: ServiceInterface

As already described in [TPS_SWCT_01844], AUTOSAR does not support the existence of optional `arguments` in a `ClientServerOperation`.

This restriction is motivated by the lack of support for optional arguments in the APIs of the RTE on the *AUTOSAR classic platform*. For the sake of interoperability between the *classic platform* and the *adaptive platform*, this restriction is observed on the *AUTOSAR adaptive platform* as well.

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

Please note that there is no obligation to have any `method`, `event`, `trigger`, 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.

3.3.1.2 Event

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

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

[constr_1494]{DRAFT} **Initial value for `event`** ⌈A `ServiceInterface.event` shall **not** have an `initValue` **`at the time before the generation of the ara API starts`**.⌋*()*

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

Table 3.33: VariableDataPrototype

3.3.1.3 Trigger

[TPS_MANI_03291]{DRAFT} **Semantics of `ServiceInterface.trigger`** ⌈A `trigger` represents a special kind of an event without any data that is transmitted from a server to one or more client(s) and at which occurrence the Service Consumer shall react in a particular manner.⌋ *(RS_MANI_00003)*

Class	**Trigger**			
Package	M2::AUTOSARTemplates::CommonStructure::TriggerDeclaration			
Note	The Trigger represents a special kind of an event (without data) at which occurrence the Service Consumer shall react in a particular manner.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, BswModuleDescription.releasedTrigger, BswModuleDescription.required Trigger, ServiceInterface.trigger, TriggerInterface.trigger			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.34: Trigger

Please note that the `trigger` is processed in the queued manner, i.e. the `trigger`s are stored in a queue and are processed in “first in first out” order.

3.3.1.4 Field

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- `hasGetter`
- `hasSetter`
- `hasNotifier`

shall exist and at least one of the attributes shall be set to `true` **at the time before the generation of the ara API starts**.⌋*()*

Please note that [constr_1673] allows that a `Field` may be defined with a notifier but without the two defined methods `get()` and `set()`.

As described above a subscriber to a `field` notifier will get the current value of the `Field` immediately after the subscription.

This functionality makes a `Field` without `get()`/`set()` methods useful in some functional cases compared to the usage of an `event` where the value would only be sent after the event is triggered.

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.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationInterface.attribute, *AtpClassifier*.atpFeature, ServiceInterface.field			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
hasGetter	Boolean	0..1	attr	This attribute controls whether read access is foreseen to this field.
hasNotifier	Boolean	0..1	attr	This attribute controls whether a notification semantics is foreseen to this field.
hasSetter	Boolean	0..1	attr	This attribute controls whether write access is foreseen to this field.

Table 3.35: Field

[constr_10129]{DRAFT} **Multiplicity of attribute `Field.hasGetter`** ⌈For each `Field`, the attribute `hasGetter` shall exist **at the time before the generation of the ara API starts**.⌋*()*

[constr_10130]{DRAFT} **Multiplicity of attribute `Field.hasSetter`** ⌈For each `Field`, the attribute `hasSetter` shall exist **at the time before the generation of the ara API starts**.⌋*()*

[constr_10131]{DRAFT} **Multiplicity of attribute `Field.hasNotifier`** ⌈For each `Field`, the attribute `hasNotifier` shall exist **at the time before the generation of the ara API starts**.⌋*()*

3.3.1.5 Method

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

Class	**ClientServerOperation**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An operation declared within the scope of a client/server interface.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationInterface.command, *AtpClassifier*.atpFeature, ClientServerInterface.operation, DiagnosticDataElementInterface.read, DiagnosticDataIdentifierInterface.read, DiagnosticDataIdentifierInterface.write, DiagnosticRoutineInterface.requestResult, DiagnosticRoutineInterface.start, DiagnosticRoutineInterface.stop, PhmRecoveryActionInterface.recovery, ServiceInterface.method			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument (ordered)	ArgumentDataPrototype	*	aggr	An argument of this ClientServerOperation **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=argument.shortName, argument.variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime
fireAndForget	Boolean	0..1	attr	This attribute defines whether this method is a fire&forget method (true) or not (false).
possibleApError	ApApplicationError	*	ref	This reference identifies AdaptivePlatformApplication Errors as a possible error raised by the enclosing Client ServerOperation.
possibleApError Set	ApApplicationErrorSet	*	ref	This reference represents the ability to refer to an entire group of ApApplicationErrors as one model element instead of having to refer to all the represented Ap ApplicationErrors separately.

Table 3.36: ClientServerOperation

3.3.1.5.1 Fire and Forget Method

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

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

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

In other words, the “fire & forget” method conveys the data and the occurrence of the data **from a client to a server**. For comparison, the `event` is used to convey

information in combination with the occurrence of the information from **a server to a client**.

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

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

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

[constr_3374]{DRAFT} **`method` with attribute `fireAndForget` set to true shall not have any inout or out arguments** ⌈A `method` that has the value of attribute `fireAndForget` set to `true` is not allowed to have any `argument`s with `direction` `inout` or `out` **at the time before the generation of the ara API starts**.⌋*()*

[constr_3375]{DRAFT} **`method` with attribute `fireAndForget` set to true shall not reference an `ApApplicationError`** ⌈A `method` that has the value of attribute `fireAndForget` set to `true` is not allowed to reference

- an `ApApplicationError` in role `possibleApError` and/or
- an `ApApplicationErrorSet` in the role `possibleApErrorSet`.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

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

3.3.1.6 Versioning of `ServiceInterface`s

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

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

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

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

It is expected that if using different versions of the same ServiceInterface in one Executable then different *namespace*s shall be used for each ServiceInterface version.

The attributes ServiceInterface.majorVersion and ServiceInterface.minorVersion provide the possibility to define version information at the level of the ServiceInterface.

[TPS_MANI_03616]{DRAFT} **Semantic versioning of ServiceInterface.majorVersion and ServiceInterface.minorVersion** ⌈Service contract versioning rules:

- for backwards-incompatible interface or behavior changes the majorVersion number shall be increased and the minorVersion number shall be set to 0
- for backwards-compatible interface or behavior changes the majorVersion number shall be unchanged and the minorVersion number shall be increased.

⌋*(RS_MANI_00064)*

Note that it is expected that the decision about backwards compatibility is made by the service designer. In other words AUTOSAR does not define formal criteria for the backwards compatibility of ServiceInterfaces.

As for the modeling of several versions of a ServiceInterface, the fully qualified shortNames of the ServiceInterfaces have to be different. The ServiceInterfacePedigree allows to collect the set of ServiceInterfaces which form the collection of different versions of the same *Service*.

Class	**ServiceInterfacePedigree**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	Collection of ServiceInterfaces that belong to the same versioning. **Tags:** atp.recommendedPackage=ServiceInterfacePedigrees			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
serviceInterface	ServiceInterface	*	ref	Reference to the ServiceInterfaces which belong to the same versioning.

Table 3.37: ServiceInterfacePedigree

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.

3.3.1.6.1 Versioning driven by 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 ServiceInterfaces 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.

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

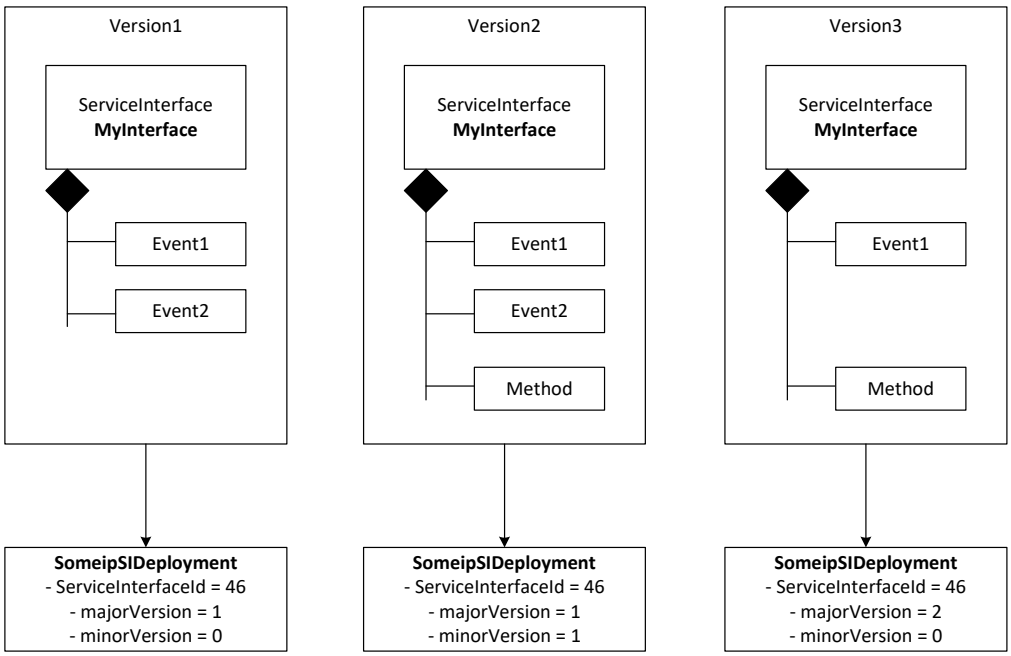


Figure 3.15: Example for different versions of the same ServiceInterface

During the ServiceInterfaceDeployment the ServiceInterface is mapped to a middleware transport layer where the necessary middleware transport layer specific configuration settings are performed, as described in chapter 11.2.

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

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

Please note that the compatibility rules for SOME/IP are described in [9].

3.3.1.7 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 3.16.

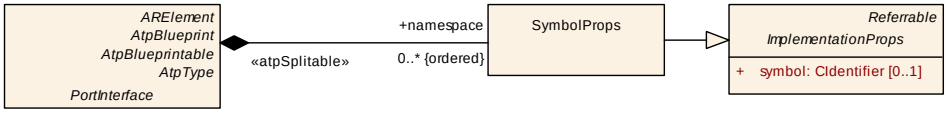


Figure 3.16: Specification of namespaces in PortInterfaces

[TPS_MANI_01004]{DRAFT} **Semantics of ServiceInterface.namespace** ⌈The aggregation ServiceInterface.namespace shall be used to define the namespace

to be used for the source code that corresponds to the given ServiceInterface.⌋ *(RS_MANI_00003)*

```
<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>
```

Listing 3.1: Example for the definition of a namespace for a given ServiceInterface

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

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

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

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

This means that the order of individual elements to the collection of namespaces has a semantical relevance[4].

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

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

[4]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*			
Subclasses	*AbstractRawDataStreamInterface*, *AbstractSynchronizedTimeBaseInterface*, ClientServerInterface, *CryptoInterface*, *DataInterface*, *DiagnosticPortInterface*, FirewallStateSwitchInterface, *IdsmAbstractPortInterface*, LogAndTraceInterface, ModeSwitchInterface, NetworkManagementPortInterface, *PersistencyInterface*, *PlatformHealthManagementInterface*, ServiceInterface, *StateManagementPortInterface*, TriggerInterface			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**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=namespace.shortName

Table 3.38: PortInterface

Class	**SymbolProps**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This meta-class represents the ability to contribute a part of a namespace.			
Base	*ARObject*, *ImplementationProps*, *Referrable*			
Aggregated by	Allocator.namespace, ApApplicationErrorDomain.namespace, *AtomicSwComponentType*.symbolProps, *CppImplementationDataType*.namespace, ImplementationDataType.symbolProps, *PortInterface*.namespace, SecurityEventDefinition.eventSymbolName			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.39: SymbolProps

The Listing 3.1 exemplifies the statement made by [TPS_MANI_01006], i.e. the resulting name space in e.g. C++ would look like sketched in Listing 3.2.

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

        }
    }
}
```

Listing 3.2: Resulting namespace for the example `ServiceInterface`

3.3.1.8 Error Handling

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

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

[TPS_MANI_01190]{DRAFT} **Semantics of `ApApplicationError`** ⌈Meta-class `ApApplicationError` represents the ability to define the existence of an error during the execution of a `method` independently of the scope of a `ServiceInterface` or `ClientServerOperation`.⌋*(RS_MANI_00002)*

Class	**ApApplicationError**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the ability to formally specify the semantics of an application error on the AUTOSAR adaptive platform **Tags:** atp.recommendedPackage=ApplicationErrors			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
errorCode	Integer	0..1	attr	This attribute has the ability to specify the error code value within the enclosing AdaptivePlatformApplication Error.
errorDomain	ApApplicationError Domain	0..1	ref	This reference represents the error domain of the Ap ApplicationError.

Table 3.40: ApApplicationError

[constr_10132]{DRAFT} **Multiplicity of attribute `ApApplicationError.errorCode`** ⌈For each `ApApplicationError`, the attribute `errorCode` shall exist **at the time before the generation of the ara API starts**.⌋*()*

[constr_1625]{DRAFT} **Existence of reference `ApApplicationError.errorDomain`** ⌈For each `ApApplicationError`, the reference `errorDomain` shall exist **at the time before the generation of the ara API starts**.

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

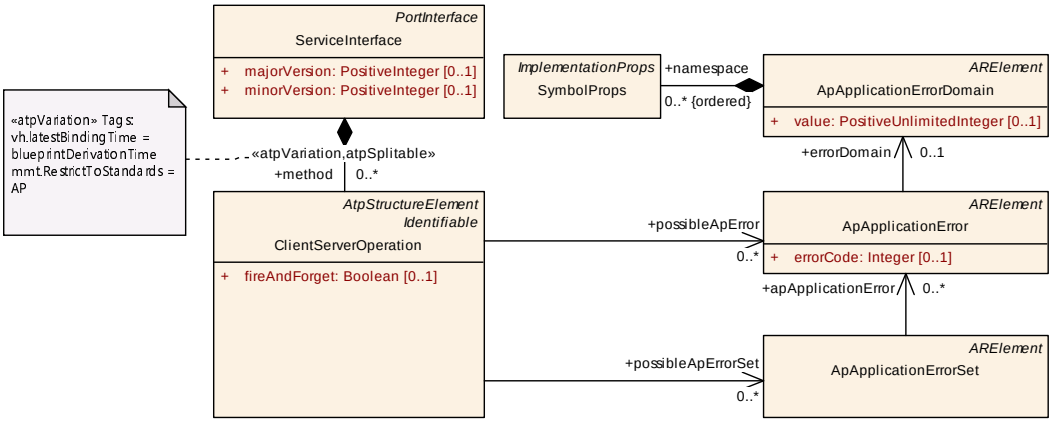


Figure 3.17: Modeling of `ApApplicationError` on the *AUTOSAR adaptive platform*

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

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

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

The decision whether an ApApplicationErrorSet is defined and referenced from specific ClientServerOperations has to be done on an individual basis. AUTOSAR just wants to make this business as straightforward as possible.

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

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

Table 3.41: ApApplicationErrorSet

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

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

- zero or more ApApplicationError(s) in the role possibleApError and/or
- zero or more ApApplicationErrorSet(s) in the role possibleApErrorSet

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

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

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

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

[TPS_MANI_01863]{DRAFT} **Supported value range for attribute ApApplicationErrorDomain.value** ⌈The supported value range of attribute ApApplicationErrorDomain.value is given by [constr_2637].⌋(*RS_MANI_00026*)

Please note that [constr_2637] can be found in the document TPS Generic Structure Template [6].

[constr_10133]{DRAFT} **Multiplicity of attribute ApApplicationErrorDomain.value** ⌈For each ApApplicationErrorDomain, the attribute value shall exist **at the time before the generation of the ara API starts**.⌋()

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

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

Class	**ApApplicationErrorDomain**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the ability to define a global error domain for an ApApplicationError. **Tags:** atp.recommendedPackage=ApplicationErrorDomains			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
namespace (ordered)	SymbolProps	*	aggr	This aggregation defines the namespace of the Ap ApplicationErrorDomain
value	PositiveUnlimitedInteger	0..1	attr	This attribute identifies the error category.

Table 3.42: ApApplicationErrorDomain

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

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

3.3.1.9 Service Interface Data Type Mapping

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

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

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

It is therefore necessary to provide the definition of an AbstractImplementationDataType for each of the used ApplicationDataType. In the meta-model, this correspondence is implemented by means of the meta-class DataTypeMappingSet[5].

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

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

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

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

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

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

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

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

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

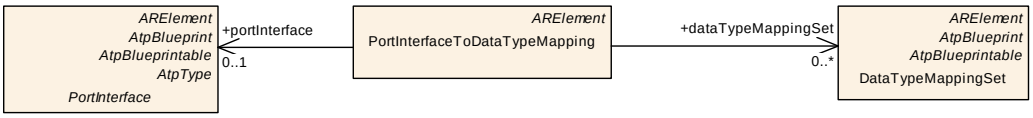


Figure 3.18: Modeling of `PortInterfaceToDataTypeMapping`

[constr_1507]{DRAFT} **`PortInterfaceToDataTypeMapping` is only applicable to `ServiceInterface` or `PersistencyKeyValueStorageInterface`** ⌈`PortInterfaceToDataTypeMapping.portInterface` shall only refer to **either** a `ServiceInterface` **or** a `PersistencyKeyValueStorageInterface` **at the time before the generation of the ara API starts**.⌋*()*

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.recommendedPackage=PortInterfaceToDataTypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataType MappingSet	DataTypeMappingSet	*	ref	This represents the reference to the applicable data TypemappingSet **Tags:** atp.StatusComment=Reserved for adaptive platform
portInterface	PortInterface	0..1	ref	This represents the reference to the applicable Port Interface **Tags:** atp.StatusComment=Reserved for adaptive platform

Table 3.43: PortInterfaceToDataTypeMapping

[constr_10134]{DRAFT} **Multiplicity of reference in the role `PortInterfaceToDataTypeMapping.dataTypeMappingSet`** ⌈For each `PortInterfaceToDataTypeMapping`, the reference in the role `dataTypeMappingSet` shall exist at least once **at the time before the generation of the ara API starts**.⌋*()*

[constr_10135]{DRAFT} **Multiplicity of reference in the role `PortInterfaceToDataTypeMapping.portInterface`** ⌈For each `PortInterfaceToDataTypeMapping`, the reference in the role `portInterface` shall exist **at the time before the generation of the ara API starts**.⌋*()*

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*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataTypeMap	DataTypeMap	*	aggr	This is one particular association between an Application DataType and its AbstractImplementationDataType.
modeRequest TypeMap	ModeRequestTypeMap	*	aggr	This is one particular association between an Mode DeclarationGroup and its AbstractImplementationData Type.

Table 3.44: DataTypeMappingSet

Class	**DataTypeMap**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	This class represents the relationship between ApplicationDataType and its implementing Abstract ImplementationDataType.			
Base	*ARObject*			
Aggregated by	DataTypeMappingSet.dataTypeMap			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
applicationData Type	ApplicationDataType	0..1	ref	This is the corresponding ApplicationDataType
implementation DataType	AbstractImplementation DataType	0..1	ref	This is the corresponding AbstractImplementationData Type.

Table 3.45: DataTypeMap

3.3.1.10 Communication Group pattern

NOTE: *Communication Groups have been set to obsolete and will be removed or replaced in the next AUTOSAR release(s).*

The Communication Group defines a specific pattern of the usage of a `ServiceInterface` in a bi-directional way.

The details can be found in SWS_CommunicationManagement [10]. In order to define a Communication Group several `ServiceInterface.category` values are defined.

[TPS_MANI_03628]{OBSOLETE} **Standardized values of `ServiceInterface.category`** ⌈The AUTOSAR Standard reserves the following values for attribute `ServiceInterface.category`:

- `COMMUNICATION_GROUP`
- `COMMUNICATION_GROUP_SERVER`
- `COMMUNICATION_GROUP_CLIENT`

It is possible to use a custom, non-standardized value for the attribute ServiceInterface.category but this option comes with the obligation to use a value that is guaranteed to not clash with possible future extensions of the collection of standardized values, e.g. use company name in the category value.⌋(*RS_MANI_00003*)

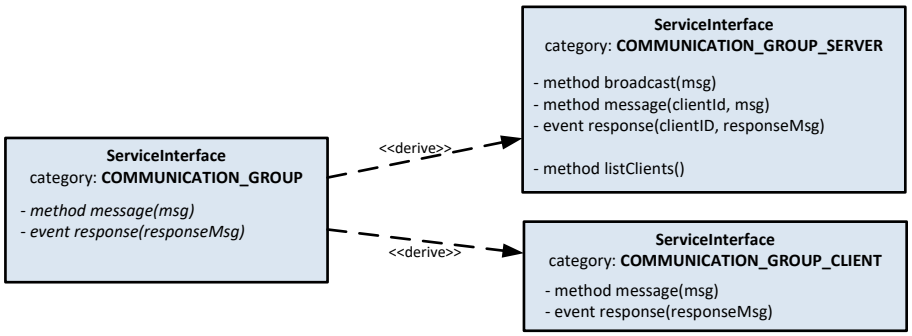


Figure 3.19: Example of Communication Group categories

The general idea of the Communication Group pattern is that a ServiceInterface of category COMMUNICATION_GROUP is created to describe the information to be transported (the msg and responseMsg data types). There will not be any instance of this ServiceInterface of category COMMUNICATION_GROUP in the system, it is just a design artifact.

Out of the ServiceInterface of category COMMUNICATION_GROUP the two ServiceInterfaces for the server (category COMMUNICATION_GROUP_SERVER) and the client (category COMMUNICATION_GROUP_CLIENT) roles are created. The rules how this creation shall be done are defined in SWS_CommunicationManagement [10].

3.3.2 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.

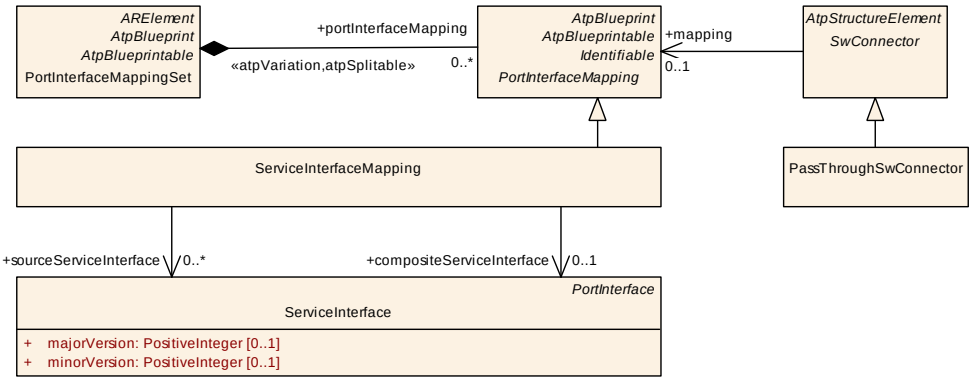


Figure 3.20: Modeling of the `ServiceInterfaceMapping`

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

- `ServiceInterface`s explicitly designed for `VFB`-based communication and
- `ServiceInterface`s explicitly designed for network-based communication

meta-class `ServiceInterfaceMapping` is available to map

- (fine-grained) `ServiceInterface`s for the `VFB`-communication to
- (coarse-grained) `ServiceInterface`s for network communication.

⌋*(RS_MANI_00017)*

[TPS_MANI_01032]{DRAFT} **Usage of `ServiceInterfaceMapping`** ⌈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 `PortPrototype`s for each of the `ServiceInterface`s.

Other `SwComponentPrototype`s could then "connect" to the `PortPrototype`s typed by `ServiceInterface`s referenced in the role `sourceServiceInterface`.

This means that the `PortPrototype` typed by the `ServiceInterface` referenced in the role `compositeServiceInterface` is used for external communication.

`PassThroughSwConnector`s can be used to describe in the modeled *facade* `CompositionSwComponentType` which "fine-grained" Ports are combined to a "coarse-grained" Port that is used for network communication. The mapping of Service Interface elements of the "fine-grained" Ports to the Service Interface elements of the "course-grained" Port is described with the `ServiceInterfaceMapping` or rather `ServiceInterfaceElementMapping`.⌋*(RS_MANI_00017)*

Please note that the modeling of a *facade* `SwComponentType` does not make any assumptions about the implementation and about the realization of such a *facade* functionality. The *facade* may be realized by an Adaptive Software Component/Application or it may be realized by a "Network-Daemon". AUTOSAR does not define any instructions for the implementation of such a functionality and the decision is project specific.

The behavioral aspects of such a “facade” (e.g. when is the coarse-grained ServiceInstance offered) are also project-specific and are not predefined by AUTOSAR.

Figure 3.21 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 SWC2 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.

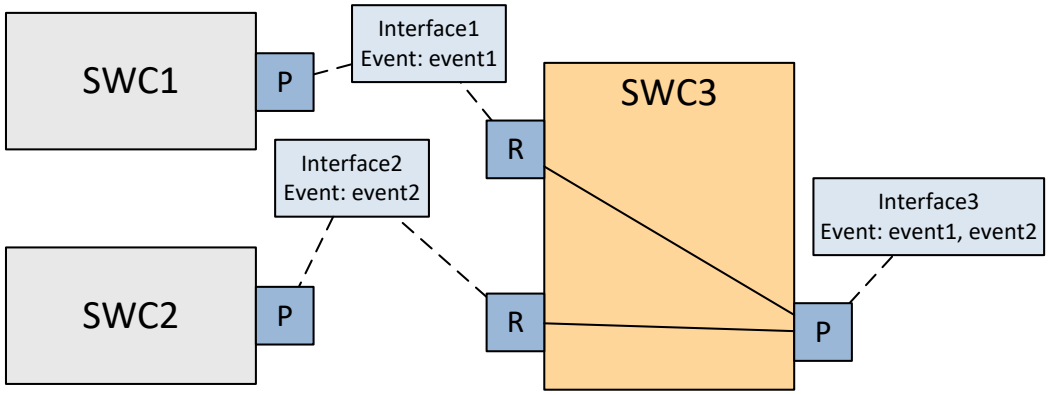


Figure 3.21: Concept of a facade software-component

[constr_5056]{DRAFT} **Restriction of sub-class of CompositionSwComponentType.connector** ⌈In the context of a CompositionSwComponentType.connector (transitively) referenced by a Executable.rootSwComponentPrototype, the only supported sub-class of SwConnector is PassThroughSwConnector.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

[constr_5057]{DRAFT} **PassThroughSwConnector and ServiceInterfaceMapping** ⌈If a PassThroughSwConnector is defined between two Ports in a CompositionSwComponentType either:

- a ServiceInterfaceMapping between the ServiceInterfaces of these two Ports shall be defined and the PassThroughSwConnector shall reference the relevant ServiceInterfaceMapping in the role mapping or
- ServiceInterfaceElementMappings for elements of ServiceInterfaces of the two Ports shall be defined and the PassThroughSwConnector shall reference the relevant ServiceInterfaceElementMappings in the role serviceInterfaceElementMapping.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

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

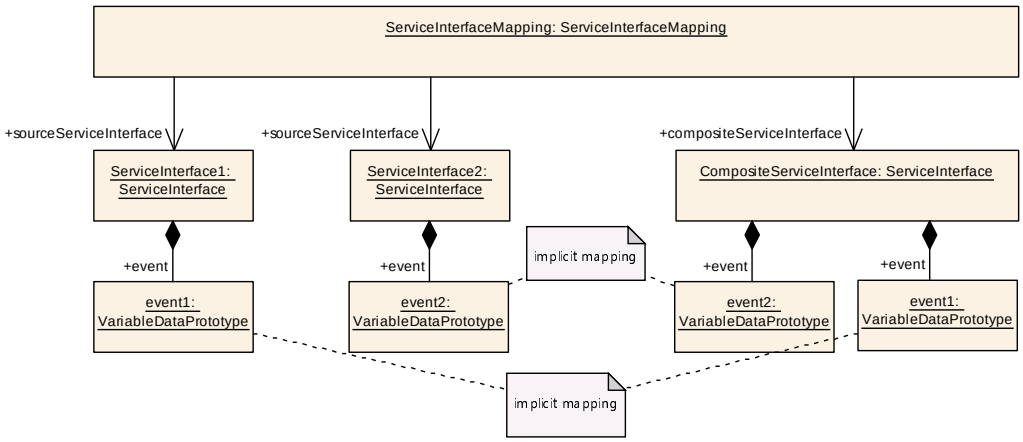


Figure 3.22: Example for the application of a `ServiceInterfaceMapping`

The regulation stated in [TPS_MANI_01022] is exemplified in Figure 3.22.

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 `ServiceInterface`s 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).

[TPS_MANI_01003]{DRAFT} **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)*

Class	**ServiceInterfaceMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping
Note	Specifies one ServiceInterfaceMapping that allows to define that a ServiceInterface is composite of several other ServiceInterfaces. **Tags:** atp.recommendedPackage=ServiceInterfaceMappings

▽

△

Class	**ServiceInterfaceMapping**			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *PortInterfaceMapping*, *Referrable*			
Aggregated by	PortInterfaceMappingSet.portInterfaceMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compositeServiceInterface	ServiceInterface	0..1	ref	This represents the composite ServiceInterface.
sourceServiceInterface	ServiceInterface	*	ref	ServiceInterface that is mapped into the composite ServiceInterface.

Table 3.46: ServiceInterfaceMapping

[constr_10136]{DRAFT} **Multiplicity of reference in the role `ServiceInterfaceMapping.compositeServiceInterface`** ⌈For each `ServiceInterfaceMapping`, the reference in the role `compositeServiceInterface` shall exist **`at the time before the generation of the ara API starts`**.⌋*()*

[constr_10137]{DRAFT} **Multiplicity of reference in the role `ServiceInterfaceMapping.sourceServiceInterface`** ⌈For each `ServiceInterfaceMapping`, the reference in the role `sourceServiceInterface` shall exist at least once **`at the time before the generation of the ara API starts`**.⌋*()*

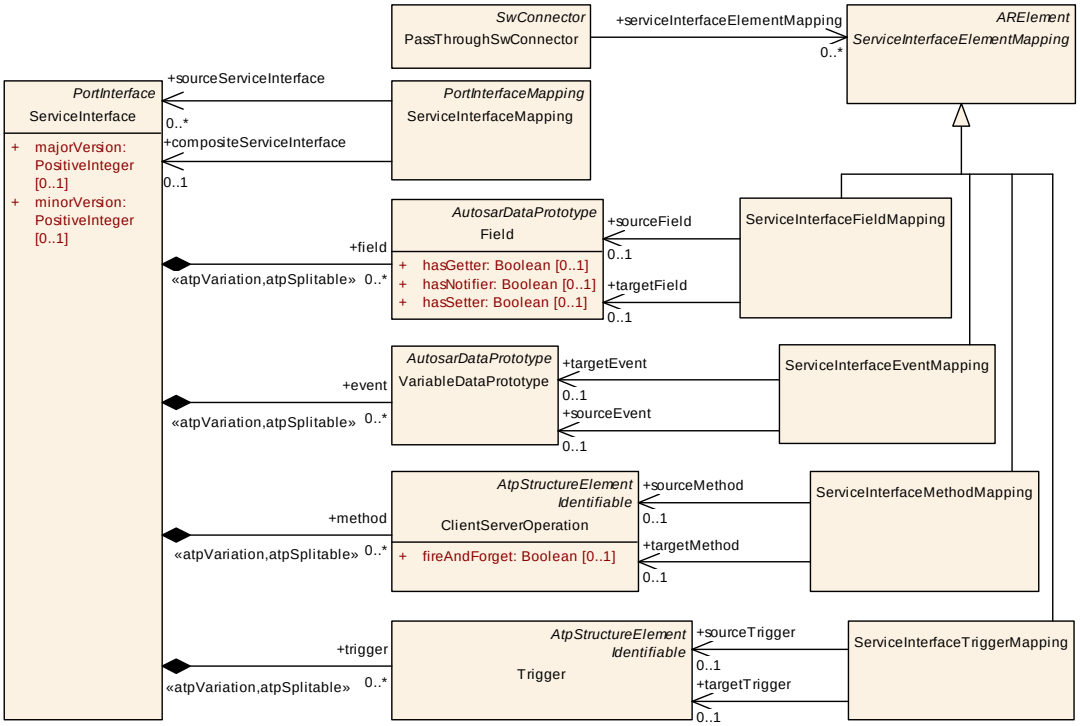


Figure 3.23: Overview of the modeling of the `ServiceInterfaceMapping` and `ServiceInterfaceElementMapping`

Note that the `ServiceInterfaceMapping` is not an up-front association (by means of `SwConnector`s) between communication ends in the sense of section 3.3.1.6.

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].

Class	**PassThroughSwConnector**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	This kind of SwConnector can be used inside a CompositionSwComponentType to connect two delegation PortPrototypes.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SwConnector*			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.connector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
providedOuterPort	AbstractProvidedPortPrototype	0..1	ref	This represents the provided outer delegation Port Prototype of the PassThroughSwConnector.
requiredOuterPort	AbstractRequiredPortPrototype	0..1	ref	This represents the required outer delegation Port Prototype of the PassThroughSwConnector.
serviceInterfaceElementMapping	ServiceInterfaceElementMapping	*	ref	Reference to a ServiceInterfaceElementMapping specifying the mapping of unequal named Service Interface elements of the two different ServiceInterfaces typing the two PortPrototypes which are referenced by the PassThroughSwConnector.

Table 3.47: PassThroughSwConnector

3.3.3 Service Interface Element Mapping

3.3.3.1 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.

An example of the definition of a ServiceInterfaceEventMapping can be found in Appendix B.2.2.1.

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

- the given pair of ServiceInterfaces is referenced by a ServiceInterfaceMapping, where one ServiceInterface is referenced in the role sourceServiceInterface and the other ServiceInterface is referenced in the role compositeServiceInterface.
- an arbitrary mixture of the following options exists:
  - an event aggregated by one of the given ServiceInterfaces is referenced by a ServiceInterfaceEventMapping in the role sourceEvent and one events aggregated by the other given ServiceInterface is referenced by the same ServiceInterfaceEventMapping in the role targetEvent.
  - a trigger aggregated by one of the given ServiceInterfaces is referenced by a ServiceInterfaceTriggerMapping in the role sourceTrigger and one trigger aggregated by the other given ServiceInterface is referenced by the same ServiceInterfaceTriggerMapping in the role targetTrigger.
  - 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.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

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.

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).

Class	***ServiceInterfaceElementMapping*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	This abstract meta-class acts as base class for the mapping of specific elements of a ServiceInterface.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	ServiceInterfaceEventMapping, ServiceInterfaceFieldMapping, ServiceInterfaceMethodMapping, ServiceInterfaceTriggerMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.48: ServiceInterfaceElementMapping

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

[TPS_MANI_03277]{DRAFT} **`ServiceInterfaceElementMapping`s for a subset of elements of a single `ServiceInterface`** ⌈If elements of a source `ServiceInterface` are mapped to target `ServiceInterface` elements by `ServiceInterfaceElementMapping`s it is allowed that only a subset of those source `ServiceInterface` elements are mapped to the target `ServiceInterface` elements.⌋ *(RS_MANI_00017)*

With [TPS_MANI_03277] use cases are supported as shown in Figure 3.24 where the event4 and method2 of ServiceInterface2 that is provided by AppM1_2 are used only locally on the Machine1 and event3 of the same `ServiceInterface` is provided together with the ServiceInterface1 elements as new SOME/IP Service over the network.

Please note that the modeling of a facade `SwComponentType` does not make any assumptions about the implementation and about the realization of such a facade functionality.

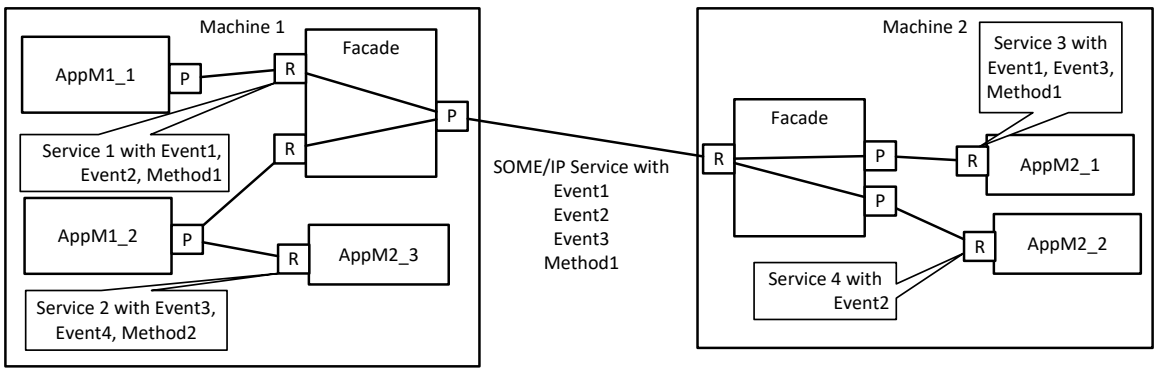


Figure 3.24: Example for `ServiceInterfaceElementMapping` for a subset of elements of a single `ServiceInterface` on the provider side

AUTOSAR does not define any instructions for the implementation of such a functionality and the decision is project specific.

The behavioral aspects of such a “facade” (e.g. when is the coarse-grained `ProvidedApServiceInstance` offered) are also project-specific and are not predefined by AUTOSAR.

Similar use cases are applicable on the Consumer side as well as shown in Figure 3.25 where the Facade component is the service consumer of a SOME/IP service with event1, event2 and event3, and method1 but only event1, event3 and method1 are used by Software that is deployed on Machine2.

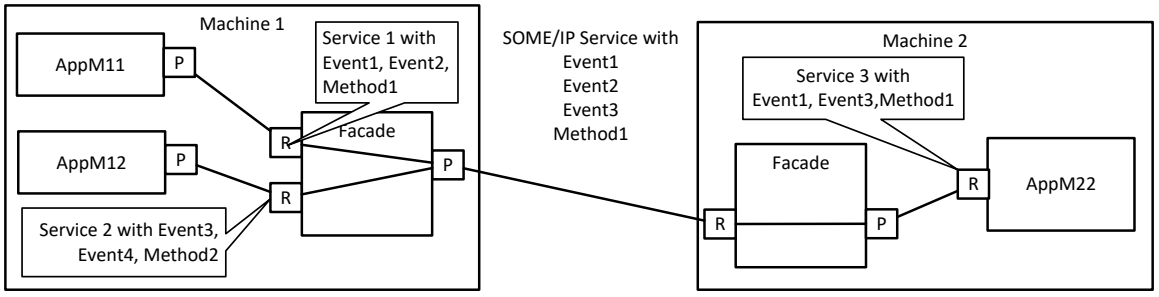


Figure 3.25: Example for `ServiceInterfaceElementMapping` for a subset of elements of a single `ServiceInterface` on the consumer side

3.3.3.2 Service Interface Event Mapping

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

Figure 3.26: Modeling of the `ServiceInterfaceEventMapping`

Class	**ServiceInterfaceEventMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	This meta-class allows to define a mapping between events of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:** atp.recommendedPackage=ServiceInterfaceElementMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *ServiceInterfaceElementMapping*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	ServiceInterfaceEventMapping			
sourceEvent	VariableDataPrototype	0..1	ref	Reference to an event that is contained in the source ServiceInterface.
targetEvent	VariableDataPrototype	0..1	ref	Reference to an event that is contained in the composite ServiceInterface.

Table 3.49: ServiceInterfaceEventMapping

[constr_10138]{DRAFT} **Multiplicity of reference in the role ServiceInterfaceEventMapping.sourceEvent** ⌈For each ServiceInterfaceEventMapping, the reference in the role sourceEvent shall exist **at the time before the generation of the ara API starts**.⌋*()*

[constr_10139]{DRAFT} **Multiplicity of reference in the role ServiceInterfaceEventMapping.targetEvent** ⌈For each ServiceInterfaceEventMapping, the reference in the role targetEvent shall exist **at the time before the generation of the ara API starts**.⌋*()*

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

It is also possible to map a given event of a given ServiceInterface multiple times in different roles to the ServiceInterface that aggregates the targetEvent, as exemplified by Figure B.17.

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 B.24.

3.3.3.3 Service Interface Trigger Mapping

[TPS_MANI_03289]{DRAFT} **Semantics of ServiceInterfaceTriggerMapping** ⌈Meta-class ServiceInterfaceTriggerMapping has the ability to map a ServiceInterface.trigger referenced in the role sourceTrigger explicitly to another ServiceInterface.trigger referenced in the role targetTrigger.⌋*(RS_MANI_00017)*

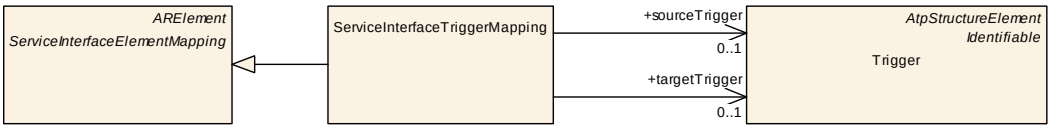


Figure 3.27: Modeling of the ServiceInterfaceTriggerMapping

Class	**ServiceInterfaceTriggerMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	This meta-class allows to define a mapping between triggers of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:** atp.recommendedPackage=ServiceInterfaceElementMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInterfaceElementMapping*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sourceTrigger	Trigger	0..1	ref	Reference to a trigger that is contained in the source ServiceInterface.
targetTrigger	Trigger	0..1	ref	Reference to a trigger that is contained in the target ServiceInterface.

Table 3.50: ServiceInterfaceTriggerMapping

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

It is also possible to map a given `trigger` of a given `ServiceInterface` multiple times in different roles to the `ServiceInterface` that aggregates the `targetTrigger`.

Please note that the mapping of one `sourceTrigger` to different `targetTrigger`s does **not** represent a *fan-out* of any kind.

It only means that the `sourceTrigger` will be used in different roles, as specified in the deployment.

3.3.3.4 Service Interface Field Mapping

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

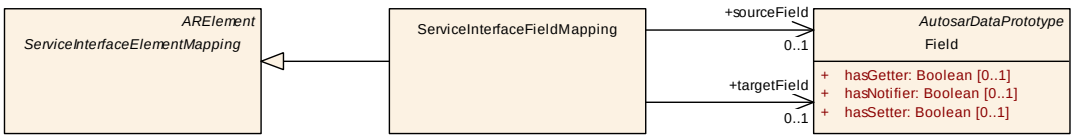


Figure 3.28: Modeling of the `ServiceInterfaceFieldMapping`

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 B.18.

Class	**ServiceInterfaceFieldMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ServiceInterfaceMapping			
Note	This meta-class allows to define a mapping between fields of ServiceInterfaces that are mapped to each other by the ServiceInterfaceMapping. **Tags:** atp.recommendedPackage=ServiceInterfaceElementMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInterfaceElementMapping*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sourceField	Field	0..1	ref	Reference to a field that is contained in the source ServiceInterface.
targetField	Field	0..1	ref	Reference to a field that is contained in the composite ServiceInterface.

Table 3.51: ServiceInterfaceFieldMapping

[constr_10140]{DRAFT} **Multiplicity of reference in the role `ServiceInterfaceFieldMapping.sourceField`** ⌈For each `ServiceInterfaceFieldMapping`, the reference in the role `sourceField` shall exist **`at the time before the generation of the ara API starts`**.⌋*()*

[constr_10141]{DRAFT} **Multiplicity of reference in the role `ServiceInterfaceFieldMapping.targetField`** ⌈For each `ServiceInterfaceFieldMapping`, the reference in the role `targetField` shall exist **`at the time before the generation of the ara API starts`**.⌋*()*

Please note that the mapping of one `sourceField` to different `targetField`s 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 `ServiceInterface`s works in Figure B.24.

3.3.3.5 Service Interface Method Mapping

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

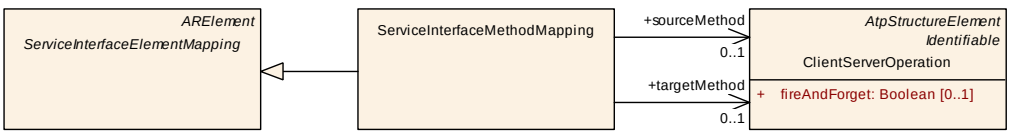


Figure 3.29: Modeling of the `ServiceInterfaceMethodMapping`

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.recommendedPackage=ServiceInterfaceElementMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInterfaceElementMapping*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sourceMethod	ClientServerOperation	0..1	ref	Reference to a method that is contained in the source ServiceInterface.
targetMethod	ClientServerOperation	0..1	ref	Reference to a method that is contained in the composite ServiceInterface.

Table 3.52: ServiceInterfaceMethodMapping

[constr_10142]{DRAFT} **Multiplicity of reference in the role ServiceInterfaceMethodMapping.sourceMethod** ⌈For each ServiceInterfaceMethodMapping, the reference in the role sourceMethod shall exist **at the time before the generation of the ara API starts**.⌋*()*

[constr_10143]{DRAFT} **Multiplicity of reference in the role ServiceInterfaceMethodMapping.targetMethod** ⌈For each ServiceInterfaceMethodMapping, the reference in the role targetMethod shall exist **at the time before the generation of the ara API starts**.⌋*()*

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

It is also possible to map a given method of a given ServiceInterface multiple times in different roles to the ServiceInterface that aggregates the targetMethod, as exemplified by Figure B.19.

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 B.24.

3.3.4 State Management Interface

3.3.4.1 Overview

PortInterfaces defined in the context of State Management are supposed to be used by the modeling of the State Management itself, i.e. the PortInterfaces described in this chapter shall be used to type PortPrototypes in the context of one or more Executable(s) that implement(s) the State Management.

There are two kinds of PortInterfaces used in the context of State Management:

- ServiceInterface, see section 3.3.1
- sub-classes of StateManagementPortInterface, see Figure 3.30

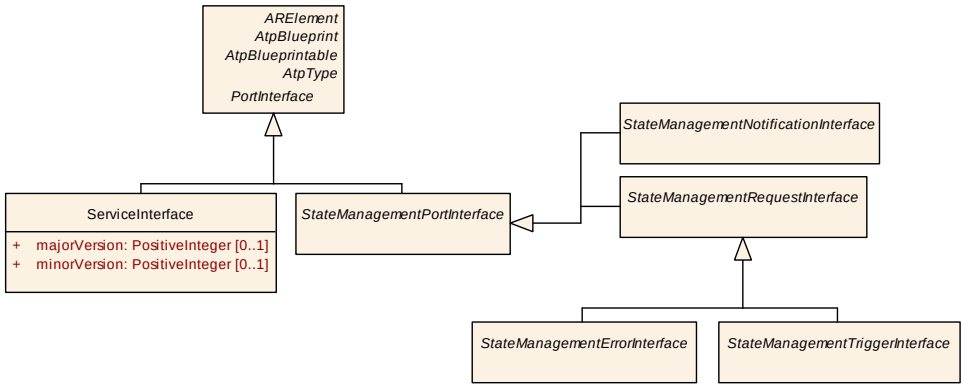


Figure 3.30: Big picture of State Management interface modeling

Class	***StateManagementPortInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This abstract class acts as a base class for PortInterfaces that are used in the context of state management on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	*StateManagementNotificationInterface*, *StateManagementRequestInterface*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.53: StateManagementPortInterface

Furthermore, the subclasses of StateManagementPortInterface are subdivided in PortInterfaces for the processing of requests to State Management (represented by abstract class StateManagementRequestInterface) and the notification of state switches, modeled as abstract class StateManagementNotification-Interface.

The “big picture” of the modeling of PortInterfaces for State Management is depicted in Figure 3.30.

In terms of requests, State Management interacts with a variety of function cluster modules and the interaction patterns, as well as the underlying APIs are very diverse. Therefore, it is necessary (for the use cases that don’t utilize ServiceInterfaces) to define a collection of different PortInterfaces separately for the two main request use cases:

- Processing of “triggers”, see section 3.3.4.3
- Processing of “errors”, see section 3.3.4.4

Class	***StateManagementNotificationInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::StateManagement			
Note	This meta-class acts as an abstract base class for state management notification interfaces. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementPortInterface*			
Subclasses	StateManagementFunctionGroupSwitchNotificationInterface			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.54: StateManagementNotificationInterface

Class	***StateManagementRequestInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::StateManagement			
Note	This meta-class acts as an abstract base class for state management request interfaces. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementPortInterface*			
Subclasses	*StateManagementErrorInterface*, *StateManagementTriggerInterface*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.55: StateManagementRequestInterface

Consequently, further abstract base classes are introduced to organize the definition of `PortInterface`s for the different State Management use cases:

- `StateManagementTriggerInterface`
- `StateManagementErrorInterface`

Class	***StateManagementTriggerInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	The usage of this meta-class for typing a PortPrototype indicates that the PortPrototype is used for the trigger provision in the context of state management on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementPortInterface*, *StateManagementRequestInterface*			
Subclasses	StateManagementDiagTriggerInterface			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.56: StateManagementTriggerInterface

Class	***StateManagementErrorInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::StateManagement			
Note	The usage of this meta-class for typing a PortPrototype indicates that the PortPrototype is used for the error provision in the context of state management on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementPortInterface*, *StateManagementRequestInterface*			
Subclasses	StateManagemenPhmErrorInterface, StateManagementEmErrorInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.57: StateManagementErrorInterface

3.3.4.2 State Management Interfaces for Notifications

As depicted by Figure 3.31, there are two kinds of notification that the State Management needs to support:

- Notification of `Function Group` state switch. This notification is typically directed at the execution management, it is formalized as `StateManagementFunctionGroupSwitchNotificationInterface`.
- Notification of the switch of an “internal” state, represented by a `field` owned by a `ServiceInterface`.

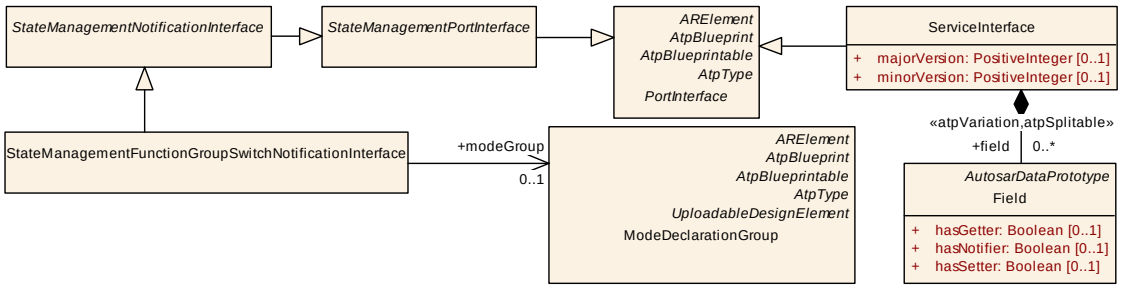


Figure 3.31: Modeling of `StateManagementPortInterface`s for state switch notifications

Class	**StateManagementFunctionGroupSwitchNotificationInterface**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::StateManagement
Note	The usage of this meta-class for typing a PortPrototype indicates that the PortPrototype is used for sending out a notification of a function group state change in the context of state management on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft atp.recommendedPackage=StateManagementPortInterfaces

△

Class	**StateManagementFunctionGroupSwitchNotificationInterface**			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementNotificationInterface*, *StateManagementPortInterface*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
modeGroup	ModeDeclarationGroup	0..1	ref	This reference identifies the ModeDeclarationGroup that defines the individual states that that can be switched to. **Tags:** atp.Status=draft

Table 3.58: StateManagementFunctionGroupSwitchNotificationInterface

[constr_10389]{DRAFT} **Existence of attribute `StateManagementFunctionGroupSwitchNotificationInterface.modeGroup`** ⌈For each `StateManagementFunctionGroupSwitchNotificationInterface`, the aggregation in the role `modeGroup` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

3.3.4.3 State Management Interfaces for Triggers

In the majority of cases, communication of triggers is done by means of `ServiceInterface`. the notable exception from this rule is the diagnostics management, which uses a dedicated subclass of `StateManagementTriggerInterface`: `StateManagementDiagTriggerInterface`.

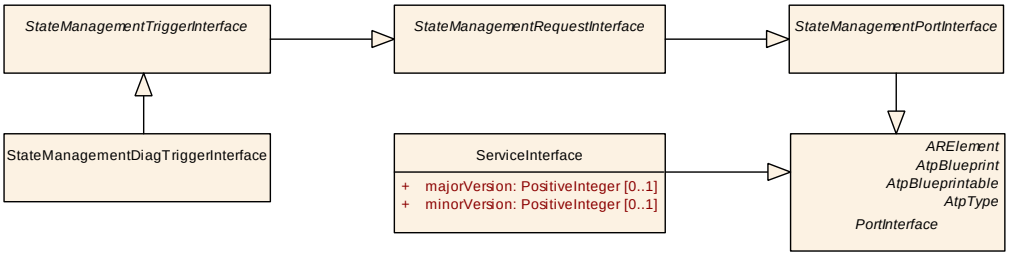


Figure 3.32: Modeling of State Management trigger interfaces

Class	**StateManagementDiagTriggerInterface**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::StateManagement
Note	This meta-class indicates that the PortPrototype that references this class is used for accepting a state change trigger from the diagnostics management. **Tags:** atp.Status=draft atp.recommendedPackage=StateManagementPortInterfaces
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementPortInterface*, *StateManagementRequestInterface*, *StateManagementTriggerInterface*
Aggregated by	ARPackage.element

▽

△

Class	**StateManagementDiagTriggerInterface**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.59: StateManagementDiagTriggerInterface

3.3.4.4 State Management Interfaces for Errors

In contrast to the communication of triggers, errors in the context of State Management are entirely handled by dedicated subclasses of `StateManagementErrorInterface`, as depicted in Figure 3.33.

As of the current state of interaction patterns with the State Management, two functional clusters report errors directly to State Management:

- For Execution Management [11], State Management exposes a `PortPrototype` typed by `StateManagementEmErrorInterface`.
- For Platform Health Management [12], State Management exposes a `PortPrototype` typed by `StateManagemenPhmErrorInterface`.

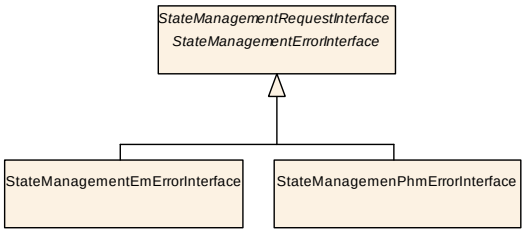


Figure 3.33: Modeling of State Management error interfaces

Class	**StateManagementEmErrorInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::StateManagement			
Note	This meta-class indicates that the PortPrototype that references this class is used for accepting a error submissions from the execution management. **Tags:** atp.Status=draft atp.recommendedPackage=StateManagementPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementErrorInterface*, *StateManagementPortInterface*, *StateManagementRequestInterface*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.60: StateManagementEmErrorInterface

Class	**StateManagemenPhmErrorInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::StateManagement			
Note	This meta-class indicates that the PortPrototype that references this class is used for accepting a error submissions from the platform health management. **Tags:** atp.Status=draft atp.recommendedPackage=StateManagementPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*, *StateManagementErrorInterface*, *StateManagementPortInterface*, *StateManagementRequestInterface*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.61: StateManagemenPhmErrorInterface

3.3.5 Persistency Interface

3.3.5.1 Overview

3.3.5.1.1 The big Picture

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 `PortPrototype`s.

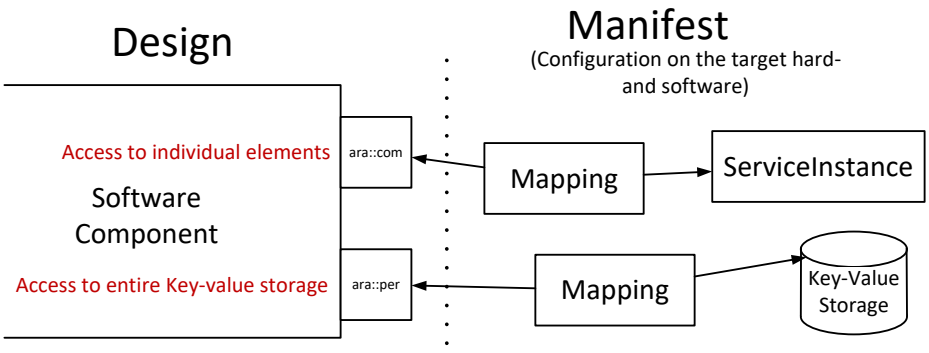


Figure 3.34: General approach for the modeling of persistency

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 key-value storage an entire key-value storage is accessible on the level of `PortPrototype`).

The aspect of deployment for the configuration of persistent data is explained in Figure 3.34.

Please note that the AUTOSAR meta-model actually defines two separate meta-classes (for more details, please refer to Figure 3.35) for the different use cases of access to persistent data (i.e. PersistencyKeyValueStorageInterface) and access to files on the file system, or maybe an emulation of one (by means of PersistencyFileStorageInterface).

3.3.5.1.2 Modeling of Persistency Interface

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

As a counterpart to the abstract base class PersistencyInterface on interface level, meta-class PersistencyInterfaceElement has been defined as an abstract base class for elements of a PersistencyInterface.

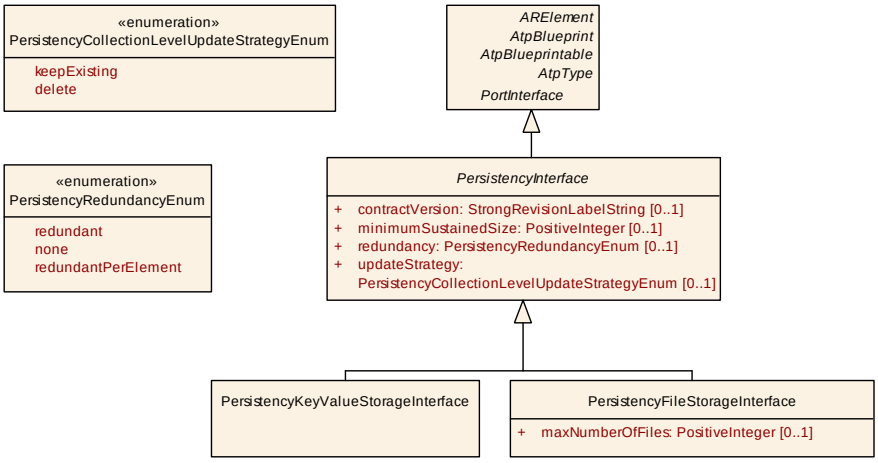


Figure 3.35: Specification of PortInterfaces for persistency use cases

Class	***PersistencyInterfaceElement*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::Persistency			
Note	This meta-class provides the abstract ability to define an element of a PortInterface for the support of persistency use cases.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	PersistencyDataElement, PersistencyFileElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
updateStrategy	PersistencyElementLevelUpdateStrategyEnum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyInterfaceElement.

Table 3.62: PersistencyInterfaceElement

[constr_10427]{DRAFT} **Multiplicity of attribute PersistencyInterfaceElement.updateStrategy** ⌈For each PersistencyInterfaceElement, the attribute updateStrategy shall exist **at the time when the creation of the manifest is finished.**⌋ *()*

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

Class	***PersistencyInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::Persistency			
Note	This meta-class provides the abstract ability to define a PortInterface for the support of persistency use cases.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	PersistencyFileStorageInterface, PersistencyKeyValueStorageInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contractVersion	StrongRevisionLabelString	0..1	attr	This attribute represents the contract version that is used to determine whether the Persistency configuration experienced structural changes and is also used for the check for data type compatibility.
minimumSustainedSize	PositiveInteger	0..1	attr	The value of this attribute represents the minimum size (unit: bytes) required at design time for the enclosing PersistencyInterface.
redundancy	PersistencyRedundancyEnum	0..1	attr	This attribute represents a requirement towards the redundancy of storage.
redundancyHandling	PersistencyRedundancyHandling	*	aggr	This aggregation represents the chosen approaches to handle redundancy for the various use cases implemented by subclasses
updateStrategy	PersistencyCollectionLevelUpdateStrategyEnum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyInterface as a whole.

Table 3.63: PersistencyInterface

[constr_10485]{DRAFT} **Existence of `PersistencyInterface.contractVersion`** ⌈In the context of `PersistencyInterface`, the attribute in the role `contractVersion` shall exist **`at the time before the generation of the ara API starts`**.⌋ *()*

3.3.5.1.3 Redundancy Handling

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

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

[TPS_MANI_01319]{DRAFT} **Modeling of redundancy in the context of `PersistencyInterface`** ⌈As an alternative to the ability to use `PersistencyInterface.redundancy` for announcing the consideration of redundancy at all, the design level

for persistency also provides the ability to provide a more detailed definition of redundant behavior for both key-value storage and files by means of the aggregation of PersistencyRedundancyHandling at PersistencyInterface.

This modeling is attached to the abstract base class PersistencyInterface in order to let both aspects of persistency (i.e. key-value storage and file storage) on the AUTOSAR adaptive platform benefit from the existence of meta-class PersistencyRedundancyHandling.⌋(*RS_MANI_00027*)

Enumeration	**PersistencyRedundancyEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec
Note	This meta-class provides a way to specify in which way redundancy shall be applied on collection level.
Aggregated by	*PersistencyInterface*.redundancy
Literal	***Description***
none	This value represents the requirement that redundancy measures are not applied on persistency storage level. **Tags:** atp.EnumerationLiteralIndex=1
redundant	This value represents the requirement that redundancy measures are applied on persistency storage level. The nature of the redundant persistent storage is not further qualified and subject to integrator decisions. **Tags:** atp.EnumerationLiteralIndex=0
redundantPer Element	This value represents the requirement that redundancy measures are applied on key-value level of a key-value storage or on file level of a file storage. The nature of the redundancy used on the persistent storage is not further qualified and subject to integrator decisions. **Tags:** atp.EnumerationLiteralIndex=2

Table 3.64: PersistencyRedundancyEnum

Class	***PersistencyRedundancyHandling*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This abstract base class represents a formal description of redundancy.			
Base	*ARObject*			
Subclasses	*PersistencyRedundancyChecksum*, PersistencyRedundancyMOutOfN			
Aggregated by	*PersistencyDeployment*.redundancyHandling, *PersistencyInterface*.redundancyHandling			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
scope	PersistencyRedundancy HandlingScopeEnum	0..1	attr	This attribute controls the scope in which the redundancy handling is applied.

Table 3.65: PersistencyRedundancyHandling

[TPS_MANI_01320]{DRAFT} **Definition of redundancy on interface level may be overruled in deployment** ⌈The modeling of redundancy by means of PersistencyInterface.redundancyHandling represents the intention of the designer of the PersistencyInterface.

While this is certainly a valuable input to the deployment phase, it is explicitly foreseen that an integrator may overrule the design decision regarding persistency based on superior knowledge only available at deployment time.⌋(*RS_MANI_00027*)

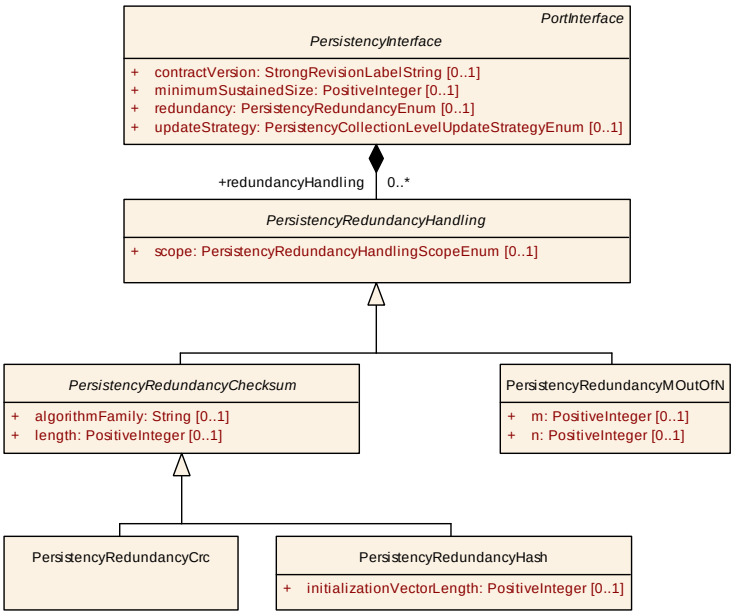


Figure 3.36: Specification of redundancy on the level of `PersistencyInterface`

[constr_1746]{DRAFT} **Mutual exclusive existence of `PersistencyInterface.redundancy` and `PersistencyInterface.redundancyHandling`** ⌈For each `PersistencyInterface`, either the attribute `redundancy` or the aggregation of `PersistencyRedundancyHandling` in the role `redundancyHandling` may exist **`at the time before the generation of the ara API starts`**.⌋*()*

Enumeration	**PersistencyRedundancyHandlingScopeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency
Note	This meta-class provides values to control the scope of redundancy measures in the persistency deployment
Aggregated by	*PersistencyRedundancyHandling*.scope
Literal	***Description***
persistency RedundancyHandlingScopeElement	The redundancy handling shall be applied on element level (key-value pair and file). **Tags:** atp.EnumerationLiteralIndex=0
persistency RedundancyHandlingScopeStorage	The redundancy handling shall be applied on storage (key-value storage and file storage) level. **Tags:** atp.EnumerationLiteralIndex=1

Table 3.66: PersistencyRedundancyHandlingScopeEnum

[TPS_MANI_01207]{DRAFT} **Standardized values of attribute `PersistencyRedundancyChecksum.algorithmFamily`** ⌈The following values of attribute `PersistencyRedundancyChecksum.algorithmFamily` are standardized by AUTOSAR:

- **`CRC_J1850`**
- **`CRC_CCITT_FALSE`**
- **`CRC_ETHERNET`**

- **CRC_0x42F0E1EBA9EA3693**
- **CRC_8H2F**
- **CRC_16ARC**
- **CRC_32P4**

⌋(*RS_MANI_00027*)

Class	***PersistencyRedundancyChecksum*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	Abstract class that defines the common attributes for implementations of redundancy.			
Base	*ARObject*, *PersistencyRedundancyHandling*			
Subclasses	PersistencyRedundancyCrc, PersistencyRedundancyHash			
Aggregated by	*PersistencyDeployment*.redundancyHandling, *PersistencyInterface*.redundancyHandling			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
algorithmFamily	String	0..1	attr	This attribute identifies the algorithm family that is used to execute the CRC/Hash.
length	PositiveInteger	0..1	attr	This attribute describes the length of the CRC/Hash in the unit bits.

Table 3.67: PersistencyRedundancyChecksum

[constr_10144]{DRAFT} **Multiplicity of reference in the role `PersistencyRedundancyChecksum.algorithmFamily`** ⌈For each `PersistencyRedundancyChecksum`, the reference in the role `algorithmFamily` shall exist **`at the time before the generation of the ara API starts`**.⌋()

[constr_10145]{DRAFT} **Multiplicity of reference in the role `PersistencyRedundancyChecksum.length`** ⌈For each `PersistencyRedundancyChecksum`, the reference in the role `length` shall exist **`at the time before the generation of the ara API starts`**.⌋()

[constr_1668]{DRAFT} **Allowed combinations of `PersistencyRedundancyChecksum.length` and `algorithmFamily`** ⌈

	8	16	32	64
CRC_J1850	x			
CRC_CCITT_FALSE		x		
CRC_ETHERNET			x	
CRC_0x42F0E1EBA9EA3693				x
CRC_8H2F	x			
CRC_16ARC		x		
CRC_32P4			x	

This rule shall be imposed **`at the time before the generation of the ara API starts`**

⌋()

Class	**PersistencyRedundancyCrc**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class formally describes the usage of a CRC for the implementation of redundancy.			
Base	*ARObject*, *PersistencyRedundancyChecksum*, *PersistencyRedundancyHandling*			
Aggregated by	*PersistencyDeployment*.redundancyHandling, *PersistencyInterface*.redundancyHandling			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.68: PersistencyRedundancyCrc

Class	**PersistencyRedundancyHash**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class formally describes the usage of a Hash for the implementation of redundancy.			
Base	*ARObject*, *PersistencyRedundancyChecksum*, *PersistencyRedundancyHandling*			
Aggregated by	*PersistencyDeployment*.redundancyHandling, *PersistencyInterface*.redundancyHandling			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initialization VectorLength	PositiveInteger	0..1	attr	Length of the initialization vector.

Table 3.69: PersistencyRedundancyHash

[constr_10046]{DRAFT} **Value of `PersistencyRedundancyMOutOfN.n`** ⌈The value of attribute `PersistencyRedundancyMOutOfN.n` shall be set at least to 2 and at most to 255, i.e. the allowed interval is [2..255].

This rule shall be imposed **at the time before the generation of the ara API starts.**⌋*()*

[constr_1751]{DRAFT} **Value of `PersistencyRedundancyMOutOfN.m`** ⌈The value of attribute `PersistencyRedundancyMOutOfN.m` shall be set at least to 1 and at most to the value of attribute `PersistencyRedundancyMOutOfN.n`, i.e. the allowed interval is [1..`PersistencyRedundancyMOutOfN.n`].

This rule shall be imposed **at the time before the generation of the ara API starts.**⌋*()*

Class	**PersistencyRedundancyMOutOfN**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class provides the ability to describe redundancy via an "M out of N" approach. In this case N is the number of copies created and M is the minimum number of identical copies to justify a reliable read access to the data.			
Base	*ARObject*, *PersistencyRedundancyHandling*			
Aggregated by	*PersistencyDeployment*.redundancyHandling, *PersistencyInterface*.redundancyHandling			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
m	PositiveInteger	0..1	attr	This attribute represents the "M" coordinate in the "M out of N" scheme.
n	PositiveInteger	0..1	attr	This attribute represents the "N" coordinate in the "M out of N" scheme.

Table 3.70: PersistencyRedundancyMOutOfN

[constr_10146]{DRAFT} **Multiplicity of reference in the role PersistencyRedundancyMOutOfN.m** ⌈For each PersistencyRedundancyMOutOfN, the reference in the role m shall exist **at the time before the generation of the ara API starts**.⌋*()*

[constr_10147]{DRAFT} **Multiplicity of reference in the role PersistencyRedundancyMOutOfN.n** ⌈For each PersistencyRedundancyMOutOfN, the reference in the role n shall exist **at the time before the generation of the ara API starts**.⌋*()*

3.3.5.1.4 Update Handling

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

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

Enumeration	**PersistencyCollectionLevelUpdateStrategyEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface
Note	This enumeration provides possible values for the update strategy on interface/storage level.
Aggregated by	*PersistencyDeployment*.updateStrategy, *PersistencyInterface*.updateStrategy
Literal	***Description***
delete	The update strategy is to delete all values on the level of the respective collection. **Tags:** atp.EnumerationLiteralIndex=1
keepExisting	The update strategy is to keep the existing values on the level of the respective collection. **Tags:** atp.EnumerationLiteralIndex=0

Table 3.71: PersistencyCollectionLevelUpdateStrategyEnum

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

Enumeration	**PersistencyElementLevelUpdateStrategyEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::Persistency
Note	This enumeration provides possible values for the update strategy on element level.
Aggregated by	*PersistencyDeploymentElement*.updateStrategy, *PersistencyInterfaceElement*.updateStrategy
Literal	***Description***

▽

△

Enumeration	**PersistencyElementLevelUpdateStrategyEnum**
delete	The update strategy is to delete the value of the respective data item. **Tags:** atp.EnumerationLiteralIndex=2
keepExisting	The update strategy is to keep the existing value of the respective data item. **Tags:** atp.EnumerationLiteralIndex=1
overwrite	The update strategy is to overwrite the respective data item. **Tags:** atp.EnumerationLiteralIndex=0

Table 3.72: PersistencyElementLevelUpdateStrategyEnum

The behavior of the software in terms of applying an update strategy is explained in detail in [13].

3.3.5.2 Persistency Key Value Storage Interface

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

Class	**PersistencyKeyValueStorageInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::Persistency			
Note	This meta-class provides the ability to implement a `PortInterface` for supporting persistency use cases for data. **Tags:** atp.recommendedPackage=PersistencyKeyValueStorageInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyInterface*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dataElement	PersistencyData Element	*	aggr	This aggregation represents the collection of Persistency DataElements in the context of the enclosing Persistency KeyValueStorageInterface.
dataTypeFor Serialization	AbstractImplementation DataType	*	ref	This reference identifies the AbstractImplementationData Types that shall be supported for storing in a key-value storage in addition to the types already determined from tha aggregation of PersistencyDataElement.
dataType Mapping	PersistencyKeyValue DataTypeMapping	0..1	aggr	This aggregation provides a collection of replacement rules for data types used in the context of the enclosing PersistencyKeyValueStorageInterface.

Table 3.73: PersistencyKeyValueStorageInterface

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

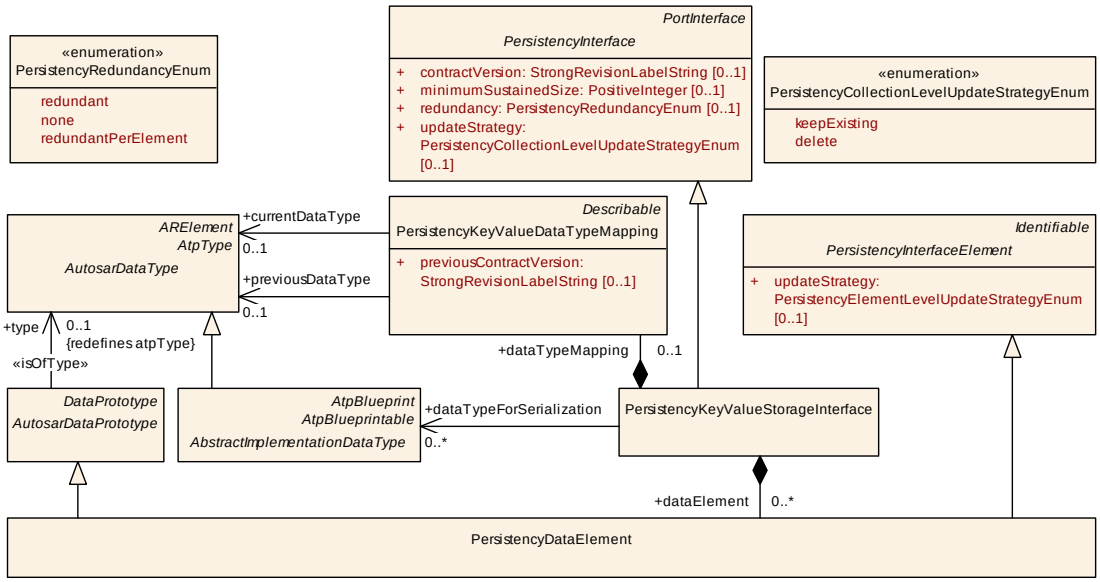


Figure 3.37: Modeling of the `PersistencyKeyValueStorageInterface`

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

[TPS_MANI_01138]{DRAFT} **Semantics of `PersistencyKeyValueStorageInterface.dataElement`** ⌈By aggregating `PersistencyDataElement` in the role `dataElement`, it is possible to explicitly model key-value pairs (and some of their properties) accessible to the application software within the context of a `PersistencyKeyValueStorageInterface`.⌋*(RS_MANI_00027)*

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

- `AbstractImplementationDataType`s referenced in the role `PersistencyKeyValueStorageInterface.dataTypeForSerialization`
- either
  - `AbstractImplementationDataType`s taken to type a `PersistencyKeyValueStorageInterface.dataElement` or
  - `AbstractImplementationDataType`s mapped to `ApplicationDataType`s taken to type a `PersistencyKeyValueStorageInterface.dataElement` by means of `PortInterfaceToDataTypeMapping.dataTypeMappingSet` that also refers to the enclosing `PersistencyKeyValueStorageInterface`.

⌋*(RS_MANI_00027)*

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

Class	**PersistencyDataElement**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::Persistency			
Note	This meta-class represents the ability to formally specify a piece of data that is subject to persistency in the context of the enclosing PersistencyKeyValueStorageInterface. PersistencyDataElement represents also a key-value pair of the deployed PersistencyKeyValueStorage and provides an initial value.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *PersistencyInterfaceElement*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, PersistencyKeyValueStorageInterface.dataElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.74: PersistencyDataElement

[constr_10425]{DRAFT} **Existence of initial values for `PersistencyDataElement`** ⌈For each `PersistencyDataElement`, if the value of attribute `PersistencyInterface.updateStrategy` is set to either

- `PersistencyElementLevelUpdateStrategyEnum.keepExisting` or
- `PersistencyElementLevelUpdateStrategyEnum.overwrite`,

then a `PersistencyDataRequiredComSpec` shall be aggregated by a `RPortPrototype` that is typed by the `PersistencyInterface` that owns the specific `PersistencyDataElement`.

The `PersistencyDataRequiredComSpec` shall

- refer to the respective `PersistencyDataElement` and
- aggregate `ValueSpecification` in the role `initValue`.

This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

[constr_10420]{DRAFT} **Restriction for the existence of initial values for `PersistencyDataElement`** ⌈For each `PersistencyKeyValuePair`, if the value of attribute `updateStrategy` is set to the value `PersistencyElementLevelUpdateStrategyEnum.delete`, then the specific `PersistencyDataElement` shall not be referenced by a `PersistencyDataRequiredComSpec` that is aggregated by a `RPortPrototype` that in turn is typed by the `PersistencyInterface` that owns the specific `PersistencyDataElement`.

This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

[TPS_MANI_01378]{DRAFT} **Semantics of meta-class `PersistencyKeyValueDataTypeMapping`** ⌈Meta-class `PersistencyKeyValueDataTypeMapping` supports the porting of a previously used data type (represented by the role `previousDataType`) used in an existing key-value pair to the currently used data type (represented by the role `currentDataType`) used in the context of the application software.

This way, application software that uses a given AutosarDataType can access a key-value-pair that was stored using a different (i.e. used in previous versions) data type as long as the mapping from the previous data type to the data type used currently in the key-value-pair is formally defined by means of the PersistencyKeyValueDataTypeMapping such that the application software can provide a suitable conversion. The conversion will typically happen in the update callback function registered by the application.⌋*(RS_MANI_00027)*

Class	**PersistencyKeyValueDataTypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::Persistency			
Note	This meta-class represents the ability to define a mapping between an existing data type in a key-value-storage stored by a previous version to a new data type used on application software level in the current version.			
Base	*ARObject*, *Describable*			
Aggregated by	PersistencyKeyValueStorageInterface.dataTypeMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
currentDataType	AutosarDataType	0..1	ref	This reference identifies the current data type for an existing key-value-pair in the context of the enclosing PersistencyKeyValueStorageInterface.
previousContractVersion	StrongRevisionLabelString	0..1	attr	This attribute identifies the contract version in which the previousDataType was used.
previousDataType	AutosarDataType	0..1	ref	This reference identifies the previous data type in a key-value-pair existing in the context of the enclosing PersistencyKeyValueStorageInterface.

Table 3.75: PersistencyKeyValueDataTypeMapping

[constr_10486]{DRAFT} **Existence of PersistencyKeyValueDataTypeMapping.previousContractVersion** ⌈In the context of PersistencyKeyValueDataTypeMapping, the attribute in the role previousContractVersion shall exist **at the time before the generation of the ara API starts**.⌋*()*

Please note that a PersistencyKeyValueDataTypeMapping needs to be complete, i.e. for each PersistencyKeyValueDataTypeMapping the two references shall exist in any case.

[constr_10377]{DRAFT} **Completeness of the modeling of PersistencyKeyValueDataTypeMapping** ⌈For each PersistencyKeyValueDataTypeMapping, the references in the roles

- previousDataType
- currentDataType

shall **both** exist **at the time before the generation of the ara API starts**.⌋*()*

It only makes sense to map types of the same data type level, which is ensured by the following two constraints:

[constr_10378]{DRAFT} **PersistencyKeyValueDataTypeMapping references AbstractImplementationDataType in the role currentDataType** ⌈Each PersistencyKeyValueDataTypeMapping that references to an AbstractImplementationDataType as part of the collection in the role currentDataType shall also refer to an AbstractImplementationDataType in the role previousDataType.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

[constr_10379]{DRAFT} **PersistencyKeyValueDataTypeMapping references ApplicationDataType in the role currentDataType** ⌈Each PersistencyKeyValueDataTypeMapping that references to an ApplicationDataType as part of the collection in the role currentDataType shall also refer to an ApplicationDataType in the role previousDataType.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

Of course, it does not make much sense if the modeling of PersistencyKeyValueDataTypeMapping.currentDataType is not even used in the context of the enclosing PersistencyKeyValueStorageInterface because this way a conversion to an unused data type is defined that would never be used in reality. This caveat is addressed by [advisory_01009].

[advisory_01009]{DRAFT} **PersistencyKeyValueDataTypeMapping.currentDataType shall refer to a data type used in the context of the PersistencyKeyValueStorageInterface** ⌈All data types referenced in the role PersistencyKeyValueDataTypeMapping.currentDataType shall be in use in the context of the enclosing PersistencyKeyValueStorageInterface. This means any data type referenced in the role currentDataType shall also be referenced in one of the following roles:

- PersistencyKeyValueStorageInterface.dataElement.type
- PersistencyKeyValueStorageInterface.dataTypeForSerialization

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

Please note that the usage of references previousDataType and currentDataType does not make any assumptions with respect to the compatibility of a pair of previousDataType and currentDataType. It is up to the application software to sort out any potential compatibility issues.

3.3.5.3 Persistency File Storage Interface

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

As far as AUTOSAR persistency is concerned, a file can have binary or text content. If it has text content then the content of the file is expected to be encoded as UTF-8 encoding with UNIX line endings.

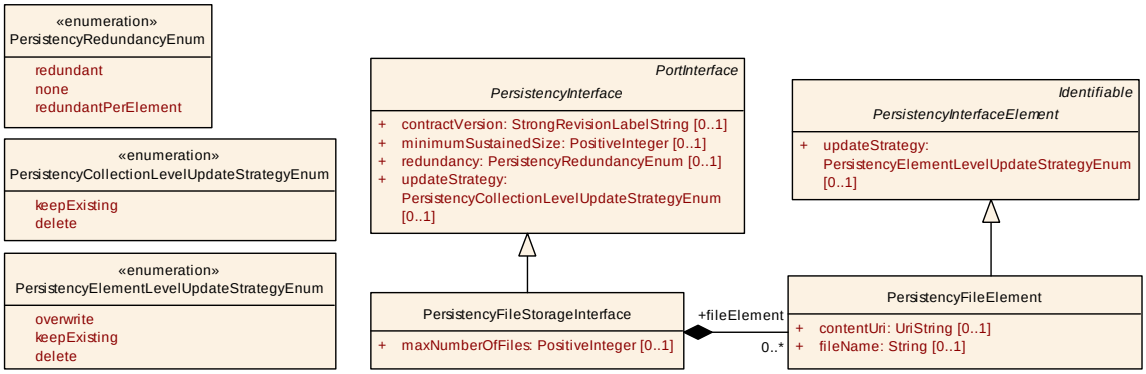


Figure 3.38: Modeling of the `PersistencyFileStorageInterface`

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

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

The value of attribute `PersistencyFileStorageInterface.maxNumberOfFiles` **includes** the explicitly modeled `PersistencyFileStorageInterface.fileElements`.⌋*(RS_MANI_00027)*

Please note that the existence of the `PersistencyFileStorageInterface` does not violate the restrictions set by the POSIX subset PSE51 defined in IEEE1003.13 [14].

Class	**PersistencyFileStorageInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::Persistency			
Note	This meta-class provides the ability to implement a PortInterface for supporting persistency use cases for files. **Tags:** atp.recommendedPackage=PersistencyFileStorageInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyInterface*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**PersistencyFileStorageInterface**			
fileElement	PersistencyFileElement	*	aggr	This aggregation represents the collection of Persistency FileStorages in the context of the enclosing Persistency FileStorageInterface.
maxNumberOf Files	PositiveInteger	0..1	attr	This attribute represents the definition of an upper bound for the handling of files at run-time in the context of the enclosing PersistencyFileStorageInterface.

Table 3.76: PersistencyFileStorageInterface

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

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

Class	**PersistencyFileElement**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::Persistency			
Note	This meta-class has the ability to represent a file at design time such that it is possible to configure the behavior for accessing the represented file at run-time.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PersistencyInterfaceElement*, *Referrable*			
Aggregated by	PersistencyFileStorageInterface.fileElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contentUri	UriString	0..1	attr	This attribute represents the URI that identifies the initial content of the PersistencyFile.
fileName	String	0..1	attr	This attribute holds the filename part of the storage location, e.g. file on the file system.

Table 3.77: PersistencyFileElement

[constr_10148]{DRAFT} **Multiplicity of reference in the role `PersistencyFileElement.contentUri`** ⌈For each `PersistencyFileElement`, the reference in the role `contentUri` shall exist if the value of attribute `updateStrategy` is set to either

- `PersistencyElementLevelUpdateStrategyEnum.keepExisting` or
- `PersistencyElementLevelUpdateStrategyEnum.overwrite`.

This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

[constr_10149]{DRAFT} **Multiplicity of reference in the role `PersistencyFileElement.fileName`** ⌈For each `PersistencyFileElement`, the reference in the role `fileName` shall exist **`at the time before the generation of the ara API starts`**.⌋*()*

[constr_1581]{DRAFT} **Value of `fileElement.fileName`** ⌈Within the scope of any given `PersistencyFileStorageInterface`, the value of all `fileElement.fileName` shall be unique **`at the time before the generation of the ara API starts`**.⌋*()*

[constr_10080]{DRAFT} **Existence of initial values for `PersistencyFileElement`** ⌈For each `PersistencyFileElement`, if the value of attribute `updateStrategy` is set to the value `delete`, then attribute `PersistencyFileElement.contentUri` shall not exist **`at the time before the generation of the ara API starts`**.⌋*()*

3.3.6 Time Synchronization Interface

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

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

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

In contrast to the service based communication, the modeling of platform software interaction using `PortPrototype`s and `PortInterface`s is less detailed. The `PortPrototype` is a placeholder for the interaction with platform software, it does not model the actually used APIs available for the interaction. The APIs to be used are formally specified in the platform software SWS document, i.e. SWS_TimeSync [15].

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

For more information, please refer to Figure 3.39.

Class	***AbstractSynchronizedTimeBaseInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the abstract ability to define a PortInterface for the interaction with Time Synchronization.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	SynchronizedTimeBaseConsumerInterface, SynchronizedTimeBaseProviderInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.78: AbstractSynchronizedTimeBaseInterface

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

[TPS_MANI_03549]{DRAFT} **Usage of PortPrototype for the interaction with Time Synchronization** ⌈Depending on the use-case the usage of RPortPrototype or PPortPrototype typed by one of the sub-classes of AbstractSynchronizedTimeBaseInterface shall be used for the interaction with the Time Synchronization.⌋ *(RS_MANI_00040)*

The application software may take the active or the passive role in the interaction with functional cluster, thus either a RPortPrototype or a PPortPrototype shall be used to represent this interaction from the application software point of view. The TimeBase Resource instance is identified using the InstanceSpecifier of the respective PortPrototype.

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

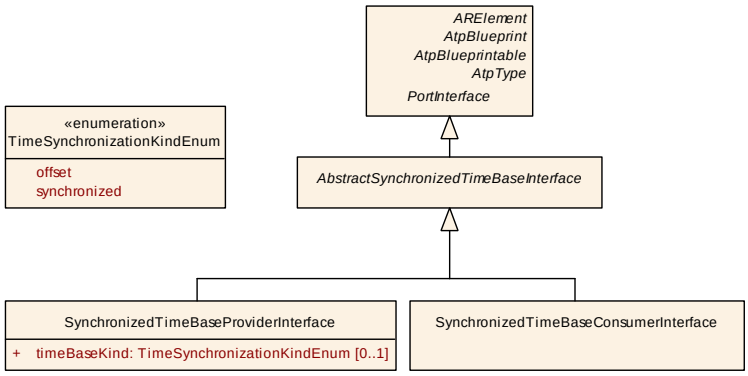


Figure 3.39: Modeling of Time Synch Interfaces

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

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

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

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

Class	**SynchronizedTimeBaseProviderInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to define a PortInterface for the interaction with a Time Synchronization Provider. **Tags:** atp.recommendedPackage=TimeSynchronizationInterfaces			
Base	*ARElement*, *ARObject*, *AbstractSynchronizedTimeBaseInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
timeBaseKind	TimeSynchronizationKindEnum	0..1	attr	Defines which kind of time base is requested at this interface.

Table 3.79: SynchronizedTimeBaseProviderInterface

[constr_10150]{DRAFT} **Multiplicity of reference in the role `SynchronizedTimeBaseProviderInterface.timeBaseKind`** ⌈For each `SynchronizedTimeBaseProviderInterface`, the reference in the role `timeBaseKind` shall exist **`at the time before the generation of the ara API starts`**.⌋*()*

Class	**SynchronizedTimeBaseConsumerInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to define a PortInterface for the interaction with a Time Synchronization Consumer. **Tags:** atp.recommendedPackage=TimeSynchronizationInterfaces			
Base	*ARElement*, *ARObject*, *AbstractSynchronizedTimeBaseInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.80: SynchronizedTimeBaseConsumerInterface

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

Table 3.81: TimeSynchronizationKindEnum

In the example in figure 3.40 the interaction of one Application with several time sync aspects are illustrated.

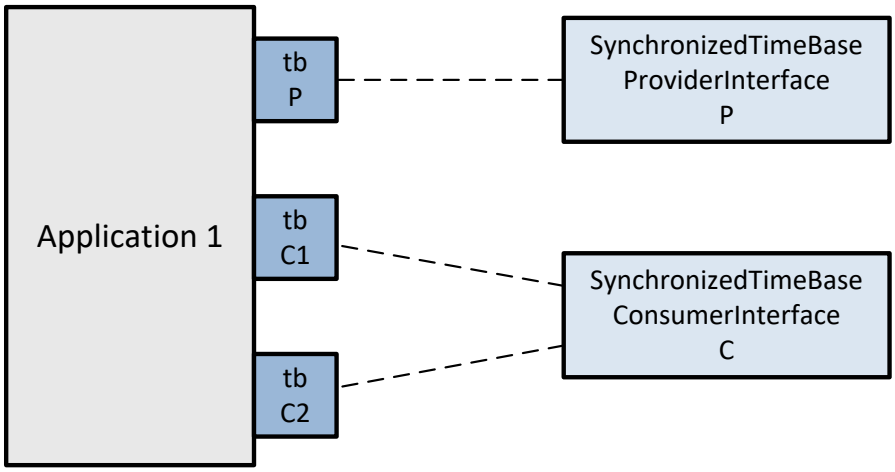


Figure 3.40: Example Application and Time Sync interaction

The interaction approach is that, for each `PortPrototype` typed by a sub-class of `AbstractSynchronizedTimeBaseInterface`, the application developer gains access to the respective kind of Time-Base Resource.

In the application code, the respective Time Base class (as defined in [15]) is constructed using the `InstanceSpecifier` representing the `PortPrototype` name.

During application deployment, those `PortPrototype`s are mapped to actual Time-Base Resources in the Time-Sync Management (see figure 10.24).

3.3.7 Platform Health Management Interface

3.3.7.1 Overview

Platform Health Management functional cluster within the Adaptive Platform is responsible to supervise the execution of applications, monitor their status, and triggering the State Management for 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 `PortPrototype`s and `PortInterface`s which are applied to many use-cases concerning the interaction of application software with platform software.

In contrast to the service based communication, the modeling of platform software interaction using `PortPrototype`s and `PortInterface`s is less detailed. The `PortPrototype` is a placeholder for the interaction with platform software, it does

not model the actually used APIs available for the interaction. The APIs to be used are formally specified in the platform software SWS document [12].

3.3.7.2 Supervised Entities and Checkpoints

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

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

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

[constr_1727]{DRAFT} **Qualified combinations of PortPrototypes and PhmSupervisedEntityInterface on application software level** ⌈Within the context of an Executable of category APPLICATION_LEVEL the usage of PhmSupervisedEntityInterface is **only** supported for an RPortPrototype.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

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

An example for the relevance of the InstanceSpecifier is given in Appendix B.5.1.

The application code then calls the *ReportCheckpoint* API (defined in [12]) of the *SupervisedEntity* (which has been constructed in the context of the respective RPortPrototype typed by the PhmSupervisedEntityInterface) in order to notify the Platform Health Management that a specific PhmCheckpoint has been reached in the program flow.

[constr_3530]{DRAFT} **Mandatory definition of checkpointId** ⌈The checkpointId shall be defined for every PhmCheckpoint element **at the time before the generation of the ara API starts**.⌋*()*

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

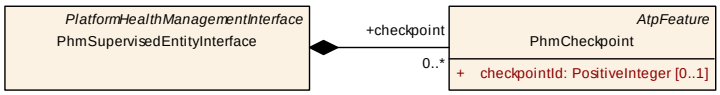


Figure 3.41: Modeling of Supervised Entities and Checkpoints

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 10.3.3.

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

Table 3.82: PhmSupervisedEntityInterface

Class	**PhmCheckpoint**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a checkpoint for interaction with the Platform Health Management Supervised Entity.			
Base	*ARObject*, *AtpFeature*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, PhmSupervisedEntityInterface.checkpoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checkpointId	PositiveInteger	0..1	attr	Defines the numeric value which is used to indicate the reporting of this Checkpoint to the Phm.

Table 3.83: PhmCheckpoint

[constr_10151]{DRAFT} **Multiplicity of reference in the role `PhmCheckpoint.checkpointId`** ⌈For each `PhmCheckpoint`, the reference in the role `checkpointId` shall exist **`at the time before the generation of the ara API starts`**.⌋*()*

3.3.7.3 Health Channels

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

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

By defining a RPortPrototype which is typed by the PhmHealthChannelInterface (see [constr_1728]) the application indicates that it wants to report the status of this PhmHealthChannelInterface.

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

The application code then calls the *ReportHealthStatus* API (defined in [12]) of the *HealthChannel* (which has been constructed in the context of the respective RPortPrototype typed by the PhmHealthChannelInterface) in order to notify the Platform Health Management that the Health Channel defined by the RPortPrototype has changed its status.

[constr_3532]{DRAFT} **Mandatory definition of statusId** ⌈The statusId shall be defined for every PhmHealthChannelStatus element **at the time before the generation of the ara API starts**.⌋*()*

[TPS_MANI_03624]{DRAFT} **Usage of statusId in application code** ⌈The application code shall only use those PhmHealthChannelStatus.statusId values which are defined as members of the PhmHealthChannelInterface.status.⌋ *(RS_MANI_00032)*

[TPS_MANI_03630]{DRAFT} **Semantics of triggersRecoveryNotification** ⌈The attribute triggersRecoveryNotification defines whether this specific PhmHealthChannelStatus shall be considered by the PHM as triggering the recovery notification.⌋*(RS_MANI_00032)*

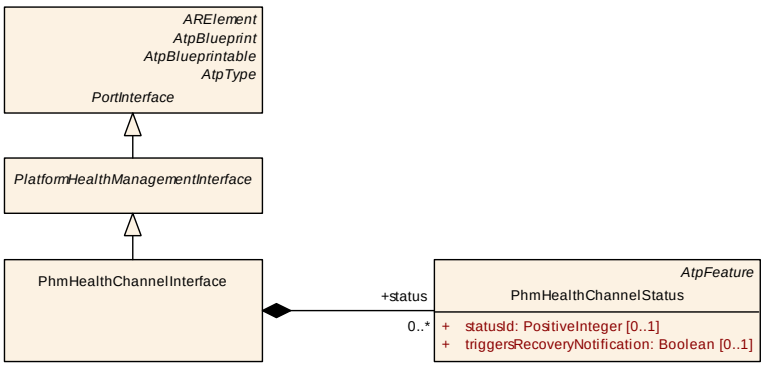


Figure 3.42: Modeling of Health Channel

[constr_1728]{DRAFT} **Qualified combinations of PortPrototypes and PhmHealthChannelInterface on application software level** ⌈Within the context of an Executable of category APPLICATION_LEVEL the usage of PhmHealthChannelInterface is **only** supported for a RPortPrototype.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

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

Table 3.84: PhmHealthChannelInterface

Class	**PhmHealthChannelStatus**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	The PhmHealthChannelStatus specifies one possible status of the health channel.			
Base	*ARObject*, *AtpFeature*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, PhmHealthChannelInterface.status			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
statusId	PositiveInteger	0..1	attr	Defines the numeric value which is used to indicate the indication of this status the Phm.
triggers Recovery Notification	Boolean	0..1	attr	Defines whether this PhmHealthChannelStatus shall cause the Phm to trigger the Health Channel recovery notification. • true: Indicates unhealthy state. Phm to trigger the Health Channel recovery notification when the Health channel status changes to this state. • false: Indicates healthy state. Phm not to trigger the Health Channel recovery notification when the Health channel status changes to this state.

Table 3.85: PhmHealthChannelStatus

3.3.7.4 Recovery notification to State Management

The Phm monitors the reporting of Supervised Entities and Checkpoints as well as the reported Health Channel status information. In case of violations the Phm can be configured to report the violation to the State Management and let the State Management deal with the recovery activities.

The example in figure 3.43 illustrates the reporting of Supervised Entities by Application 1 and 2.

The Phm is configured to perform the supervision of these reported elements. In case of violations the Phm is configured to notify the State Management application to deal with the situation.

[TPS_MANI_01280]{DRAFT} **Semantics of meta-class `PhmSupervisionRecoveryNotificationInterface`** ⌈The recovery notification of a failed Supervision by PHM does issue is to call a piece of code on State Management software level.

The mechanism for activating the code on the level of State Management software is to model a `PPortPrototype` typed by a `PhmSupervisionRecoveryNotificationInterface`.⌋*(RS_MANI_00032)*

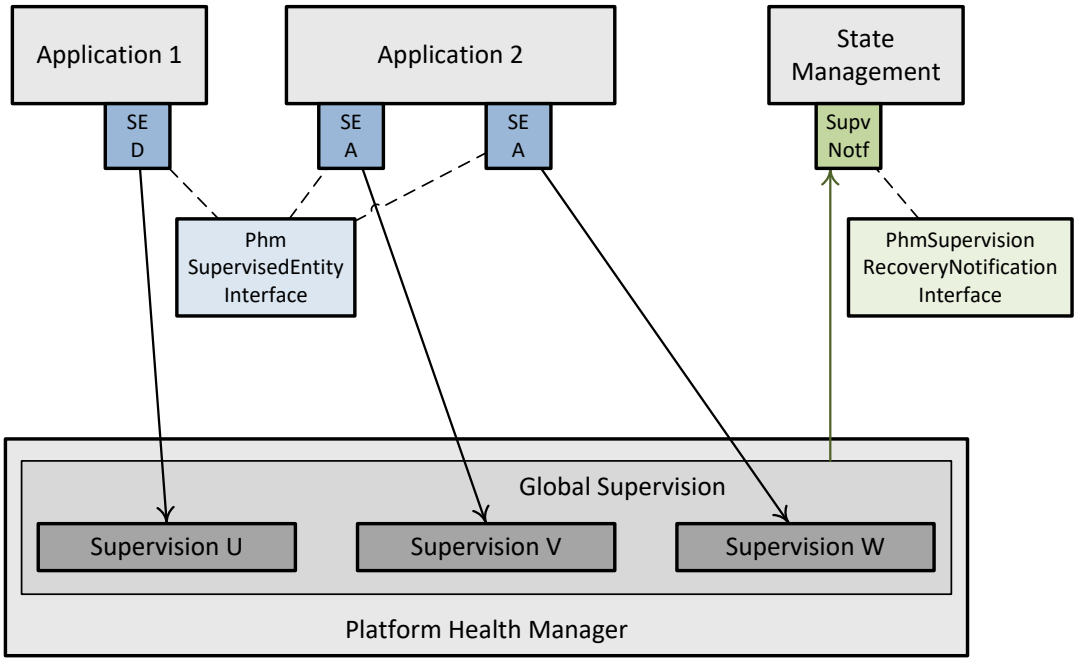


Figure 3.43: Example of a Phm monitoring and recovery setup

[TPS_MANI_03631]{DRAFT} **Semantics of meta-class `PhmHealthChannelRecoveryNotificationInterface`** ⌈The recovery notification of a failed HealthChannel monitoring by PHM does issue is to call a piece of code on State Management software level.

The mechanism for activating the code on the level of State Management software is to model a `PPortPrototype` typed by a `PhmHealthChannelRecoveryNotificationInterface`.⌋*(RS_MANI_00032)*

The operation to be called by Phm in the context of [TPS_MANI_01280] and [TPS_MANI_03631] are defined in the Platform Health Management specification document [12].

As already mentioned, the State Management is supposed to implement the recovery actions. This implies that the `PhmSupervisionRecoveryNotificationInterface` and `PhmHealthChannelRecoveryNotificationInterface` can only

be used in combination with a PPortPrototype. This aspect is clarified by [constr_1729].

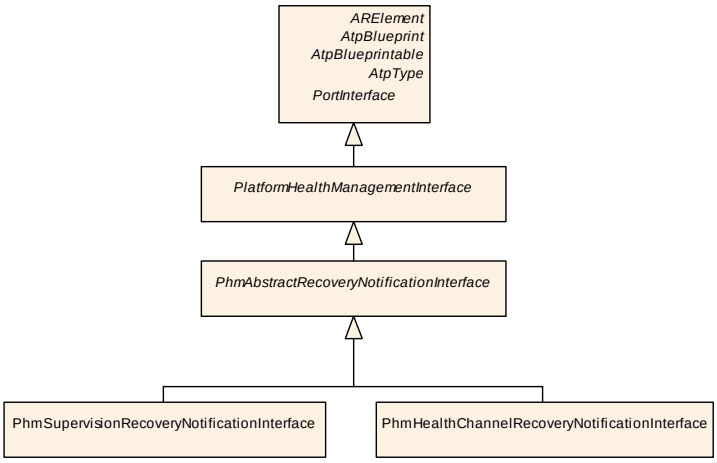


Figure 3.44: Modeling of the PhmAbstractRecoveryNotificationInterface

[constr_1729]{DRAFT} **Qualified combinations of PortPrototypes and PhmSupervisionRecoveryNotificationInterface / PhmHealthChannelRecoveryNotificationInterface on State Management software level** ⌈Within the context of an Executable of category APPLICATION_LEVEL the usage of PhmSupervisionRecoveryNotificationInterface and PhmHealthChannelRecoveryNotificationInterface is **only** supported for a PPortPrototype.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

Class	**PhmSupervisionRecoveryNotificationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents a PortInterface that can be taken for implementing a PHM Supervision notification. **Tags:** atp.recommendedPackage=PlatformHealthManagementInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PhmAbstractRecoveryNotificationInterface*, *PlatformHealthManagementInterface*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.86: PhmSupervisionRecoveryNotificationInterface

Class	**PhmHealthChannelRecoveryNotificationInterface**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface
Note	This meta-class represents a PortInterface that can be taken for implementing a PHM HealthChannel notification. **Tags:** atp.recommendedPackage=PlatformHealthManagementInterfaces

▽

△

Class	PhmHealthChannelRecoveryNotificationInterface			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, PhmAbstractRecoveryNotificationInterface, PlatformHealthManagementInterface, PortInterface, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 3.87: PhmHealthChannelRecoveryNotificationInterface

Class	PhmAbstractRecoveryNotificationInterface (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This abstract meta-class provides the abstract ability to define a PortInterface for the Recovery Notification by Platform Health Management.			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, PlatformHealthManagementInterface, Port Interface, Referrable			
Subclasses	PhmHealthChannelRecoveryNotificationInterface, PhmSupervisionRecoveryNotificationInterface			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 3.88: PhmAbstractRecoveryNotificationInterface

3.3.8 Diagnostic Interface

3.3.8.1 Overview

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

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

The interfaces on the side of the AUTOSAR `Diagnostic Manager` (and thus the part of the implementation of the `PortPrototype` that faces the AUTOSAR `Diagnostic Manager`) are **entirely proprietary**. This aspect is depicted in Figure 3.45.

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

[TPS_MANI_01242]{DRAFT} **`PortInterface`s used for communication with the AUTOSAR `Diagnostic Manager`** ⌈All `PortInterface`s used for this purpose are derived from the abstract meta-class `DiagnosticPortInterface`. A `DiagnosticPortInterface` does not implement a service-oriented communication pattern,

in particular there is no explicit service discovery on the API level involved.⌋(*RS_MANI_00061*)

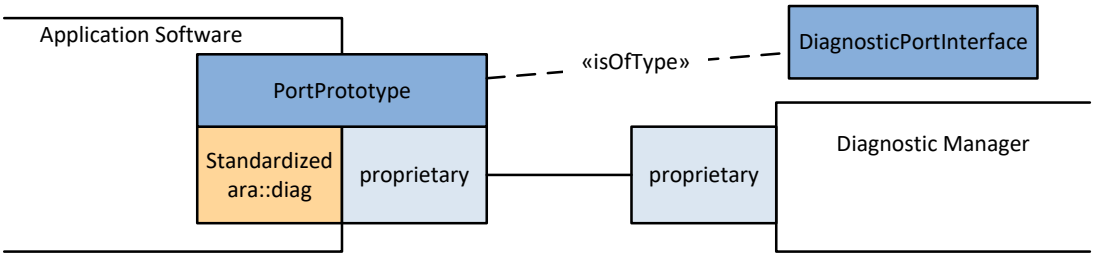


Figure 3.45: Standardized vs. proprietary parts in the implementation of `ara::diag`

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

Figure 3.46 depicts all meta-classes that directly inherit from `DiagnosticPortInterface`.

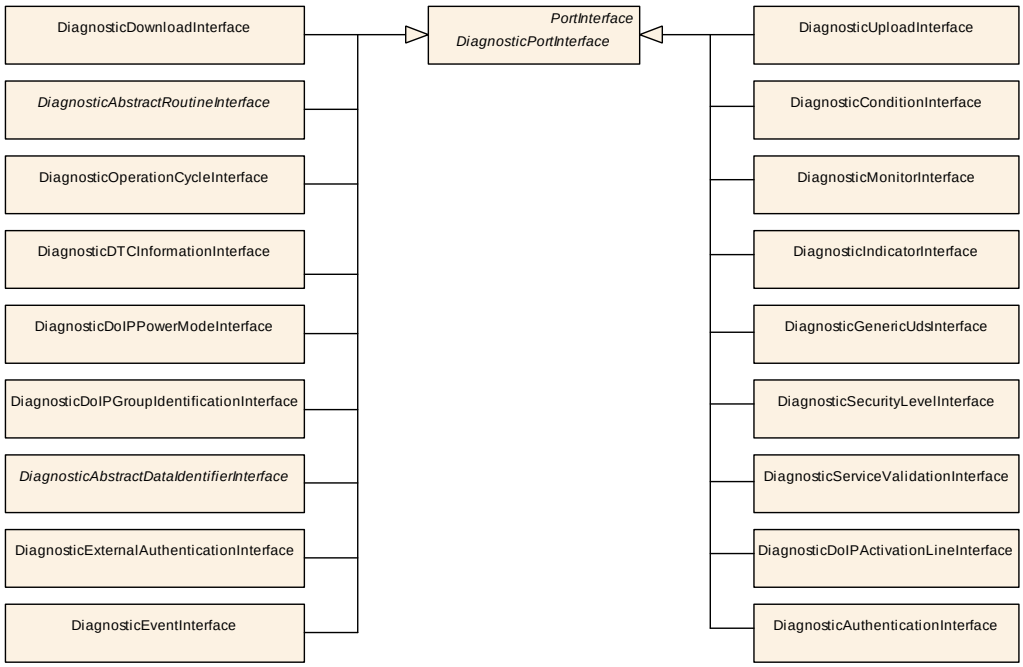


Figure 3.46: Modeling of `PortInterface`s for diagnostic purposes

Class	***DiagnosticPortInterface*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface
Note	This meta-class serves as an abstract base-class for all diagnostics-related PortInterfaces.
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*

▽

△

Class	***DiagnosticPortInterface*** (abstract)			
Subclasses	*DiagnosticAbstractDataIdentifierInterface*, *DiagnosticAbstractRoutineInterface*, DiagnosticAuthentication Interface, DiagnosticComControlInterface, DiagnosticConditionInterface, DiagnosticDTCInformation Interface, DiagnosticDoIPActivationLineInterface, DiagnosticDoIPEntityIdentificationInterface, Diagnostic DoIPGroupIdentificationInterface, DiagnosticDoIPPowerModeInterface, DiagnosticDoIPTriggerVehicle AnnouncementInterface, DiagnosticDownloadInterface, DiagnosticEcuResetInterface, DiagnosticEvent Interface, DiagnosticExternalAuthenticationInterface, DiagnosticGenericUdsInterface, Diagnostic IndicatorInterface, DiagnosticMonitorInterface, *DiagnosticMultipleResourceInterface*, Diagnostic OperationCycleInterface, DiagnosticRequestFileTransferInterface, DiagnosticSecurityLevelInterface, DiagnosticServiceValidationInterface, *DiagnosticSovdPortInterface*, DiagnosticUploadInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.89: DiagnosticPortInterface

3.3.8.2 Diagnostic Routine Interface

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

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

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

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

- `fireAndForget`
- `possibleApError`
- `possibleApErrorSet`

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

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

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

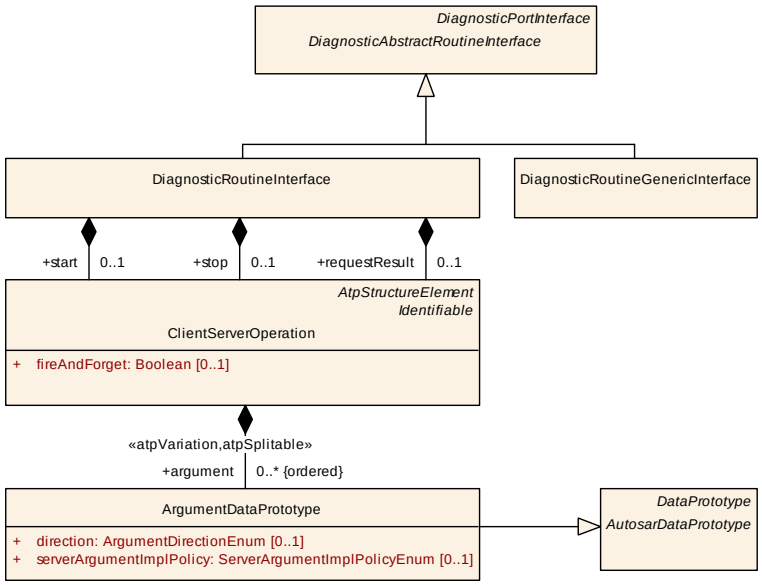


Figure 3.47: Modeling of `DiagnosticRoutineInterface`

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

Table 3.90: DiagnosticRoutineInterface

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

Table 3.91: DiagnosticRoutineGenericInterface

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

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

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

[constr_10031]{DRAFT} **Existence of DiagnosticRoutineInterface.start** ⌈Attribute DiagnosticRoutineInterface.start shall exist **at the time when the creation of the manifest is finished.**⌋*()*

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

Table 3.92: DiagnosticAbstractRoutineInterface

3.3.8.3 Interface to Data Identifier and Element of Data Identifier

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

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

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

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

[constr_10030]{DRAFT} **Existence of DiagnosticDataIdentifierInterface.read** ⌈Attribute DiagnosticDataIdentifierInterface.read shall exist **at the time when the creation of the manifest is finished.**⌋*()*

Class	**DiagnosticAbstractDataIdentifierInterface** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class serves as the abstract base class of PortInterfaces dedicated to the access of diagnostic data identifiers on the AUTOSAR adaptive platform.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	DiagnosticDataElementInterface, DiagnosticDataIdentifierGenericInterface, DiagnosticDataIdentifier Interface			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.93: DiagnosticAbstractDataIdentifierInterface

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

Table 3.94: DiagnosticDataIdentifierInterface

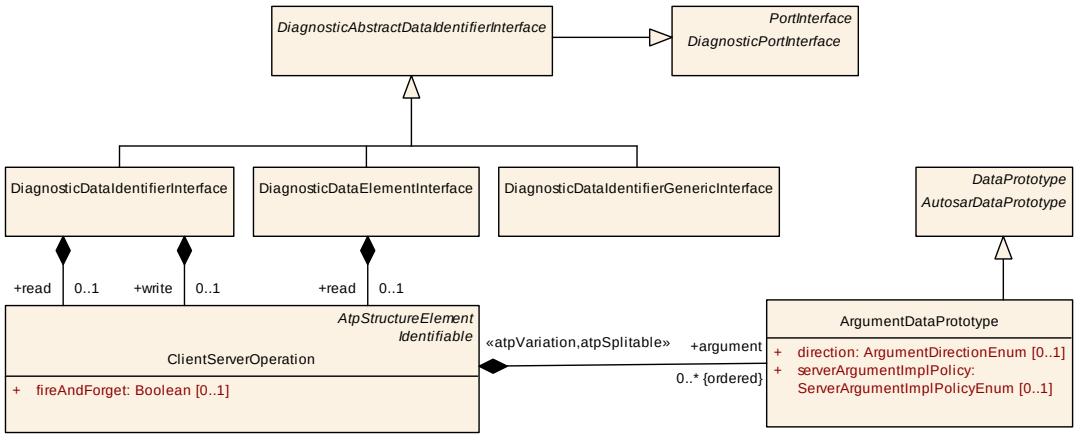


Figure 3.48: Modeling of `DiagnosticDataIdentifierInterface`

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

For this purpose, a `ClientServerOperation`s is aggregated in the role `read`.⌋
(RS_MANI_00061)

Please note that the DiagnosticDataElementInterface *intentionally* does not support a write operation because the consistency of the data in principle cannot be ensured if it is send to the application software piecemeal. Different Processes may be configured to receive the data but it cannot be ensured that all processes are up and running when data is transmitted.

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

Table 3.95: DiagnosticDataElementInterface

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

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

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

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

Table 3.96: DiagnosticDataIdentifierGenericInterface

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

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

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

This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

3.3.8.4 Interface to diagnostic Events

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

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

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

Table 3.97: DiagnosticMonitorInterface

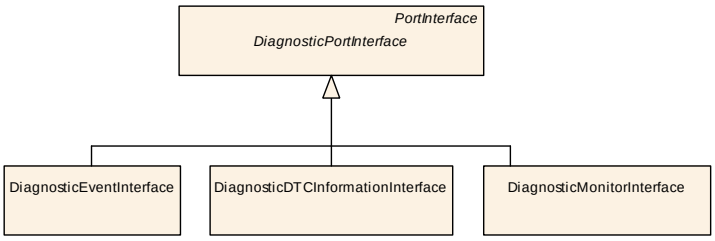


Figure 3.49: Modeling of `DiagnosticEventInterface`

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

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

Table 3.98: DiagnosticDTCInformationInterface

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

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

Table 3.99: DiagnosticEventInterface

3.3.8.5 Interface to diagnostic Condition

[TPS_MANI_01249]{DRAFT} **Semantics of `DiagnosticConditionInterface`** ⌈AUTOSAR supports different diagnostic conditions, i.e. enable condition and clear

condition. This aspect is represented in the definition of the DiagnosticConditionInterface for the *AUTOSAR adaptive platform.*⌋(*RS_MANI_00061*)

Please note that the DiagnosticConditionInterface does not get specific about what the role[6] of the condition would be.

This understanding of the role is in general not known to the application which just provides the status of the condition, but does not know whether the condition will be used as an enable condition or clear condition.

This is the reason why there are no dedicated PortInterfaces for enable condition and clear condition.

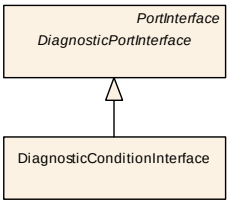


Figure 3.50: Modeling of DiagnosticConditionInterface

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

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

Table 3.100: DiagnosticConditionInterface

3.3.8.6 Interface for multiple diagnostic Resources

There are cases where software on application level needs to handle anonymous diagnostic resources, namely DiagnosticEvents and DiagnosticConditions in bulk and without knowing the identity of the specific resources at the time when the application is designed.

This means that the application would define a single PortPrototype (along with an appropriate PortInterface) over which e.g. the reporting of bulk diagnostic events is

[6] enable condition vs. clear condition

executed. This section defines the PortInterfaces used for the purpose of handling bulk diagnostic resources.

The definition of these PortInterfaces serves the special use case discussed above and therefore does not represent a “generic” replacement for the existing PortInterfaces over which single DiagnosticEvents or DiagnosticConditions are handled.

AUTOSAR defines an abstract subclass of DiagnosticPortInterface named DiagnosticMultipleResourceInterface that is acting as a base class for DiagnosticPortInterfaces that affect multiple diagnostic resources (e.g. diagnostic events) at the same time.

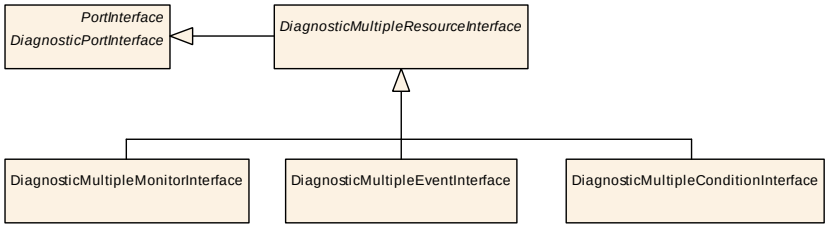


Figure 3.51: Modeling of DiagnosticMultipleResourceInterface

Class	***DiagnosticMultipleResourceInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class acts as an abstract base class for diagnostic port interfaces that affect multiple diagnostic resources (e.g. multiple events in one port interface).			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	DiagnosticMultipleConditionInterface, DiagnosticMultipleEventInterface, DiagnosticMultipleMonitorInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.101: DiagnosticMultipleResourceInterface

3.3.8.6.1 Interface for multiple diagnostic Events

Two subclasses of DiagnosticMultipleResourceInterface are dedicated to the handling of diagnostic events:

- DiagnosticMultipleMonitorInterface provides the ability to report the status of multiple diagnostic monitors in the context of one PortInterface.
- DiagnosticMultipleEventInterface provides the ability to retrieve the status of multiple diagnostic events in the context of one PortInterface.

Class	DiagnosticMultipleMonitorInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a monitor-focused PortInterface for diagnostics on the adaptive platform. In contrast to the DiagnosticMonitorInterface, the DiagnosticMultipleMonitor Interface allows for handling more than one event in the scope of a single PortPrototype. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticMultipleResourceInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.102: DiagnosticMultipleMonitorInterface

Class	DiagnosticMultipleEventInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a event-focused PortInterface for diagnostics on the adaptive platform. In contrast to the DiagnosticEventInterface, the DiagnosticMultipleMonitorInterface allows for handling more than one event in the scope of a single PortPrototype. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticMultipleResourceInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.103: DiagnosticMultipleEventInterface

3.3.8.6.2 Interface for multiple diagnostic Conditions

As documented in Figure 3.51, another subclass of `DiagnosticMultipleResourceInterface`, the `DiagnosticMultipleConditionInterface` is dedicated to the ability to handle multiple diagnostic conditions in the context of one `PortInterface`.

Class	DiagnosticMultipleConditionInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a condition-focused PortInterface for diagnostics on the adaptive platform. In contrast to the DiagnosticConditionInterface, the DiagnosticMultipleCondition Interface allows for handling more than one condition in the scope of a single PortPrototype. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticMultipleResourceInterface*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.104: DiagnosticMultipleConditionInterface

3.3.8.7 Indicator Interface

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

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

Table 3.105: DiagnosticIndicatorInterface

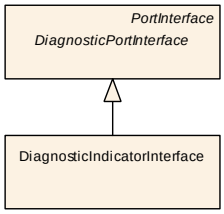


Figure 3.52: Modeling of `DiagnosticIndicatorInterface`

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

3.3.8.8 Security Level Interface

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

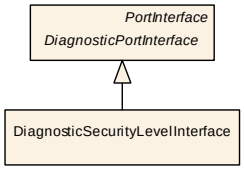


Figure 3.53: Modeling of `DiagnosticSecurityLevelInterface`

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

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

Table 3.106: DiagnosticSecurityLevelInterface

3.3.8.9 Service Validation Interface

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

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

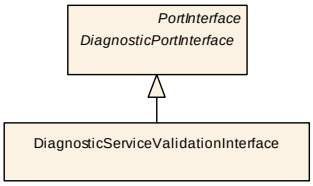


Figure 3.54: Modeling of `DiagnosticServiceValidationInterface`

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

Table 3.107: DiagnosticServiceValidationInterface

3.3.8.10 Operation Cycle Interface

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

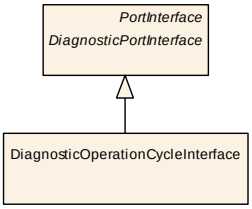


Figure 3.55: Modeling of `DiagnosticOperationCycleInterface`

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

Table 3.108: DiagnosticOperationCycleInterface

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

3.3.8.11 Generic UDS Interface

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

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

▽

△

Class	DiagnosticGenericUdsInterface			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 3.109: DiagnosticGenericUdsInterface

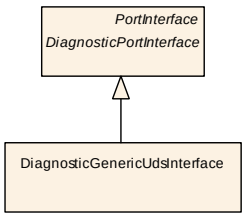


Figure 3.56: Modeling of `DiagnosticGenericUdsInterface`

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

3.3.8.12 DoIP Interfaces

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

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

- `DiagnosticDoIPGroupIdentificationInterface`
- `DiagnosticDoIPPowerModeInterface`
- `DiagnosticDoIPActivationLineInterface`
- `DiagnosticDoIPTriggerVehicleAnnouncementInterface`
- `DiagnosticDoIPEntityIdentificationInterface`

⌋*(RS_MANI_00061)*

Class	DiagnosticDoIPGroupIdentificationInterface
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIP Group Identification on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces

▽

△

Class	DiagnosticDoIPGroupIdentificationInterface			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 3.110: DiagnosticDoIPGroupIdentificationInterface

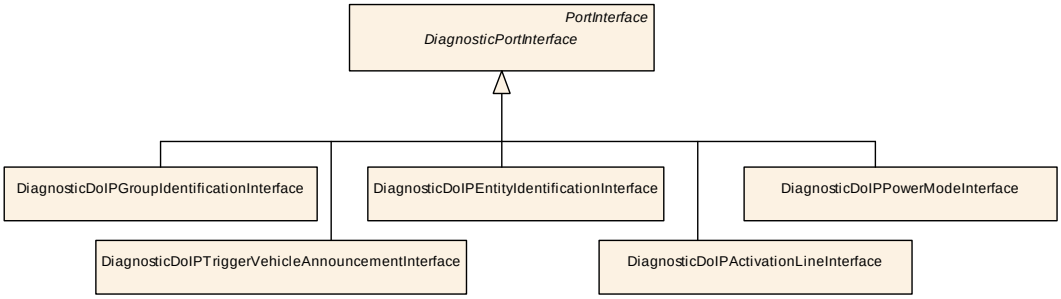


Figure 3.57: Modeling of DoIP `DiagnosticPortInterface`s

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

Table 3.111: DiagnosticDoIPPowerModeInterface

Class	DiagnosticDoIPActivationLineInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIPActivationLine on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DiagnosticPortInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 3.112: DiagnosticDoIPActivationLineInterface

Class	**DiagnosticDoIPTriggerVehicleAnnouncementInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to implement the DoIPTriggerVehicle Announcement on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 3.113: DiagnosticDoIPTriggerVehicleAnnouncementInterface

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

Table 3.114: DiagnosticDoIPEntityIdentificationInterface

The DiagnosticDoIPGroupIdentificationInterface, DiagnosticDoIP-PowerModeInterface, DiagnosticDoIPActivationLineInterface, DiagnosticDoIPEntityIdentificationInterface and DiagnosticDoIPTriggerVehicleAnnouncementInterface do not require any further details in its formalization.

3.3.8.13 Diagnostic Interfaces for Upload and Download

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

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

- DiagnosticUploadInterface
- DiagnosticDownloadInterface

⌋(*RS_MANI_00061*)

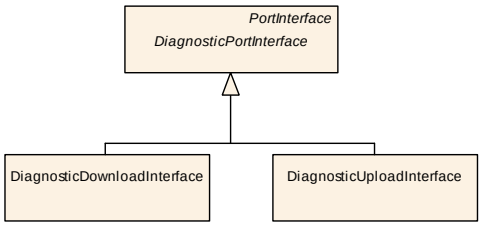


Figure 3.58: Modeling of `DiagnosticUploadInterface` and `DiagnosticDownloadInterface`

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

Table 3.115: DiagnosticUploadInterface

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

Table 3.116: DiagnosticDownloadInterface

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

3.3.8.14 Interface to support managing the EcuReset

[TPS_MANI_01332]{DRAFT} **Semantics of `DiagnosticEcuResetInterface`**
⌈Meta-class `DiagnosticEcuResetInterface` represents the ability to support the handling of a request to reset the machine.

This interface will typically be used by the state manager on the AUTOSAR adaptive platform.⌋*(RS_MANI_00061)*

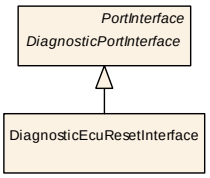


Figure 3.59: Modeling of `DiagnosticEcuResetInterface`

Class	**DiagnosticEcuResetInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic service EcuReset on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.117: DiagnosticEcuResetInterface

3.3.8.15 Diagnostic Authentication Interface

[TPS_MANI_01359]{DRAFT} **Semantics of `DiagnosticAuthenticationInterface`** ⌈The ability to support the diagnostic authentication in the application software is formalized in another sub-class of `DiagnosticPortInterface`: `DiagnosticAuthenticationInterface`.⌋*(RS_MANI_00061)*

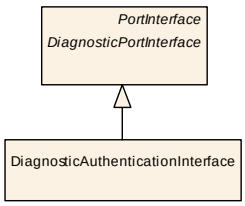


Figure 3.60: Modeling of `DiagnosticAuthenticationInterface`

Class	**DiagnosticAuthenticationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic service "authentication" on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.118: DiagnosticAuthenticationInterface

3.3.8.16 Diagnostic External Authentication Interface

[TPS_MANI_01353]{DRAFT} **Semantics of `DiagnosticExternalAuthenticationInterface`** ⌈The ability to support the authentication of a diagnostic client in the application software is formalized in another sub-class of `DiagnosticPortInterface`: `DiagnosticExternalAuthenticationInterface`.⌋*(RS_MANI_00061)*

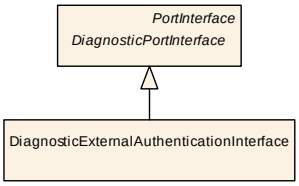


Figure 3.61: Modeling of `DiagnosticExternalAuthenticationInterface`

Class	**DiagnosticExternalAuthenticationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic client authentication (i.e. convey the Authentication state to the Diagnostic Server instance of the DM) on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.119: DiagnosticExternalAuthenticationInterface

3.3.8.17 Diagnostic Communication Control Interface

[TPS_MANI_01363]{DRAFT} **Semantics of `DiagnosticComControlInterface`** ⌈The ability to support the activation and deactivation of communication with a diagnostic client is formalized in another sub-class of `DiagnosticPortInterface`: `DiagnosticComControlInterface`.⌋*(RS_MANI_00061)*

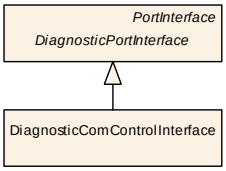


Figure 3.62: Modeling of `DiagnosticComControlInterface`

Class	**DiagnosticComControlInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a focused PortInterface for handling the diagnostic service communication control on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.120: DiagnosticComControlInterface

3.3.8.18 Diagnostic Request File Transfer Interface

[TPS_MANI_01373]{DRAFT} **Semantics of `DiagnosticRequestFileTransferInterface`** ⌈The usage of the `DiagnosticRequestFileTransferInterface` is foreseen for software that implements a handler for requests for file transfer.⌋*(RS_MANI_00061)*

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

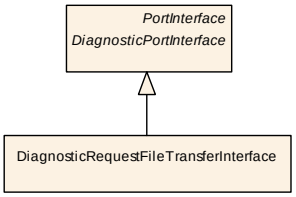


Figure 3.63: Modeling of `DiagnosticRequestFileTransferInterface`

Class	**DiagnosticRequestFileTransferInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class represents the ability to implement a PortInterface to process requests for file transfer using diagnostic channels on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.121: DiagnosticRequestFileTransferInterface

3.3.8.19 Diagnostic Port Interfaces for SOVD

[TPS_MANI_01394]{DRAFT} **Support for the authorization of SOVD clients** ⌈The ability to support the authorization of SOVD clients is formalized in meta-class `DiagnosticSovdAuthorizationInterface`, modeled as a sub-class of abstract base class `DiagnosticSovdPortInterface`. Application Software exposes a `PPortPrototype` for the support of authorization.⌋*(RS_MANI_00061, RS_MANI_00070)*

[TPS_MANI_01395]{DRAFT} **Support for the proximity challenge in the context of SOVD** ⌈The ability to support the proximity challenge in the context of SOVD is formalized in meta-class `DiagnosticSovdProximityChallengeInterface`, modeled as a sub-class of abstract base class `DiagnosticSovdPortInterface`. Application Software exposes a `PPortPrototype` for the proximity challenge.⌋*(RS_MANI_00061, RS_MANI_00070)*

[TPS_MANI_01420]{DRAFT} **Support for the transport of bulk data in the context of SOVD** ⌈The ability to transport bulk data in the context of SOVD is formalized in meta-class `DiagnosticSovdBulkDataInterface`, modeled as a sub-class of abstract base class `DiagnosticSovdPortInterface`. Application Software exposes a `PPortPrototype` for the bulk data transport.⌋*(RS_MANI_00061, RS_MANI_00070)*

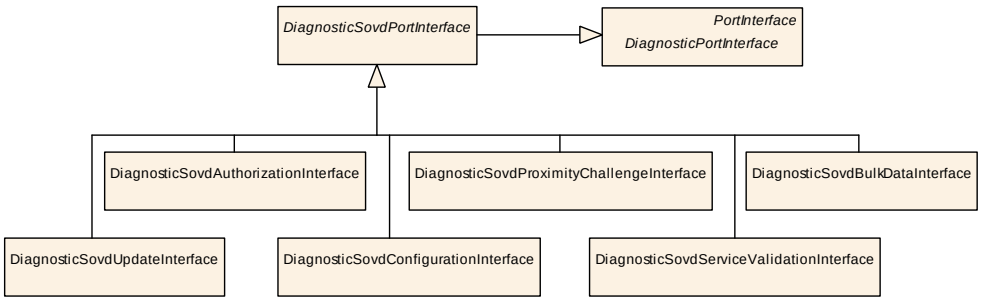


Figure 3.64: Modeling of sub-classes of `DiagnosticSovdPortInterface`

[TPS_MANI_01421]{DRAFT} **Support for service validation in the context of SOVD** ⌈The ability to execute a service validation in the context of SOVD is formalized in meta-class `DiagnosticSovdServiceValidationInterface`, modeled as a sub-class of abstract base class `DiagnosticSovdPortInterface`. Application Software exposes a `PPortPrototype` for the service validation.⌋*(RS_MANI_00061, RS_MANI_00070)*

[TPS_MANI_01422]{DRAFT} **Support for update procedures in the context of SOVD** ⌈The ability to execute update procedures in the context of SOVD is formalized in meta-class `DiagnosticSovdUpdateInterface`, modeled as a sub-class of abstract base class `DiagnosticSovdPortInterface`. Application Software exposes a `PPortPrototype` for the update procedure.⌋*(RS_MANI_00061, RS_MANI_00070)*

[TPS_MANI_01423]{DRAFT} **Support for configuration in the context of SOVD** ⌈The ability to execute update procedures in the context of SOVD is formalized in meta-class `DiagnosticSovdConfigurationInterface`, modeled as a sub-class

of abstract base class `DiagnosticSovdPortInterface`. Application Software exposes a `PPortPrototype` for configuration purposes.⌋ *(RS_MANI_00061, RS_MANI_00070)*

Class	***DiagnosticSovdPortInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This abstract meta-class acts as a base class for all diagnostics-related PortInterfaces used in the context of SOVD. **Tags:** atp.Status=candidate			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	DiagnosticSovdAuthorizationInterface, DiagnosticSovdBulkDataInterface, DiagnosticSovdConfiguration Interface, DiagnosticSovdProximityChallengeInterface, DiagnosticSovdServiceValidationInterface, DiagnosticSovdUpdateInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.122: DiagnosticSovdPortInterface

Class	**DiagnosticSovdAuthorizationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD authorization. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.123: DiagnosticSovdAuthorizationInterface

Class	**DiagnosticSovdProximityChallengeInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD proximity challenge. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.124: DiagnosticSovdProximityChallengeInterface

Class	**DiagnosticSovdUpdateInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD update procedure. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.125: DiagnosticSovdUpdateInterface

Class	**DiagnosticSovdBulkDataInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD bulk data transmission. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.126: DiagnosticSovdBulkDataInterface

Class	**DiagnosticSovdServiceValidationInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class is used to type a PPortPrototype for implementing the SOVD service validation. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticPortInterface			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.127: DiagnosticSovdServiceValidationInterface

Class	DiagnosticSovdConfigurationInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::DiagnosticPortInterface			
Note	This meta-class is used to configure a PortInterface for the exchange of configuration content. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DiagnosticPortInterface*, *DiagnosticSovdPortInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.128: DiagnosticSovdConfigurationInterface

3.3.9 Crypto Interfaces

3.3.9.1 Interaction with Crypto Software

[TPS_MANI_03253]{DRAFT} **Interaction with crypto software** ⌈Interaction with crypto software on an instance of the application software shall be modeled on the basis of the existence of `RPortPrototype`s typed by a `PortInterface` that is derived from the abstract meta-class `CryptoInterface`.⌋*(RS_MANI_00031)*

In contrast to the conventions on the AUTOSAR classic Platform, these `CryptoInterface`s are only used on the application side of this communication relation.

The Crypto API is described in [16]. The model-path to an `RPortPrototype` that is referencing a `CryptoInterface` is provided by the `ara::core::InstanceSpecifier` that defines the logical local name used by the application developer in the API call. This local `ara::core::InstanceSpecifier` is translated at runtime with the information from the deployment model to a specific crypto object, e.g. CryptoKeySlot in a CryptoKeyStorage.

Class	***CryptoInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class provides the abstract ability to define a PortInterface for the support of crypto use cases. **Tags:** atp.Status=candidate			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	CryptoCertificateInterface, CryptoKeySlotInterface, CryptoProviderInterface, CryptoTrustMasterInterface			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.129: CryptoInterface

Figure 3.65 depicts all meta-classes that directly inherit from `CryptoInterface`.

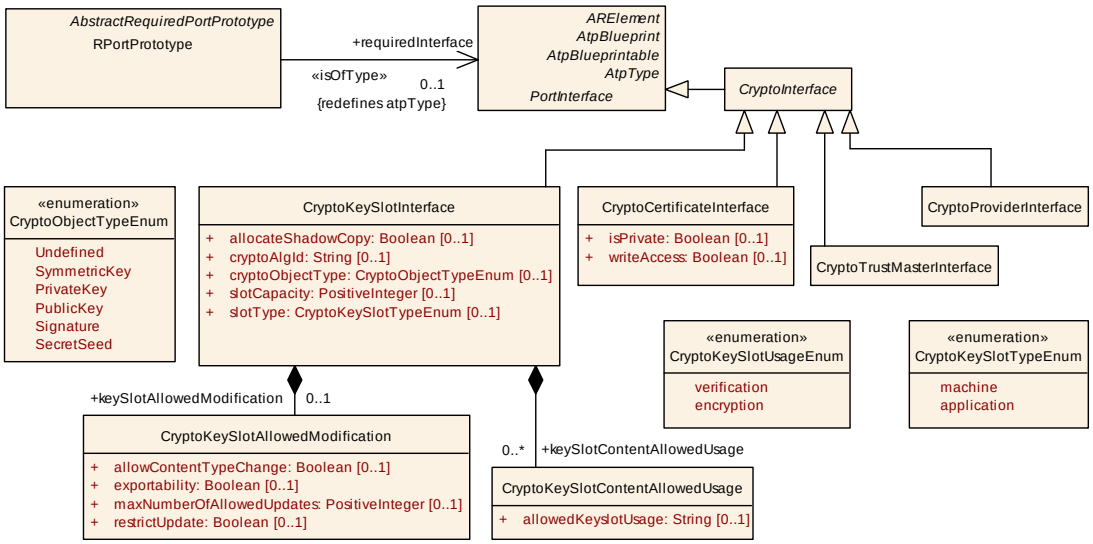


Figure 3.65: `CryptoInterface`s for modeling of the interaction of the Application with the Crypto software

3.3.9.2 Crypto Key Slot Interface

[TPS_MANI_03254]{DRAFT} **Modeling of application that uses and modifies a Crypto Key** ⌈Application software that uses and modifies a Crypto Key is modeled as a `AdaptiveApplicationSwComponentType` with an `RPortPrototype` that is typed by a `CryptoKeySlotInterface` that has the `slotType` value set to `application`.⌋*(RS_MANI_00031)*

[TPS_MANI_03255]{DRAFT} **Modeling of Key Manager application that manages a Crypto Key that is used by Stack Services** ⌈An Key Manager Application that manages a Crypto Key that is used by Stack Services like COM, Persistency or Diagnostic is modeled as a `AdaptiveApplicationSwComponentType` with an `RPortPrototype` that is typed by a `CryptoKeySlotInterface` that has the `slotType` value set to `machine`.⌋*(RS_MANI_00031)*

Class	**CryptoKeySlotInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class provides the ability to define a PortInterface for Crypto Key Slots. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**CryptoKeySlotInterface**			
allocateShadow Copy	Boolean	0..1	attr	This attribute defines whether a shadow copy of this Key Slot shall be allocated to enable rollback of a failed Key Slot update campaign (see interface BeginTransaction). **Tags:** atp.Status=candidate
cryptoAlgId	String	0..1	attr	This attribute defines a crypto algorithm restriction (kAlgId Any means without restriction). The algorithm can be specified partially: family & length, mode, padding. Future Crypto Providers can support some crypto algorithms that are not well known/ standardized today, therefore AUTOSAR doesn't provide a concrete list of crypto algorithms' identifiers and doesn't suppose usage of numerical identifiers. Instead of this a provider supplier should provide string names of supported algorithms in accompanying documentation. The name of a crypto algorithm shall follow the rules defined in the specification of cryptography for Adaptive Platform. **Tags:** atp.Status=candidate
cryptoObject Type	CryptoObjectTypeEnum	0..1	attr	Object type that can be stored in the slot. If this field contains "Undefined" then mSlotCapacity must be provided and larger then 0 **Tags:** atp.Status=candidate
keySlotAllowed Modification	CryptoKeySlotAllowed Modification	0..1	aggr	Restricts how this keySlot may be used **Tags:** atp.Status=candidate
keySlotContent AllowedUsage	CryptoKeySlotContent AllowedUsage	*	aggr	Restriction of allowed usage of a key stored to the slot. **Tags:** atp.Status=candidate
slotCapacity	PositiveInteger	0..1	attr	Capacity of the slot in bytes to be reserved by the stack vendor. One use case is to define this value in case that the cryptoObjectType is undefined and the slot size can not be deduced from cryptoObjectType and cryptoAlgId. "0" means slot size can be deduced from cryptoObject Type and cryptoAlgId. **Tags:** atp.Status=candidate
slotType	CryptoKeySlotType Enum	0..1	attr	This attribute defines whether the keySlot is exclusively used by the Application; or whether it is used by Stack Services and managed by a Key Manager Application. **Tags:** atp.Status=candidate

Table 3.130: CryptoKeySlotInterface

Enumeration	**CryptoKeySlotTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign
Note	This enumeration defines the options for the usage of a Key Slot in the platform. **Tags:** atp.Status=candidate
Aggregated by	CryptoKeySlot.slotType, CryptoKeySlotInterface.slotType
Literal	*Description*
application	KeySlot is used and modified exclusively by the Application. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=candidate

▽

△

Enumeration	**CryptoKeySlotTypeEnum**
machine	Key slot is used by platform modules only. The application manages the key but is not able to use the key. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=candidate

Table 3.131: CryptoKeySlotTypeEnum

Enumeration	**CryptoObjectTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign
Note	Enumeration of all types of crypto objects, i.e. types of content that can be stored to a key slot. **Tags:** atp.Status=candidate
Aggregated by	CryptoKeySlot.cryptoObjectType, CryptoKeySlotInterface.cryptoObjectType
Literal	***Description***
PrivateKey	cryp::PrivateKey object **Tags:** atp.EnumerationLiteralIndex=2 atp.Status=candidate
PublicKey	cryp::PublicKey object **Tags:** atp.EnumerationLiteralIndex=3 atp.Status=candidate
SecretSeed	cryp::SecretSeed object **Tags:** atp.EnumerationLiteralIndex=5 atp.Status=candidate
Signature	cryp::Signature object (asymmetric digital signature or symmetric MAC/HMAC) **Tags:** atp.EnumerationLiteralIndex=4 atp.Status=candidate
SymmetricKey	cryp::SymmetricKey object **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=candidate
Undefined	Object type unknown **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=candidate

Table 3.132: CryptoObjectTypeEnum

Please note that the assignment of a `CryptoKeySlot` to a `CryptoProvider` is described in the deployment model (Machine Manifest). With this mapping also the assignment of the `CryptoKeySlot` to a `CryptoPrimitive` of a `CryptoProvider` is established.

But the application developer is able to restrict the usage of the `CryptoKeySlot` to a specific cryptographic algorithm with the attribute `cryptoAlgId`.

To support crypto algorithms that are not well known/ standardized today, AUTOSAR doesn't provide a concrete list of crypto algorithm's identifiers and doesn't suppose usage of numerical identifiers.

Instead of this a provider supplier should provide string names of supported algorithms in accompanying documentation. The name of a crypto algorithm shall follow the rules defined in the specification of cryptography for Adaptive Platform.

In addition the application developer is able to define further requirements for the usage of the `CryptoKeySlot`. With the attribute `cryptoObjectType` the crypto objects that are allowed to be stored in the key slot can be specified.

The allowed modifications of the key slot can be specified by `keySlotAllowedModification`. The allowed usage of the key slot content can be specified by `keySlotContentAllowedUsage`.

The Integrator needs to take the defined settings in the Application Design model into account if the assignment to the `CryptoKeySlot` in the Crypto Storage is performed. Please note that the Application Design model settings are transferred into the deployment model and are therefore are also available at run-time as described in chapter 10.10.2.

Class	**CryptoKeySlotAllowedModification**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class restricts the allowed modification of a key stored in the key slot. **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	CryptoKeySlot.keySlotAllowedModification, CryptoKeySlotInterface.keySlotAllowedModification			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allowContent TypeChange	Boolean	0..1	attr	This attribute describes whether the key content type can be changed (true) or not (false), e.g. changing the key from symmetric to RSA. **Tags:** atp.Status=candidate
exportability	Boolean	0..1	attr	This attribute describes whether the key slot content is allowed to be exported or not. **Tags:** atp.Status=candidate
maxNumberOf AllowedUpdates	PositiveInteger	0..1	attr	This attribute describes the maximum updates that are allowed to the slot. **Tags:** atp.Status=candidate
restrictUpdate	Boolean	0..1	attr	This attribute defines whether restrictions on the number of updates are defined or not. • false: no restriction is placed on the number of updates. • true: restrictions are placed on the number of updates with the attribute maxNumberOfAllowedUpdates. **Tags:** atp.Status=candidate

Table 3.133: CryptoKeySlotAllowedModification

Class	**CryptoKeySlotContentAllowedUsage**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class restricts the allowed usage of a key stored in the key slot. **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	CryptoKeySlot.keySlotContentAllowedUsage, CryptoKeySlotInterface.keySlotContentAllowedUsage			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allowedKeyslot Usage	String	0..1	attr	This attribute defines for which operations the KeySlot may be used. **Tags:** atp.Status=candidate

Table 3.134: CryptoKeySlotContentAllowedUsage

[constr_5238]{DRAFT} **CryptoKeySlotAllowedModification.restrictUpdate and the relationship to maxNumberOfAllowedUpdates** ⌈If the CryptoKeySlotAllowedModification.restrictUpdate is set to true then CryptoKeySlotAllowedModification.maxNumberOfAllowedUpdates shall be set to a value **at the time before the generation of the ara API starts**.⌋*()*

[constr_5239]{DRAFT} **Predefined values for CryptoKeySlotContentAllowedUsage.allowedKeyslotUsage** ⌈The following values for CryptoKeySlotContentAllowedUsage.allowedKeyslotUsage are predefined by AUTOSAR:

- ALLOW-DATA-ENCRYPTION,
- ALLOW-DATA-DECRYPTION,
- ALLOW-SIGNATURE,
- ALLOW-VERIFICATION,
- ALLOW-KEY-AGREEMENT,
- ALLOW-KEY-DIVERSIFY,
- ALLOW-DRNG-INIT,
- ALLOW-KDF-MATERIAL,
- ALLOW-KEY-EXPORTING,
- ALLOW-KEY-IMPORTING,
- ALLOW-EXACT-MODE-ONLY,
- ALLOW-DERIVED-DATA-ENCRYPTION,
- ALLOW-DERIVED-DATA-DECRYPTION,
- ALLOW-DERIVED-SIGNATURE,
- ALLOW-DERIVED-VERIFICATION,
- ALLOW-DERIVED-DIVERSIFY,

- ALLOW-DERIVED-DRNG-INIT,
- ALLOW-DERIVED-KDF-MATERIAL,
- ALLOW-DERIVED-KEY-EXPORTING,
- ALLOW-DERIVED-KEY-IMPORTING,
- ALLOW-DERIVED-EXACT-MODE-ONLY

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

3.3.9.3 Crypto Certificate Interface

[TPS_MANI_03256]{DRAFT} **Modeling of application that accesses a Crypto Certificate** ⌈Application software that accesses a Crypto Certificate is modeled as a AdaptiveApplicationSwComponentType with an RPortPrototype that is typed by a CryptoCertificateInterface.⌋*(RS_MANI_00031)*

Class	**CryptoCertificateInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class provides the ability to define a PortInterface for a CryptoCertificate. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
isPrivate	Boolean	0..1	attr	This attribute controls the possibility to access the content of the CryptoCertificateSlot by Find() interfaces of the X509 Provider. **Tags:** atp.Status=candidate
writeAccess	Boolean	0..1	attr	This attribute defines whether the application has write-access to the CryptoCertificate (true) or only read-access (false). **Tags:** atp.Status=candidate

Table 3.135: CryptoCertificateInterface

3.3.9.4 Crypto Provider Interface

[TPS_MANI_03257]{DRAFT} **Modeling of application that accesses a Crypto Provider** ⌈Application software that accesses a Crypto Provider is modeled as a AdaptiveApplicationSwComponentType with an RPortPrototype that is typed by a CryptoProviderInterface.⌋*(RS_MANI_00031)*

Please note that the CryptoProviderInterface shall be used if the Application needs to access a Crypto Provider to execute keyless operations, e.g. Hashing, Random Number Generation. For cryptographic transformations that require keys the CryptoKeySlotInterface may be used.

Class	**CryptoProviderInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class provides the ability to define a PortInterface for a CryptoProvider. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 3.136: CryptoProviderInterface

3.3.9.5 Crypto TrustMaster Interface

[TPS_MANI_03258]{DRAFT} **Modeling of application designed as trust-master** ⌈Application software designed as trust-master is modeled as a AdaptiveApplicationSwComponentType with an RPortPrototype that is typed by a CryptoTrustMasterInterface.⌋ *(RS_MANI_00031)*

An Application requires TrustMaster privileges to set global (machine-wide) root-of-trust certificates. Note: such a certificate may not be private.

Class	**CryptoTrustMasterInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class provides the ability to define a PortInterface for TrustMaster. **Tags:** atp.Status=candidate atp.recommendedPackage=CryptoInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *CryptoInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 3.137: CryptoTrustMasterInterface

3.3.9.6 Linking of Crypto Certificate to a Crypto Key Slot

It is possible to model a link between a Crypto Certificate and a Crypto KeySlot in the Application Design with the meta-class SwcServiceDependency that aggregates

CryptoCertificateKeySlotNeeds in the role serviceNeeds and RoleBasedPortAssignments that refer to an RPortPrototype that is typed by a CryptoCertificateInterface and an RPortPrototype that is typed by a CryptoKeySlotInterface.

[TPS_MANI_03259]{DRAFT} **Linking of Crypto Certificate to a Crypto Key Slot** ⌈

ServiceNeeds kind CryptoCertificateKeySlotNeeds

RoleBasedPortAssignment valid roles:

- CryptoKeySlotInterface [1]
- CryptoCertificateInterface [1]

RoleBasedDataAssignment
N/A

RepresentedPortGroups
N/A

⌋(*RS_MANI_00031*)

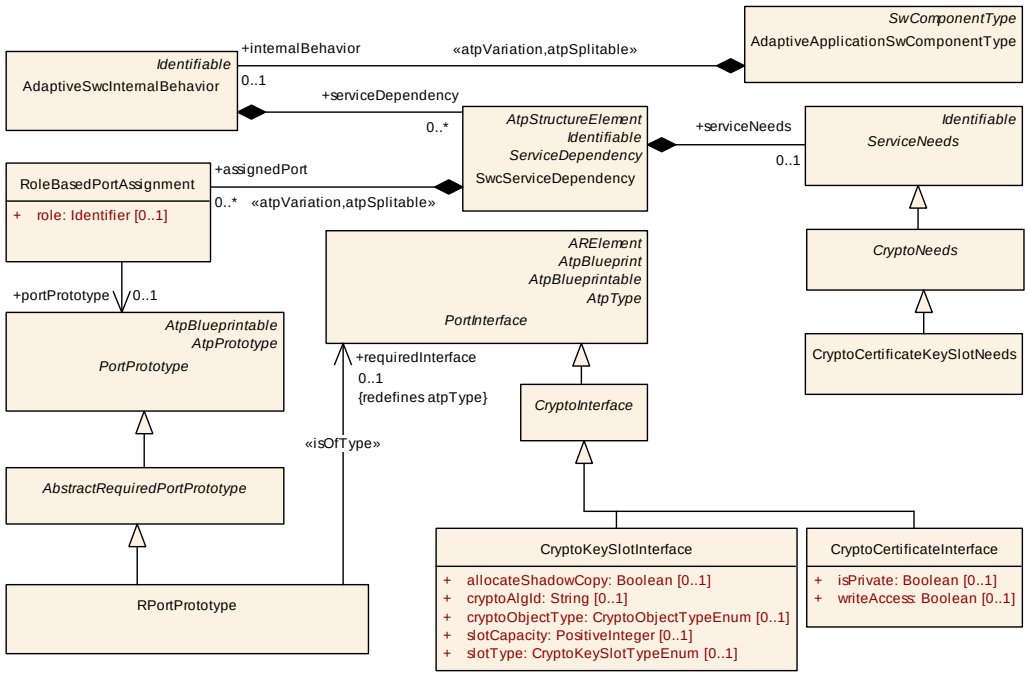


Figure 3.66: Linking of Crypto Certificate with Crypto Key Slot in Application Design

Class	***CryptoNeeds*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign
Note	Specifies the abstract needs on the configuration of Crypto. **Tags:** atp.Status=candidate

▽

△

Class	***CryptoNeeds*** (abstract)			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds*			
Subclasses	CryptoCertificateKeySlotNeeds			
Aggregated by	BswServiceDependency.serviceNeeds, SwcServiceDependency.serviceNeeds			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.138: CryptoNeeds

Class	**CryptoCertificateKeySlotNeeds**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CryptoDesign			
Note	This meta-class shall be taken to indicate that the SwcServiceDependecy modeled with this kind of ServiceNeeds defines a relationship between a CryptoKeySlot and a CryptoCertificate. **Tags:** atp.Status=candidate			
Base	*ARObject*, *CryptoNeeds*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds*			
Aggregated by	BswServiceDependency.serviceNeeds, SwcServiceDependency.serviceNeeds			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.139: CryptoCertificateKeySlotNeeds

An example how the `SwcServiceDependency` is used to create a relation between a Crypto Certificate and a Crypto KeySlot can be found in Appendix B.7.1.

3.3.10 Raw Data Stream Interface

In some cases it is necessary for the application software to be able to process raw binary data streams sent over a communication channel. Obviously, SOME/IP serialization does not make sense in such a scenario, as would the modeling of `AutosarDataType`s, i.e. the creation of a `ServiceInterface`.

Therefore, a different mechanism that actively supports the requirements of raw data streaming is available on the *AUTOSAR adaptive platform*.

As far as the application software is concerned, the interaction with a raw data stream is based on the usage of an `RPortPrototype` typed by either a `RawDataStreamClientInterface` or a `RawDataStreamServerInterface`.

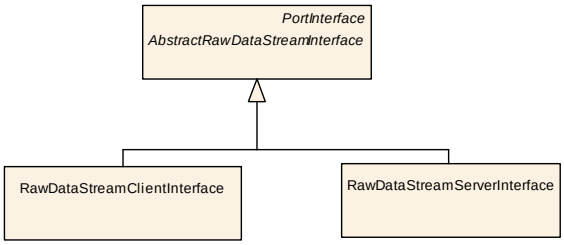


Figure 3.67: Modeling of `PortInterface`s for raw data stream

This kind of PortInterface does neither support nor require any elements with a modeled data type, i.e. an AutosarDataType.

Class	**AbstractRawDataStreamInterface** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class serves as an abstract base class for PortInterfaces related to raw data streams.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	RawDataStreamClientInterface, RawDataStreamServerInterface			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.140: AbstractRawDataStreamInterface

Class	**RawDataStreamClientInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the necessary capabilities for raw data streaming on the client side, i.e. the streaming of data that do not undergo any serialization. Each RawDataStreamClientInterface supports the following capabilities without further modeling: • connect: set up the communication channel • shutdown: close the communication channel • write: send data down the communication channel • read: access incoming data on the communication channel **Tags:** atp.recommendedPackage=RawDataStreamInterfaces			
Base	*ARElement*, *ARObject*, *AbstractRawDataStreamInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.141: RawDataStreamClientInterface

Class	**RawDataStreamServerInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the necessary capabilities for raw data streaming on the server side, i.e. the streaming of data that do not undergo any serialization. Each RawDataStreamServerInterface supports the following capabilities without further modeling: • waitForConnection: wait until a communication channel is set up. • shutdown: close the communication channel • write: send data down the communication channel • read: access incoming data on the communication channel **Tags:** atp.recommendedPackage=RawDataStreamInterfaces			
Base	*ARElement*, *ARObject*, *AbstractRawDataStreamInterface*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.142: RawDataStreamServerInterface

3.3.11 Intrusion Detection Interface

This section provides documentation for PortInterfaces used in the context of intrusion detection. These PortInterfaces are modeled as sub-classes of the abstract base class IdsmAbstractPortInterface.

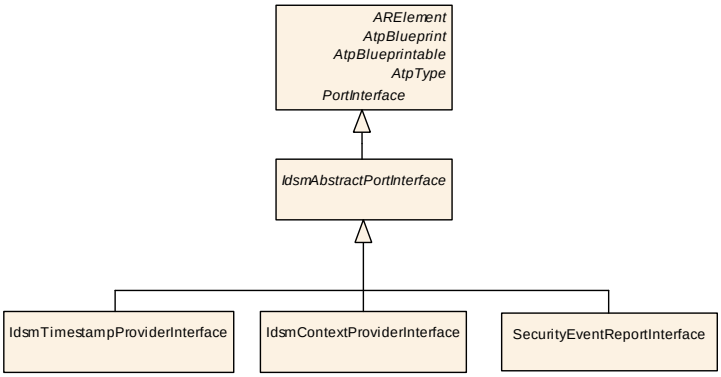


Figure 3.68: Abstract base-class IdsmAbstractPortInterface

Class	***IdsmAbstractPortInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This abstract meta-class acts as a base class for all kinds of PortInterfaces related to security event handling.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	IdsmContextProviderInterface, IdsmTimestampProviderInterface, SecurityEventReportInterface			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.143: IdsmAbstractPortInterface

3.3.11.1 Security Event Report Interface

On the *AUTOSAR adaptive platform*, a dedicated PortInterface for the interaction of application-layer software with the AUTOSAR Intrusion Detection System Manager is defined.

The name of this sub-class of abstract meta-class PortInterface is SecurityEventReportInterface.

Class	SecurityEventReportInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to define a PortInterface for the reporting of security events in the context of the intrusion detection system. **Tags:** atp.Status=candidate atp.recommendedPackage=SecurityEventReportInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *IdsmAbstractPortInterface*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.144: SecurityEventReportInterface

[TPS_MANI_01340]{DRAFT} **Semantics of `SecurityEventReportInterface`** ⌈Each `RPortPrototype` typed by a `SecurityEventReportInterface` is able to report exactly one security event.⌋*(RS_MANI_00068)*

3.3.11.2 Intrusion Detection Timestamp Provider Interface

[TPS_MANI_01439]{DRAFT} **Semantics of meta-class `IdsmTimestampProviderInterface`** ⌈A `PPortPrototype` that is typed by a `IdsmTimestampProviderInterface` implements a *time-stamp provider* for the *Intrusion Detection System Management.*⌋*()*

Class	IdsmTimestampProviderInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to define a PortInterface for providing a timestamp for security events in the context of the intrusion detection system. **Tags:** atp.recommendedPackage=IdsmPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *IdsmAbstractPortInterface*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.145: IdsmTimestampProviderInterface

3.3.11.3 Intrusion Detection Context Provider Interface

[TPS_MANI_01442]{DRAFT} **Semantics of meta-class `IdsmContextProviderInterface`** ⌈A `PPortPrototype` that is typed by a `IdsmContextProviderInterface` implements a *context provider* for the *Intrusion Detection System Management.*⌋*()*

Class	IdsmContextProviderInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to define a PortInterface for providing a Context for security events in the context of the intrusion detection system. **Tags:** atp.recommendedPackage=IdsmPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *IdsmAbstractPortInterface*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 3.146: IdsmContextProviderInterface

3.3.12 Log And Trace Interface

On the *AUTOSAR adaptive platform*, a dedicated `PortInterface` named `LogAndTraceInterface` is defined for the interaction of application-layer software with the AUTOSAR Logging and Tracing Functional Cluster.

Class	LogAndTraceInterface			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a PortInterface for support of Logging or Tracing. **Tags:** atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 3.147: LogAndTraceInterface

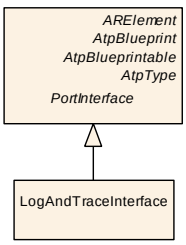


Figure 3.69: Specification of the `LogAndTraceInterface`

[TPS_MANI_03284]{DRAFT} **Semantics of `LogAndTraceInterface`** ⌈Each `RPortPrototype` typed by a `LogAndTraceInterface` is able to forward logging information onto the external Dlt Log Viewer.⌋ *(RS_MANI_00037)*

[constr_5290]{DRAFT} **`PPortPrototype` is not allowed to be typed by `LogAndTraceInterface`** ⌈A `PPortPrototype` is not allowed to reference a `LogAndTraceInterface` in the role `providedInterface` **at the time before the generation of the ara API starts**.⌋ *()*

Please note that the mapping of an RPortPrototype typed by a LogAndTraceInterface to a DltLogSink is described in the deployment model in chapter 10.6.2.

Class	**LTMessageCollectionToPortPrototypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::LogAndTrace			
Note	This mapping element assigns a collection of Log or Trace messages to a PortPrototype of an application. **Tags:** atp.recommendedPackage=LTMessageCollectionToPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
logAndTrace Message CollectionSet	LogAndTraceMessage CollectionSet	0..1	ref	Reference to a Collection of Log or Trace messages
rPortPrototype	RPortPrototype	0..1	ref	Reference to the RPortPrototype to which Log or Trace messages are assigned.

Table 3.148: LTMessageCollectionToPortPrototypeMapping

Class	**LogAndTraceMessageCollectionSet**			
Package	M2::AUTOSARTemplates::LogAndTraceExtract			
Note	Collection of DltMessages **Tags:** atp.recommendedPackage=LogAndTraceMessageCollectionSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dltMessage	DltMessage	*	aggr	Collection of DltMessages in the DltMessageCollection Set. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dltMessage.shortName, dltMessage.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime

Table 3.149: LogAndTraceMessageCollectionSet

[constr_5291]{DRAFT} **Allowed usage of LTMessageCollectionToPortPrototypeMapping.rPortPrototype** ⌈An LTMessageCollectionToPortPrototypeMapping shall (in the role rPortPrototype) only refer to a RPortPrototype that is typed by a LogAndTraceInterface.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

[TPS_MANI_03285]{DRAFT} **Semantics of LTMessageCollectionToPortPrototypeMapping** ⌈With the LTMessageCollectionToPortPrototypeMapping it is possible to describe at application design time a collection of Log or Trace messages that will be used by the application.⌋*(RS_MANI_00023)*

3.3.13 Firewall Interface

On the *AUTOSAR adaptive platform*, a dedicated `PortInterface` named `FirewallStateSwitchInterface` is defined for the interaction of application-layer software with the Firewall Cluster.

Class	**FirewallStateSwitchInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a PortInterface for interaction with the Firewall mode. **Tags:** atp.Status=candidate atp.recommendedPackage=FirewallStateSwitchPortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
firewallState Machine	ModeDeclarationGroup Prototype	*	aggr	The state machine of this firewall interface. **Tags:** atp.Status=candidate

Table 3.150: FirewallStateSwitchInterface

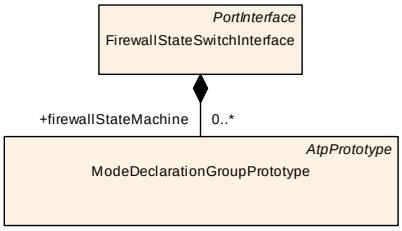


Figure 3.70: Specification of the `FirewallStateSwitchInterface`

[TPS_MANI_03321] Semantics of `FirewallStateSwitchInterface` ⌈The `RPortPrototype` typed by a `FirewallStateSwitchInterface` provides the functionality to switch the firewall state.⌋*()*

[TPS_MANI_03295] Semantics of `FirewallStateSwitchInterface.firewallStateMachine` ⌈The `ModeDeclarationGroupPrototype` aggregated in the `FirewallStateSwitchInterface` in the role `firewallStateMachine` defines the state machine of the Firewall Cluster.⌋*()*

3.3.14 Network Management Interface

[TPS_MANI_01419]{DRAFT} **Semantics of `NetworkManagementPortInterface`** ⌈The `NetworkManagementPortInterface` is supposed to be used by state management to interact with the network management for cases that are intentionally (e.g. because of the complexity of the rules to be applied for a specific case) not modeled in the context of `StateManagementModuleInstantiation`.⌋*()*

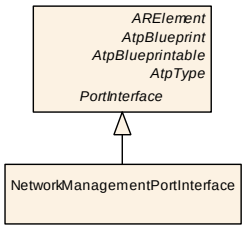


Figure 3.71: Specification of the `NetworkManagementPortInterface`

Class	**NetworkManagementPortInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This PortInterface shall be used to submit triggers to the state management **Tags:** atp.recommendedPackage=NetworkManagementInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.151: NetworkManagementPortInterface

3.4 Data Type

3.4.1 Overview

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

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

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

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

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

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

3.4.2 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 ApplicationData Types only.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*ApplicationCompositeDataType*, ApplicationPrimitiveDataType			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.152: ApplicationDataType

3.4.2.1 String Data Type

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

For more information about the modeling of string data types on the level of CppImplementationDataType please refer to section 3.4.3.4.

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

The meta-classes used to define an ApplicationPrimitiveDataType of category STRING are summarized in Figure 3.72.

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*.

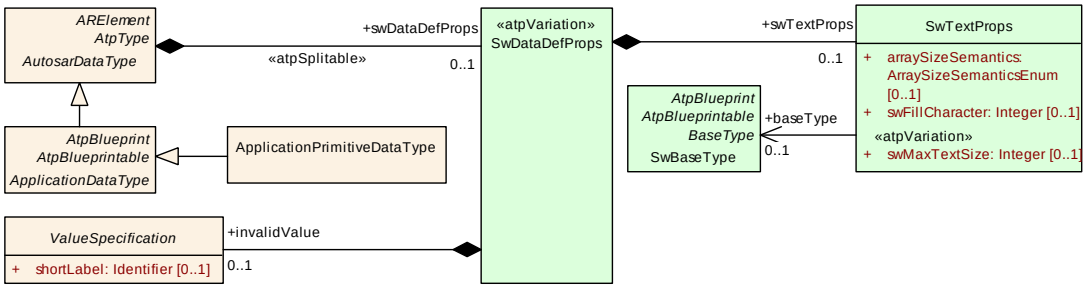


Figure 3.72: Specification of textual strings

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

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

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

Table 3.153: 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*			
Aggregated by	SwDataDefProps.swTextProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls the semantics of the arraysize for the array representing the string in an Implementation DataType. It is there to support a safe conversion between ApplicationDatatype and ImplementationDatatype, even for variable length strings as required e.g. for Support of SAE J1939.

▽

△

Class	**SwTextProps**			
baseType	SwBaseType	0..1	ref	This is the base type of one character in the string. In particular this baseType denotes the intended encoding of the characters in the string on level of ApplicationData Type. **Tags:** xml.sequenceOffset=30
swFillCharacter	Integer	0..1	attr	Filler character for text parameter to pad up to the maximum length swMaxTextSize. The value will be interpreted according to the encoding specified in the associated base type of the data object, e.g. 0x30 (hex) represents the ASCII character zero as filler character and 0 (dec) represents an end of string as filler character. The usage of the fill character depends on the arraySize Semantics. **Tags:** xml.sequenceOffset=40
swMaxTextSize	Integer	0..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 3.154: SwTextProps

Enumeration	**ArraySizeSemanticsEnum**
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes
Note	This type controls how the information about the number of elements in an ApplicationArrayDataType is to be interpreted.
Aggregated by	ApplicationArrayElement.arraySizeSemantics, DiagnosticDataElement.arraySizeSemantics, ImplementationDataTypeElement.arraySizeSemantics, SwTextProps.arraySizeSemantics
Literal	*Description*
fixedSize	This means that the ApplicationArrayDataType will always have a fixed number of elements. **Tags:** atp.EnumerationLiteralIndex=0
variableSize	This implies that the actual number of elements in the ApplicationArrayDataType might vary at run-time. The value of arraySize represents the maximum number of elements in the array. **Tags:** atp.EnumerationLiteralIndex=1

Table 3.155: ArraySizeSemanticsEnum

Class	**SwBaseType**			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This meta-class represents a base type used within ECU software. **Tags:** atp.recommendedPackage=BaseTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *BaseType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 3.156: SwBaseType

3.4.2.2 Associative Map Data Type

[TPS_MANI_01027]{DRAFT} **Semantics of `ApplicationAssocMapDataType`** ⌈An `ApplicationAssocMapDataType` represents an associative data structure, i.e. a data structure where so-called *key*s (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)*

Figure 3.73 depicts an example of the structure of an `ApplicationAssocMapDataType`.

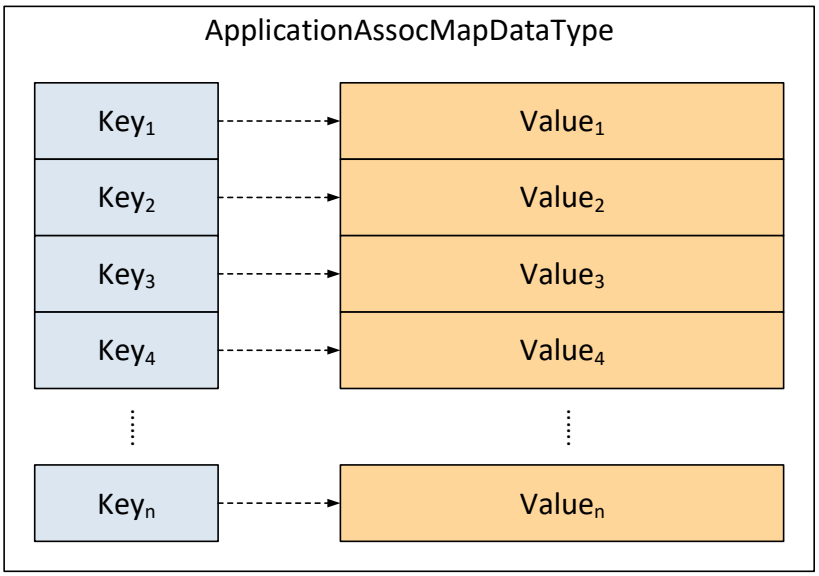


Figure 3.73: Example `ApplicationAssocMapDataType` on the *AUTOSAR adaptive platform*

[constr_3349]{DRAFT} **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*.

This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

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

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

As can be deduced from looking at Figure 3.73, the concept of an ApplicationDataType of category MAP shall not be confused with an ApplicationAssocMapDataType[7].

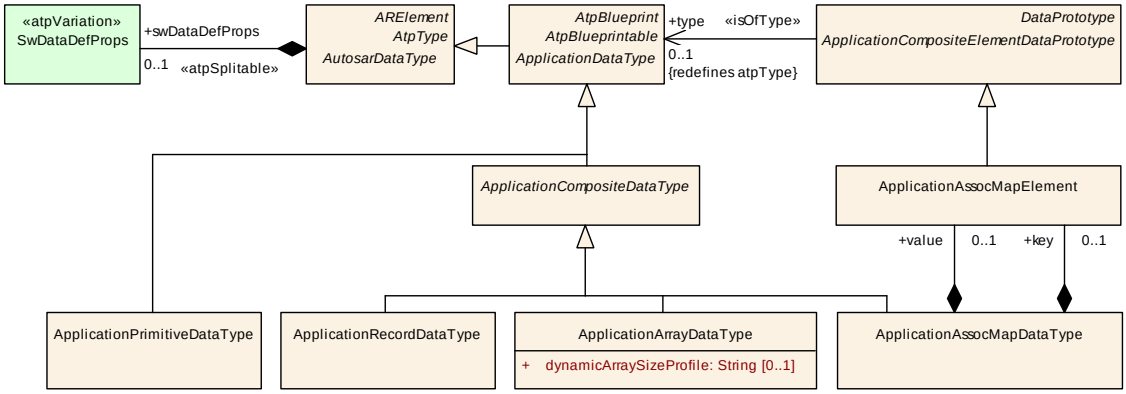


Figure 3.74: Formal model of ApplicationAssocMapDataType

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

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

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

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

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

Class	**ApplicationAssocMapDataType**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType
Note	An application data type which is a map and consists of a key and a value **Tags:** atp.recommendedPackage=ApplicationDataTypes

▽

[7] 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

△

Class	ApplicationAssocMapDataType			
Base	ARElement, ARObject, ApplicationCompositeDataType, ApplicationDataType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
key	ApplicationAssocMapElement	0..1	aggr	Key element of the map that is used to uniquely identify the value of the map.
value	ApplicationAssocMapElement	0..1	aggr	Value element of the map that stores the content associated to a key.

Table 3.157: ApplicationAssocMapDataType

[constr_10124]{DRAFT} **Multiplicity of attribute ApplicationAssocMapDataType.key** ⌈For each ApplicationAssocMapDataType, the attribute key shall exist **at the time before the generation of the ara API starts**.⌋*()*

[constr_10125]{DRAFT} **Multiplicity of attribute ApplicationAssocMapDataType.value** ⌈For each ApplicationAssocMapDataType, the attribute value shall exist **at the time before the generation of the ara API starts**.⌋*()*

Class	ApplicationAssocMapElement			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType			
Note	Describes the properties of the elements of an application map data type.			
Base	ARObject, ApplicationCompositeElementDataPrototype, AtpFeature, AtpPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable			
Aggregated by	ApplicationAssocMapDataType.key, ApplicationAssocMapDataType.value, *AtpClassifier*.atpFeature			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.158: ApplicationAssocMapElement

An example of the definition of an ApplicationAssocMapDataType can be found in Appendix B.1.1.

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

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

[constr_1488]{DRAFT} **Initialization of a DataPrototype typed by an ApplicationAssocMapDataType** ⌈A DataPrototype typed by an ApplicationAssocMapDataType shall only be initialized by an ApplicationAssocMapValueSpecification **at the time before the generation of the ara API starts**.⌋*()*

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

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

[constr_1489]{DRAFT} **Uniqueness of `ApplicationAssocMapValueSpecification.mapElementTuple.key`** ⌈The value of all `mapElementTuple.key` elements in the context of a given `ApplicationAssocMapValueSpecification` shall be unique **`at the time before the generation of the ara API starts`**.⌋()

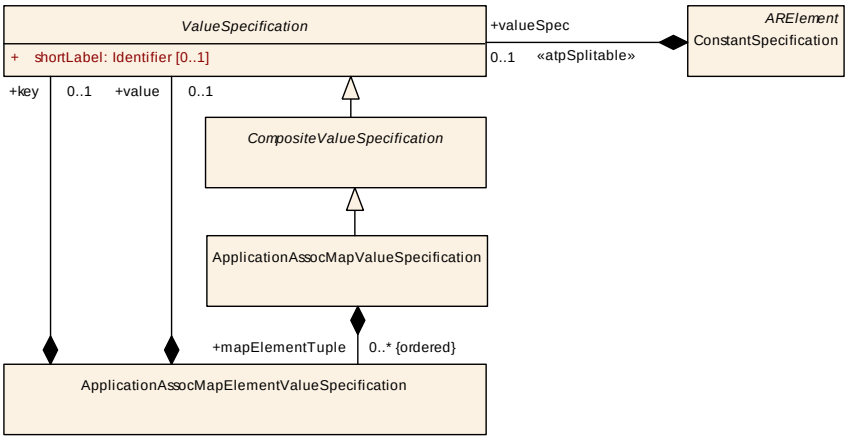


Figure 3.75: Formal model of the initialization of an `ApplicationAssocMapDataType`

Class	**ApplicationAssocMapValueSpecification**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType			
Note	This meta-class represents the ability to define the initialization of an ApplicationAssocMapDataType.			
Base	*ARObject*, *CompositeValueSpecification*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValueSpecification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, CompositeRuleBasedValueSpecification.argument, ConstantSpecification.valueSpec, CryptoServiceKey.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElementCondition.compareValue, FieldSenderComSpec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, SomeipEventDeployment.eventReceptionDefaultValue, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableDataPrototype.initValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
mapElementTuple (ordered)	ApplicationAssocMapElementValueSpecification	*	aggr	This aggregation represents the initial values for the elements of the ApplicationAssocMapValueSpecification.

Table 3.159: ApplicationAssocMapValueSpecification

Class	**ApplicationAssocMapElementValueSpecification**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationDataType			
Note	This meta-class represents the ability to define the initialization of the elements of an ApplicationAssoc MapDataType.			
Base	*ARObject*			
Aggregated by	ApplicationAssocMapValueSpecification.mapElementTuple			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
key	ValueSpecification	0..1	aggr	This aggregation represents the initialization of the key part of an AssociativeElementValueSpecification.
value	ValueSpecification	0..1	aggr	This aggregation represents the initialization of the value part of an AssociativeElementValueSpecification.

Table 3.160: ApplicationAssocMapElementValueSpecification

[constr_10126]{DRAFT} **Multiplicity of attribute `ApplicationAssocMapElementValueSpecification.key`** ⌈For each `ApplicationAssocMapElementValueSpecification`, the attribute `key` shall exist **`at the time before the generation of the ara API starts`**.⌋*()*

[constr_10127]{DRAFT} **Multiplicity of attribute `ApplicationAssocMapElementValueSpecification.value`** ⌈For each `ApplicationAssocMapElementValueSpecification`, the attribute `value` shall exist **`at the time before the generation of the ara API starts`**.⌋*()*

3.4.2.3 Attributes of SwDataDefProps

This section provides clarification about which attributes of meta-class `SwDataDefProps` used in the context of `ApplicationDataType`s relevant for the *AUTOSAR adaptive platform*.

A consequence of [constr_1478] is that the [constr_1478] only lists the values of `category` that are exclusive 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].

[constr_1478]{DRAFT} **SwDataDefProps applicable to ApplicationDataTypes exclusive to the *AUTOSAR adaptive platform*** ⌈

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

This rule shall be imposed **at the time before the generation of the ara API starts**

⌋()

3.4.3 CppImplementationDataType

3.4.3.1 Overview

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

Therefore, AUTOSAR implements[8] a multi-level approach for the modeling of data types. One of the described levels, the so-called *Implementation Data Level* aims at a modeling on a level that could be described as "language binding" in the parlor of the *AUTOSAR adaptive platform*.

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

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

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

While it would technically be feasible to extend the semantics of `ImplementationDataType` for a support of a C++ language binding this would significantly water down the clarity and expressiveness of `ImplementationDataType`[9].

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

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

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

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

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

[8] As explained in [1]

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

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

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

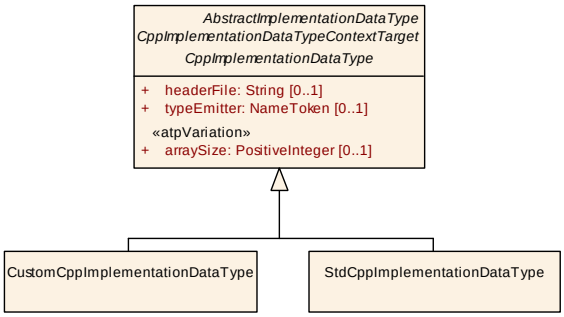


Figure 3.76: Specializations of `CppImplementationDataType`

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

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

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

[TPS_MANI_01167]{DRAFT} **`AbstractImplementationDataType`** ⌈Meta-class `CppImplementationDataType` inherits from abstract base class `AbstractImplementationDataType` in order to become a valid target for specific references from other meta-classes that want to refer to "`ImplementationDataType` in general".⌋ *(RS_MANI_00039)*

Class	***CppImplementationDataType*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class represents the way to specify a reusable data type definition taken as a the basis for a C++ language binding			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	CustomCppImplementationDataType, StdCppImplementationDataType			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	CppImplementationDataType (abstract)			
arraySize	PositiveInteger	0..1	attr	This attribute can be used to specify the array size if the enclosing CppImplementationDataType has array semantics. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
headerFile	String	0..1	attr	Configuration of the Header File with the custom class declaration.
namespace (ordered)	SymbolProps	*	aggr	This aggregation allows for the definition an own namespace for the enclosing CppImplementationData Type.
subElement (ordered)	CppImplementation DataTypeElement	*	aggr	This represents the collection of sub-elements of the enclosing CppImplementationDataType
template Argument (ordered)	CppTemplateArgument	*	aggr	This aggregation allows for the specification of properties of template arguments
typeEmitter	NameToken	0..1	attr	This attribute can be taken to control how the respective CppImplementationDataType is contributed to the language binding.
typeReference	CppImplementation DataType	0..1	ref	This reference shall be defined to define a type reference (a.k.a. typedef).

Table 3.161: CppImplementationDataType

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

Table 3.162: StdCppImplementationDataType

Class	CustomCppImplementationDataType			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class represents the way to specify a data type definition that is taken as the basis for a C++ language binding to a custom implementation that is declared in the configured header file. The Short Name of this CustomCppImplementationDataType defines the Class-Name of the custom implementation. **Tags:** atp.recommendedPackage=CppImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataType*, *CppImplementationData TypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 3.163: CustomCppImplementationDataType

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

Table 3.164: AbstractImplementationDataType

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

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

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

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

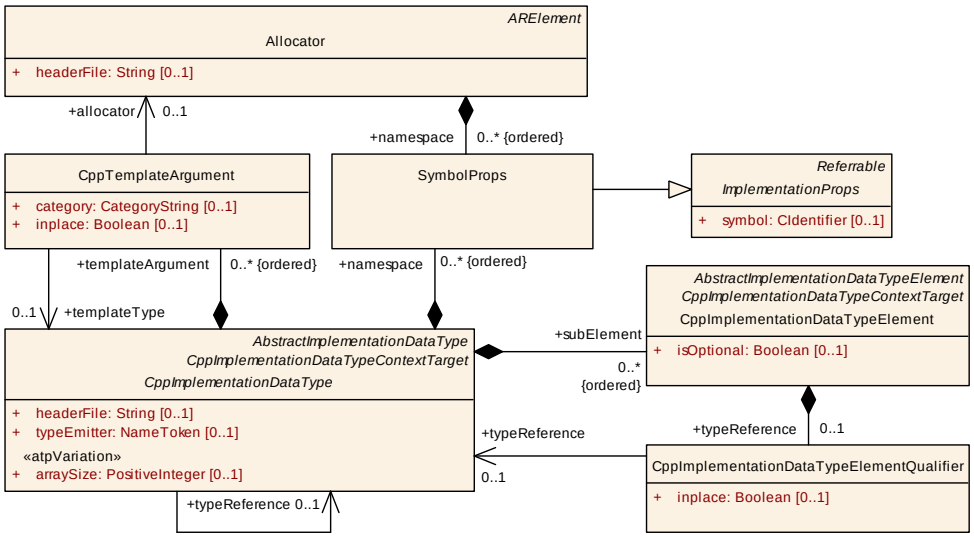


Figure 3.77: CppImplementationDataType overview

[constr_3443]{DRAFT} **Specification of a namespace for a StdCppImplementationDataType** ⌈The definition of a namespace for a StdCppImplementationDataType of category VALUE is not allowed.

For this value of category the std namespace is already assumed by the usage of the StdCppImplementationDataType.

This rule shall be imposed **at the time before the generation of the ara API starts**⌋*()*

[TPS_MANI_01309]{DRAFT} **Semantics of attribute CppImplementationDataType.headerFile** ⌈The attribute CppImplementationDataType.headerFile shall be used to specify the name of the corresponding header file in two cases:

- A CustomCppImplementationDataType shall set the value of the attribute to the name of the header file that defines the C++ code for the CustomCppImplementationDataType.
- A platform data type (modeled as a StdCppImplementationDataType) shall set the attribute to the name of the applicable header file (e.g. "cstdint") from the C++ standard library.

⌋*(RS_MANI_00016)*

[constr_1743]{DRAFT} **CppImplementationDataType.headerFile vs. CppImplementationDataType.typeEmitter** ⌈The two attributes CppImplementationDataType.headerFile and CppImplementationDataType.typeEmitter shall always be used mutually exclusive.

In other words, a subclass of CppImplementationDataType shall either use headerFile or typeEmitter. The simultaneous usage of both attributes is not supported.

This rule shall be imposed **at the time before the generation of the ara API starts**⌋*()*

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

- TYPE_EMITTER_ARA
- FUNDAMENTAL_TYPE: this value is only applicable for the platform types bool, float, and double.

⌋*(RS_MANI_00039)*

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

- If the attribute typeEmitter is set to the value TYPE_EMITTER_ARA, the ARA generator shall generate the corresponding data type definition.
- If the attribute typeEmitter is set to any value other than TYPE_EMITTER_ARA, the ARA generator shall silently **not** generate the corresponding data type definition.

⌋(*RS_MANI_00039*)

In the context of [TPS_MANI_01177], [TPS_MANI_01309] and [constr_1743] apply.

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

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

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

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

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

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

- `CppTemplateArgument.templateType` or
- `CppTemplateArgument.allocator`

may exist **`at the time before the generation of the ara API starts`**.⌋()

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

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

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

⌋(*RS_MANI_00039*)

Class	**CppTemplateArgument**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class has the ability to define properties for template arguments.			
Base	*ARObject*			
Aggregated by	*CppImplementationDataType*.templateArgument			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allocator	Allocator	0..1	ref	This reference identifies the applicable allocator.
category	CategoryString	0..1	attr	This attribute shall be used to contribute further clarification regarding the semantics of the enclosing Cpp TemplateArgument.
inplace	Boolean	0..1	attr	This attribute specifies whether the shortName of the referenced templateType is used in the code generation and the type declaration is defined outside of the enclosing CppImplementationDataType (true) or whether the type definition is embedded inside of the enclosing CppImplementationDataType and the shortName is ignored (false).
templateType	CppImplementation DataType	0..1	ref	This reference identifies the data type of the specific template argument required for the language binding.

Table 3.165: CppTemplateArgument

[constr_1578]{DRAFT} Applicable data categories ⌈

Category	Applicable to ...								Description
	ApplicationArrayDataType	ApplicationRecordDataType	ApplicationPrimitiveDataType	ApplicationRecordElement	ApplicationArrayElement	ApplicationValueSpecification	StdCppImplementationDataType	CustomCppImplementationDataType	
VALUE			x	x	x	x	x		Contains a single value. See also [TPS_MANI_03192].
TYPE_REFERENCE							x		The element is defined via reference to another data type (via CppImplementationDataType.typeReference.
STRUCTURE		x		x	x		x		Holds one or several further elements which can have different AutosarDataTypes. See also [TPS_MANI_03180].
VARIANT							x	x	Can hold values of different data types. It is similar to STRUCTURE except that all of its members start at the same location in memory. A VARIANT data prototype can contain only one of its elements at a time and represents a type-safe union. The size of the VARIANT is at least the size of the largest member. See also [TPS_MANI_03189].
ARRAY	x			x	x		x	x	A fixed-sized array of sub-elements of the same data type. See also [TPS_MANI_03169].
VECTOR							x	x	An array of elements of the same data type that is able to grow at run-time. See also [TPS_MANI_03174].

▽

△

Category	Applicable to ...								Description
	ApplicationArrayDataType	ApplicationRecordDataType	ApplicationPrimitiveDataType	ApplicationRecordElement	ApplicationArrayElement	ApplicationValueSpecification	StdCppImplementationDataType	CustomCppImplementationDataType	
ASSOCIATIVE_MAP							x	x	An associative array of key-value pairs. See also [TPS_MANI_03183].
STRING			x	x	x	x	x		Contains a text string. See also [TPS_MANI_03178].
BOOLEAN			x	x	x	x			Contains one boolean state. Depending on the CPU direct addressing of single bits may not be available. So a byte or a word can be used to store only one logical state.

This rule shall be imposed **at the time before the generation of the ara API starts**

⌋()

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

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

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

Class	**CppImplementationDataTypeElement**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	Declares a data object which is locally aggregated. Such an element can only be used within the scope where it is aggregated. A CppImplementationDataTypeElement is used to represent an element of a structure, defining its type.			
Base	*ARObject*, *AbstractImplementationDataTypeElement*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *CppImplementationDataType*.subElement			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

▽

△

Class	CppImplementationDataTypeElement			
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing CppImplementationDataTypeElement as optional. This means the that, at runtime, the Cpp ImplementationDataTypeElement may or may not have a valid value and shall therefore be ignored. The underlying runtime software provides means to set the CppImplementationDataTypeElement as not valid at the sending end of a communication and determine its validity at the receiving end.
typeReference	CppImplementation DataTypeElement Qualifier	0..1	aggr	This aggregation defines the type of the Cpp ImplementationDataTypeElement and determines whether in C++ the CppImplementationDataTypeElement is defined inside or outside of the enclosing Cpp ImplementationDataType.

Table 3.166: CppImplementationDataTypeElement

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

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

The existence of attributes of `CppImplementationDataType` depending on the usage of data categories eligible for `CppImplementationDataType` according to [constr_1578] is documented in [constr_10417].

[constr_10417]{DRAFT} **Existence of attributes of `CppImplementationDataType`s depending on the `category`** ⌈

Attributes of CppImplementationDataType	Attribute Existence per Category							
	VALUE	TYPE_REFERENCE	STRUCTURE	VARIANT	ARRAY	VECTOR	ASSOCIATIVE_MAP	STRING
subElement			1..*					
templateArgument				1..*	1	1..*	2..*	0..1
typeReference		1						
arraySize					1	0..1[10]		

This rule shall be imposed **at the time before the generation of the ara API starts**

⌋()

[10] This depends on the existence of an allocator, see [TPS_MANI_03186].

Please note that [constr_10417] only includes the attributes of CppImplementationDataType that depend on the usage of category. All attributes of CppImplementationDataType that are not mentioned in [constr_10417] are applicable for all values of category.

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

On the *AUTOSAR adaptive platform*, it is not allowed to reference an ImplementationDataType from within SwDataDefProps. This aspect is expressed in a general form by [constr_1579].

As a consequence of [constr_1579], type-references have to be done differently on the *AUTOSAR adaptive platform*. For this purpose dedicated references are available.

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

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

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

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

Table 3.167: CppImplementationDataTypeElementQualifier

[constr_10128]{DRAFT} **Multiplicity of attribute CppImplementationDataTypeElementQualifier.typeReference** ⌈For each CppImplementationDataTypeElementQualifier, the attribute typeReference shall exist **at the time before the generation of the ara API starts**.⌋*()*

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

Specifically, the following rules shall apply:

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

⌋*(RS_MANI_00039)*

Please note that Figure B.7 shows an example of a Structure where the `typeReference` of one `subElement` is classified as `inplace`.

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

- `ARRAY`
- `VECTOR`
- `ASSOCIATIVE_MAP`
- `VARIANT`
- `STRUCTURE`
- `STRING`
- `TYPE_REFERENCE`, if the `CppImplementationDataType` refers to a `CompuMethod` of `category` `TEXTTABLE`

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

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

- StdCppImplementationDataType of category VALUE
- CustomCppImplementationDataType
- CppImplementationDataType of category TYPE_REFERENCE, unless the CppImplementationDataType refers to a CompuMethod of category TEXTTABLE

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

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

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

Specifically, the following rules shall apply:

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

⌋*(RS_MANI_00039)*

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

- ARRAY
- VECTOR
- ASSOCIATIVE_MAP
- VARIANT
- STRUCTURE

- STRING

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋()

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

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

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

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

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

[constr_1708]{DRAFT} **Combination of CppImplementationDataTypeElement.isOptional and CppImplementationDataTypeElementQualifier.inplace** ⌈If a CppImplementationDataTypeElement is typed by a CppImplementationDataType of category STRUCTURE then the combination of attribute CppImplementationDataTypeElement.isOptional set to true and CppImplementationDataTypeElement.typeReference.inplace set to true is not allowed **at the time before the generation of the ara API starts**.⌋()

Rationale for the existence of [constr_1708]: the “optional” semantics is implemented via a template and it is not possible to pass an “inplace” structure as a template argument.

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

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

Class	**Allocator**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class represents the ability to specify an optional custom C++ allocator for a C++ type which may dynamically grow beyond it's initial allocated size during it's lifetime. Any storage principles are defined in the implementation of the allocator itself, which should implement the ISO C++ std::allocator_ traits interface. **Tags:** atp.recommendedPackage=Allocators			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
headerFile	String	0..1	attr	Configuration of the Header File with the custom class declaration
namespace (ordered)	SymbolProps	*	aggr	This aggregation allows for the definition of a namespace of an Allocator.

Table 3.168: Allocator

[constr_3446]{DRAFT} **`CppTemplateArgument` with `allocator` reference and the `inplace` flag** ⌈A `CppTemplateArgument` that points with an `allocator` reference to an `Allocator` shall not have the `inplace` flag set to a value **`at the time before the generation of the ara API starts`**.⌋*()*

ISO C++ permits the usage of an `Allocator` only for those `StdCppImplementationDataType`s which may dynamically grow in size, thus, it is necessary to restrict the other `StdCppImplementationDataType`s. This restriction ([constr_6905]) is deliberately not imposed on `CustomCppImplementationDataType`s because of their custom nature.

[constr_6905]{DRAFT} **`CppTemplateArgument.allocator`** ⌈The reference `CppTemplateArgument.allocator` shall only exist if the `StdCppImplementationDataType.category` is set to either of the values:

- `STRING`
- `VECTOR`
- `ASSOCIATIVE_MAP`

This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

3.4.3.2 Attributes of SwDataDefProps

In contrast to the situation on the *AUTOSAR classic platform* (see [constr_1009]), the usage of `SwDataDefProps` plays a significantly less prominent role in the definition of `CppImplementationDataType` on the *AUTOSAR adaptive platform*.

This aspect is clarified by [constr_1579].

[constr_1579]{DRAFT} **SwDataDefProps applicable to CppImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*** ⌈

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

This rule shall be imposed **at the time before the generation of the ara API starts**

⌋()

A consequence of [constr_1578] is that [constr_1579] 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].

The invalidValue is applicable to Primitive Data Types and defines one specific value (in the range of that Primitive Data Type) which indicates that the respective value is not valid.

A typical use case is a composite data type that contains the values of all 4 wheel speeds. If one of the wheel speed sensors fails, and is no longer able to provide useful data, it does still make sense to provide the other 3 wheel speed values.

In such a scenario the one wheel speed value would then be set to the invalidValue. The receivers are able to check for each individual element of the data composition whether the value corresponds to the invalidValue and take corresponding actions.

[constr_3569]{DRAFT} **Applicability of attribute invalidValue on CppImplementationDataType of category TYPE_REFERENCE** ⌈If a CppImplementationDataType of category TYPE_REFERENCE has an invalidValue defined, then the referenced CppImplementationDataType (via typeReference) shall eventually be of category VALUE **at the time before the generation of the ara API starts**.⌋*()*

Please note that the following rationale exists for the support of invalidValue for specific categorys of data types:

- The usage of category VALUE on the *AUTOSAR adaptive platform* boils down to the usage of the standard types. There is no use case to define an invalidValue for a standard data type because **all usages** of the standard data type would be characterized by the same invalidValue.
- The definition of an invalidValue for a container (except STRING) is not supported because there are no known use cases for supporting an invalidValue.
- The definition of an invalidValue on a data type of category STRING is accepted because it is also supported on the *AUTOSAR classic platform* and it is necessary to sustain interoperability between the *AUTOSAR classic platform* and the *AUTOSAR adaptive platform*.
- The definition of an invalidValue on an StdCppImplementationDataType of category TYPE_REFERENCE represents the main-stream use case for the definition of an invalidValue.

3.4.3.3 Primitive Data Types

[TPS_MANI_03192]{DRAFT} **CppImplementationDataType of category VALUE** ⌈The primitive data types like Boolean, fixed-width integer data types and floating-point

data types are described as CppImplementationDataTypes of category VALUE.⌋ *(RS_MANI_00039)*

[TPS_MANI_01411]{DRAFT} **Initialization of a DataPrototype typed by a StdCppImplementationDataType of category VALUE that represents a Boolean** ⌈A DataPrototype typed by a StdCppImplementationDataType of category VALUE that represents a Boolean shall be initialized by a NumericalValueSpecification that contains one of the following values:

- 0
- 1
- true
- false

⌋*(RS_MANI_00039)*

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

3.4.3.4 String Data Type

[TPS_MANI_03178]{DRAFT} **StdCppImplementationDataType of category STRING** ⌈A StdCppImplementationDataType of category STRING represents a container data type for a sequence of characters.⌋*(RS_MANI_00039)*

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

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

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

Since C++ strings are not containerized types, they do not receive a containerized typename in the template<> argument list on code level. It is therefore necessary in this case to restrict the referenced templateType in CppTemplateArgument.

[constr_6815]{DRAFT} **Existence of CppTemplateArgument.templateType for CppImplementationDataType of category STRING** ⌈In a CppImplementationDataType of category STRING, the reference templateType shall not exist **at the time before the generation of the ara API starts**.⌋*()*

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

An example of the definition of a string data type can be found in Appendix B.1.2.

3.4.3.5 Array Data Type

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

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

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

⌋*(RS_MANI_00039)*

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

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

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

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

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

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

Examples of the definition of CppImplementationDataType of category ARRAY can be found in Appendix B.1.3.

3.4.3.6 Vector Data Type

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

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

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

⌋*(RS_MANI_00039)*

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

[constr_3434]{DRAFT} **Aggregation of templateArguments for a VECTOR** ⌈A CppImplementationDataType of category VECTOR that boils down to `ara::core::Vector` shall aggregate at least one and at most two templateArgument where the first one shall define the type of elements contained in the CppImplementationDataType of category VECTOR with the templateType reference **at the time before the generation of the ara API starts**.⌋*()*

The semantics of the second templateArgument is described in [constr_10416].

[constr_10416]{DRAFT} **Aggregation of optional templateArgument that defines an Allocator for a VECTOR** ⌈If a CppImplementationDataType of category VECTOR that boils down to `ara::core::Vector` aggregates (in addition to

the templateArgument mentioned in [constr_3434]) one additional templateArgument, then this additional templateArgument shall reference the Allocator by means of the allocator reference **at the time before the generation of the ara API starts**.⌋()

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

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

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

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

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

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

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

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

Examples of the definition of CppImplementationDataType of category VECTOR can be found in Appendix B.1.4.

3.4.3.7 Struct Data Type

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

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

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

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

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

Examples of the definition of `CppImplementationDataType` of `category STRUCTURE` can be found in Appendix B.1.5.

3.4.3.8 Enumeration Data Type

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

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

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

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

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

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

An example of the definition of an enumeration can be found in Appendix B.1.6.

3.4.3.9 Map Data Type

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

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

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

⌋*(RS_MANI_00039)*

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

- one CppTemplateArgument shall have the category ASSOC_MAP_KEY and shall reference a CppImplementationDataType with the templateType reference. This CppTemplateArgument represents the role that corresponds to ApplicationAssocMapDataType.key and defines the respective data type details.
- one CppTemplateArgument shall shall have the category ASSOC_MAP_VALUE and shall reference a CppImplementationDataType with the templateType reference. This CppTemplateArgument represents the role that corresponds to ApplicationAssocMapDataType.value and defines the respective data type details.
- one additional optional CppTemplateArgument is allowed to reference an Allocator with the allocator reference.

⌋*(RS_MANI_00039)*

Examples of the definition of CppImplementationDataType of category ASSOCIATIVE_MAP can be found in Appendix B.1.7.

3.4.3.10 Variant Data Type

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

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

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

⌋*(RS_MANI_00039)*

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

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

[TPS_MANI_01393]{DRAFT} **Initialization of a data object typed by a CppImplementationDataType of category VARIANT** ⌈The initialization of a data object that is typed by a CppImplementationDataType of category VARIANT is done by using a RecordValueSpecification with two fields:

- The first field identifies the index of the element inside the variant data type. The index of the first element shall be 1 and growing from there.
- The second field identifies the "payload" of the initialization and consists of a ValueSpecification that matches the data type of the element in the variant data type that is supposed to be initialized.

⌋*(RS_MANI_00039)*

Please note the convention to let the index be 1-based is in place to achieve the same initialization semantics as on the *AUTOSAR classic platform*.

This means that – in principle – the same RecordValueSpecification can be used to initialize a data object typed by a variant data type (*AUTOSAR adaptive platform*) and a corresponding data object typed by a Wrapped Union Data Type on the *AUTOSAR classic platform*.

The conversion of the index value to a numerical value used in an expression of the language binding of the variant data type is out of scope of the TPS Manifest Specification.

In other words, if the actual initialization uses 0-based index value then the value provided in the model has to be decremented by 1 accordingly.

[constr_10399]{DRAFT} **Allowed interval of the "index" `field` according to the initialization rule for data object typed by a `CppImplementationDataType` of `category VARIANT`** ⌈The allowed value range of the "index" `field` of a `RecordValueSpecification` according to [TPS_MANI_01393] goes from 1 to the number of `templateArgument`s owned by the `CppImplementationDataType` of `category VARIANT`.

This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

Examples of the definition of `CppImplementationDataType` of `category VARIANT` can be found in Appendix B.1.8.

3.4.3.11 Bitfield Data Type

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

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

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

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

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

3.4.4 Compatibility of ApplicationDataType and CppImplementationDataType

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

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

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

Several rules depend on the category of the data types:

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

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

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

   This rule shall be imposed **at the time before the generation of the ara API starts**⌋*()*

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

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

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

3. **[constr_5034]**{DRAFT} **Compatibility of data types with category BOOLEAN** ⌈An ApplicationDataType of category BOOLEAN can only be mapped to a CppImplementationDataType of category VALUE **at the time before the generation of the ara API starts**.⌋*()*

4. **[constr_5035]**{DRAFT} **Compatibility of data types with `category STRING`** ⌈A `CppImplementationDataType` where attribute `category` is set to the value `STRING` can only be mapped to an `ApplicationDataType`

   - where attribute `category` is set to the value `STRING` **and**
   - where attribute `swDataDefProps.swTextProps.baseType.baseTypeDefinition.baseTypeEncoding` is set to the value `UTF-8`.

   This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

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

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

   This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

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

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

   - a `CppImplementationDataType` of `category VECTOR`

   This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

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

   - a `CppImplementationDataType` of `category ARRAY`

   This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

8. **[constr_5039]**{DRAFT} **Compatibility of data types with `category STRUCTURE`** ⌈An `ApplicationDataType` of `category STRUCTURE` can only be mapped to a `CppImplementationDataType` of `category STRUCTURE` **`at the time before the generation of the ara API starts`**.⌋*()*

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

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

   This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

10. **[constr_5041]**{DRAFT} **Compatibility of data types with `category ASSOCIATIVE_MAP`** ⌈An `ApplicationDataType` of `category ASSOCIATIVE_MAP` can only be mapped to a `CppImplementationDataType` of `category ASSOCIATIVE_MAP` **`at the time before the generation of the ara API starts`**.⌋*()*

11. **[constr_5042]**{DRAFT} **No data type mapping for `CppImplementationDataType` of `category VARIANT`** ⌈An `ApplicationDataType` shall never be mapped to a `CppImplementationDataType` of `category VARIANT` **`at the time before the generation of the ara API starts`**.⌋*()*

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

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

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

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

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

[constr_5044]{DRAFT} **`DataTypeMap` for composite data types** ⌈In the context of a given `ServiceInterface`, all pairs of `ApplicationDataType` and `CppImplementationDataType` used in the context of the definition of an `ApplicationCompositeDataType` used in the context of an `event`, `field`, `method` shall be described in a `DataTypeMap` that is contained in one of the `DataTypeMappingSet`s

that are referenced in a PortInterfaceToDataTypeMapping that also references the mentioned ServiceInterface.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋()

3.4.5 Optional Members in complex Data Structures

3.4.5.1 Background

The *AUTOSAR adaptive platform* supports the usage of a TLV[11] data encoding on the SOME/IP transport layer. TLV is typically used where at least a part of the transmitted data is only *optionally* existing and filled with meaningful values.

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

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

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

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

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

AutosarDataType In this case the optionality that is primarily only needed for communication purposes would still be existing in all other usages of data types. AUTOSAR still sees use cases for implementing this option, especially in the context of the *AUTOSAR classic platform*.

Admittedly, the definition of different optionality configurations for the same data type may lead to the existence of a bunch of structurally identical data types that only vary in terms of optionality. The existence of variation points may help to mitigate this effect, though.

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

This would lead to an increased effort for the definition of C++ data types in the context of the same ServiceInterface. Additional constraints have been identified in the context of the *AUTOSAR classic platform* that finally render this option as not viable.

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

ComSpec In this case the definition of optionality would even be more specific in comparison to the definition of optionality on the level of ServiceInterfaces.

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

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

3.4.5.2 Definition of Optionality

As mentioned before, the concrete definition of optionality on the level of an AutosarDataType is done by the indication of individual elements of the composite AutosarDataType.

More specifically, the definition of optionality needs to be supported for subclasses of AutosarDataType, namely on the level of ApplicationDataType as well as on the level of CppImplementationDataType.

In other words, if ApplicationDataTypes with optional elements are used to define a ServiceInterface then it is still necessary to convey the optionality down to the level of data type definition that directly affects the language binding of the AUTOSAR model.

Figure 3.78 shows the modeling of optionality on the level of ApplicationDataType.

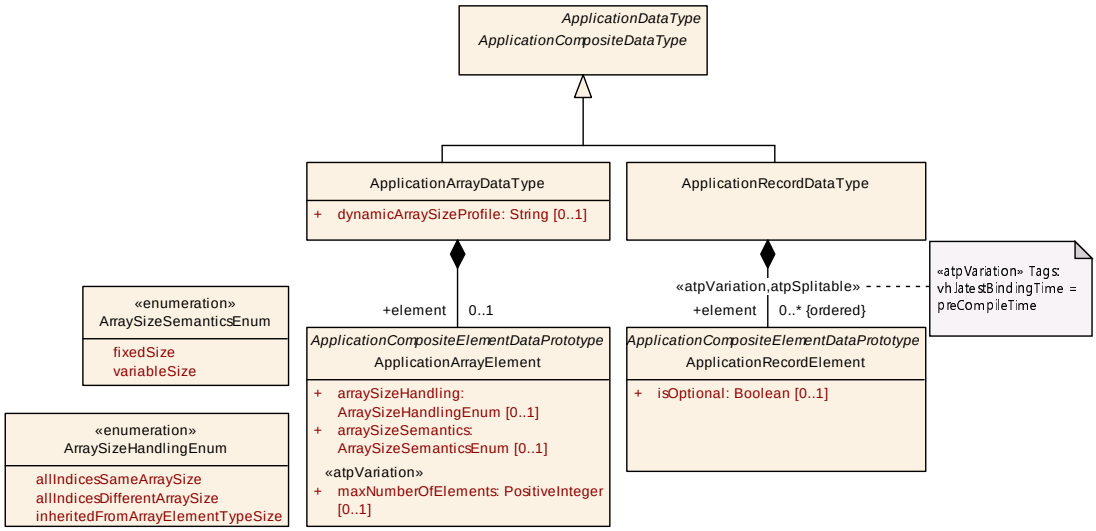


Figure 3.78: Modeling of optionality on the level of ApplicationDataType

[TPS_MANI_01184]{DRAFT} **Definition of optional elements on the level of ApplicationDataType** ⌈The modeling approach for the definition of optional elements on the level of ApplicationDataType is to set the attribute ApplicationRecordElement.isOptional to the value true.

If the attribute is not set or set to the value `false` then the respective `ApplicationRecordElement` **shall be considered mandatory**.⌋*(RS_MANI_00030)*

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*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
element (ordered)	ApplicationRecordElement	*	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 ApplicationrecordData Type. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=element.shortName, element.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table 3.169: ApplicationRecordDataType

Class	**ApplicationRecordElement**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Describes the properties of one particular element of an application record data type.			
Base	*ARObject*, *ApplicationCompositeElementDataPrototype*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationRecordDataType.element, *AtpClassifier*.atpFeature			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing ApplicationRecordElement as optional. This means the that, at runtime, the ApplicationRecord Element may or may not have a valid value and shall therefore be ignored. The underlying runtime software provides means to set the ApplicationRecordElement as not valid at the sending end of a communication and determine its validity at the receiving end.

Table 3.170: ApplicationRecordElement

On top of that, it is still possible to use `CppImplementationDataType` directly for the definition of a `ServiceInterface`.

[TPS_MANI_01185]{DRAFT} **Definition of optional elements on the level of `CppImplementationDataType`** ⌈The modeling approach for the definition of optional elements on the level of `CppImplementationDataType` is to set the attribute `CppImplementationDataTypeElement.isOptional` to the value `true`.

If the attribute is not set or set to the value `false` then the respective `CppImplementationDataTypeElement` **shall be considered mandatory**.⌋*(RS_MANI_00030)*

The attribute NotAvailableValueSpecification.defaultPattern has no meaning for the initialization of DataPrototypes on the *AUTOSAR adaptive platform*. This aspect is covered by [TPS_MANI_01333]:

[TPS_MANI_01333]{DRAFT} **Attribute NotAvailableValueSpecification.defaultPattern is not applicable** ⌈The attribute NotAvailableValueSpecification.defaultPattern (if defined) shall be ignored by the adaptive platform.

The rationale for ignoring the defaultPattern is that the optional data is technically not accessible from application code in case it has not been received.⌋ *(RS_MANI_00030)*

3.5 Serialization Of Data

In Adaptive AUTOSAR, the serialization code is generated out of the service description and is compiled and executed in the application context.

The meta-class TransformationPropsToServiceInterfaceElementMapping defines the serialization for a ServiceInterface element and provides the necessary serialization settings with the TransformationProps element.

The existence of a TransformationPropsToServiceInterfaceElementMapping demands the existence of serialization code that is linked with the application component object file to an application binary.

The serialization of SOME/IP is based on the ServiceInterface specification. If an AutosarDataPrototype that is used within a ServiceInterface is composite like a structure, union or array then SOME/IP supports the configuration of length fields that will be put in front of the serialized data.

AUTOSAR supports the configuration of such serialization settings on two different levels:

- Modeling on ServiceInterface element level that is valid for all available occurrences of a DataPrototype in the ServiceInterface element. This case is described in detail in chapter 3.5.1.
- Fine granular modeling on the level of DataPrototypes described in this chapter. This case is described in detail in chapter 3.5.2.

3.5.1 Default Values for Serialization Properties

[TPS_MANI_03101]{DRAFT} **SOME/IP serialization** ⌈The ApSomeipTransformationProps meta-class that is referenced by the TransformationPropsToServiceInterfaceElementMapping in the role transformationProps provides the ability to define a SOME/IP serialization settings for ServiceInterface elements

that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, trigger, methodCall, methodReturn, or field.⌋(*RS_MANI_00008, RS_MANI_00025*)

[constr_3395]{DRAFT} **TransformationPropsToServiceInterfaceElementMapping is restricted to one single ServiceInterface** ⌈All ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, trigger, methodCall, methodReturn, or field shall be aggregated by the same ServiceInterface in the role event, trigger, method or field.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋()

[TPS_MANI_03103]{DRAFT} **Default size for all array and map length fields** ⌈The attribute sizeOfArrayLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available variable size arrays (vectors), fixed size arrays and associative_maps defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, methodCall, methodReturn, or field.⌋(*RS_MANI_00008, RS_MANI_00025*)

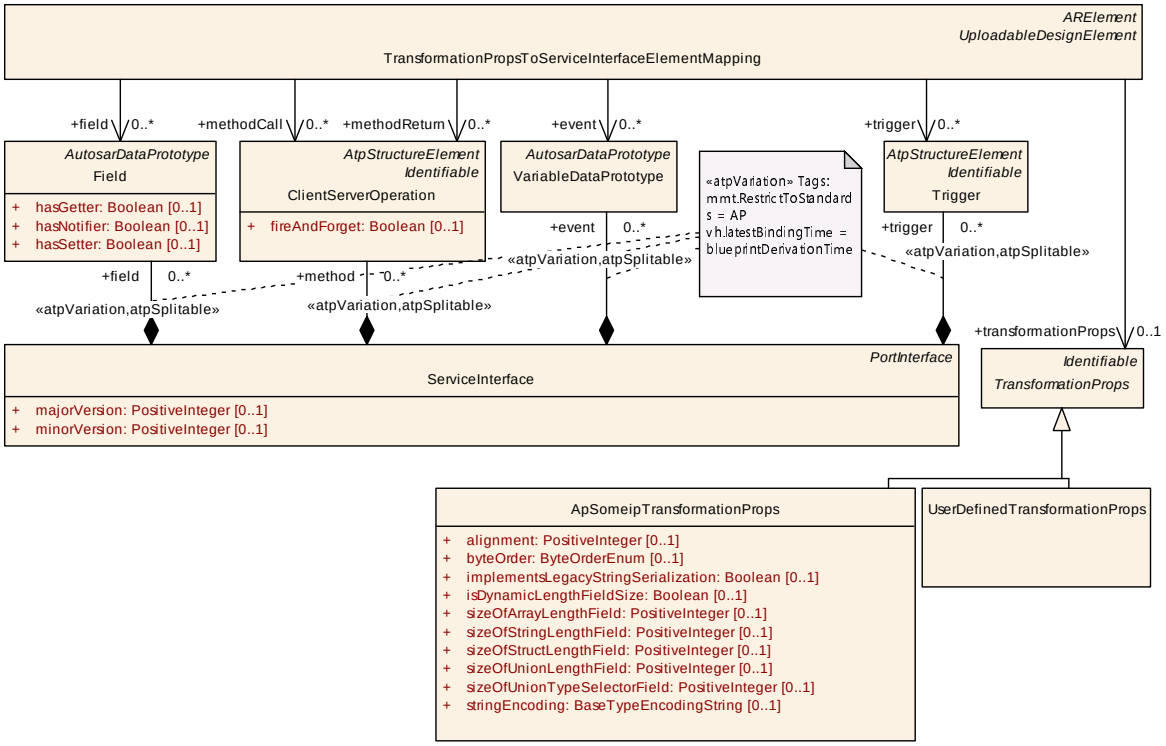


Figure 3.79: Association of serialization properties with a ServiceInterface

[TPS_MANI_03104]{DRAFT} **Default size for all structure length fields** ⌈The attribute sizeOfStructLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available structures defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, methodCall, methodReturn, or field.⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03117]{DRAFT} **Default size for all string length fields** ⌈The attribute sizeOfStringLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available strings defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, methodCall, methodReturn, or field.⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03105]{DRAFT} **Default size for all union length fields** ⌈The attribute sizeOfUnionLengthField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a length field generated by SOME/IP in front of all available unions defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, methodCall, methodReturn, or field.⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03106]{DRAFT} **Default size for all union type selector fields** ⌈The attribute sizeOfUnionTypeSelectorField of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the size of a type field generated by SOME/IP in front of all available unions defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, methodCall, methodReturn, or field.⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03107]{DRAFT} **Default alignment for all dynamic DataPrototypes** ⌈The attribute alignment of ApSomeipTransformationProps referenced by TransformationPropsToServiceInterfaceElementMapping in the role transformationProps defines the padding for alignment purposes that will be added by SOME/IP after the serialized data of all variable data length data elements defined in ServiceInterface elements that are referenced by the TransformationPropsToServiceInterfaceElementMapping in the role event, methodCall, methodReturn, or field.⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_03108]{DRAFT} **Default Byte Order for all `DataPrototype`s** ⌈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`, `methodCall`, `methodReturn`, or `field`.⌋*(RS_MANI_00008, RS_MANI_00025)*

[TPS_MANI_01210]{DRAFT} **Default encoding for all `DataPrototype`s typed by `CppImplementationDataType` of `category STRING`** ⌈The attribute `stringEncoding` of a `ApSomeipTransformationProps` referenced by `TransformationPropsToServiceInterfaceElementMapping` in the role `transformationProps` defines the string encoding in the serialized data stream resulting from `ServiceInterface` elements that are referenced by the `TransformationPropsToServiceInterfaceElementMapping` in the role `event`, `methodCall`, `methodReturn`, or `field`.⌋*(RS_MANI_00008, RS_MANI_00025)*

[constr_1675]{DRAFT} **Existence of attribute `ApSomeipTransformationProps.stringEncoding`** ⌈The attribute `TransformationPropsToServiceInterfaceElementMapping.transformationProps.stringEncoding` shall only exist for a `event`, `methodCall`, `methodReturn`, or `field` (referenced by the same `TransformationPropsToServiceInterfaceElementMapping`) that consists of or contains a `DataPrototype` typed by a `CppImplementationDataType` of `category STRING`.

This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

Please note that more details about `ApSomeipTransformationProps` can be found in chapter 3.5.2.

[constr_1678]{DRAFT} **Allowed values for attribute `ApSomeipTransformationProps.stringEncoding`** ⌈Imposed by technical restrictions in the definition of the SOME/IP message format [9], only two possible values of attribute `ApSomeipTransformationProps.stringEncoding` are allowed:

- `UTF-8`: UCS Transformation Format 8
- `UTF-16`: Character encoding for Unicode *code points* based on 16 bit *code units* [17]

This rule shall be imposed **`at the time before the generation of the ara API starts`**.⌋*()*

Class	**ApSomeipTransformationProps**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::SerializationProperties
Note	SOME/IP serialization properties.
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TransformationProps*

▽

△

Class	**ApSomeipTransformationProps**			
Aggregated by	TransformationPropsSet.transformationProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
alignment	PositiveInteger	0..1	attr	Defines the padding for alignment purposes that will be added by the SOME/IP transformer after the serialized data of the variable data length data element. The alignment shall be specified in Bits.
byteOrder	ByteOrderEnum	0..1	attr	Specifies the byte order of data in the serialized data stream.
implements LegacyString Serialization	Boolean	0..1	attr	This attribute indicates that Strings in the SOME/IP message shall NOT be serialized according to the SOME/IP specification for Strings. If this attribute is set to true, BOM and null-termination shall NOT be added in the serialization for Strings in the payload. If this attribute is set to false (or not set) BOM and null-termination shall be added in the serialization for Strings in the payload according to the SOME/IP specification for Strings. NOTE! This attribute is not future safe, and will be removed in an upcoming AUTOSAR release! **Tags:** atp.Status=obsolete
isDynamic LengthFieldSize	Boolean	0..1	attr	This attribute represents the ability to control the setting of the wire type for TLV encoding. If the attribute is set to true then wire type 5-7 shall be used. If the attribute does not exist or is set to false then wire type 4 shall be used.
sizeOfArray LengthField	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of a variable size Array (Vector), fixed-size Array or an Associative_Map. It describes the size of the length field (in Bytes) that will be put in front of the Array or Associative_Map in the SOME/IP message.
sizeOfString LengthField	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.
sizeOfStruct LengthField	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.
sizeOfUnion LengthField	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.
sizeOfUnion TypeSelector Field	PositiveInteger	0..1	attr	Configures the SOME/IP serialization for the referenced dataPrototype in case of a Union. It describes the size of the type selector field (in Bytes) that will be put in front of the Union in the SOME/IP message.
stringEncoding	BaseTypeEncoding String	0..1	attr	Configures the encoding for SOME/IP serialization for the referenced dataPrototype in case of an String.

Table 3.171: ApSomeipTransformationProps

Enumeration	**ByteOrderEnum**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	When more than one byte is stored in the memory the order of those bytes may differ depending on the architecture of the processing unit. If the least significant byte is stored at the lowest address, this architecture is called little endian and otherwise it is called big endian. ByteOrder is very important in case of communication between different PUs or ECUs.
Aggregated by	ApSomeipTransformationProps.byteOrder, BaseTypeDirectDefinition.byteOrder, DiagnosticCommonProps.defaultEndianness, ISignalToIPduMapping.packingByteOrder, MultiplexedIPdu.selectorFieldByteOrder, PduToFrameMapping.packingByteOrder, SegmentPosition.segmentByteOrder, SOMEIPTransformationDescription.byteOrder, System.containerIPduHeaderByteOrder
Literal	*Description*
mostSignificantByteFirst	Most significant byte shall come at the lowest address (also known as BigEndian or as Motorola-Format) **Tags:** atp.EnumerationLiteralIndex=0
mostSignificantByteLast	Most significant byte shall come highest address (also known as LittleEndian or as Intel-Format) **Tags:** atp.EnumerationLiteralIndex=1
opaque	For opaque data endianness conversion has to be configured to Opaque. See AUTOSAR COM Specification for more details. **Tags:** atp.EnumerationLiteralIndex=2

Table 3.172: ByteOrderEnum

Primitive	**BaseTypeEncodingString**
Package	M2::MSR::AsamHdo::BaseTypes
Note	This is the string denotion of a BaseType encoding. It may be refined by specific use-cases. **Tags:** xml.xsd.customType=BASE-TYPE-ENCODING-STRING xml.xsd.type=string

Table 3.173: BaseTypeEncodingString

[TPS_MANI_03102]{DRAFT} **UserDefined serialization** ⌈The `UserDefinedTransformationProps` meta-class that is referenced by the `TransformationPropsToServiceInterfaceElementMapping` in the role `transformationProps` provides the ability to define a User defined serialization for `ServiceInterface` elements that are referenced by the `TransformationPropsToServiceInterfaceElementMapping` in the role `event`, `methodCall`, `methodReturn`, or `field`.⌋*(RS_MANI_00014, RS_MANI_00025)*

Please note that `UserDefinedTransformationProps` is derived from meta-class `Identifiable` and therefore has the ability to describe special data (`sdg`) by which it is possible to define custom structural extensions of an AUTOSAR model in a generic way. For more information about special data please refer to [6].

Class	**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.recommendedPackage=TransformationPropsToServiceInterfaceElementMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
event	VariableDataPrototype	*	ref	This represents the reference to one or several events of one ServiceInterface.
field	Field	*	ref	This represents the reference to one or several fields of one ServiceInterface.
methodCall	ClientServerOperation	*	ref	This represents the reference to one or several method calls of one ServiceInterface.
methodReturn	ClientServerOperation	*	ref	This represents the reference to one or several method return of one ServiceInterface.
tlvDataId Definition	TlvDataIdDefinitionSet	*	ref	This reference identifies the TlvDataIdDefinitions relevant for the enclosing TransformationPropsToServiceInterface Mapping.
transformation Props	TransformationProps	0..1	ref	This represents the reference to the applicable Serialization properties.
trigger	Trigger	*	ref	This represents the reference to one or several triggers of one ServiceInterface.

Table 3.174: TransformationPropsToServiceInterfaceElementMapping

Class	**UserDefinedTransformationProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	The class UserDefinedTransformationProps specifies specific configuration properties of a user defined serializer.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TransformationProps*			
Aggregated by	TransformationPropsSet.transformationProps			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 3.175: UserDefinedTransformationProps

3.5.2 Individual Definition of Serialization Properties

[TPS_MANI_03109]{DRAFT} **`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 `TransformationPropsToServiceInterfaceElementMapping`.⌋*(RS_MANI_00025)*

[constr_3361]{DRAFT} **Selective definition of serialization settings** ⌈If a `SomeipDataPrototypeTransformationProps` is defined for a composite `DataPrototype` of an element of a `ServiceInterface` (`method`, `field`, `event`) and if the

reference someipTransformationProps exists then SomeipDataPrototype-TransformationProps that define the reference someipTransformationProps shall be defined for all other composite DataPrototypes of the ServiceInterface element as well.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋()

Consider the following example to illustrate [constr_3361]: Let's assume five composite data types (e.g., ApplicationCompositeDataTypes) DT_0, DT_1, DT_2, DT_3, and DT_4 consisting of DataPrototypes (el_0 to el_10) which in turn are also typed by composite data types, see Figure 3.80.

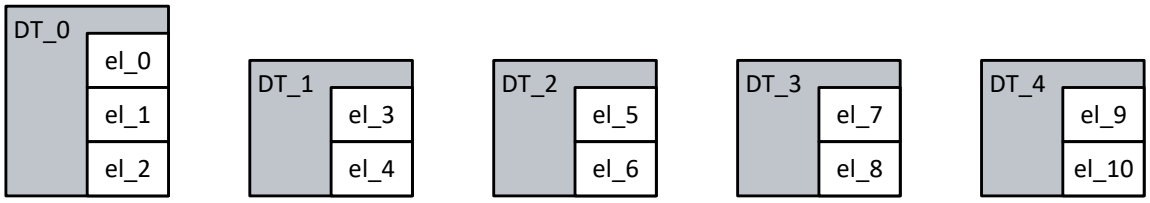


Figure 3.80: Definition of data types used in the example

Further consider a ServiceInterface SI_0 consisting of two events EV_0 and EV_ typed by DT_0 and DT_1 respectively.

Additionally, SI_0 provides two methods M_0 and M_1 where M_0 has two arguments ARG_0 and ARG_1 typed by DT_2 and DT_3 and M_1 has one argument ARG_2 typed by DT_4. This yields the following hierarchy for SI_0, see Figure 3.81.

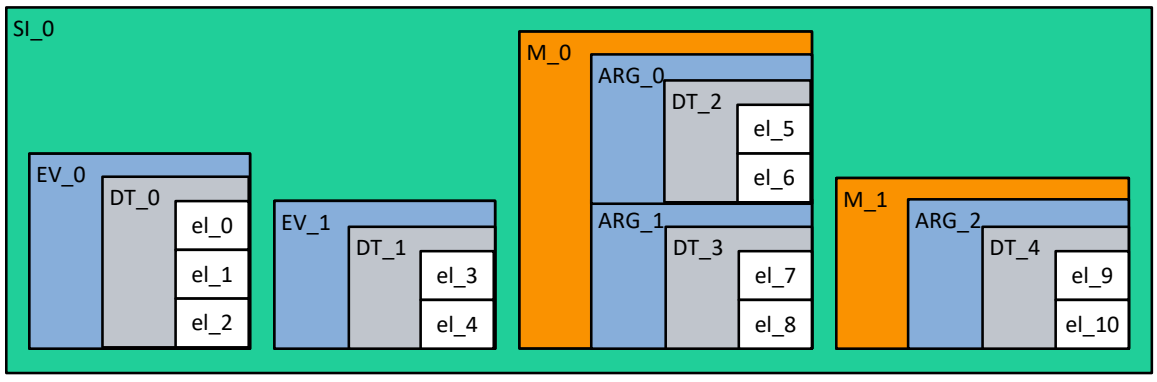


Figure 3.81: Definition of service interface used in the example

In this setup [constr_3361] would mandate that if a SomeipDataPrototypeTrans-formationProps were defined for el_2, then SomeipDataPrototypeTransfor-mationProps have to be defined for el_0 and el_1 as well since el_0 and el_1 are composite DataPrototypes of the same ServiceInterface element EV_0 as el_2.

Conversely, [constr_3361], would not mandate that if a SomeipDataPrototype-TransformationProps were defined for el_2, then SomeipDataPrototype-TransformationProps have to be defined for el_3 and el_4 as well since el_3 and el_4 are composite DataPrototypes of a different ServiceInterface element (EV_1 instead of EV_0) than el_2.

Similar for fields (since the setter and getter methods of a field are not explicitly modeled as ClientServerOperation (with their input and output argument) and thus there is only the Field itself (which derives from AutosarDataPrototype) to reason about.

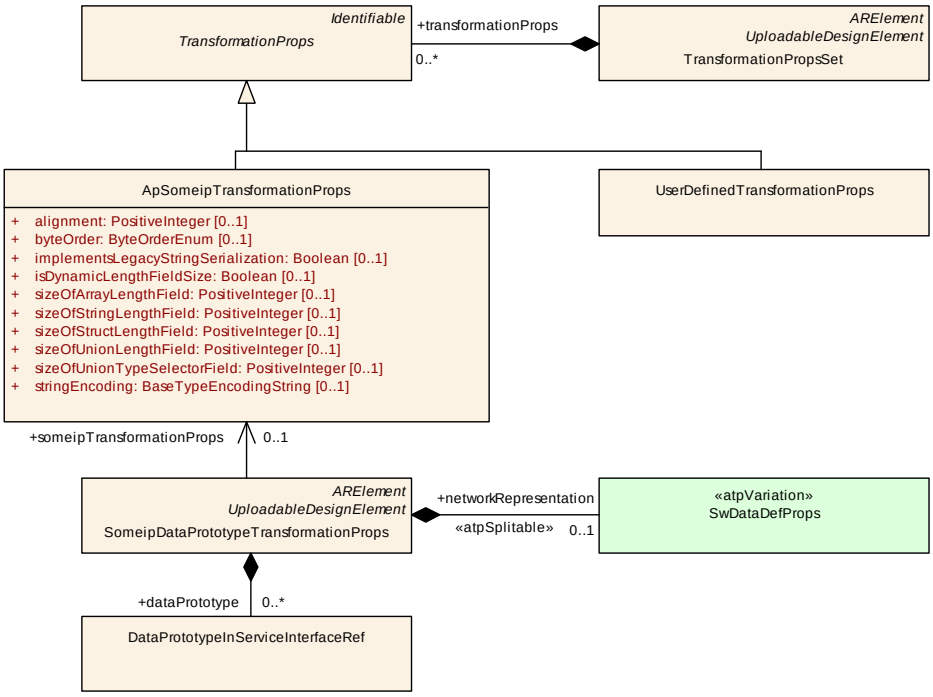


Figure 3.82: Overview about SOME/IP Serialization Properties

As far as methods are concerned, [constr_3361] would mandate the following: If in the above example a SomeipDataPrototypeTransformationProps were defined for el_5, then SomeipDataPrototypeTransformationProps have to be defined for el_6, el_7, and el_8 as well since el_6, el_7, and el_8 are composite DataPrototypes of the same ServiceInterface element M_0 as el_5.

Conversely, [constr_3361] would not mandate that if a SomeipDataPrototype-TransformationProps were defined for el_5, then SomeipDataPrototype-TransformationProps have to be defined for el_9 and el_10 as well since el_9 and el_10 are composite DataPrototypes of a different ServiceInterface element M_1 (instead of M_0) than el_5.

[TPS_MANI_03070]{DRAFT} **Size of a length field for a chosen array or map** ⌈The attribute sizeOfArrayLengthField of ApSomeipTransformationProps defines the size of a length field generated by SOME/IP in front of a variable size array (vector), fixed size array or associative_map for which the SomeipDataPrototypeTrans-formationProps is defined, i.e. the variable size array (vector), fixed size array or

associative_map that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.⌋(*RS_MANI_00008, RS_MANI_00024*)

Class	**TransformationPropsSet**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	Collection of TransformationProps. **Tags:** atp.recommendedPackage=TransformationPropsSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
transformation Props	TransformationProps	*	aggr	Transformer specific configuration properties.

Table 3.176: TransformationPropsSet

[constr_3353]{DRAFT} **Restriction in usage of ApSomeipTransformationProps.sizeOfArrayLengthField** ⌈The value of the attribute sizeOfArrayLengthField shall be either 0, 1, 2 or 4 **at the time before the generation of the ara API starts**.⌋()

[constr_3447]{DRAFT} **ApSomeipTransformationProps.sizeOfArrayLengthField that equals 0** ⌈The sizeOfArrayLengthField value of 0 is only allowed to be used if a fixed size array for which the SomeipDataPrototypeTransformationProps is defined is referenced within the aggregated DataPrototypeInServiceInterfaceRef **at the time before the generation of the ara API starts**.⌋()

The setting of sizeOfArrayLengthField for fixed size arrays supports a backward compatible extension of such arrays with additional array elements.

[TPS_MANI_03071]{DRAFT} **Size of a length field for a chosen structure** ⌈The attribute sizeOfStructLengthField of ApSomeipTransformationProps defines the size of a length field generated by SOME/IP in front of a structure for which the SomeipDataPrototypeTransformationProps is defined, i.e. the structure that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.⌋(*RS_MANI_00008, RS_MANI_00024*)

[constr_3354]{DRAFT} **Restriction in usage of ApSomeipTransformationProps.sizeOfStructLengthField** ⌈The value of the attribute sizeOfStructLengthField shall be either 0, 1, 2 or 4 **at the time before the generation of the ara API starts**.⌋()

[TPS_MANI_03116]{DRAFT} **Size of a length field for a chosen string** ⌈The attribute sizeOfStringLengthField of ApSomeipTransformationProps defines the size of a length field generated by SOME/IP in front of a String for which the SomeipDataPrototypeTransformationProps is defined, i.e. the String that is referenced within the aggregated DataPrototypeInServiceInterfaceRef.⌋(*RS_MANI_00008, RS_MANI_00024*)

[constr_3372]{DRAFT} **Restriction in usage of `ApSomeipTransformationProps.sizeOfStringLengthField`** ⌈The value of the attribute `sizeOfStringLengthField` shall be either 0, 1, 2 or 4 **at the time before the generation of the ara API starts**.⌋*()*

[TPS_MANI_03217]{DRAFT} **On-the-wire encoding for a chosen string** ⌈The attribute `stringEncoding` of `ApSomeipTransformationProps` defines the on-the-wire encoding of a String for which the `SomeipDataPrototypeTransformationProps` is defined, i.e. the String that is referenced within the aggregated `DataPrototypeInServiceInterfaceRef`.⌋*(RS_MANI_00008, RS_MANI_00024)*

[TPS_MANI_03072]{DRAFT} **Size of a length field for a chosen union** ⌈The attribute `sizeOfUnionLengthField` of `ApSomeipTransformationProps` defines the size of a length field generated by SOME/IP in front of a union for which the `SomeipDataPrototypeTransformationProps` is defined, i.e. the union that is referenced within the aggregated `DataPrototypeInServiceInterfaceRef`.⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3355]{DRAFT} **Restriction in usage of `ApSomeipTransformationProps.sizeOfUnionLengthField`** ⌈The value of the attribute `sizeOfUnionLengthField` shall be either 0, 1, 2 or 4 **at the time before the generation of the ara API starts**.⌋*()*

[TPS_MANI_03073]{DRAFT} **Alignment of a dynamic DataPrototype** ⌈The attribute `alignment` of `ApSomeipTransformationProps` defines the padding for alignment purposes that will be added by SOME/IP after the serialized data of the variable data length data element for which the `SomeipDataPrototypeTransformationProps` is defined, i.e. the variable data length DataPrototype that is referenced within the aggregated `DataPrototypeInServiceInterfaceRef`.⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3356]{DRAFT} **Restriction in usage of `ApSomeipTransformationProps.alignment`** ⌈The value of the attribute `alignment` shall be either 8, 16, 32, 64, 128, or 256 **at the time before the generation of the ara API starts**.⌋*()*

[TPS_MANI_03074]{DRAFT} **Size of a type selector field for a chosen union** ⌈The attribute `sizeOfUnionTypeSelectorField` of `ApSomeipTransformationProps` defines the size of a type selector field generated by SOME/IP in front of a union for which the `SomeipDataPrototypeTransformationProps` is defined, i.e. the union that is referenced within the aggregated `DataPrototypeInServiceInterfaceRef`.⌋*(RS_MANI_00008, RS_MANI_00024)*

[constr_3357]{DRAFT} **Restriction in usage of `ApSomeipTransformationProps.sizeOfUnionTypeSelectorField`** ⌈The value of the attribute `sizeOfUnionTypeSelectorField` shall be either 1, 2 or 4 **at the time before the generation of the ara API starts**.⌋*()*

[TPS_MANI_03278]{DRAFT} **Usage of `ApSomeipTransformationProps.byteOrder`** ⌈The `byteOrder` attribute defined in an `ApSomeipTransformationProps`

that is referenced by SomeipDataPrototypeTransformationProps shall be ignored for the dataPrototypes for which the SomeipDataPrototypeTransformationProps apply.⌋(*RS_MANI_00008*, *RS_MANI_00024*)

The byteOrder attribute defines the byte order of the complete payload in the SOME/IP message and therefore the configuration via TransformationPropsToServiceInterfaceElementMapping is the only valid option to define the byte order for a ServiceInterface element. Please note that according to SOME/IP, the header is encoded in network byte order (Big Endian).

Class	**SomeipDataPrototypeTransformationProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::SerializationProperties			
Note	This meta-class represents the ability to define data transformation props specifically for a SOME/IP serialization for a given DataPrototype. **Tags:** atp.recommendedPackage=SomeipDataPrototypeTransformationPropss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataPrototype	DataPrototypeInServiceInterfaceRef	*	aggr	Collection of DataPrototypes for which the settings in SomeipDataPrototypeTransformationProps are valid. For reuse reasons the SomeipDataPrototypeTransformationProps is able to aggregate several DataPrototypes.
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 AbstractImplementationDataType shall be used for the serialization/deserialization. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=networkRepresentation
someipTransformationProps	ApSomeipTransformationProps	0..1	ref	This reference represents the ability to define data transformation props specifically for a SOME/IP serialization.

Table 3.177: SomeipDataPrototypeTransformationProps

The modeling of the reference to a DataPrototype in the context of a PortInterface that is typed by an ApplicationDataType or by a CppImplementationDataType is depicted in Figure 11.17.

[TPS_MANI_01136]{DRAFT} **AutosarDataPrototype is the target of the DataPrototypeInServiceInterfaceRef** ⌈If the target of an DataPrototypeInServiceInterfaceRef is an AutosarDataPrototype the role DataPrototypeInServiceInterfaceRef.dataPrototype shall be used to describe the reference **independently** of whether the AutosarDataPrototype is typed by an ApplicationDataType or a CppImplementationDataType and even **independently** of whether the AutosarDataType of the AutosarDataPrototype represents a composite data type.⌋(*RS_MANI_00008*, *RS_MANI_00024*)

[TPS_MANI_01137]{DRAFT} **Possible use cases for the usage of DataPrototypeInServiceInterfaceRef** ⌈

Use case	Role
AutosarDataPrototype typed by an ApplicationDataType	dataPrototype
DataPrototype in AutosarDataPrototype typed by an ApplicationCompositeDataType	dataPrototype
AutosarDataPrototype typed by a CppImplementationDataType	dataPrototype
DataPrototype in AutosarDataPrototype typed by a CppImplementationDataType	elementInImplDatatype

Table 3.178: Possible use cases for the usage of DataPrototypeInServiceInterfaceRef

⌋*(RS_MANI_00008, RS_MANI_00024)*

From a careful observation of [TPS_MANI_01137], it should be clear that there is no valid use case to simultaneously use the two roles dataPrototype and elementInImplDatatype in the context of the same DataPrototypeInServiceInterfaceRef.

[constr_1551]{DRAFT} **Existence of DataPrototypeInServiceInterfaceRef.dataPrototype vs. DataPrototypeInServiceInterfaceRef.elementInImplDatatype** ⌈For every given DataPrototypeInServiceInterfaceRef, either the aggregation DataPrototypeInServiceInterfaceRef.dataPrototype or DataPrototypeInServiceInterfaceRef.elementInImplDatatype shall exist **at the time before the generation of the ara API starts**.⌋*()*

The usage of the SomeipDataPrototypeTransformationProps.networkRepresentation is explained in more detail in the System Template [18] in [TPS_SYST_02136] and [TPS_SYST_02137].

3.5.3 Assignment of TLV properties

3.5.3.1 Assignment of TLV Data IDs

[TPS_MANI_01097]{DRAFT} **Assignment of TLV data ids** ⌈The assignment of TLV data ids is done in the context of the specification of TransformationPropsToServiceInterfaceElementMapping, namely by means of the attribute TransformationPropsToServiceInterfaceElementMapping.tlvDataIdDefinition.id.⌋*(RS_MANI_00030)*

This approach takes benefit from the fact that the TlvDataIdDefinition is able to create references to relevant model elements.

The assignment of the TLV data id is therefore done by creating such a reference and assigning a TLV data id to it by means of the attribute TlvDataIdDefinition.id.

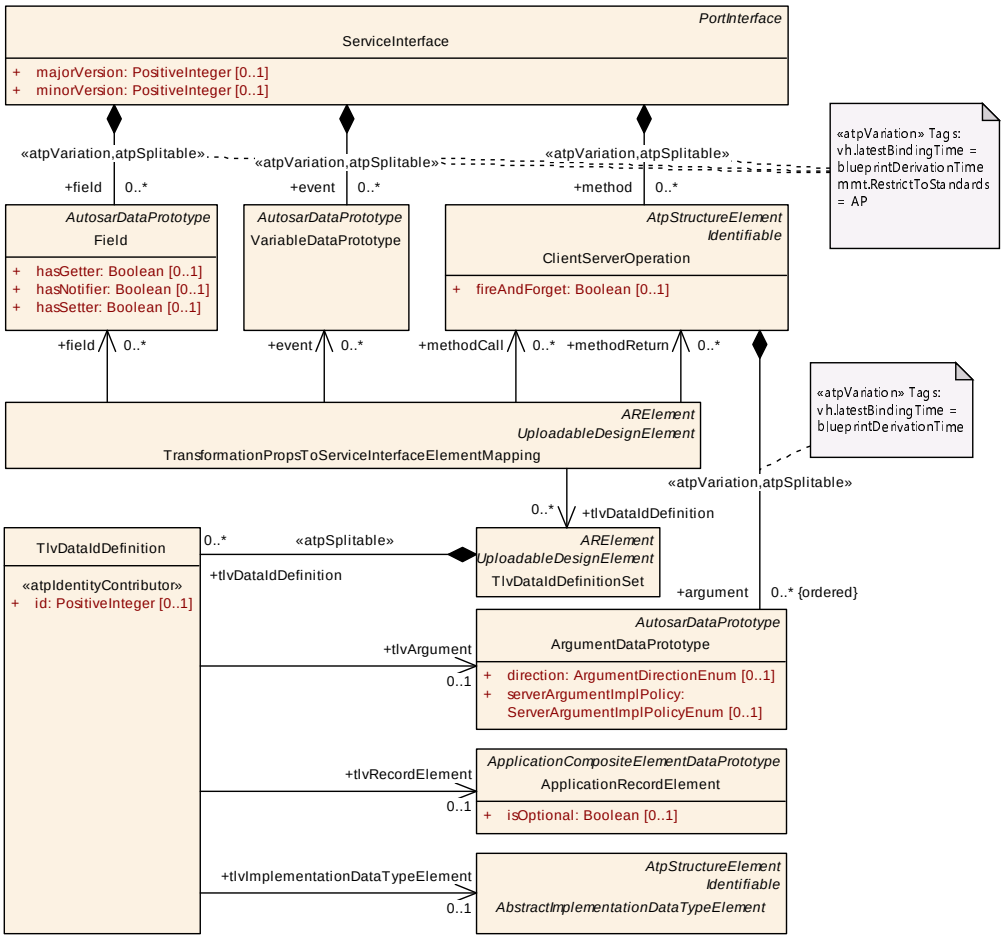


Figure 3.83: Modeling of the TLV data id

Please note that the assignment of TLV data ids is compulsory for an entire data structure that has at least one optional member. In a nutshell, this conclusion (that is also backed by [PRS_SOMEIP_00230], see [9]) is the motivation for the existence of [constr_1594], and [constr_1595].

Please note further that the assignment of TLV data ids is not restricted to data structures with optional members. There is also a use case to support sending the elements of a specific data structure in arbitrary order even if none of the elements is considered optional.

[TPS_MANI_01270]{DRAFT} **Reference from `TransformationPropsToServiceInterfaceElementMapping` to `TlvDataIdDefinitionSet`** ⌈The reference from `TransformationPropsToServiceInterfaceElementMapping` to `TlvDataIdDefinitionSet` means that it is in the hand of the creator of a model to decide whether a global scope should be assumed or whether the definition needs to be customized for a specific case.⌋ *(RS_MANI_00030)*

[constr_1594]{DRAFT} **Consistent assignment of TLV data ids to `ApplicationRecordDataType`** ⌈For every `ApplicationRecordDataType` where direct members set the attribute `ApplicationRecordElement.isOptional` to the value `true`

references to **all direct members** of this ApplicationRecordDataType shall be created on the basis of the definition of TlvDataIdDefinition.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋()

Class	**TlvDataIdDefinitionSet**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	This meta-class acts as a container of TlvDataIdDefinitions to be used in a given context **Tags:** atp.recommendedPackage=TlvDataDefinitionSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
tlvDataId Definition	TlvDataIdDefinition	*	aggr	This aggregation represents the collection of TlVDataTid Definitions aggregated by the TlvDataIdDefinitionSet **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=tlvDataIdDefinition.id

Table 3.179: TlvDataIdDefinitionSet

[constr_1595]{DRAFT} **Consistent assignment of TLV data ids to CppImplementationDataType or CppImplementationDataTypeElement** ⌈For every CppImplementationDataType of category STRUCTURE where direct members set the attribute CppImplementationDataTypeElement.isOptional to the value true references to **all direct members** of this CppImplementationDataType shall be created on the basis of the definition of TlvDataIdDefinition.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋()

The definition of a TlvDataIdDefinition that refers to an eligible model element is not limited to scenarios where optional elements are defined. It is also possible to define TlvDataIdDefinition for arbitrary methods or data structures.

A typical use case could be to prepare the argument list or sub-elements for future extensions. However, if one argument or sub-element is referenced then it is necessary to define references from TlvDataIdDefinitions to all other arguments or sub-elements as well.

[constr_1593]{DRAFT} **Completeness of the existence of a set of TlvDataIdDefinition.tlvArguments** ⌈If the reference TlvDataIdDefinition.tlvArgument exists for one argument of a given ClientServerOperation then further TlvDataIdDefinition.tlvArgument shall exist **for all** arguments of the given ClientServerOperation and all affected TlvDataIdDefinition shall be aggregated by the same TransformationPropsToServiceInterfaceElementMapping.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋()

Although it would be possible to apply an optimization in the definition of the TLV configuration such that the TLV configuration could be defined direction-specific[12], AUTOSAR defines that such a mixed TLV configuration shall not be used.

[constr_1603]{DRAFT} **Completeness of the existence of a set of TlvDataIdDefinition.tlvRecordElements** ⌈If the reference TlvDataIdDefinition.tlvRecordElement exists for one element of a given ApplicationRecordDataType then further TlvDataIdDefinition.tlvRecordElement shall exist **for all** elements of the given ApplicationRecordDataType and all affected TlvDataIdDefinition shall be aggregated by the same TransformationPropsToServiceInterfaceElementMapping.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋()

[constr_1604]{DRAFT} **Completeness of the existence of a set of TlvDataIdDefinition.tlvImplementationDataTypeElements** ⌈If the reference TlvDataIdDefinition.tlvImplementationDataTypeElement exists for one subElement of a given CppImplementationDataType or CppImplementationDataTypeElement then further TlvDataIdDefinition.tlvImplementationDataTypeElement shall exist **for all** subElements of the given CppImplementationDataType or CppImplementationDataTypeElement and all affected TlvDataIdDefinition shall be aggregated by the same TransformationPropsToServiceInterfaceElementMapping.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋()

Class	**TlvDataIdDefinition**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	This meta-class represents the ability to define the tlvDataId.			
Base	*ARObject*			
Aggregated by	TlvDataIdDefinitionSet.tlvDataIdDefinition			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
id	PositiveInteger	0..1	attr	This attribute represents the definition of the value of the TlvDataId **Stereotypes:** atpIdentityContributor
tlvArgument	ArgumentDataPrototype	0..1	ref	This reference assigns a tlvDataId to a given argument of a ClientServerOperation.
tlv Implementation DataType Element	AbstractImplementation DataTypeElement	0..1	ref	This reference associates the definition of a TLV data id with a given AbstractImplementationDataTypeElement.
tlvRecord Element	ApplicationRecord Element	0..1	ref	This reference associates the definition of a TLV data id with a given ApplicationRecordElement.

Table 3.180: TlvDataIdDefinition

[12]For example, the usage of TLV encoding for arguments with attribute direction set to in, but **not** for arguments with attribute direction set to out.

The definition of a TlvDataIdDefinition.id has the purpose to provide means to unambiguously identify the argument or sub-element. For this purpose, the value of the id needs to be unique in the respective context.

[constr_1596]{DRAFT} **Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to ArgumentDataPrototype** ⌈For all TlvDataIdDefinition that are referencing ArgumentDataPrototypes of a given ClientServerOperation in the role tlvArgument, the attribute TlvDataIdDefinition.id **shall exist and have a unique value per communication direction**, i.e. in the context of the collection of all

- arguments where attribute direction is set to either in or inout
- arguments where attribute direction is set to either out or inout
- arguments where attribute direction is set to inout (if the method **only** has arguments where attribute direction is set to inout)

of the respective enclosing ClientServerOperation.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

Rationale for the existence of [constr_1596]: arguments where attribute direction is set to either in or inout are never sent in the same SOME/IP message as arguments where attribute direction is set to either out or inout.

[constr_1597]{DRAFT} **Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to ApplicationRecordElement** ⌈For all TlvDataIdDefinition that are referencing ApplicationRecordElements of a given ApplicationDataType in the role tlvRecordElement the attribute TlvDataIdDefinition.id **shall exist and have a unique value** in the context of respective enclosing ApplicationRecordDataType.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

[constr_1598]{DRAFT} **Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to CppImplementationDataTypeElement** ⌈For all TlvDataIdDefinition that are referencing CppImplementationDataTypeElements of a given CppImplementationDataType/CppImplementationDataTypeElement in the role tlvImplementationDataTypeElement the attribute TlvDataIdDefinition.id **shall exist and have a unique value** in the context of respective enclosing CppImplementationDataType or CppImplementationDataTypeElement.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋*()*

Obviously, it is necessary to avoid ambiguity with respect to the definition of TLV data ids. Each model element that can be assigned such an id shall only be assigned one id.

[constr_1599]{DRAFT} **TlvDataIdDefinition referencing ArgumentDataPrototype** ⌈Each ArgumentDataPrototype shall be referenced **at most once** in the role tlvArgument in the context of the same TransformationPropsToServiceInterfaceElementMapping **at the time before the generation of the ara API starts**.⌋()

[constr_1600]{DRAFT} **TlvDataIdDefinition referencing ApplicationRecordElement** ⌈Each ApplicationRecordElement shall be referenced **at most once** in the role tlvRecordElement in the context of the same TransformationPropsToServiceInterfaceElementMapping **at the time before the generation of the ara API starts**.⌋()

[constr_1601]{DRAFT} **TlvDataIdDefinition referencing CppImplementationDataTypeElement** ⌈Each CppImplementationDataTypeElement shall be referenced **at most once** in the role tlvImplementationDataTypeElement in the context of the same TransformationPropsToServiceInterfaceElementMapping **at the time before the generation of the ara API starts**.⌋()

[constr_1748]{DRAFT} **Existence of references TlvDataIdDefinition.tlvArgument, TlvDataIdDefinition.tlvRecordElement, and TlvDataIdDefinition.tlvImplementationDataTypeElement** ⌈For each TlvDataIdDefinition, only one out of the following references shall exist:

- reference to an ArgumentDataPrototype in the role tlvArgument
- reference to an ApplicationRecordElement in the role tlvRecordElement
- reference to an AbstractImplementationDataTypeElement in the role tlvImplementationDataTypeElement.

This rule shall be imposed **at the time before the generation of the ara API starts**.⌋()

Class	***AbstractImplementationDataTypeElement*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	This meta-class represents the ability to act as an abstract base class for specific derived meta-classes that support the modeling of ImplementationDataTypes for a particular language binding.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	CppImplementationDataTypeElement, ImplementationDataTypeElement			
Aggregated by	*AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.181: AbstractImplementationDataTypeElement

[constr_1628]{DRAFT} **Definition of static length field sizes in case of TLV usage** ⌈If the aggregation tlvDataIdDefinition exists for a given TransformationPropsToServiceInterfaceElementMapping then attributes

- sizeOfArrayLengthField,

- sizeOfStringLengthField,
- sizeOfStructLengthField, and
- sizeOfUnionLengthField

shall have a value greater than 0 **at the time before the generation of the ara API starts**.⌋()

Rationale for the existence of [constr_1628]: The TLV serialization requires the usage of length fields:

- If wire type 4 is used (for more details, please refer to [TPS_MANI_01186]) then the length field size shall be statically configured.
- If wire types 5-7 are used (see [TPS_MANI_01186]) then the static configuration of the length field size shall also be present since not all length fields are preceded by a tag, e.g. structures contained in an array or the top-level structure contained in a SOME/IP event.

Without demanding the existence of length fields in such a case the result of a serialization could be ambiguous, i.e. make it impossible for the de-serializer to figure out the data layout[13].

[constr_1629]{DRAFT} **Identical sizes of length fields in case of TLV usage** ⌈If the aggregation tlvDataIdDefinition exists for a given TransformationPropsToServiceInterfaceElementMapping then attributes

- sizeOfArrayLengthField,
- sizeOfStringLengthField,
- sizeOfStructLengthField, and
- sizeOfUnionLengthField

shall have an identical value **at the time before the generation of the ara API starts**.⌋()

Rationale for the existence of [constr_1629]: if wire type 4 is used (for more details, please refer to [TPS_MANI_01186]) and if the receiver encounters a member of a structure or an argument with an unknown tag the de-serializer cannot determine the actual data type of the member of the structure or argument.

[constr_1630]{DRAFT} **No definition of length field sizes on DataPrototype level in case of TLV usage** ⌈If the reference in the role tlvDataIdDefinition exists for a given TransformationPropsToServiceInterfaceElementMapping then attributes

- sizeOfArrayLengthField,
- sizeOfStringLengthField,

[13] If a structure consists only of optional elements, it would be hard to detect the case where an array element carries such a structure that happens to set all elements to non-available.

- sizeOfStructLengthField, and
- sizeOfUnionLengthField

shall not be individually defined on the level of a DataPrototype (i.e. by means of the reference SomeipDataPrototypeTransformationProps.someipTransformationProps) but only on the level of a ServiceInterface (i.e. by means of the reference TransformationPropsToServiceInterfaceElementMapping.transformationProps).

This rule shall be imposed **at the time before the generation of the ara API starts**⌋*()*

Rationale for the existence of [constr_1630]: if wire type 4 is used (for more details, please refer to [TPS_MANI_01186]) and if the receiver encounters a member or argument with an unknown tag the de-serializer needs to know the size of the length field.

The most reliable way to achieve this is to demand the definition of the size of the length field on the level of the ServiceInterface.

3.5.3.2 Assignment of Wire Type Selection

The TLV encoding supports the definition of a so-called wire type that controls how the information about the length of length fields shall be interpreted.

The meaning of specific settings of the wire type is defined in [9, PRS SOME/IP Protocol].

[TPS_MANI_01186]{DRAFT} **Definition of the applicable wire type** ⌈Attribute ApSomeipTransformationProps.isDynamicLengthFieldSize shall be used to define the applicable wire type.

If the value of attribute ApSomeipTransformationProps.isDynamicLengthFieldSize is set to true then wire type 5-7 shall be used.

If the value of attribute ApSomeipTransformationProps.isDynamicLengthFieldSize does not exist or is set to false then wire type 4 shall be used.⌋ *(RS_MANI_00030)*

3.6 Identity and Access Management

The definition of intents (for example: ClientComSpec.clientIntent) as described in chapter 3.2.3.5.4 is used to express the intention of the software designer to use (or refrain from using) specific APIs in the application software.

The definition of intents represents one aspect of the identity and Access Management (IAM). Another aspect of the IAM configuration is the definition of the actual permissions granted by the platform software.

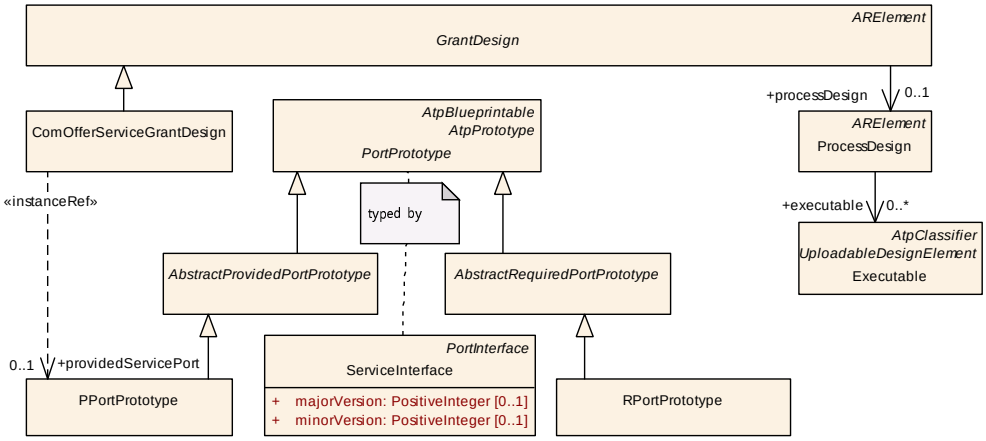


Figure 3.84: Modeling of grant designs for service discovery

The modeling of such grants is done on two levels:

- the definition of `GrantDesign` allows for the pre-specification of grants already on the design level. The modeling of `GrantDesign` is described in this chapter.
- the definition of `Grant` allows for the actual and final specification of grants from the perspective of the platform software. The modeling of Grant is described in chapter 10.9.

Class	***GrantDesign*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign			
Note	This meta-class serves as an abstract base class for the description of grants on design level. **Tags:** atp.Status=candidate			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	*ComGrantDesign*, ComOfferServiceGrantDesign, RawDataStreamGrantDesign			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
processDesign	ProcessDesign	0..1	ref	This reference identifies the corresponding Process Design that gives context to the GrantDesing. **Tags:** atp.Status=candidate

Table 3.182: GrantDesign

Abstract meta-class `GrantDesign` acts as the base class for the definition of grants on the design level.

Grants are specific for a given `Process`. In other words, two `Process`es created from the same `Executable` may be assigned different sets of grants. This specific relation shall also be available on the design level.

[TPS_MANI_01231]{DRAFT} **GrantDesign references ProcessDesign** ⌈Meta-class GrantDesign references ProcessDesign as a means to design the set of Grants for the given Process.⌋*(RS_MANI_00060)*

3.6.1 Com Grant Design

Subclasses of GrantDesign are created to cover specific aspects of grants for communication on the *AUTOSAR adaptive Platform*.

[TPS_MANI_01232]{DRAFT} **Semantics of meta-class ComOfferServiceGrantDesign** ⌈The existence of a ComOfferServiceGrantDesign that references a specific AbstractProvidedPortPrototype in the role providedServicePort indicates that the design foresees that the referenced AbstractProvidedPortPrototype shall be granted rights to offer the respective service.⌋*(RS_MANI_00060)*

Please note that there is no explicitly modeled intent that corresponds to the existence of the ComOfferServiceGrantDesign. The understanding is that the mere existence of an AbstractProvidedPortPrototype typed by a ServiceInterface indicates the intent to offer a service.

The understanding is that the mere existence of an AbstractRequiredPortPrototype typed by a ServiceInterface indicates the intent to find a service.

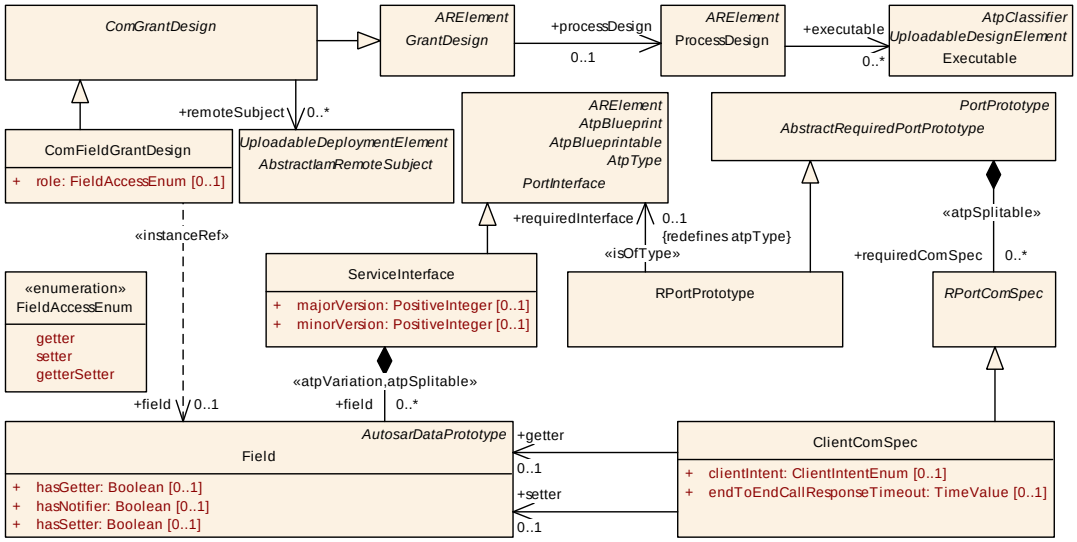


Figure 3.85: Modeling of grant designs for field

Class	ComOfferServiceGrantDesign			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class represents the ability to define a Grant for offering a service. **Tags:** atp.Status=candidate atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
providedServicePort	PPortPrototype	0..1	iref	This instanceRef identifies the PPortPrototype on which the service shall be offered. **Tags:** atp.Status=candidate **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef

Table 3.183: ComOfferServiceGrantDesign

[TPS_MANI_01234]{DRAFT} **Semantics of `ComFieldGrantDesign`** ⌈The existence of a `ComFieldGrantDesign` that references a specific `Field` in the role `field` indicates that the design foresees that the application software shall be granted rights to access the respective `Field`. The nature of the access, i.e. get vs. set is specified by means of the attribute role.⌋ *(RS_MANI_00060)*

[constr_10155]{DRAFT} **Multiplicity of reference in the role `ComOfferServiceGrantDesign.providedServicePort`** ⌈For each `ComOfferServiceGrantDesign`, the reference in the role `providedServicePort` shall exist **`at the time when the GrantDesign is complete`**.⌋ *()*

Class	*ComGrantDesign* (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class serves as an abstract base class for the description of com grants on design level. **Tags:** atp.Status=candidate			
Base	*ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	ComEventGrantDesign, ComFieldGrantDesign, ComMethodGrantDesign, ComTriggerGrantDesign			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
remoteSubject	AbstractIamRemote Subject	*	ref	This optional reference defines the remoteSubject that is allowed to access the defined Object via the Grant. **Tags:** atp.Status=candidate

Table 3.184: ComGrantDesign

[constr_3720]{DRAFT} **Upper multiplicity of reference in the role `ComGrantDesign.remoteSubject`** ⌈In the context of `ComGrantDesign`, the reference in the role `remoteSubject` shall exist at most once **`at the time when the GrantDesign is complete`**.⌋ *()*

Class	**ComFieldGrantDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class represents the ability to define a Grant for a ServiceInterface.field. **Tags:** atp.Status=candidate atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrantDesign*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
field	Field	0..1	iref	Reference to the affected Field in the context of an Executable. **Tags:** atp.Status=candidate **InstanceRef implemented by:** FieldInExecutable InstanceRef
role	FieldAccessEnum	0..1	attr	This attribute provides the ability to further specify the access to the ServiceInterface.field from a design perspective. **Tags:** atp.Status=candidate

Table 3.185: ComFieldGrantDesign

[constr_10157]{DRAFT} **Multiplicity of reference in the role `ComFieldGrantDesign.field`** ⌈For each `ComFieldGrantDesign`, the reference in the role `field` shall exist **`at the time when the GrantDesign is complete`**.⌋*()*

[constr_10158]{DRAFT} **Multiplicity of reference in the role `ComFieldGrantDesign.role`** ⌈For each `ComFieldGrantDesign`, the reference in the role `role` shall exist **`at the time when the GrantDesign is complete`**.⌋*()*

Enumeration	**FieldAccessEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant
Note	This meta-class provides values that qualify access to a field.
Aggregated by	ComFieldGrant.role, ComFieldGrantDesign.role
Literal	***Description***
getter	Access to the getter of the Field. **Tags:** atp.EnumerationLiteralIndex=0
getterSetter	Access to getter and setter of the field **Tags:** atp.EnumerationLiteralIndex=2
setter	Access to the setter of the Field. **Tags:** atp.EnumerationLiteralIndex=1

Table 3.186: FieldAccessEnum

[TPS_MANI_01235]{DRAFT} **Semantics of `ComEventGrantDesign`** ⌈The existence of a `ComEventGrantDesign` that references a specific `VariableDataPrototype` that is aggregated in the role `event` by the enclosing `ServiceInterface` indicates that the design foresees that the application software shall be granted rights to access the respective `event`.⌋*(RS_MANI_00060)*

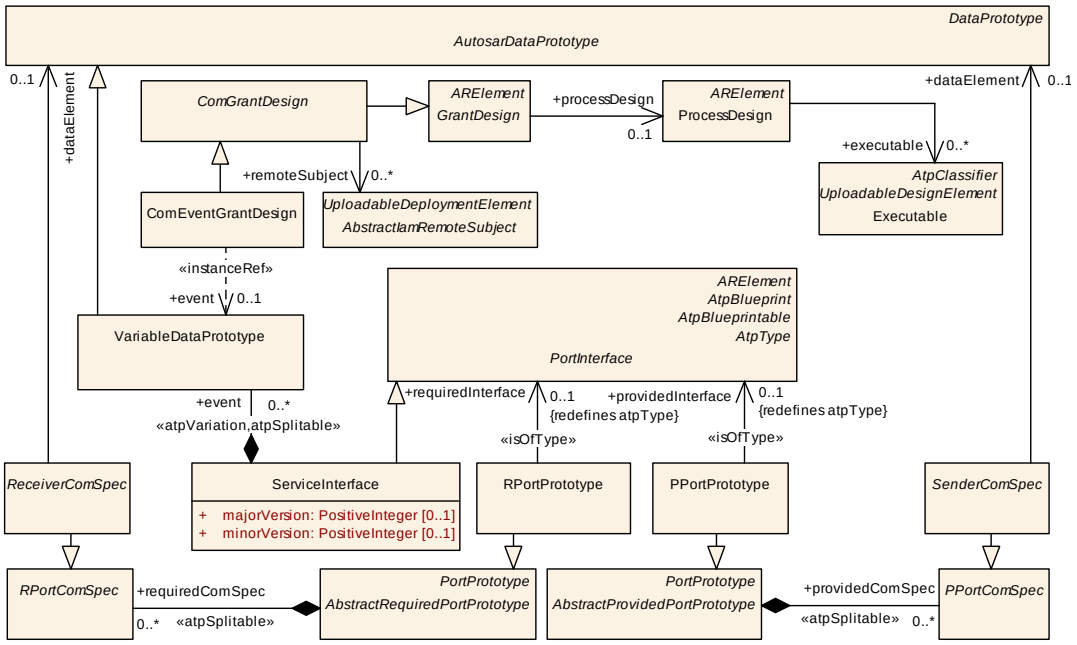


Figure 3.86: Modeling of grant designs for `event`

[TPS_MANI_03290]{DRAFT} **Semantics of `ComTriggerGrantDesign`** ⌈The existence of a `ComTriggerGrantDesign` that references a specific `Trigger` that is aggregated in the role `trigger` by the enclosing `ServiceInterface` indicates that the design foresees that the application software shall be granted rights to access the respective `trigger`.⌋ *(RS_MANI_00060)*

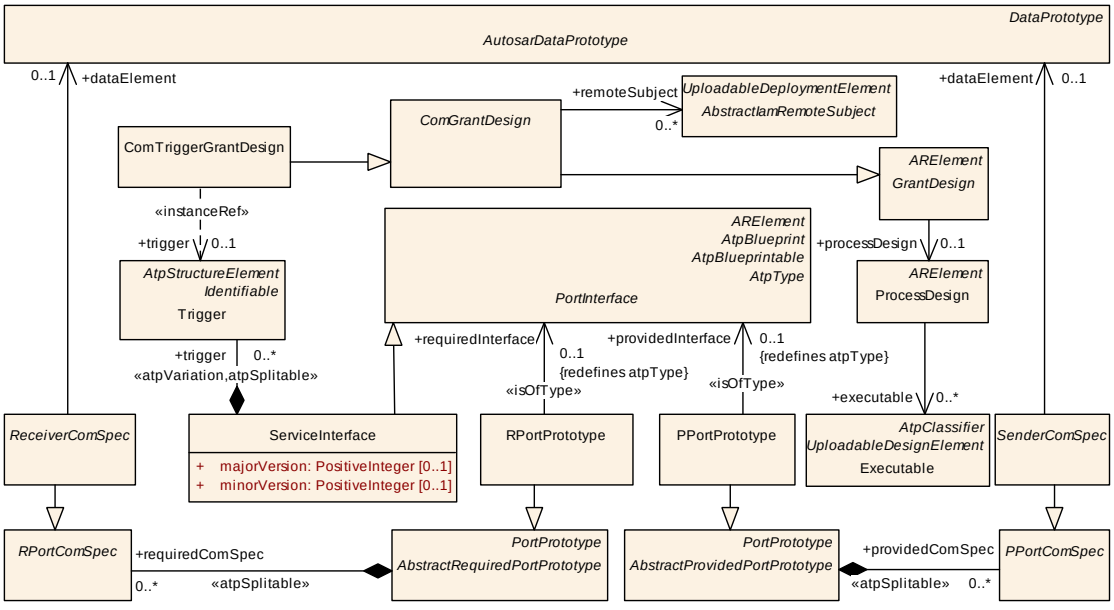


Figure 3.87: Modeling of grant designs for `trigger`

Class	ComEventGrantDesign			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class represents the ability to define a Grant for a ServiceInterface.event. **Tags:** atp.Status=candidate atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrantDesign*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
event	VariableDataPrototype	0..1	iref	This reference represents the affected event. **Tags:** atp.Status=candidate **InstanceRef implemented by:** EventInExecutable InstanceRef

Table 3.187: ComEventGrantDesign

[constr_10159]{DRAFT} **Multiplicity of reference in the role `ComEventGrantDesign.event`** ⌈For each `ComEventGrantDesign`, the reference in the role `event` shall exist **`at the time when the GrantDesign is complete`**.⌋*()*

Class	ComTriggerGrantDesign			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class represents the ability to define a Grant for a ServiceInterface.trigger. **Tags:** atp.Status=candidate atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrantDesign*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
trigger	Trigger	0..1	iref	This reference represents the affected trigger. **Tags:** atp.Status=candidate **InstanceRef implemented by:** TriggerInExecutable InstanceRef

Table 3.188: ComTriggerGrantDesign

[constr_10160]{DRAFT} **Multiplicity of reference in the role `ComTriggerGrantDesign.trigger`** ⌈For each `ComTriggerGrantDesign`, the reference in the role `trigger` shall exist **`at the time when the GrantDesign is complete`**.⌋*()*

[TPS_MANI_01236]{DRAFT} **Semantics of `ComMethodGrantDesign`** ⌈The existence of a `ComMethodGrantDesign` that references a specific `ClientServerOperation` that is aggregated in the role `method` by the enclosing `ServiceInterface` indicates that the design foresees that the application software shall be granted rights to call the respective `method`.⌋*(RS_MANI_00060)*

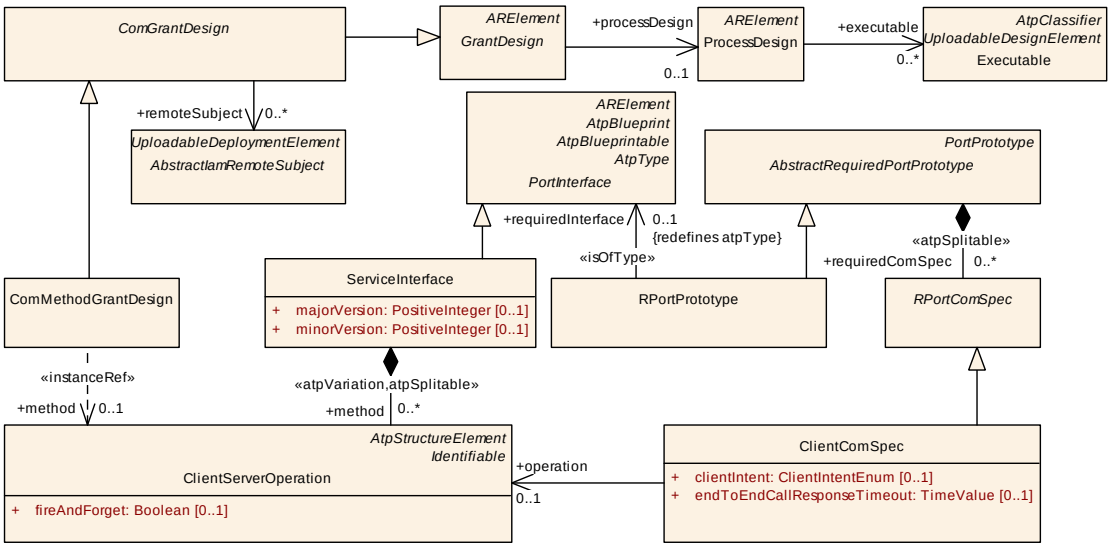


Figure 3.88: Modeling of grant designs for `method`

Class	**ComMethodGrantDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::ComGrant			
Note	This meta-class represents the ability to define a Grant for a ServiceInterface.method. **Tags:** atp.Status=candidate atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrantDesign*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
method	ClientServerOperation	0..1	iref	This reference identifies the corresponding method. **Tags:** atp.Status=candidate **InstanceRef implemented by:** RequiredMethodInExecutableInstanceRef

Table 3.189: ComMethodGrantDesign

[constr_10161]{DRAFT} **Multiplicity of reference in the role `ComMethodGrantDesign.method`** ⌈For each `ComMethodGrantDesign`, the reference in the role `method` shall exist **`at the time when the GrantDesign is complete`**.⌋*()*

3.6.2 Grant Design for Raw Streaming Data

The usage of a raw data stream is subject to restrictions imposed by the IAM. Therefore, meta-class `RawDataStreamGrantDesign` exists to support this use case.

[TPS_MANI_01284]{DRAFT} **Granularity of meta-class `RawDataStreamGrantDesign`** ⌈The granularity of the `RawDataStreamGrantDesign` is the entire `AbstractRawDataStreamInterface`. It is not expected that a definition of an IAM policy makes sense on a smaller level, i.e. on the level of `ClientServerOperation` aggregated by a `AbstractRawDataStreamInterface`.⌋*(RS_MANI_00060)*

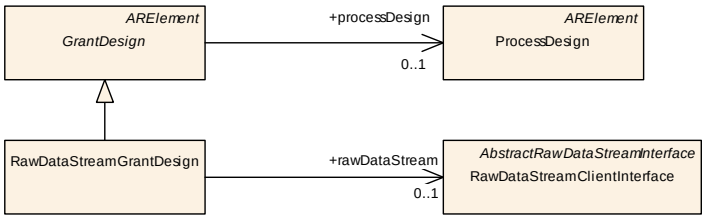


Figure 3.89: Modeling of the `RawDataStreamGrantDesign`

Class	**RawDataStreamGrantDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::GrantDesign::RawDataStreamGrant			
Note	This meta-class represents the ability to define the IAM configuration for a RawDataStream on design level. **Tags:** atp.Status=candidate atp.recommendedPackage=GrantDesigns			
Base	*ARElement*, *ARObject*, *CollectableElement*, *GrantDesign*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
rawDataStream	RawDataStreamClient Interface	0..1	ref	This reference identifies the applicable RawDataStream Interface. **Tags:** atp.Status=candidate

Table 3.190: RawDataStreamGrantDesign

3.6.3 Remote access control

The definition of the deployment for the *Identity and Access Manager* and the definition of grants relies on the local enforcement of identity and access permissions.

In other words it is possible for example to define that a particular `method` of a `ServiceInterface` is allowed to be called by a local `Process` on the local `Machine`.

But it is not possible to restrict the remote `Machine`s that are allowed to call this `method`.

The fact that the `Machine` on which the service is running has no mean to make additional checks on the incoming requests enables processing of wrongly issued requests by a healthy remote `Machine` as well as escalation of privileges by an attacker via issuing arbitrary request towards services from a compromised remote `Machine`.

Most of the times it is not possible for a `Machine` to recognize that its communicating peer is compromised because the attacker has access to all the resources of that `Machine` and can run in stealth mode.

An effective way to minimize the damage of a compromised remote `Machine` is to enforce additional checks on the incoming requests at the receiver side ensuring that remote `Machine` cannot go beyond what they could request in a healthy state.

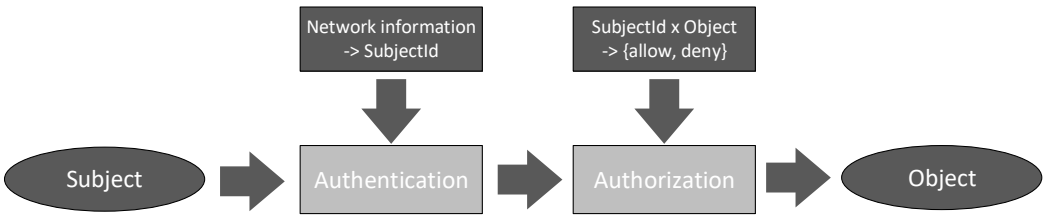


Figure 3.90: Access policy enforcement based on the Subject ID from the network binding

The access control process aims at enforcing policies on the relation between a "Subject" and an "Object". In the example where an remote `Machine` makes calls to a service interface, the remote `Machine` is the Subject and the `method` of a `ServiceInterface` is the Object.

The access control process comprises of the two main operations, namely, Authentication and Authorization, which are mostly independent.

During the authentication process the identity of the subject is verified and an authentic identifier is resolved. Authentication is an essential part of the chain to ensure that different subjects cannot impersonate each other.

In the next step, during Authorization, the identity of the Subject is checked upon the rules and policies defined for the accessing the Object to verify if the Subject's request is legitimate. These policies shall be defined by the system or the resource owner.

The authentication of the remote subject is based on the network binding. When a secure channel is established, the remote peer has already gone through an authentication protocol.

Therefore, the identity information can be forwarded to the IAM to apply the corresponding defined policies that are defined for the requests coming from that channel as depicted in Figure 3.90.

The remote subject is modeled as a specialization of `AbstractIamRemoteSubject`. The different specializations will be presented in the following sections.

Class	***AbstractIamRemoteSubject*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM			
Note	This abstract meta-class defines the proxy information about the remote node. **Tags:** atp.Status=candidate			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	IPSecIamRemoteSubject, IpIamRemoteSubject, TlsIamRemoteSubject			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 3.191: AbstractIamRemoteSubject

With the modeling of ComGrantDesigns the permissions that are granted by the platform software are defined. As an option a ComGrantDesign is able to reference the AbstractIamRemoteSubject in the role remoteSubject.

[TPS_MANI_03238]{DRAFT} **Definition of ComMethodGrantDesign.remoteSubject** ⌈If the ComMethodGrantDesign references one or several AbstractIamRemoteSubjects in the role remoteSubject then the design foresees that only the defined remoteSubjects shall be granted rights to access the ClientServerOperation that is referenced in the role method by the same ComMethodGrantDesign.⌋*(RS_MANI_00060)*

[TPS_MANI_03239]{DRAFT} **Definition of ComEventGrantDesign.remoteSubject** ⌈If the ComEventGrantDesign references one or several AbstractIamRemoteSubjects in the role remoteSubject then the design foresees that only the defined remoteSubjects shall be granted rights to access the VariableDataPrototype that is referenced in the role event by the same ComEventGrantDesign.⌋*(RS_MANI_00060)*

[TPS_MANI_03251]{DRAFT} **Definition of ComFieldGrantDesign.remoteSubject** ⌈If the ComFieldGrantDesign references one or several AbstractIamRemoteSubjects in the role remoteSubject then the design foresees that only the defined remoteSubjects shall be granted rights to access the Field that is referenced in the role field by the same ComFieldGrantDesign.⌋*(RS_MANI_00060)*

3.6.3.1 Remote subject in case of TLS

This chapter defines how a AbstractIamRemoteSubject is modeled in case of a TLS-based secure channel.

[TPS_MANI_03240]{DRAFT} **Modeling of a remote peer in case of TLS-based secure channel** ⌈In case of TLS-based secure channel the remote peer is modeled as TlsIamRemoteSubject that is identified either by

- a CryptoServiceCertificate that is referenced by the TlsIamRemoteSubject in the role acceptedRemoteCertificate,
- a Pre-shared Key that is referenced by the TlsIamRemoteSubject via TlsCryptoCipherSuite in the role acceptedCryptoCipherSuiteWithPsk.

⌋*(RS_MANI_00036, RS_MANI_00060)*

Please note that the security of a pre-shared key as authentication in TLS protocol depends on the number of entities sharing the same key. If multiple Machines are using the same shared key, one cannot reliably distinguish between those Machines because any of them can impersonate the others.

It can only be ensured that no other Machine without the knowledge of the pre-shared key can established a secure channel.

The TlsIamRemoteSubject may be identified by using certificates in two ways. First, it is possible to directly specify the certificates that the TlsIamRemoteSubject may to use by referencing them and setting derivedCertificateAccepted to false.

This approach requires the presence of the remote certificate on the local Machine. Secondly, by setting derivedCertificateAccepted to true it is possible to specify the Common Name (as given in the X509 Certificate) of the TlsIamRemoteSubject.

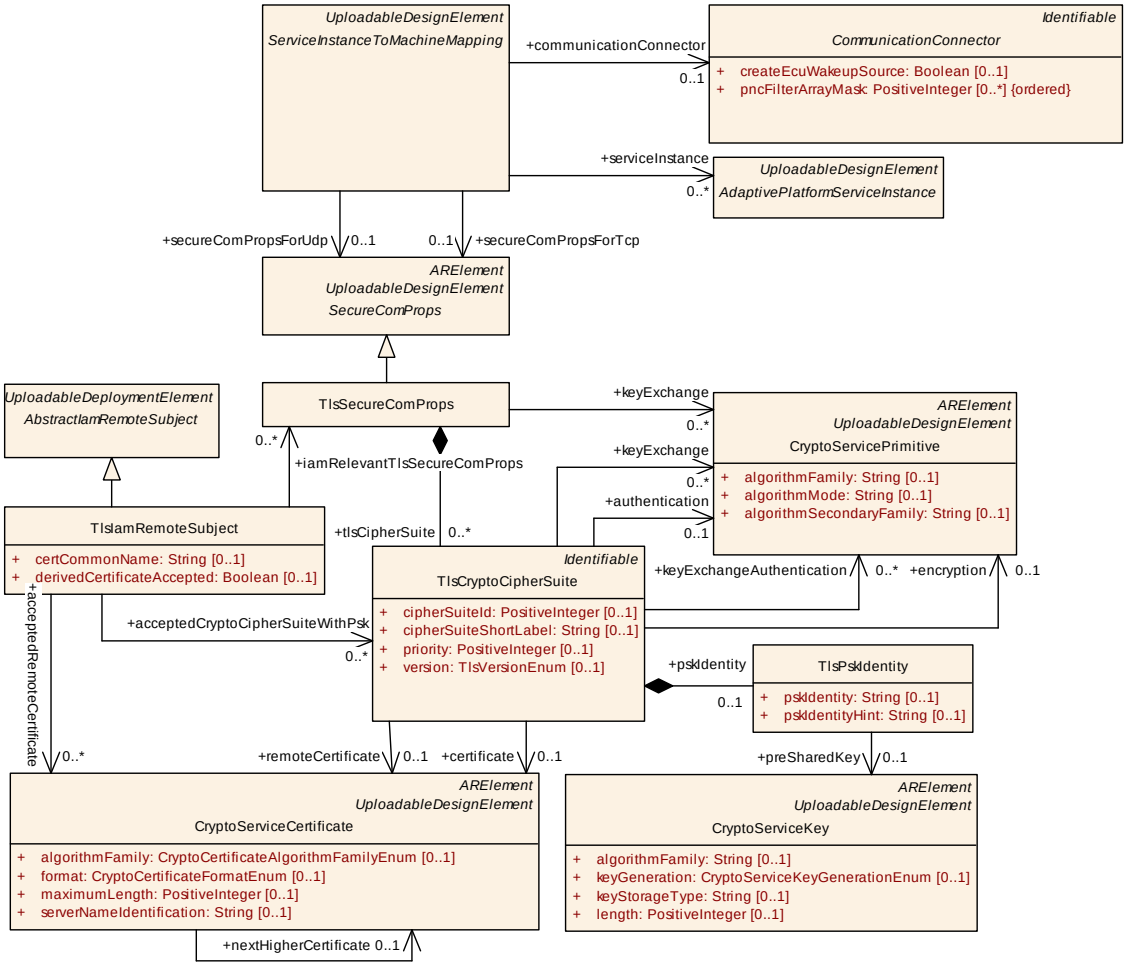


Figure 3.91: Proxy information about the remote node in case of TLS

In that case, the acceptedRemoteCertificates define the set of allowed root certificates for the certificate presented by the TlsIamRemoteSubject.

The reason for the upper multiplicity is that the OEM may have multiple suppliers for a Machine and it shall be allowed to define that in such a case all these Machines are allowed to connect even though they have different certificate chains.

[TPS_MANI_03241]{DRAFT} **Modeling of relevant TlsSecureComProps for TlsIamRemoteSubject** ⌈With the TlsIamRemoteSubject.iamRelevantTlsSecureComProps reference it is possible to define all TlsSecureComProps that the TlsIamRemoteSubject supports to establish a secure channel.⌋ *(RS_MANI_00036, RS_MANI_00060)*

Class	**TlsIamRemoteSubject**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM			
Note	This meta-class defines the proxy information about the remote node in case of TLS. **Tags:** atp.Status=candidate atp.recommendedPackage=IamRemoteSubjects			
Base	*ARElement*, *ARObject*, *AbstractIamRemoteSubject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
acceptedCryptoCipherSuiteWithPsk	TlsCryptoCipherSuite	*	ref	This reference is used to identify a remote node by means of the preshared Key. **Tags:** atp.Status=candidate
acceptedRemoteCertificate	CryptoServiceCertificate	*	ref	This reference is used to identify a remote node by means of the certificate. **Tags:** atp.Status=candidate
certCommonName	String	0..1	attr	This attribute defines the common name (CN) of the certificate of the remote peer. **Tags:** atp.Status=candidate
derivedCertificateAccepted	Boolean	0..1	attr	This attribute defines whether a derivedCertificate is accepted (true) or not (false). **Tags:** atp.Status=candidate
iamRelevantTlsSecureComProps	TlsSecureComProps	*	ref	This reference defines the local TlsSecureComProps that are relevant for IAM. **Tags:** atp.Status=candidate

Table 3.192: TlsIamRemoteSubject

3.6.3.2 Remote subject in case of IPsec

This chapter defines how a `AbstractIamRemoteSubject` is modeled in case of a IPsec-based secure channel.

[TPS_MANI_03242]{DRAFT} **Modeling of a remote peer in case of IPsec-based secure channel** ⌈In case of IPsec-based secure channel the remote peer is modeled as `IPSecIamRemoteSubject` that is identified by `IPSecRule`s that are referenced by `localIpSecRule`. The `IPSecRule`s define all secure connections that the remote peer is allowed to establish.⌋*(RS_MANI_00036, RS_MANI_00060)*

Please note that the local IP Address of the remote peer is defined by the `Network-Endpoint` that aggregates the `IPSecRule`s.

Class	IPSeclamRemoteSubject			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM			
Note	This meta-class defines the proxy information about the remote node in case of IPsec. **Tags:** atp.Status=candidate atp.recommendedPackage=IamRemoteSubjects			
Base	ARElement, ARObject, AbstractIamRemoteSubject, CollectableElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable, UploadableDeploymentElement, UploadablePackage Element			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
localIpSecRule	IPSecRule	*	ref	This reference is used to describe theRemoteSubjects local IPSecRules. **Tags:** atp.Status=candidate

Table 3.193: IPSeclamRemoteSubject

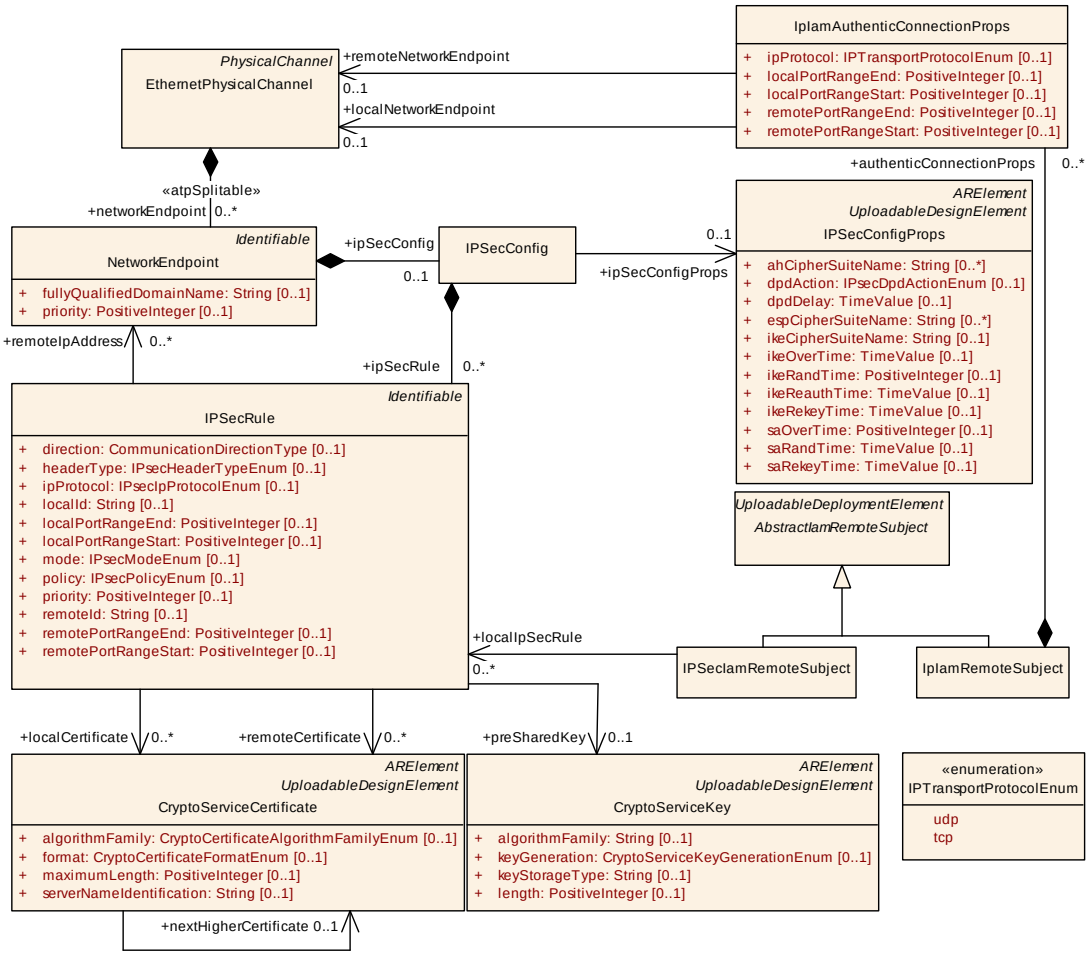


Figure 3.92: Proxy information about the remote node in case of IPsec

3.6.3.3 Remote subject in case of IP communication

Please note that it is possible to define a `AbstractIamRemoteSubject` that is based on the general IP communication. In this case no details about how the communication is secured are given and actually securing the communication (e.g., cryptographically, via hardware mechanism, or appropriate network and switch design) is not part of the model. A `IpIamRemoteSubject` is identified by a combination of a local and a remote IP address, local and remote port ranges, and a transport protocol.

[TPS_MANI_03244]{DRAFT} **Modeling of a remote peer in case of a general IP communication** ⌈In case of a general IP communication the remote peer is modeled as `IpIamRemoteSubject` that is identified by the `NetworkEndpoint` that is referenced by the `localNetworkEndpoint` reference. The defined remote peer is allowed to establish IP connections to the `remoteNetworkEndpoint` over the `ipProtocol` and the defined local port range and remote port range.⌋ *(RS_MANI_00060)*

Class	**IpIamRemoteSubject**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM			
Note	This meta-class defines the proxy information about the remote node in case of general IP communication. **Tags:** atp.Status=candidate atp.recommendedPackage=IamRemoteSubjects			
Base	*ARElement*, *ARObject*, *AbstractIamRemoteSubject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authentic Connection Props	IpIamAuthentic ConnectionProps	*	aggr	Definition of IP rules assigned to the IpIamRemote Subject. **Tags:** atp.Status=candidate

Table 3.194: IpIamRemoteSubject

Enumeration	**IPTransportProtocolEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM
Note	Transport Layer protocols in the Internet protocol suite. **Tags:** atp.Status=candidate
Aggregated by	IpIamAuthenticConnectionProps.ipProtocol
Literal	***Description***
tcp	Transmission Control Protocol (TCP) **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=candidate
udp	User Datagram Protocol (UDP). **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=candidate

Table 3.195: IPTransportProtocolEnum

Class	**IplamAuthenticConnectionProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SCREIAM			
Note	This meta-class defines a set of properties for IP connections in the context of IAM configuration. **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	IplamRemoteSubject.authenticConnectionProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ipProtocol	IPTransportProtocol Enum	0..1	attr	This attribute defines the relevant IP protocol. **Tags:** atp.Status=candidate
localNetwork Endpoint	EthernetPhysical Channel	0..1	ref	This reference defines an authentic local Network Endpoint in terms of IAM configuration. **Tags:** atp.Status=candidate
localPortRange End	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines an end value for the local port range. **Tags:** atp.Status=candidate
localPortRange Start	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines a start value for the local port range. **Tags:** atp.Status=candidate
remoteNetwork Endpoint	EthernetPhysical Channel	0..1	ref	This reference defines an authentic remote Network Endpoint in terms of IAM configuration. **Tags:** atp.Status=candidate
remotePort RangeEnd	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines an end value for the remote port range. **Tags:** atp.Status=candidate
remotePort RangeStart	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines a start value for the remote port range. **Tags:** atp.Status=candidate

Table 3.196: IplamAuthenticConnectionProps

3.6.3.4 Remote subject in case of SecOC communication

The identity information in the case of SecOC depends on the group of `Machine`s that are sharing the same cryptographic key.

In other words, if a valid SecOC message is received with a given key it is given that only remote `Machine`s that “know the key” were able to send the message. The key is associated with a DataId and defines the “object” in the access control model. If a message received for a given DataID cannot be validated, then it will be dropped. Therefore, the access control between the remote subject and local object is taking place.

To summarize, the modeling of a Remote subject in case of SecOC cannot provide additional benefit neither by increasing the granularity of the subject identification nor providing new enforcement of rules on the object.

3.7 Intrusion Detection Design

[TPS_MANI_01338]{DRAFT} **Semantics of `SecurityEventReportToSecurityEventDefinitionMapping`** ⌈The modeling of the association between a specific security event and the corresponding `RPortPrototype` typed by a `SecurityEventReportInterface` is created by means of the `SecurityEventReportToSecurityEventDefinitionMapping`.⌋*(RS_MANI_00068)*

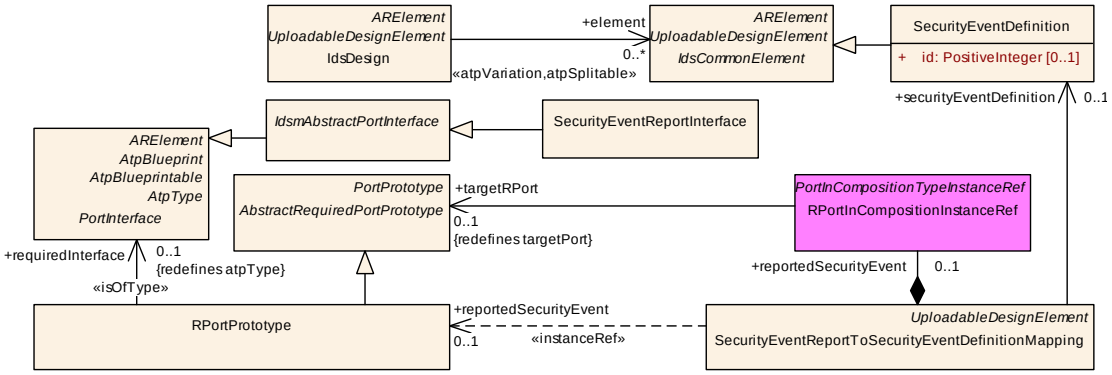


Figure 3.93: Specification of the `SecurityEventReportToSecurityEventDefinitionMapping`

This meta-class maps the `RPortPrototype` to a `SecurityEventDefinition` that itself is part of the so-called `Security Extract`.

[TPS_MANI_01339]{DRAFT} **Existence of the `SecurityEventReportToSecurityEventDefinitionMapping` is motivated by the AUTOSAR methodology** ⌈The existence of the `SecurityEventReportToSecurityEventDefinitionMapping` is motivated by the AUTOSAR methodology. At the point in time when a given `SecurityEventReportInterface` is defined it could be that the corresponding `SecurityEventDefinition` is not yet defined.

So it is possible to add this association later. Another reason for the existence of the mapping class is that a specific piece of application software may report different specific security events defined by different OEMs, depending on the deployment of the application software.

Of course, the semantics of the security event all always be either identical or at least comparable, it could still happen that the Id of a security event might change depending on the specific project or simply because different OEMs use different Ids for semantically identical security events.⌋*(RS_MANI_00068)*

Class	**SecurityEventReportToSecurityEventDefinitionMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::SecurityDesign			
Note	This meta-class represents the ability to map a PortPrototype for reporting a security event to the actual security event that shall be reported by this PortPrototype. **Tags:** atp.Status=candidate atp.recommendedPackage=SecurityEventReportToSecurityEventDefinitionMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
reported SecurityEvent	AbstractRequiredPort Prototype	0..1	iref	This identifies the mapped security event. **Tags:** atp.Status=candidate **InstanceRef implemented by:** RPortInComposition InstanceRef
securityEvent Definition	SecurityEventDefinition	0..1	ref	This reference identifies the definition of the security event. **Tags:** atp.Status=candidate

Table 3.197: SecurityEventReportToSecurityEventDefinitionMapping

4 Diagnostic Design

The configuration of diagnostics on the *AUTOSAR adaptive platform* will typically be done by creating a Diagnostic Extract documented within the specification of the TPS Diagnostic Extract Template [2] that is also used on the *AUTOSAR classic platform*.

Therefore, concepts within the Diagnostic Extract should be similarly applicable to models on both platforms uniformly.

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.

4.1 Extensions to the Diagnostic Common Props

The definition of the `DiagnosticCommonProps` is extended exclusively for the use on the *AUTOSAR adaptive platform*. The extension improves the description of authentication of diagnostic clients as well as the specification of diagnostic addresses.

Class	<<atpVariation>> **DiagnosticCommonProps**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticCommonProps			
Note	This meta-class aggregates a number of common properties that are shared among a diagnostic extract. **Tags:** vh.latestBindingTime=codeGenerationTime			
Base	*ARObject*			
Aggregated by	DiagnosticContributionSet.commonProperties			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authentication Timeout	TimeValue	0..1	attr	This attribute defines the time (in seconds) that the authentication state is maintained in default-session if there is no communication from the authenticated client.
debounce AlgorithmProps	DiagnosticDebounce AlgorithmProps	*	aggr	Defines the used debounce algorithms relevant in the context of the enclosing DiagnosticCommonProps. Usually, there is a variety of debouncing algorithms to take into account and therefore the multiplicity of this aggregation is set to 0..*.
default Endianness	ByteOrderEnum	0..1	attr	Defines the default endianness of the data belonging to a DID or RID which is applicable if the DiagnosticData Element does not define the endianness via the swData DefProps.baseType attribute.
diagnostic Address	SoftwareCluster DiagnosticAddress	*	aggr	"This aggregation represents the collection of diagnostic addresses that apply for the SoftwareClusterDesign. Note: This atpSplitable property has no atp.Splitkey due to atpVariation (PropertySetPattern). **Stereotypes:** atpSplitable **Tags:** xml.namePlural=DIAGNOSTIC-ADDRESSES
event Combination Reporting Behavior	DiagnosticEvent CombinationReporting BehaviorEnum	0..1	attr	In case of EventCombination on Retrieval, this attribute specifies if a specific order of reporting is to be maintained.

△

Class	<<atpVariation>> **DiagnosticCommonProps**			
external Authentication	DiagnosticExternal Authentication Identification	*	aggr	This aggregation supports the configuration of the authentication of diagnostic clients. Note: This atpSplitable property has no atp.Splitkey due to atpVariation (PropertySetPattern). **Stereotypes:** atpSplitable
maxNumberOf Request Correctly Received Response Pending	PositiveInteger	0..1	attr	Maximum number of negative responses with response code 0x78 (requestCorrectlyReceived-ResponsePending) allowed per request. DCM will send a negative response with response code 0x10 (generalReject), in case the limit value gets reached. Value 0xFF means that no limit number of NRC 0x78 response apply.
occurrence Counter Processing	DiagnosticOccurrence CounterProcessing Enum	0..1	attr	This attribute defines the consideration of the fault confirmation process for the occurrence counter.
resetConfirmed BitOnOverflow	Boolean	0..1	attr	This attribute defines, whether the confirmed bit is reset or not while an event memory entry will be displaced.
resetPendingBit OnOverflow	Boolean	0..1	attr	This attribute defines, whether the pending bit is reset or not while an event memory entry will be displaced. In order to be compliant to ISO 14229-1 [1], this parameter needs to be set to "false".
responseOnAll RequestSids	Boolean	0..1	attr	If set to FALSE the DCM will not respond to diagnostic request that contains a service ID which is in the range from 0x40 to 0x7F or in the range from 0xC0 to 0xFF (Response IDs).
responseOn Second Declined Request	Boolean	0..1	attr	Defines the reaction upon a second request (ClientB) that can not be processed (e.g. due to priority assessment). TRUE: when the second request (Client B) can not be processed, it shall be answered with NRC21 BusyRepeat Request. FALSE: when the second request (Client B) can not be processed, it shall not be responded.
typeOfEvent Combination Supported	DiagnosticEvent CombinationBehavior Enum	0..1	attr	Select type of Event Combination support.

Table 4.1: DiagnosticCommonProps

[constr_10488]{DRAFT} **Existence of attribute `DiagnosticCommonProps.authenticationTimeout`** ⌈Attribute `DiagnosticCommonProps.authenticationTimeout` shall exist **at the time when the manifest is complete.**⌋*()*

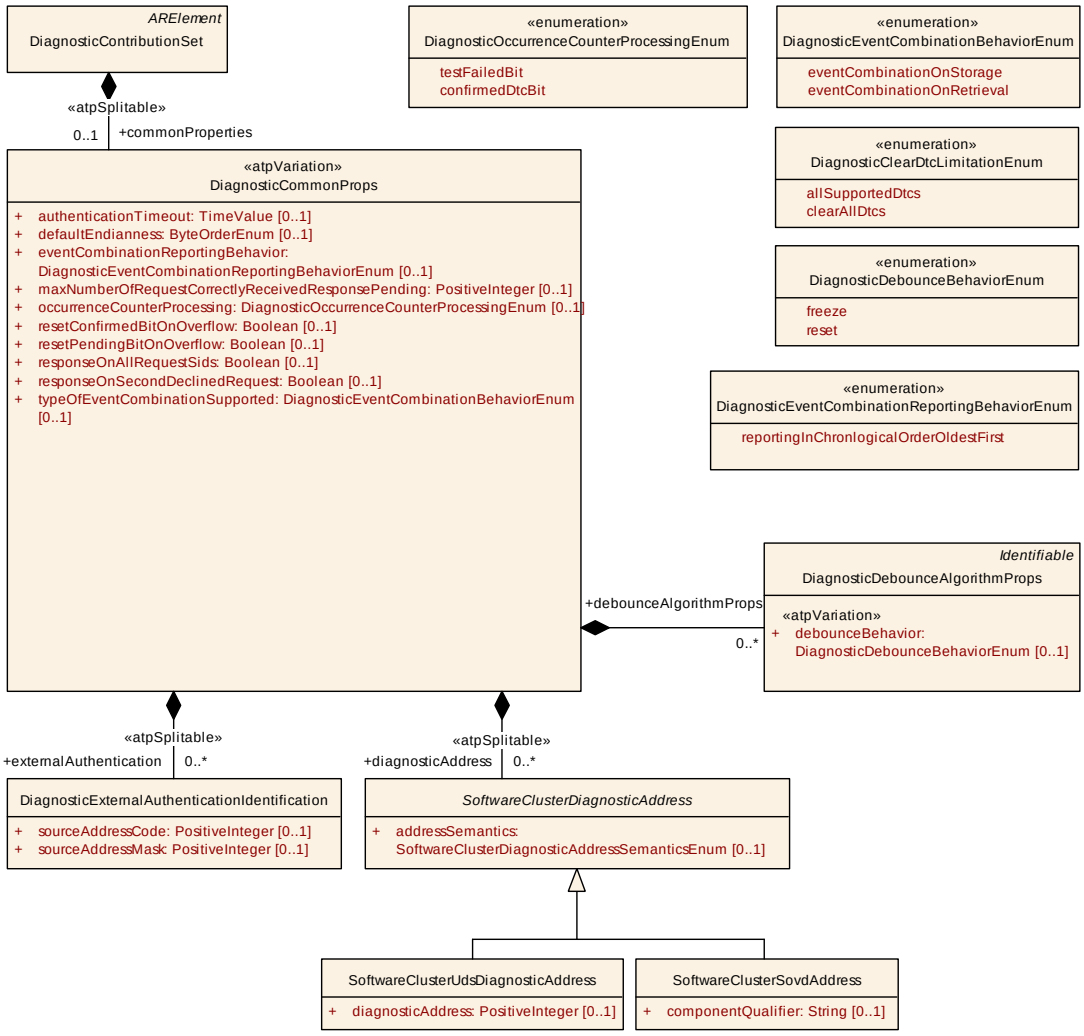


Figure 4.1: Modeling of `DiagnosticCommonProps` on the *AUTOSAR adaptive platform*

4.1.1 Diagnostic Address

[TPS_MANI_01434]{DRAFT} **Diagnostic Address defined in the context of a `DiagnosticCommonProps`** ⌈It is possible to front-load the definition of diagnostic addressed as part of the the configuration of diagnostic management.

Therefore, the definition of the `DiagnosticCommonProps` 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 `DiagnosticCommonProps.diagnosticAddress`.

A `DiagnosticCommonProps` 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)*

The goal of the specific modeling decision described in [TPS_MANI_01434] has been to make the modeling of the diagnostic address extensible such that the idiomatic ways to define diagnostic addresses can also be supported by adding respective subclasses of SoftwareClusterDiagnosticAddress.

Class	***SoftwareClusterDiagnosticAddress*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
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. ...).			
Base	*ARObject*			
Subclasses	SoftwareClusterSovdAddress, SoftwareClusterUdsDiagnosticAddress			
Aggregated by	DiagnosticCommonProps.diagnosticAddress			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
address Semantics	SoftwareCluster DiagnosticAddress SemanticsEnum	0..1	attr	This attribute clarifies whether the address value shall be interpreted as a physical or a functional address.

Table 4.2: SoftwareClusterDiagnosticAddress

[constr_10487]{DRAFT} **Only one physical address per DiagnosticCommonProps** ⌈Each DiagnosticCommonProps shall only aggregate one SoftwareClusterDiagnosticAddress where the value of attribute addressSemantics is set to SoftwareClusterDiagnosticAddressSemanticsEnum.physicalAddress.⌋*()*

[TPS_MANI_01405]{DRAFT} **Semantics of sub-classes of SoftwareClusterDiagnosticAddress** ⌈The definition of the actual diagnostic address is done by means of the applicable sub-class of the abstract base-class SoftwareClusterDiagnosticAddress:

- If the diagnostic session is executed on the basis of **UDS**, then the diagnostic address shall be specified by means of SoftwareClusterUdsDiagnosticAddress.diagnosticAddress.
- If the diagnostic session is executed on the basis of **SOVD**, then the diagnostic address shall be specified by means of SoftwareClusterSovdAddress.componentQualifier.

⌋*(RS_MANI_00005, RS_MANI_00070)*

Class	**SoftwareClusterSovdAddress**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define a diagnostic address specifically for the SOVD case. **Tags:** atp.Status=candidate			
Base	*ARObject*, *SoftwareClusterDiagnosticAddress*			
Aggregated by	DiagnosticCommonProps.diagnosticAddress			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**SoftwareClusterSovdAddress**			
component Qualifier	String	0..1	attr	This attribute is used to specify the component qualifier for the usage in an SOVD query. **Tags:** atp.Status=candidate

Table 4.3: SoftwareClusterSovdAddress

[constr_10404]{DRAFT} **Existence of `SoftwareClusterSovdAddress.componentQualifier`** ⌈For each `SoftwareClusterSovdAddress`, attribute `componentQualifier` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10236]{DRAFT} **Multiplicity of attribute `SoftwareClusterDiagnosticAddress.addressSemantics`** ⌈For each `SoftwareClusterDiagnosticAddress`, the attribute `addressSemantics` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

Enumeration	**SoftwareClusterDiagnosticAddressSemanticsEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This meta-class defines a list of semantics for the interpretation of diagnostic addresses in the context of a SoftwareCluster.
Aggregated by	*SoftwareClusterDiagnosticAddress*.addressSemantics
Literal	*Description*
functionalAddress	This address represents a functional address. **Tags:** atp.EnumerationLiteralIndex=1
physicalAddress	This address represents a physical address. **Tags:** atp.EnumerationLiteralIndex=0

Table 4.4: SoftwareClusterDiagnosticAddressSemanticsEnum

Class	**SoftwareClusterUdsDiagnosticAddress**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define a diagnostic address specifically for the UDS case			
Base	*ARObject*, *SoftwareClusterDiagnosticAddress*			
Aggregated by	DiagnosticCommonProps.diagnosticAddress			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
diagnostic Address	PositiveInteger	0..1	attr	This attribute represents the collection of diagnostic addresses the SoftwareCluster occupies.

Table 4.5: SoftwareClusterUdsDiagnosticAddress

4.1.2 Diagnostic Authentication Configuration

[TPS_MANI_01435] Semantics of meta-class `DiagnosticExternalAuthenticationIdentification` ⌈Meta-class `DiagnosticExternalAuthenticationIdentification` is used to define source address information for the authentication of diagnostic clients.⌋*()*

Please note that it is possible to support diagnostic clients with a fixed source address or with a range of source addresses by means of the aggregation of DiagnosticExternalAuthenticationIdentification at DiagnosticCommonProps with a multiplicity of 0..*.

Class	**DiagnosticExternalAuthenticationIdentification**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to support the authentication of diagnostic clients.			
Base	*ARObject*			
Aggregated by	DiagnosticCommonProps.externalAuthentication			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sourceAddress Code	PositiveInteger	0..1	attr	This attribute represents the value to exactly match after applying the mask to the source address of a client's request.
sourceAddress Mask	PositiveInteger	0..1	attr	This attribute represents the mask to apply to the source address of a client's request.

Table 4.6: DiagnosticExternalAuthenticationIdentification

4.2 Diagnostic Mapping

4.2.1 Overview

In terms of applicability to this document, one specific 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.

In order to exemplify the approach, the diagram depicted in Figure 4.2 describes a very simplistic situation where two different PPortPrototypes typed by possibly two different DiagnosticDataIdentifierInterface exposed by an AdaptiveApplicationSwComponentType is accessed by the AUTOSAR Adaptive Diagnostic Management on the *AUTOSAR adaptive platform* with the purpose of accessing an entire DID.

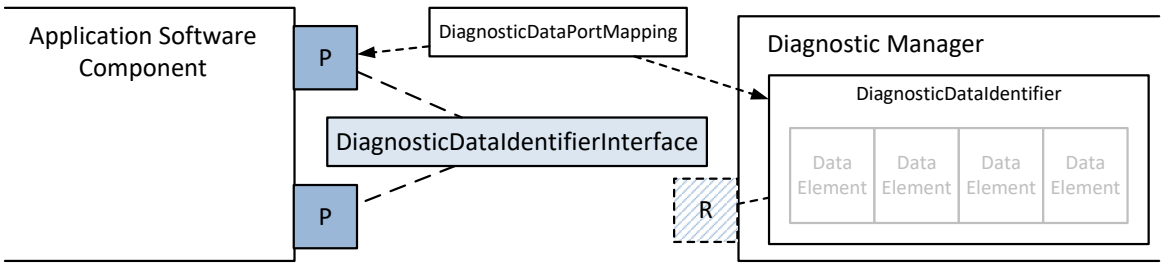


Figure 4.2: Example data exchange for diagnostic purpose

In particular, a subclass of DiagnosticSwMapping (in this specific case: DiagnosticDataPortMapping) formalizes the "connection" between both ends of the communication.

Class	***DiagnosticSwMapping*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	This represents the ability to define a mapping between a diagnostic information (at this point there is no way to become more specific about the semantics) to a software-component.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticAuthenticationPortMapping, DiagnosticClearConditionPortMapping, DiagnosticDataPort Mapping, DiagnosticEnableConditionPortMapping, DiagnosticEventPortMapping, DiagnosticExternal AuthenticationPortMapping, DiagnosticFimFunctionMapping, DiagnosticIndicatorPortMapping, DiagnosticMemoryDestinationPortMapping, DiagnosticMonitorPortMapping, *DiagnosticMultipleResource PortMapping*, DiagnosticOperationCyclePortMapping, DiagnosticSecurityLevelPortMapping, Diagnostic ServiceDataMapping, DiagnosticServiceGenericMapping, DiagnosticServiceSwMapping, Diagnostic ServiceValidationMapping, DiagnosticSovdAuthorizationPortMapping, DiagnosticSovdBulkDataPort Mapping, DiagnosticSovdConfigurationPortMapping, DiagnosticSovdProximityChallengePortMapping, DiagnosticSovdServiceValidationPortMapping, DiagnosticSovdUpdatePortMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.7: DiagnosticSwMapping

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 `PortPrototype`s).

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*, *DiagnosticCommon Element*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	DiagnosticParameter	*	aggr	This is the dataElement associated with the Diagnostic DataIdentifier. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataElement.bitOffset, data Element.ident.shortName, dataElement.variation Point.shortLabel vh.latestBindingTime=postBuild
didSize	PositiveInteger	0..1	attr	This attribute indicates the size in bytes of the Diagnostic DataIdentifier.
representsVin	Boolean	0..1	attr	This attributes indicates whether the specific Diagnostic DataIdentifier represents the vehicle identification.
supportInfoByte	DiagnosticSupportInfo Byte	0..1	aggr	This attribute represents the supported information associated with the DiagnosticDataIdentifier.

Table 4.8: DiagnosticDataIdentifier

Class	**DiagnosticSupportInfoByte**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class defines the support information (typically byte A) to declare the usability of the Data Elements within the so-called packeted PIDs (e.g. PID$68).			
Base	*ARObject*			
Aggregated by	DiagnosticDataIdentifier.supportInfoByte, DiagnosticParameterIdentifier.supportInfoByte			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
position	PositiveInteger	0..1	attr	This represents the position of the supportInfo in the PID. Unit: byte.
size	PositiveInteger	0..1	attr	This represents the size of the supportInfo within the PID. Unit: byte.

Table 4.9: DiagnosticSupportInfoByte

The response to this situation on the *AUTOSAR classic platform* has been the definition of the SwcServiceDependency that allows for associating several PortPrototypes in specific roles to a given use-case.

On the other hand, there is a clear difference between the design of PortInterfaces for diagnostic usage on the *AUTOSAR classic platform* (where generic ClientServerInterfaces, SenderReceiverInterfaces, and ModeSwitchInterfaces are used) and the *AUTOSAR adaptive platform* (where the PortInterfaces are specifically tailored to the respective diagnostic use case).

In other words, it specifically tailored PortInterfaces were used, the need for an "umbrella" that defines the scope of the modeling of a given service use case would be drastically reduced.

On top of that, the usage of the InstanceSpecifier as the element of identification is strictly bound to the PortPrototype rather than the SwcServiceDependency. In other words, the diagnostic management uses ≪instanceRef≫s to identify the "endpoint" in the application software.

The ≪instanceRef≫ and the corresponding InstanceSpecifier are strongly related to each other. If one is known the other can be deduced. The consequence of this conclusion is that it would be possible to configure the diagnostic management in the presence of mappings to the application software model that itself does not have to be accessible at the time of creating the configuration of diagnostic management.

This means that an ≪instanceRef≫ originating from a subclass of DiagnosticSwMapping needs to have an additional decoration in the form of the stereotype ≪atpUriDef≫.

Please note that the mapping targets[1] within a set of diagnostic mappings may exist in several instances at run-time.

This kind of multiple instantiation is formalized by the existence of meta-class Process (which in turn is represented by meta-class ProcessDesign on design level), see chapter 3.2.4.

[1] on the end of the application software

It is very typical that different instances of a piece of application software could require a different diagnostic mapping and the modeling needs to accommodate to this requirement, i.e. a relation between a diagnostic mapping and the ProcessDesign needs to be established.

[constr_10002]{DRAFT} **Only one mapping per PortPrototype** ⌈If one instance of the following sub-classes of DiagnosticSwMapping – that refers to a given ProcessDesign – refers to a PortPrototype, then no other instance of DiagnosticSwMapping that refers to the same ProcessDesign shall refer to the same PortPrototype:

- DiagnosticEventPortMapping that is associated with a RPortPrototype typed by a DiagnosticMonitorInterface or a DiagnosticEventInterface.
- DiagnosticOperationCyclePortMapping that is associated with a RPortPrototype typed by a DiagnosticOperationCycleInterface.
- DiagnosticEnableConditionPortMapping that is associated with a RPortPrototype typed by a DiagnosticConditionInterface.
- DiagnosticClearConditionPortMapping that is associated with a RPortPrototype typed by a DiagnosticConditionInterface.
- DiagnosticIndicatorPortMapping that is associated with a RPortPrototype typed by a DiagnosticIndicatorInterface.
- DiagnosticMemoryDestinationPortMapping that is associated with an RPortPrototype typed by a DiagnosticDTCInformationInterface.
- DiagnosticSecurityLevelPortMapping that is associated with an PPortPrototype typed by a DiagnosticSecurityLevelInterface.
- DiagnosticDataPortMapping that is associated with a PPortPrototype typed by a DiagnosticDataIdentifierInterface.
- DiagnosticSecurityLevelPortMapping that is associated with a PPortPrototype typed by a DiagnosticSecurityLevelInterface.
- DiagnosticServiceValidationMapping that is associated with a PPortPrototype typed by a DiagnosticServiceValidationInterface.

This rule shall be imposed **at the time when the diagnostic design is complete**.⌋*()*

The rationale for the existence of [constr_10002] is that the respective PortPrototype has a clearly defined functionality. For example, it can only provide the content of one DID, but it cannot provide the content of an arbitrary number of DIDs.

For such a case, the DiagnosticServiceGenericMapping (see section 4.2.3.3.1) shall be applied.

Please note the [constr_10002] does not apply to the DiagnosticServiceGenericMapping, i.e. a PortPrototype that is not subject to [constr_10002] can be referenced by multiple DiagnosticServiceGenericMapping.

In other words, the ability for several DiagnosticServiceGenericMappings to refer to the same PortPrototype is what makes the DiagnosticServiceGenericMapping *generic*.

In addition, [constr_10002] is intentionally not applied for PPortPrototypes typed by DiagnosticDataElementInterface because the operations executed in such context are stateless and can therefore be called from several clients without restrictions.

4.2.1.1 Mapping of multiple Diagnostic Resources to a PortPrototype

The “regular” mapping of diagnostic resources foresees a 1:1 mapping of a specific diagnostic resource to a PortPrototype.

There are, however, use cases where the intention is to map multiple diagnostic resources to a single PortPrototype. The following use cases exist:

- The software wants to report the status of multiple events to the DM.
- The software wants to retrieve the status of multiple events from the DM.
- The software wants to set multiple diagnostic conditions.

Using the “regular” mapping approach, the software would need to define a dedicated PortPrototype for, e.g. reporting the status of each event. This could mean that a large number of PortPrototypes has to be defined in such a case.

If the PortPrototype is typed by a DiagnosticMultipleResourceInterface, then the PortPrototype can become the target of references from multiple DiagnosticMultipleResourcePortMappings.

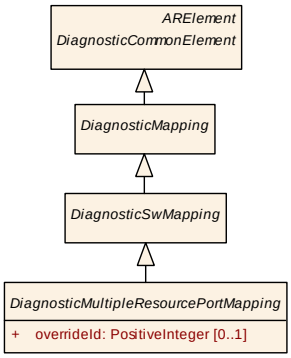


Figure 4.3: Modeling of DiagnosticMultipleResourcePortMapping

Class	***DiagnosticMultipleResourcePortMapping*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This abstract base class enables the mapping of diagnostic PortInterfaces that deal with multiple diagnostic resources.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticMultipleConditionPortMapping, DiagnosticMultipleEventPortMapping, DiagnosticMultiple MonitorPortMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
overrideId	PositiveInteger	0..1	attr	This attribute shall be used to define the value of a manually override of the automatic generated handle Id value.

Table 4.10: DiagnosticMultipleResourcePortMapping

This means, for example, that for each `DiagnosticEvent` that shall have its status reported to the DM, a single `DiagnosticMultipleMonitorPortMapping` (please refer to section 4.2.2.3) shall be defined.

In other words, this approach minimizes the number of `PortPrototype`s that have to be defined, but has no impact on the overall number of diagnostic mappings.

Please note that each `DiagnosticMultipleResourcePortMapping` can only map **one** `PortPrototype` to **one** diagnostic resource (e.g. a `DiagnosticEvent`).

But it is possible to define multiple `DiagnosticMultipleResourcePortMapping` that refer to the **same** `PortPrototype` and a **different** diagnostic resources of the same kind. Thereby the multiple resource mapping semantics is achieved.

[TPS_MANI_01415]{DRAFT} **Semantics of attribute `DiagnosticMultipleResourcePortMapping.overrideId`** ⌈The DM will assign an ID for each diagnostic resource mapped to a `PortPrototype` that is typed by a `DiagnosticMultipleResourceInterface`.

If the user prefers to override the algorithm applied for the assignment of the ID, it is possible to assign a dedicated ID by means of setting the optional attribute `overrideId` to the intended value. The DM will respect this setting for the assignment of the ID.⌋*()*

4.2.2 Fault-Memory-related Diagnostic Mapping

This section contains a documentation of `DiagnosticMapping`s in the context of fault memory configuration.

4.2.2.1 Diagnostic Monitor to Port Mapping

[TPS_MANI_01351]{DRAFT} **Reporting the status of a `DiagnosticEvent` on the *AUTOSAR adaptive platform*** ⌈For the purpose of reporting the status of a diagnostic

event on the *AUTOSAR adaptive platform*, the relation between a DiagnosticEvent and one RPortPrototypes is created by using the DiagnosticMonitorPortMapping that refers to

- a DiagnosticEvent in the role diagnosticEvent,
- a ProcessDesign in the role process, and
- an RPortPrototype typed by a DiagnosticMonitorInterface in the role rPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype will use at run-time.

⌋(*RS_MANI_00005, RS_MANI_00061*)

Class	**DiagnosticMonitorPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service port the Diagnostic Monitor is mapped. **Tags:** atp.recommendedPackage=DiagnosticPortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to the DiagnosticEvent that is assigned to SWC service ports.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic MonitorPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 4.11: DiagnosticMonitorPortMapping

The use case that supports the reporting of a diagnostic event is depicted in Figure 4.4.

[constr_10047]{DRAFT} **Restriction for the applicability of DiagnosticMonitorPortMapping** ⌈If an RPortPrototype is referenced by a DiagnosticMonitorPortMapping, then the RPortPrototype shall be typed by a DiagnosticMonitorInterface **at the time when the diagnostic design is complete**.⌋*()*

[constr_10048]{DRAFT} **Existence of reference from DiagnosticMonitorPortMapping to DiagnosticEvent** ⌈Each DiagnosticEvent shall only be referenced by exactly one DiagnosticMonitorPortMapping or DiagnosticMultipleMonitorPortMapping **at the time when the diagnostic design is complete**.⌋*()*

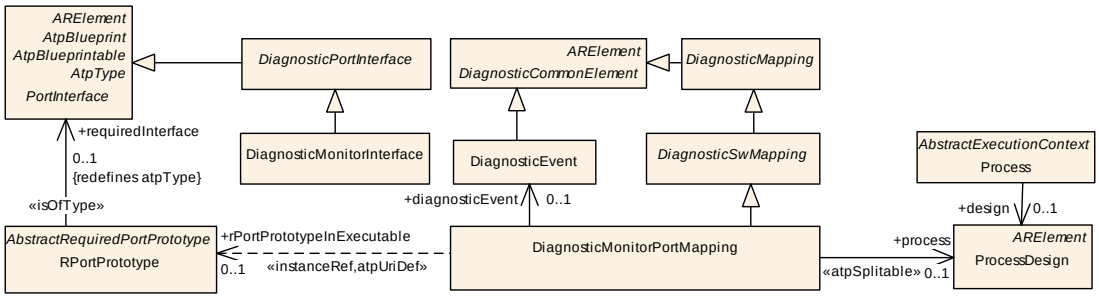


Figure 4.4: Modeling of `DiagnosticMonitorPortMapping` for reporting the status of diagnostic events on the *AUTOSAR adaptive platform*

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*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
associated Event Identification	PositiveInteger	0..1	attr	This attribute represents the identification number that is associated with the enclosing DiagnosticEvent and allows to identify it when placed into a snapshot record or extended data record storage. This value can be reported as internal data element in snapshot records or extended data records.
clearEvent Allowed Behavior	DiagnosticClearEvent AllowedBehaviorEnum	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
confirmation Threshold	PositiveInteger	0..1	attr	This attribute defines the number of operation cycles with a failed result before a confirmed DTC is set to 1. The semantic of this attribute is a by "1" increased value compared to the confirmation threshold of the "trip counter" mentioned in ISO 14229-1 in figure D.4. A value of "1" defines the immediate confirmation of the DTC along with the first reported failed. This is also sometimes called "zero trip DTC". A value of "2" defines a DTC confirmation in the operation cycle after the first occurred failed. A value of "2" is typically used in the US for OBD DTC confirmation. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
connected Indicator	DiagnosticConnected Indicator	*	aggr	Event specific description of Indicators. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=connectedIndicator.shortName, connected Indicator.variationPoint.shortLabel vh.latestBindingTime=postBuild
prestorage FreezeFrame	Boolean	0..1	attr	This attribute describes whether the Prestorage of Freeze Frames is supported by the assigned event or not. true: Prestorage of FreezeFrames is supported fFalse: Prestorage of FreezeFrames is not supported

△

Class	**DiagnosticEvent**			
prestored Freezeframe StoredInNvm	Boolean	0..1	attr	If the Event uses a prestored freeze-frame (using the operations PrestoreFreezeFrame and ClearPrestored FreezeFrame of the service interface DiagnosticMonitor) this attribute indicates if the Event requires the data to be stored in non-volatile memory. TRUE = Dem shall store the prestored data in non-volatile memory, FALSE = Data can be lost at shutdown (not stored in Nvm)
recoverableIn SameOperation Cycle	Boolean	0..1	attr	If the attribute is set to true then reporting PASSED will reset the indication of a failed test in the current operation cycle. If the attribute is set to false then reporting PASSED will be ignored and not lead to a reset of the indication of a failed test.

Table 4.12: DiagnosticEvent

Enumeration	**DiagnosticClearEventAllowedBehaviorEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent
Note	This enumeration defines the possible behavior for clear event allowed
Aggregated by	DiagnosticEvent.clearEventAllowedBehavior
Literal	*Description*
noStatusByte Change	The event status byte keeps unchanged. **Tags:** atp.EnumerationLiteralIndex=0
onlyThisCycleAnd Readiness	The OperationCycle and readiness bits of the event status byte are reset. **Tags:** atp.EnumerationLiteralIndex=1

Table 4.13: DiagnosticClearEventAllowedBehaviorEnum

Class	**DiagnosticConnectedIndicator**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticEvent			
Note	Description of indicators that are defined per DiagnosticEvent.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticEvent.connectedIndicator			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
behavior	DiagnosticConnected IndicatorBehaviorEnum	0..1	attr	Behavior of the linked indicator.
healingCycle	DiagnosticOperation Cycle	0..1	ref	The deactivation of indicators per event is defined as healing of a diagnostic event. The operation cycle in which the warning indicator will be switched off is defined here.
healingCycle Counter Threshold	PositiveInteger	0..1	attr	This attribute defines the number of healing cycles for the WarningIndicatorOffCriteria **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
indicator	DiagnosticIndicator	0..1	ref	Reference to the used indicator.
indicatorFailure CycleCounter Threshold	PositiveInteger	0..1	attr	This attribute defines the number of failure cycles for the WarningIndicatorOnCriteria. Please note that this attribute is not relevant for the Adaptive Platform.

Table 4.14: DiagnosticConnectedIndicator

4.2.2.1.1 Multiple Diagnostic Monitor to Port Mapping

[TPS_MANI_01416]{DRAFT} **Reporting the status of multiple `DiagnosticEvent`s in the context of one `PortInterface` on the *AUTOSAR adaptive platform*** ⌈For the purpose of reporting the status of multiple diagnostic events on the *AUTOSAR adaptive platform*, the relation between a collection of `DiagnosticEvent`s and one `RPortPrototype`s is created by using the `DiagnosticMultipleMonitorPortMapping` that refers to

- a collection of `DiagnosticEvent`s in the role `diagnosticEvent`,
- a `ProcessDesign` in the role `process`, and
- an `RPortPrototype` typed by a `DiagnosticMultipleMonitorInterface` in the role `rPortPrototypeInExecutable`. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding `InstanceSpecifier` that the `PortPrototype` will use at run-time.

⌋*(RS_MANI_00005, RS_MANI_00061)*

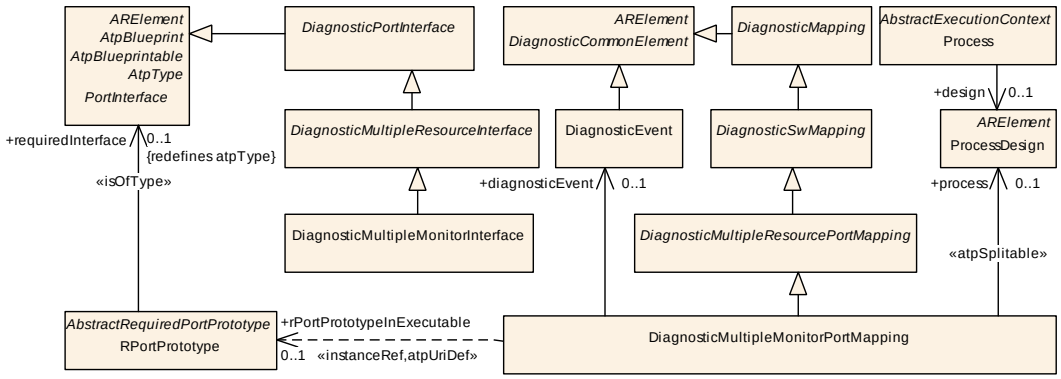


Figure 4.5: Modeling of `DiagnosticMultipleMonitorPortMapping` for reporting the status of multiple diagnostic events on the *AUTOSAR adaptive platform*

Class	**DiagnosticMultipleMonitorPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service port that can handle a collection of monitors the specific event is mapped **Tags:** atp.recommendedPackage=DiagnosticPortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticMultipleResourcePortMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to the DiagnosticEvent that is assigned to a SWC service port.

▽

△

Class	DiagnosticMultipleMonitorPortMapping			
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic MonitorMultipleMonitorPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 4.15: DiagnosticMultipleMonitorPortMapping

[constr_10436]{DRAFT} **Restriction for the applicability of `DiagnosticMultipleMonitorPortMapping`** ⌈If an `RPortPrototype` is referenced by a `DiagnosticMultipleMonitorPortMapping`, then the `RPortPrototype` shall be typed by a `DiagnosticMultipleMonitorInterface` **at the time when the diagnostic design is complete**.⌋*()*

4.2.2.2 Diagnostic Event to Port Mapping

[TPS_MANI_01048]{DRAFT} **Retrieving the status of a `DiagnosticEvent` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈For the purpose of reporting the status of a diagnostic event on the *AUTOSAR adaptive platform*, the relation between a `DiagnosticEvent` and one `RPortPrototype`s is created by using the `DiagnosticEventPortMapping` that refers to

- a `DiagnosticEvent` in the role `diagnosticEvent`,
- a `ProcessDesign` in the role `process`, and
- an `RPortPrototype` typed by a `DiagnosticEventInterface` in the role `rPortPrototypeInExecutable`. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding `InstanceSpecifier` that the `PortPrototype` will use at run-time.

⌋*(RS_MANI_00005, RS_MANI_00061)*

[constr_10049]{DRAFT} **Restriction for the applicability of `DiagnosticEventPortMapping`** ⌈If an `RPortPrototype` is referenced by a `DiagnosticEventPortMapping`, then the `RPortPrototype` shall be typed by a `DiagnosticEventInterface` **at the time when the diagnostic design is complete**.⌋*()*

The use case that supports the retrieval of information about a diagnostic event is depicted in Figure 4.6.

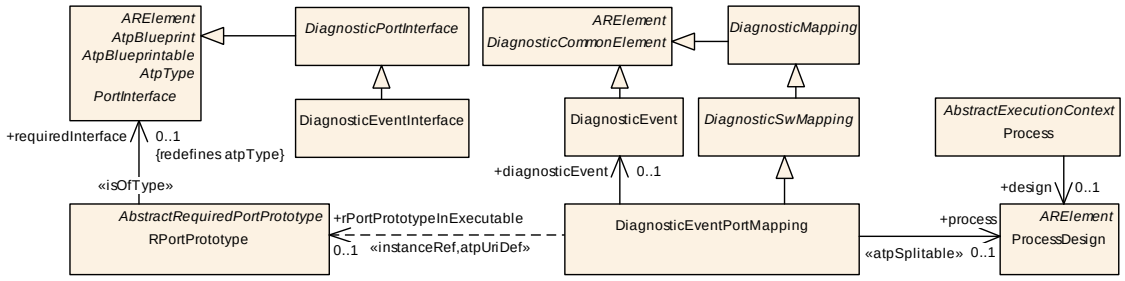


Figure 4.6: Modeling of `DiagnosticEventPortMapping` to retrieve information about a diagnostic event on the *AUTOSAR adaptive platform*

Class	**DiagnosticEventPortMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Defines to which SWC service ports the DiagnosticEvent is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to the DiagnosticEvent that is assigned to SWC service ports.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic EventPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 4.16: DiagnosticEventPortMapping

4.2.2.3 Multiple Diagnostic Event to Port Mapping

[TPS_MANI_01417]{DRAFT} **Retrieving the status of multiple `DiagnosticEvent`s to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈For the purpose of retrieving the status of a collection of diagnostic events on the *AUTOSAR adaptive platform*, the relation between the collection of `DiagnosticEvent`s and one `RPortPrototype` is created by using the `DiagnosticMultipleEventPortMapping` that refers to

- a collection of `DiagnosticEvent`s in the role `diagnosticEvent`,
- a `ProcessDesign` in the role `process`, and
- an `RPortPrototype` typed by a `DiagnosticEventInterface` in the role `rPortPrototypeInExecutable`. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself,

however, is required for constructing the corresponding `InstanceSpecifier` that the `PortPrototype` will use at run-time.

⌋(*RS_MANI_00005*, *RS_MANI_00061*)

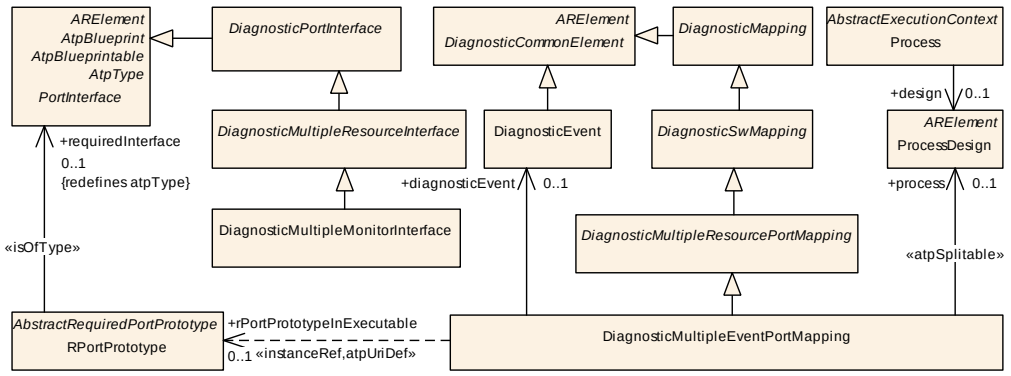


Figure 4.7: Modeling of `DiagnosticMultipleEventPortMapping` for retrieving the status of multiple diagnostic events on the *AUTOSAR adaptive platform*

Class	**DiagnosticMultipleEventPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service port that can handle a collection of event status the specific event is mapped. **Tags:** atp.recommendedPackage=DiagnosticPortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticMultipleResourcePortMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnosticEvent	DiagnosticEvent	0..1	ref	Reference to the DiagnosticEvent that is assigned to a SWC service port.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic MonitorMultipleEventPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 4.17: DiagnosticMultipleEventPortMapping

[constr_10437]{DRAFT} **Restriction for the applicability of `DiagnosticMultipleEventPortMapping`** ⌈If an `RPortPrototype` is referenced by a `DiagnosticMultipleEventPortMapping`, then the `RPortPrototype` shall be typed by a `DiagnosticMultipleEventInterface` **at the time when the diagnostic design is complete**.⌋()

4.2.2.4 Diagnostic Operation Cycle to Port Mapping

[TPS_MANI_01049]{DRAFT} **Mapping of `DiagnosticOperationCycle` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticOperationCycle` and one `RPortPrototype` is created by using the `DiagnosticOperationCyclePortMapping` that refers to

- a `DiagnosticOperationCycle` in the role `operationCycle`,
- a `ProcessDesign` in the role `process`, and
- an `RPortPrototype` typed by a `DiagnosticOperationCycleInterface` in the role `rPortPrototypeInExecutable`. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding `InstanceSpecifier` that the `PortPrototype` will use at run-time.

⌋*(RS_MANI_00005, RS_MANI_00061)*

[constr_10050]{DRAFT} **Restriction for the applicability of `DiagnosticOperationCyclePortMapping`** ⌈If an `RPortPrototype` is referenced by a `DiagnosticOperationCyclePortMapping`, then the `RPortPrototype` shall be typed by a `DiagnosticOperationCycleInterface` **at the time when the diagnostic design is complete**.⌋*()*

[constr_10051]{DRAFT} **Existence of reference from `DiagnosticOperationCyclePortMapping` to `DiagnosticOperationCycle`** ⌈Each `DiagnosticOperationCycle` shall only be referenced by exactly one `DiagnosticOperationCyclePortMapping` **at the time when the diagnostic design is complete**.⌋ *()*

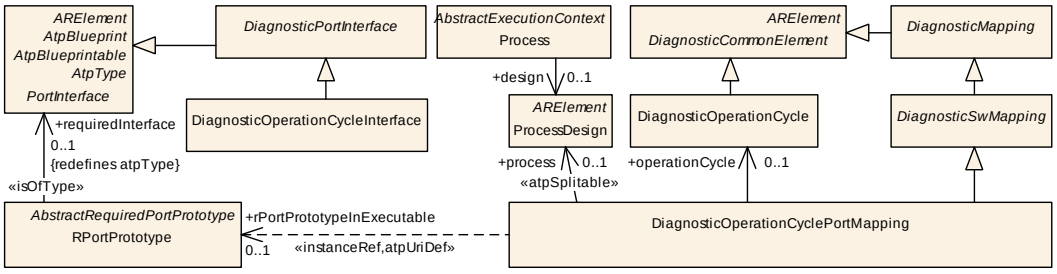


Figure 4.8: Modeling of `DiagnosticOperationCyclePortMapping` for the usage on the *AUTOSAR adaptive platform*

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

▽

△

Class	**DiagnosticOperationCycle**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
type	DiagnosticOperationCycleTypeEnum	0..1	attr	Operation cycles types for the Dem.

Table 4.18: DiagnosticOperationCycle

Enumeration	**DiagnosticOperationCycleTypeEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticOperationCycle
Note	Operation cycles types used to identify certain Operation cycles with a certain semantics.
Aggregated by	DiagnosticOperationCycle.type
Literal	***Description***
ignition	Ignition ON / OFF cycle **Tags:** atp.EnumerationLiteralIndex=0
obdDrivingCycle	OBD Driving cycle **Tags:** atp.EnumerationLiteralIndex=1
other	further operation cycle **Tags:** atp.EnumerationLiteralIndex=2
warmup	OBD Warm up cycle **Tags:** atp.EnumerationLiteralIndex=5

Table 4.19: DiagnosticOperationCycleTypeEnum

Class	**DiagnosticOperationCyclePortMapping**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Defines to which SWC service ports the DiagnosticOperationCycle is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
operationCycle	DiagnosticOperationCycle	0..1	ref	Reference to the DiagnosticOperationCycle that is assigned to SWC service ports.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototypeInExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic OperationCyclePortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeInExecutableInstanceRef

Table 4.20: DiagnosticOperationCyclePortMapping

4.2.2.5 Diagnostic Enable Condition to Port Mapping

[TPS_MANI_01050]{DRAFT} **Mapping of `DiagnosticEnableCondition` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticEnableCondition` and one `RPortPrototype`is created by using the `DiagnosticEnableCondition-PortMapping` that refers to

- a `DiagnosticEnableCondition` in the role `enableCondition`,
- a `ProcessDesign` in the role `process`, and
- an `RPortPrototype` typed by a `DiagnosticConditionInterface` in the role `rPortPrototypeInExecutable`. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding `InstanceSpecifier` that the `PortPrototype` will use at run-time.

⌋*(RS_MANI_00005, RS_MANI_00061)*

[constr_10052]{DRAFT} **Restriction for the applicability of `DiagnosticEnableConditionPortMapping`** ⌈If an `RPortPrototype` is referenced by a `DiagnosticEnableConditionPortMapping`, then the `RPortPrototype` shall be typed by a `DiagnosticConditionInterface` **`at the time when the diagnostic design is complete`**.⌋*()*

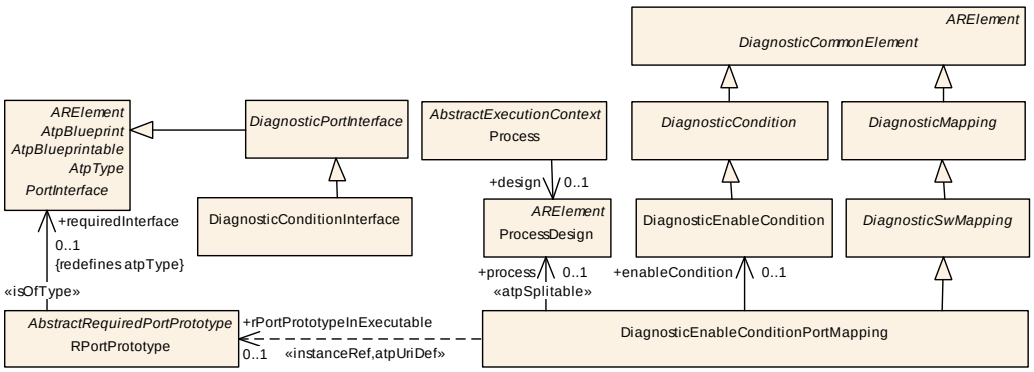


Figure 4.9: Modeling of `DiagnosticEnableConditionPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_10053]{DRAFT} **Existence of reference from `DiagnosticEnableConditionPortMapping` to `DiagnosticEnableCondition`** ⌈Each `DiagnosticEnableCondition` shall only be referenced by at most one `DiagnosticEnableConditionPortMapping` **`at the time when the diagnostic design is complete`**.⌋*()*

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

Table 4.21: DiagnosticEnableCondition

Class	DiagnosticEnableConditionPortMapping			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Defines to which SWC service ports the DiagnosticEnableCondition is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
enableCondition	DiagnosticEnable Condition	0..1	ref	Reference to the EnableCondition which is mapped to a SWC service port.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic EnableConditionPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 4.22: DiagnosticEnableConditionPortMapping

4.2.2.6 Diagnostic Clear Condition to Port Mapping

[TPS_MANI_01259]{DRAFT} **Mapping of `DiagnosticClearCondition` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticClearCondition` and one `RPortPrototype`is created by using the `DiagnosticClearConditionPortMapping` that refers to

- a `DiagnosticClearCondition` in the role `clearCondition`,
- a `ProcessDesign` in the role `process`, and
- an `RPortPrototype` typed by a `DiagnosticConditionInterface` in the role `rPortPrototypeInExecutable`. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference

itself, however, is required for constructing the corresponding `InstanceSpecifier` that the `PortPrototype` will use at run-time.

⌋(*RS_MANI_00005, RS_MANI_00061*)

[constr_10054]{DRAFT} **Restriction for the applicability of `DiagnosticClearConditionPortMapping`** ⌈If an `RPortPrototype` is referenced by a `DiagnosticClearConditionPortMapping`, then the `RPortPrototype` shall be typed by a `DiagnosticConditionInterface` **at the time when the diagnostic design is complete**.⌋()

[constr_10055]{DRAFT} **Existence of reference from `DiagnosticClearConditionPortMapping` to `DiagnosticClearCondition`** ⌈Each `DiagnosticClearCondition` shall only be referenced by at most one `DiagnosticClearConditionPortMapping` **at the time when the diagnostic design is complete**.⌋()

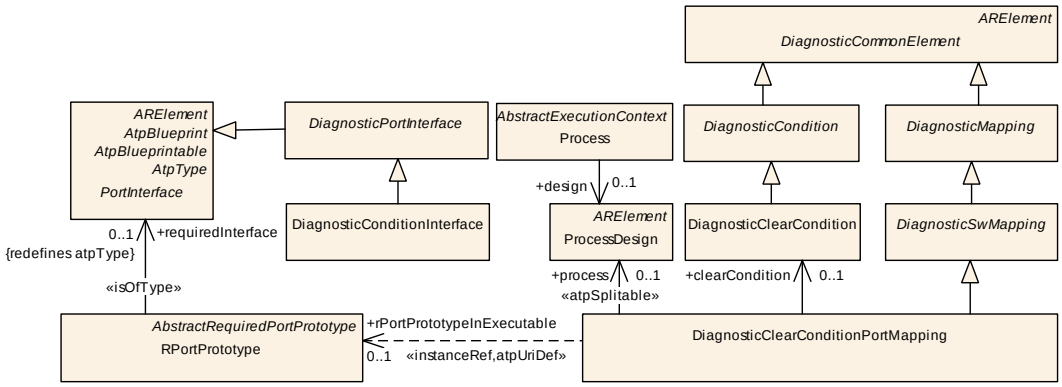


Figure 4.10: Modeling of `DiagnosticClearConditionPortMapping` for the usage on the *AUTOSAR adaptive platform*

Class	**DiagnosticClearConditionPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports the DiagnosticClearCondition is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clearCondition	DiagnosticClear Condition	0..1	ref	Reference to the ClearCondition which is mapped to a SWC service port.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process

▽

△

Class	**DiagnosticClearConditionPortMapping**			
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic ClearConditionMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 4.23: DiagnosticClearConditionPortMapping

[constr_10162]{DRAFT} **Multiplicity of reference in the role `DiagnosticClearConditionPortMapping.clearCondition`** ⌈For each `DiagnosticClearConditionPortMapping`, the reference in the role `clearCondition` shall exist **`at the time when the diagnostic design is complete`**.⌋*()*

4.2.2.7 Multiple Diagnostic Condition to Port Mapping

[TPS_MANI_01418]{DRAFT} **Mapping of multiple `DiagnosticCondition`s to a `PortPrototype` on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticCondition` and one `RPortPrototype`is created by using the `DiagnosticMultipleConditionPortMapping` that refers to

- a `DiagnosticCondition`s in the role `diagnosticCondition`,
- a `ProcessDesign` in the role `process`, and
- an `RPortPrototype` typed by a `DiagnosticConditionInterface` in the role `rPortPrototypeInExecutable`. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding `InstanceSpecifier` that the `PortPrototype` will use at run-time.

⌋*(RS_MANI_00005, RS_MANI_00061)*

Please note that the nature of the reference `DiagnosticMultipleConditionPortMapping.diagnosticCondition` allows for referencing a mixed collection of enable conditions and clear conditions, if applicable.

But since the concrete reference only points to the concrete sub-classes of `DiagnosticCondition`, the semantics of each individual reference can easily be determined.

Class	**DiagnosticMultipleConditionPortMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping
Note	Defines to which SWC service port that can handle a collection of diagostic conditions the specific condition is mapped. **Tags:** atp.recommendedPackage=DiagnosticPortMappings

▽

△

Class	**DiagnosticMultipleConditionPortMapping**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticMultipleResourcePortMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnostic Condition	DiagnosticCondition	0..1	ref	Reference to the DiagnosticCondition which is mapped to a SWC service port.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic ConditionPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 4.24: DiagnosticMultipleConditionPortMapping

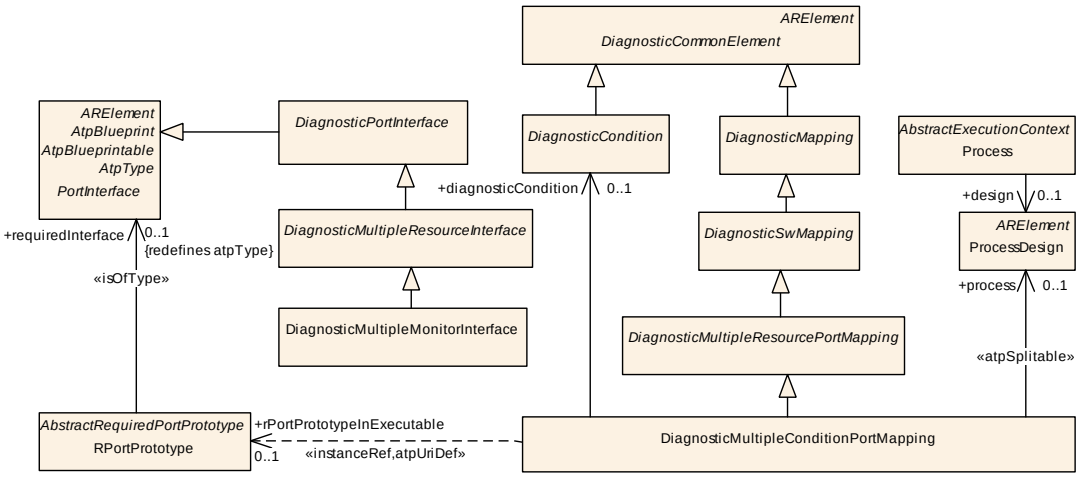


Figure 4.11: Modeling of `DiagnosticMultipleConditionPortMapping` for reporting the status of multiple diagnostic events on the *AUTOSAR adaptive platform*

[constr_10438]{DRAFT} **Restriction for the applicability of `DiagnosticMultipleConditionPortMapping`** ⌈If an `RPortPrototype` is referenced by a `DiagnosticMultipleConditionPortMapping`, then the `RPortPrototype` shall be typed by a `DiagnosticMultipleConditionInterface`.⌋*()*

4.2.2.8 Diagnostic Indicator to Port Mapping

[TPS_MANI_01260]{DRAFT} **Mapping of `DiagnosticIndicator` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticIndicator` and one `RPortPrototype`is created by using the `DiagnosticIndicatorPortMapping` that refers to

- a DiagnosticIndicator in the role indicator,
- a ProcessDesign in the role process, and
- an RPortPrototype typed by a DiagnosticIndicatorInterface in the role rPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype will use at run-time.

⌋(*RS_MANI_00005, RS_MANI_00061*)

[constr_10056]{DRAFT} **Restriction for the applicability of DiagnosticIndicatorPortMapping** ⌈If an RPortPrototype is referenced by a DiagnosticIndicatorPortMapping, then the RPortPrototype shall be typed by a DiagnosticIndicatorInterface **at the time when the diagnostic design is complete**.⌋()

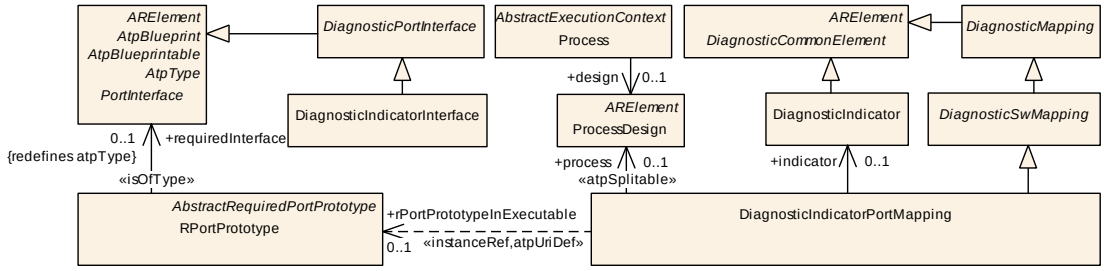


Figure 4.12: Modeling of DiagnosticIndicatorPortMapping for the usage on the *AUTOSAR adaptive platform*

Class	**DiagnosticIndicatorPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	Defines to which SWC service ports the DiagnosticIndicator is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
indicator	DiagnosticIndicator	0..1	ref	Reference to the DiagnosticIndicator which is mapped to a SWC service port.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic IndicatorMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 4.25: DiagnosticIndicatorPortMapping

[constr_10163]{DRAFT} **Multiplicity of reference in the role DiagnosticIndicatorPortMapping.indicator** ⌈For each DiagnosticIndicatorPortMapping, the reference in the role indicator shall exist **at the time when the diagnostic design is complete.**⌋*()*

4.2.2.9 Diagnostic Memory Destination to Port Mapping

[TPS_MANI_01261]{DRAFT} **Mapping of DiagnosticMemoryDestination to PortPrototype(s) on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a DiagnosticMemoryDestination and one RPortPrototypeis created by using the DiagnosticMemoryDestinationPortMapping that refers to

- a DiagnosticMemoryDestination in the role memoryDestination,
- a ProcessDesign in the role process, and
- an RPortPrototype typed by a DiagnosticDTCInformationInterface in the role rPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype will use at run-time.

⌋*(RS_MANI_00005, RS_MANI_00061)*

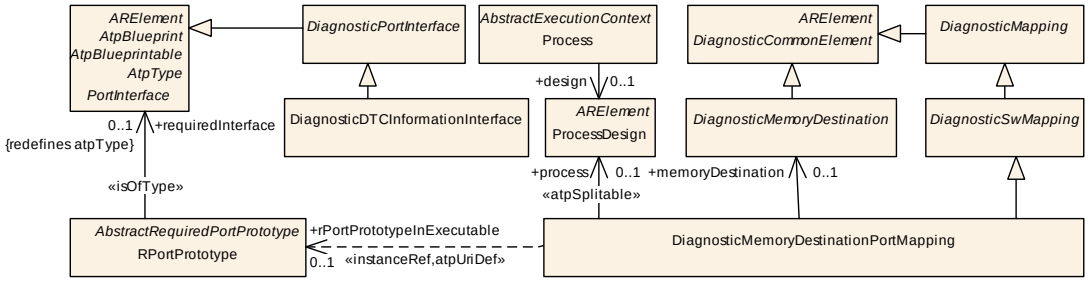


Figure 4.13: Modeling of DiagnosticMemoryDestinationPortMapping for the usage on the *AUTOSAR adaptive platform*

[constr_10057]{DRAFT} **Restriction for the applicability of DiagnosticMemoryDestinationPortMapping** ⌈If an RPortPrototype is referenced by a DiagnosticMemoryDestinationPortMapping, then the RPortPrototype shall be typed by a DiagnosticDTCInformationInterface **at the time when the diagnostic design is complete.**⌋*()*

Class	**DiagnosticMemoryDestinationPortMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping
Note	Defines to which SWC service ports the DiagnosticMemoryDestination. **Tags:** atp.recommendedPackage=DiagnosticMappings

▽

△

Class	**DiagnosticMemoryDestinationPortMapping**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
memory Destination	DiagnosticMemory Destination	0..1	ref	Reference to the MemoryDestination which is mapped to a SWC service port.
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic MemoryDestinationMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 4.26: DiagnosticMemoryDestinationPortMapping

[constr_10164]{DRAFT} **Multiplicity of reference in the role `DiagnosticMemoryDestinationPortMapping.memoryDestination`** ⌈For each `DiagnosticMemoryDestinationPortMapping`, the reference in the role `memoryDestination` shall exist **`at the time when the diagnostic design is complete`**.⌋*()*

Class	***DiagnosticMemoryDestination*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination			
Note	This abstract meta-class represents a possible memory destination for a diagnostic event.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticMemoryDestinationPrimary, DiagnosticMemoryDestinationUserDefined			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
agingRequires TestedCycle	Boolean	0..1	attr	Defines whether the aging cycle counter is processed every aging cycles or else only tested aging cycle are considered. If the attribute is set to TRUE: only tested aging cycle are considered for aging cycle counter. If the attribute is set to FALSE: aging cycle counter is processed every aging cycle. On the adaptive platform, the value of this attribute can be different for each DiagnosticMemoryDestination.
clearDtc Limitation	DiagnosticClearDtc LimitationEnum	0..1	attr	Defines the scope of the DEM_ClearDTC Api. On the adaptive platform, the value of this attribute can be different for each DiagnosticMemoryDestination.
dtcStatus AvailabilityMask	PositiveInteger	0..1	attr	Mask for the supported DTC status bits by the Dem.
event Displacement Strategy	DiagnosticEvent DisplacementStrategy Enum	0..1	attr	This attribute defines, whether support for event displacement is enabled or not, and which displacement strategy is followed.

▽

△

Class	***DiagnosticMemoryDestination*** (abstract)			
maxNumberOf EventEntries	PositiveInteger	0..1	attr	This attribute fixes the maximum number of event entries in the fault memory.
memoryEntry StorageTrigger	DiagnosticMemoryEntry StorageTriggerEnum	0..1	attr	Describes the trigger to allocate an event memory entry.
statusBit HandlingTest FailedSinceLast Clear	DiagnosticStatusBit HandlingTestFailed SinceLastClearEnum	0..1	attr	This attribute defines, whether the aging and displacement mechanism shall be applied to the "Test FailedSinceLastClear" status bits. On the adaptive platform, the value of this attribute can be different for each DiagnosticMemoryDestination.
statusBit StorageTest Failed	Boolean	0..1	attr	This parameter is used to activate/deactivate the permanent storage of the "TestFailed" status bits. true: storage activated false: storage deactivated On the adaptive platform, this attribute shall always be set to true.
typeOfFreeze FrameRecord Numeration	DiagnosticTypeOf FreezeFrameRecord NumerationEnum	0..1	attr	This attribute defines the type of assigning freeze frame record numbers for event-specific freeze frame records.

Table 4.27: DiagnosticMemoryDestination

Enumeration	**DiagnosticClearDtcLimitationEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination
Note	Scope of the DEM_ClearDTC Api.
Aggregated by	*DiagnosticMemoryDestination*.clearDtcLimitation
Literal	***Description***
allSupportedDtcs	DEM_ClearDtc API accepts all supported DTC values. **Tags:** atp.EnumerationLiteralIndex=0
clearAllDtcs	DEM_ClearDtc API accepts ClearAllDTCs only. **Tags:** atp.EnumerationLiteralIndex=1

Table 4.28: DiagnosticClearDtcLimitationEnum

Enumeration	**DiagnosticEventDisplacementStrategyEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination
Note	Defines the displacement strategy.
Aggregated by	*DiagnosticMemoryDestination*.eventDisplacementStrategy
Literal	***Description***
full	Event memory entry displacement is enabled, by consideration of priority active/passive status, and occurrence. **Tags:** atp.EnumerationLiteralIndex=0
none	Event memory entry displacement is disabled. **Tags:** atp.EnumerationLiteralIndex=1
prioOcc	Event memory entry displacement is enabled, by consideration of priority and occurrence (but without active/passive status). **Tags:** atp.EnumerationLiteralIndex=2

Table 4.29: DiagnosticEventDisplacementStrategyEnum

Enumeration	**DiagnosticMemoryEntryStorageTriggerEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination
Note	Trigger types to allocate an event memory entry.
Aggregated by	*DiagnosticMemoryDestination*.memoryEntryStorageTrigger
Literal	***Description***
confirmed	Status information of UDS DTC status bit 3 **Tags:** atp.EnumerationLiteralIndex=0
fdcThreshold	Threshold to allocate an event memory entry and to capture the Freeze Frame. **Tags:** atp.EnumerationLiteralIndex=1
testFailed	Status information of UDS DTC status bit 0. **Tags:** atp.EnumerationLiteralIndex=3

Table 4.30: DiagnosticMemoryEntryStorageTriggerEnum

Enumeration	**DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode
Note	This enumeration controls whether the aging and displacement mechanism shall be applied to the 'TestFailedSinceLastClear' status bits.
Aggregated by	*DiagnosticMemoryDestination*.statusBitHandlingTestFailedSinceLastClear
Literal	***Description***
statusBitAgingAnd Displacement	The "TestFailedSinceLastClear" status bits are reset to 0, if aging or displacement applies. **Tags:** atp.EnumerationLiteralIndex=0
statusBitNormal	Aging and displacement has no impact on the "TestFailedSinceLastClear" status bits. **Tags:** atp.EnumerationLiteralIndex=1

Table 4.31: DiagnosticStatusBitHandlingTestFailedSinceLastClearEnum

Enumeration	**DiagnosticTypeOfFreezeFrameRecordNumerationEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticMemoryDestination
Note	FreezeFrame record numeration type
Aggregated by	*DiagnosticMemoryDestination*.typeOfFreezeFrameRecordNumeration
Literal	***Description***
calculated	Freeze frame records will be numbered consecutive starting by 1 in their chronological order. **Tags:** atp.EnumerationLiteralIndex=0
configured	Freeze frame records will be numbered based on the given configuration in their chronological order. **Tags:** atp.EnumerationLiteralIndex=1

Table 4.32: DiagnosticTypeOfFreezeFrameRecordNumerationEnum

4.2.3 UDS-related Diagnostic Mapping

This section contains a documentation of `DiagnosticMapping`s in the context of UDS service configuration.

4.2.3.1 Diagnostic Security to Port Mapping

[TPS_MANI_01262]{DRAFT} **Mapping of `DiagnosticSecurityLevel` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticSecurityLevel` and one `PPortPrototype`is created by using the `DiagnosticSecurityLevelPortMapping` that refers to

- a `DiagnosticSecurityLevel` in the role `securityLevel`,
- a `ProcessDesign` in the role `process`, and
- an `PPortPrototype` typed by a `DiagnosticSecurityLevelInterface` in the role `pPortPrototypeInExecutable`. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding `InstanceSpecifier` that the `PortPrototype` on the application layer will use at run-time.

⌋*(RS_MANI_00005, RS_MANI_00061)*

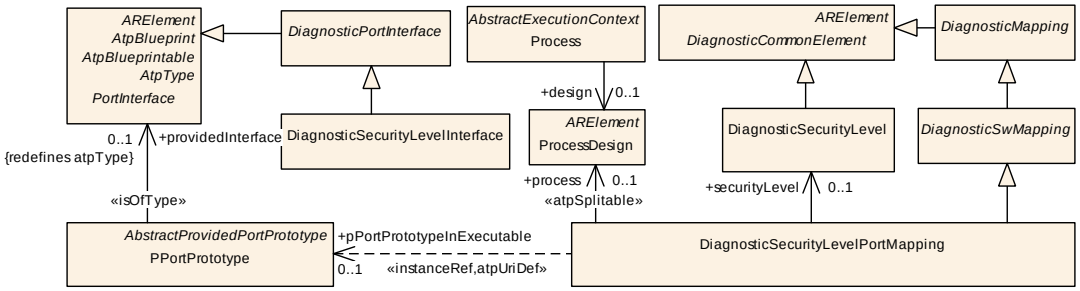


Figure 4.14: Modeling of `DiagnosticSecurityLevelPortMapping` for the usage on the *AUTOSAR adaptive platform*

[constr_10058]{DRAFT} **Restriction for the applicability of `DiagnosticSecurityLevelPortMapping`** ⌈If a `PPortPrototype` is referenced by a `DiagnosticSecurityLevelPortMapping`, then the `PPortPrototype` shall be typed by a `DiagnosticSecurityLevelInterface` **at the time when the diagnostic design is complete**.⌋*()*

[constr_10059]{DRAFT} **Existence of reference from `DiagnosticSecurityLevelPortMapping` to `DiagnosticSecurityLevel`** ⌈Each `DiagnosticSecurityLevel` shall only be referenced by exactly one `DiagnosticSecurityLevelPortMapping` **at the time when the diagnostic design is complete**.⌋*()*

Class	**DiagnosticSecurityLevelPortMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping
Note	Defines to which SWC service ports the DiagnosticSecurityLevel is mapped. **Tags:** atp.recommendedPackage=DiagnosticMappings

▽

△

Class	**DiagnosticSecurityLevelPortMapping**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SecurityLevelMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
securityLevel	DiagnosticSecurityLevel	0..1	ref	Reference to the SecurityLevel which is mapped to a SWC service port.

Table 4.33: DiagnosticSecurityLevelPortMapping

Class	**DiagnosticSecurityLevel**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
Note	This meta-class represents the ability to define a security level considered for diagnostic purposes. **Tags:** atp.recommendedPackage=DiagnosticSecurityLevels			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
accessData RecordSize	PositiveInteger	0..1	attr	This represents the size of the AccessDataRecord used in GetSeed. Unit:byte.
keySize	PositiveInteger	0..1	attr	This represents the size of the security key. Unit: byte.
numFailed SecurityAccess	PositiveInteger	0..1	attr	This represents the number of failed security accesses after which the delay time is activated.
securityDelay Time	TimeValue	0..1	attr	This represents the delay time after a failed security access. Unit: second.
seedSize	PositiveInteger	0..1	attr	This represents the size of the security seed. Unit: byte.

Table 4.34: DiagnosticSecurityLevel

4.2.3.2 Diagnostic Data Identifier to Port Mapping

The DM on the *AUTOSAR adaptive platform* has the ability to access entire `DiagnosticDataIdentifier`s at once. For supporting this ability, a dedicated mapping class named `DiagnosticDataPortMapping` is introduced.

[TPS_MANI_01263]{DRAFT} **Mapping of `DiagnosticDataIdentifier` or `DiagnosticDataElement` to `PortPrototype`(s) on the *AUTOSAR adaptive platform***

⌈On the *AUTOSAR adaptive platform*, the relation between a DiagnosticDataIdentifier resp. DiagnosticDataElement and one PPortPrototype is created by using the DiagnosticDataPortMapping that refers to either

- a DiagnosticDataIdentifier in the role diagnosticDataIdentifier or
- a DiagnosticDataElement in the role diagnosticDataElement.

⌋*(RS_MANI_00005, RS_MANI_00061)*

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*, *DiagnosticServiceMappingDiagTarget*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*DiagnosticAbstractParameter*.dataElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls the meaning of the value of the array size.
maxNumberOf Elements	PositiveInteger	0..1	attr	The existence of this attribute turns the data instance into an array of data. The attribute determines the size of the array in terms of how many elements the array can take.
scalingInfoSize	PositiveInteger	0..1	attr	Size in bytes of scaling information for the DiagnosticData Element if used with DiagnosticReadScalingDataBy Identifier
swDataDef Props	SwDataDefProps	0..1	aggr	This property allows to specify data definition properties in order to support the definition of e.g. computation formulae and data constraints. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=swDataDefProps

Table 4.35: DiagnosticDataElement

As depicted in Figure 4.15, DiagnosticDataPortMapping has the ability to handle access to **either** an entire DID **or** to just an element of a DID.

Therefore, the existence of [constr_10003] is required to enforce that just one of the references is actually used for any given DiagnosticDataPortMapping.

[constr_10003]{DRAFT} **Restriction for the existence of DiagnosticDataPortMapping.diagnosticDataIdentifier vs. DiagnosticDataPortMapping.diagnosticDataElement** ⌈For each DiagnosticDataPortMapping, **either** the reference in the role diagnosticDataIdentifier **or** diagnosticDataElement shall exist **at the time when the diagnostic design is complete**.⌋*()*

[constr_10060]{DRAFT} **PortInterface of PPortPrototype referenced by DiagnosticDataPortMapping** ⌈Any particular PPortPrototype that is referenced in the role DiagnosticDataPortMapping.pPortPrototypeInExecutable shall be typed by either of

- DiagnosticDataIdentifierInterface
- DiagnosticDataElementInterface

- DiagnosticDataIdentifierGenericInterface

This rule shall be imposed **at the time when the diagnostic design is complete**.⌋*()*

[constr_10061]{DRAFT} **Mapping to DiagnosticDataIdentifierInterface, DiagnosticDataElementInterface, or DiagnosticDataIdentifierGenericInterface** ⌈All PPortPrototypes typed by either

- DiagnosticDataIdentifierInterface
- DiagnosticDataElementInterface
- DiagnosticDataIdentifierGenericInterface

shall **only** be referenced by a DiagnosticDataPortMapping. No other subclass of DiagnosticSwMapping is eligible for this purpose.

This rule shall be imposed **at the time when the diagnostic design is complete**.⌋*()*

[TPS_MANI_01347]{DRAFT} **Definition of a DiagnosticDataElement used in the context of a DID obtained from a PPortPrototype typed by a DiagnosticDataElementInterface** ⌈If a DiagnosticDataElement is aggregated by a DiagnosticParameter that in turn is aggregated by a DiagnosticDataIdentifier and the DiagnosticDataElement is also referenced by a DiagnosticDataPortMapping then the referenced DiagnosticDataElement (by way of SwDataDefProps) shall refer to a SwBaseType with attribute baseTypeSize set to either of the values

- 8,
- 16, or
- 32

The usage of attribute baseTypeEncoding is regulated by [constr_10483].

In the case addressed by this spec item, it is only possible to define the DiagnosticDataElement as a scalar or an array (see [TPS_DEXT_01001], [TPS_DEXT_01002]). The ability to define a Variable-Size Array shall only be used for the last element of the DID.⌋*(RS_MANI_00061)*

[TPS_MANI_01348]{DRAFT} **Definition of a DiagnosticDataElement used in the context of a DID obtained from a PPortPrototype typed by a DiagnosticDataIdentifierInterface** ⌈If a DiagnosticDataIdentifier is referenced by a DiagnosticDataPortMapping then the aggregated DiagnosticDataIdentifier.dataElement.dataElement (by way of SwDataDefProps) shall refer to a SwBaseType with attribute baseTypeSize set to either

- 8,
- 16, or

- 32

The usage of attribute `baseTypeEncoding` is regulated by [constr_10483].

In the case addressed by this spec item, it is only possible to define the `DiagnosticDataElement` as a scalar or an array (see [TPS_DEXT_01001], [TPS_DEXT_01002]). The ability to define a `Variable-Size Array` shall only be used for the last element of the DID.⌋*(RS_MANI_00061)*

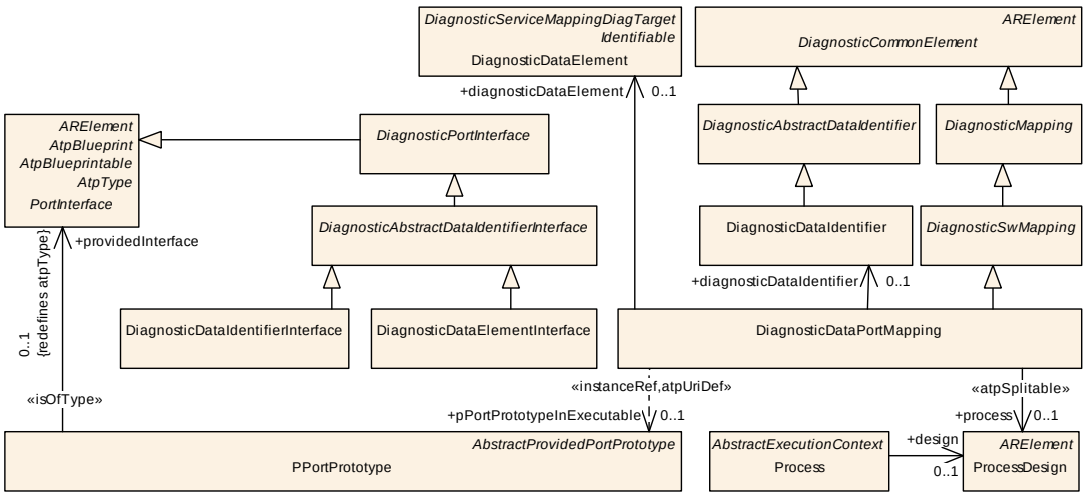


Figure 4.15: Modeling of `DiagnosticDataPortMapping` for the usage on the *AUTOSAR adaptive platform*

Class	**DiagnosticDataPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This meta-class provides the ability to define a diagnostic access to an entire DID. **Tags:** atp.recommendedPackage=DiagnosticServiceMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnosticData Element	DiagnosticDataElement	0..1	ref	This reference represents the applicable DiagnosticData Element.
diagnosticData Identifier	DiagnosticDataIdentifier	0..1	ref	This reference represents the applicable DiagnosticData Identfiier.
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This reference identifies the applicable PPortPrototype from which that data is obtained. The reference has the ability to point into the component hierarchy (under possible consideration of the rootSoftwareComposition). **Stereotypes:** atpUriDef **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process

Table 4.36: DiagnosticDataPortMapping

[constr_10165]{DRAFT} **Multiplicity of reference in the role DiagnosticDataPortMapping.process** ⌈For each DiagnosticDataPortMapping, the reference in the role process shall exist **at the time when the diagnostic design is complete**.⌋*()*

[constr_10490]{DRAFT} **DiagnosticDataElement shall be directly or indirectly the target of a reference owned by a subclass of DiagnosticMapping** ⌈Each DiagnosticDataElement shall be directly or indirectly considered by a DiagnosticMapping. Therefore, exactly one of the following conditions shall be fulfilled:

- A DiagnosticProvidedDataMapping exists that refers to the respective DiagnosticDataElement in the role dataElement.
- A DiagnosticDataPortMapping exists that refers to the respective DiagnosticDataElement in the role diagnosticDataElement.
- A DiagnosticDataPortMapping exists that refers in the role diagnosticDataIdentifier to the DiagnosticDataIdentifier that aggregates the respective DiagnosticDataElement in the role dataElement.dataElement.

This rule shall be imposed **at the time when the diagnostic design is complete**.⌋*()*

4.2.3.3 Diagnostic Generic UDS Service Handler to Port Mapping

It is possible to associate a collection of UDS services to a given PPortPrototype with the intention that the PPortPrototype can handle the associated services.

By creating a dedicated association between generic UDS handlers and the services they can take it is possible to use multiple generic UDS handlers and let each take only the associated services.

Technically, a possible alternative to the documented modeling of generic UDS handling would be to avoid the mapping at all and foresee the existence of a catch-all generic UDS handler.

This, to a large extent, contradicts the idea of having modular software installations on the basis of the definition of SoftwareClusters (see section 15.3).

4.2.3.3.1 Diagnostic Generic Mapping

[TPS_MANI_01326]{DRAFT} **Generic Mapping to a DiagnosticServiceInstance on the AUTOSAR Adaptive Platform** ⌈On the AUTOSAR adaptive platform, the relation between a DiagnosticServiceInstance and one PPortPrototype for select diagnostic services is created by using the meta-class DiagnosticServiceGenericMapping that refers to

- a DiagnosticServiceInstance in the role diagnosticServiceInstance,
- the PPortPrototype in the role pPortPrototypeInExecutable, and
- a ProcessDesign in the role process.

⌋(*RS_MANI_00005*, *RS_MANI_00061*)

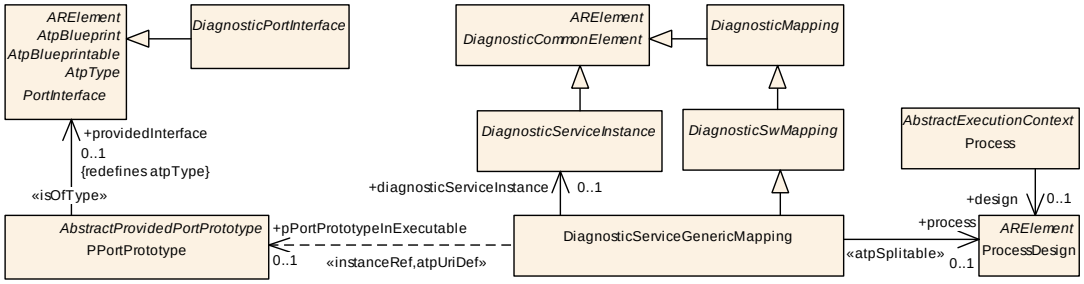


Figure 4.16: Modeling of the diagnostic generic mapping

[constr_10062]{DRAFT} **DiagnosticServiceInstances that can be mapped by a DiagnosticServiceGenericMapping** ⌈DiagnosticServiceGenericMapping shall only be used for the following list of DiagnosticServiceInstances:

- DiagnosticEcuReset
- DiagnosticComControl
- DiagnosticRoutineControl
- DiagnosticCustomServiceInstance
- DiagnosticRequestUpload
- DiagnosticRequestDownload
- DiagnosticRequestFileTransfer

This rule shall be imposed **at the time when the diagnostic design is complete**.⌋()

Please note that services DiagnosticDataTransfer and DiagnosticTransferExit are mapped dynamically according to the UDS request sequence for upload and download.

Class	**DiagnosticEcuReset**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EcuReset
Note	This represents an instance of the "ECU Reset" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticEcuResets
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*
Aggregated by	ARPackage.element

▽

△

Class	**DiagnosticEcuReset**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
customSub Function Number	PositiveInteger	0..1	attr	This attribute shall be used to define a custom sub-function number if none of the standardized values of category shall be used.
ecuResetClass	DiagnosticEcuReset Class	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticEcuReset in the given context.

Table 4.37: DiagnosticEcuReset

Class	**DiagnosticEcuResetClass**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::EcuReset			
Note	This meta-class contains attributes shared by all instances of the "Ecu Reset" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticEcuResets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
respondTo Reset	DiagnosticResponseTo EcuResetEnum	0..1	attr	This attribute defines whether the response to the Ecu Reset service shall be transmitted before or after the actual reset.

Table 4.38: DiagnosticEcuResetClass

Class	**DiagnosticComControl**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommunicationControl			
Note	This represents an instance of the "Communication Control" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticCommunicationControls			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
comControl Class	DiagnosticComControl Class	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticComControl in the given context.
customSub Function Number	PositiveInteger	0..1	attr	This attribute shall be used to define a custom sub-function number if none of the standardized values of category shall be used.

Table 4.39: DiagnosticComControl

Class	**DiagnosticComControlClass**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommunicationControl			
Note	This meta-class contains attributes shared by all instances of the "Communication Control" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticCommunicationControls			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allChannels	CommunicationCluster	*	ref	This reference represents the semantics that all available channels shall be affected. It is still necessary to refer to individual CommunicatuionClusters because there could be private CommunicationClusters in the System Extract that are not subject to the service "communication control". By referring to the applicable CommunicationClusters it can be made sure that only the affected Communication Clusters are accessed.
allPhysical Channels	EthernetPhysical Channel	*	ref	This reference represents the semantics that all available channels shall be affected. It is still necessary to refer to individual EthernetPhysicalChannels because there could be private VLANs (and thus private EthernetPhysical Channels) in the System Extract that are not subject to the service "communication control". By referring to the applicable EthernetPhysicalChannels it can be made sure that only the affected EthernetPhysical Channels are accessed. **Tags:** xml.namePlural=ALL-PHYSICAL-CHANNELS
specificChannel	DiagnosticComControl SpecificChannel	*	aggr	This represents the ability to add additional attributes to the case that only specific channels are supposed to be considered,
subNode Channel	DiagnosticComControl SubNodeChannel	*	aggr	This attribute represents the ability to add further attributes to the definition of a specific sub-node channel that is subject to the diagnostic service "communication control".

Table 4.40: DiagnosticComControlClass

Class	**DiagnosticRoutineControl**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::RoutineControl			
Note	This represents an instance of the "Routine Control" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticRoutineControls			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
routine	DiagnosticRoutine	0..1	ref	This refers to the applicable DiagnosticRoutine.
routineControl Class	DiagnosticRoutine ControlClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticRoutineControl in the given context.

Table 4.41: DiagnosticRoutineControl

Class	DiagnosticRoutineControlClass			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::RoutineControl			
Note	This meta-class contains attributes shared by all instances of the "Routine Control" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticRoutineControls			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.42: DiagnosticRoutineControlClass

Class	DiagnosticCustomServiceInstance			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CustomServiceInstance			
Note	This meta-class has the ability to define an instance of a custom diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticCustomInstances			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
customService Class	DiagnosticCustom ServiceClass	0..1	ref	Reference to the corresponding DiagnosticCustom ServiceClass.

Table 4.43: DiagnosticCustomServiceInstance

Class	DiagnosticCustomServiceClass			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommonService			
Note	This represents the ability to define a custom diagnostic service class and assign an ID to it. Further configuration is not foreseen from the point of view of the diagnostic extract and consequently needs to be done on the level of ECUC. **Tags:** atp.recommendedPackage=DiagnosticCustomServiceClasses			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
customService Id	PositiveInteger	0..1	attr	This attribute may only be used for the definition of custom services. The values shall not overlap with existing standardized service IDs.

Table 4.44: DiagnosticCustomServiceClass

Class	DiagnosticRequestUpload
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::MemoryByAddress
Note	This represents an instance of the "Request Upload" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticMemoryByAdresss
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMemory AddressableRangeAccess*, *DiagnosticMemoryByAddress*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*
Aggregated by	ARPackage.element

▽

△

Class	DiagnosticRequestUpload			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
requestUpload Class	DiagnosticRequest UploadClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticRequestUpload in the given context.

Table 4.45: DiagnosticRequestUpload

Class	DiagnosticRequestUploadClass			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::MemoryByAddress			
Note	This meta-class contains attributes shared by all instances of the "Request Upload" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticMemoryByAdresss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.46: DiagnosticRequestUploadClass

Class	DiagnosticRequestDownload			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::MemoryByAddress			
Note	This represents an instance of the "Request Download" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticMemoryByAdresss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMemory AddressableRangeAccess*, *DiagnosticMemoryByAddress*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
request DownloadClass	DiagnosticRequest DownloadClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticRequestDownload in the given context.

Table 4.47: DiagnosticRequestDownload

Class	DiagnosticRequestDownloadClass			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::MemoryByAddress			
Note	This meta-class contains attributes shared by all instances of the "Request Download" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticMemoryByAdresss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.48: DiagnosticRequestDownloadClass

Class	DiagnosticServiceGenericMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This meta-class represents the ability to implement a generic generic mapping for select diagnostics services on the adaptive platform. **Tags:** atp.recommendedPackage=DiagnosticServiceMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
diagnostic ServiceInstance	DiagnosticService Instance	0..1	ref	Reference to the ServiceInstance mapped to a SWC service port.
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic ServiceGenericMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process

Table 4.49: DiagnosticServiceGenericMapping

Class	DiagnosticRequestFileTransfer			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::RequestFileTransfer			
Note	This diagnostic service instance implements the UDS service 0x38. **Tags:** atp.recommendedPackage=DiagnosticRequestFileTransfers			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
requestFile TransferClass	DiagnosticRequestFile TransferClass	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticRequestFile Transfer in the given context.

Table 4.50: DiagnosticRequestFileTransfer

Class	DiagnosticRequestFileTransferClass			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::RequestFileTransfer			
Note	This meta-class contains attributes shared by all instances of the "Request File transfer" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticRequestFileTransfers			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.51: DiagnosticRequestFileTransferClass

4.2.3.4 Diagnostic Service Validation Mapping

[TPS_MANI_01352]{DRAFT} **Definition of DiagnosticServiceValidationMapping** ⌈On the *AUTOSAR adaptive platform*, meta-class DiagnosticServiceValidationMapping exists to indicate that the referenced PPortPrototype is used for the execution of a manufacturer-specific or supplier-specific check prior to the execution of a diagnostic service.

The number of PPortPrototypes that implement such a checking routine is arbitrary and the order in which the individual checks are executed is subject to a decision made on deployment level.

This is the main reason why the DiagnosticServiceValidationMapping (beside the reference to the ProcessDesign) has no reference into another design object in the context of the DEXT.⌋*(RS_MANI_00005, RS_MANI_00061)*

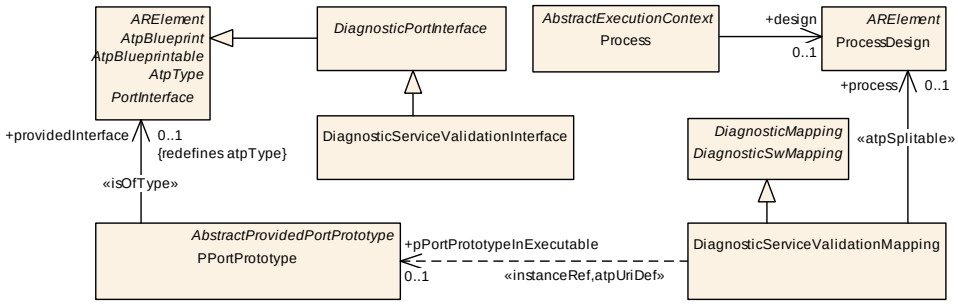


Figure 4.17: Modeling of the diagnostic validation mapping

[constr_10063]{DRAFT} **Possible values for DiagnosticServiceValidationMapping.category** ⌈The value of attribute DiagnosticServiceValidationMapping.category is restricted to the following values:

MANUFACTURER_VALIDATION The enclosing DiagnosticServiceValidationMapping represents a validation defined by the manufacturer.

SUPPLIER_VALIDATION The enclosing DiagnosticServiceValidationMapping represents a validation defined by the supplier.

This rule shall be imposed **at the time when the diagnostic design is complete**.⌋*()*

[constr_10064]{DRAFT} **Existence of DiagnosticServiceValidationMapping.pPortPrototypeInExecutable** ⌈A PPortPrototype referenced in the role DiagnosticServiceValidationMapping.pPortPrototypeInExecutable shall be typed by a DiagnosticServiceValidationInterface **at the time when the diagnostic design is complete**.⌋*()*

Please note that the final decision about the order in which the validations are executed is decided in the deployment phase, see section 15.3.2.3.

Class	DiagnosticServiceValidationMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This meta-class provides the ability to specify manufacturer/supplier checks to be executed before diagnostic services can be processed. **Tags:** atp.recommendedPackage=DiagnosticPortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This mapping identifies a PortPrototype typed by a DiagnosticValidationInterface in which a manufacturer/ supplier-specific check is executed. **Stereotypes:** atpUriDef **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process

Table 4.52: DiagnosticServiceValidationMapping

4.2.3.5 Diagnostic Authentication Mapping

[TPS_MANI_01360]{DRAFT} **Creation of two diagnostic mappings the fulfill different roles in the context of authenticating a diagnostic client** ⌈The support for the diagnostic service Authentication (0x29) on the *AUTOSAR adaptive platform* requires the creation of two diagnostic mappings the fulfill different roles in the context of authenticating a diagnostic client:

- The meta-class `DiagnosticAuthenticationPortMapping` (described in this section) is used to configure the forwarding of the request for authentication from the DM to some application software that acts as an authentication manager.
- The meta-class `DiagnosticExternalAuthenticationPortMapping` is used to convey the authentication state of a diagnostic client to the diagnostic server instance of DM.

⌋(*RS_MANI_00005*, *RS_MANI_00061*)

Class	DiagnosticAuthenticationPortMapping
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping
Note	This mapping class identifies the PortPrototype in the application software that handles the client authentication. **Tags:** atp.recommendedPackage=DiagnosticPortMapppings
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	**DiagnosticAuthenticationPortMapping**			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnostic Authentication	Diagnostic Authentication	0..1	ref	Reference to the DiagnosticAuthentication that is assigned to a SWC service port.
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic AuthenticationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process

Table 4.53: DiagnosticAuthenticationPortMapping

Please note that meta-class `DiagnosticExternalAuthenticationPortMapping` is described in section 4.2.3.6.

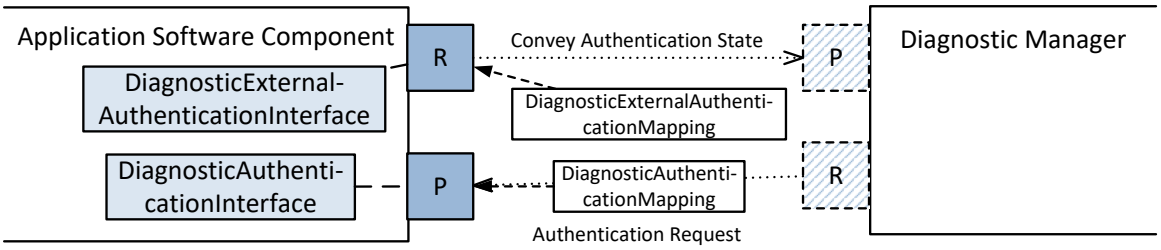


Figure 4.18: Example data exchange for diagnostic authentication

The two different roles mentioned in [TPS_MANI_01360] for the authentication of a diagnostic client are depicted in Figure 4.18.

[TPS_MANI_01361]{DRAFT} **Support the authentication request of a diagnostic client** ⌈For the purpose of supporting the authentication request of a diagnostic client on the *AUTOSAR adaptive platform*, the relation between a `DiagnosticAuthentication` and one `PPortPrototype` is created by using the `DiagnosticAuthenticationPortMapping` that refers to

- a `DiagnosticAuthentication` in the role `diagnosticAuthentication`,
- a `ProcessDesign` in the role `process`, and
- a `PPortPrototype` typed by a `DiagnosticAuthenticationInterface` in the role `pPortPrototypeInExecutable`. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding `InstanceSpecifier` that the `PortPrototype` will use at run-time.

⌋(*RS_MANI_00005*, *RS_MANI_00061*)

[constr_10093]{DRAFT} **Existence of reference from `DiagnosticAuthenticationPortMapping` to `DiagnosticAuthentication`** ⌈Each `DiagnosticAuthentication` shall only be referenced by exactly one `DiagnosticAuthenticationPortMapping` **at the time when the diagnostic design is complete**.⌋()

Class	***DiagnosticAuthentication*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::Authentication			
Note	This meta-class represents the ability to configure the usage of the UDS service Authentication in the Diagnostic extract.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticAuthTransmitCertificate, DiagnosticAuthenticationConfiguration, DiagnosticDeAuthentication, DiagnosticProofOfOwnership, DiagnosticVerifyCertificateBidirectional, DiagnosticVerifyCertificate Unidirectional			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authentication Class	Diagnostic AuthenticationClass	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 DiagnosticService Instance. The subclasses that affected by this pattern implement references to the applicable "class"-role that substantiate this abstract reference.

Table 4.54: DiagnosticAuthentication

Class	**DiagnosticAuthenticationClass**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::Authentication			
Note	This meta-class contains configuration shared by all instances of the Authentication diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticAuthentications			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.55: DiagnosticAuthenticationClass

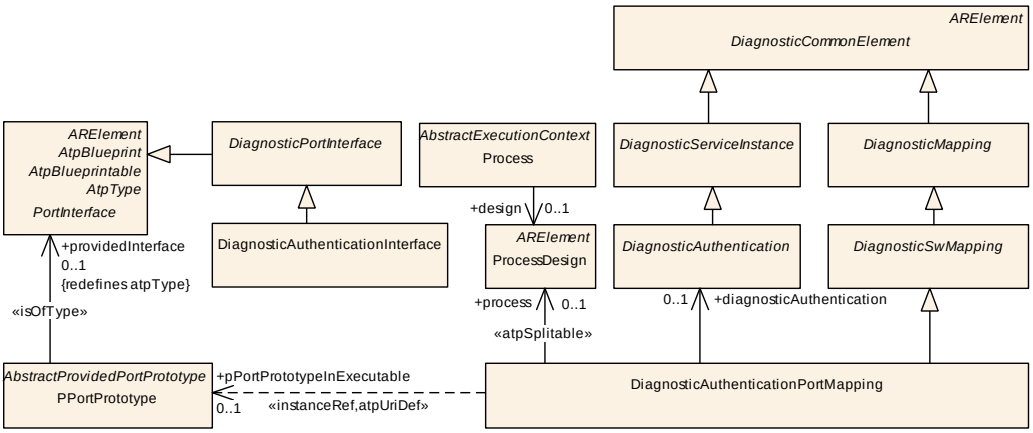


Figure 4.19: Modeling of `DiagnosticAuthenticationPortMapping` to forward the authentication request of a diagnostic client to an authentication manager

[constr_10092]{DRAFT} **Restriction for the applicability of `DiagnosticAuthenticationPortMapping`** ⌈If a `PPortPrototype` is referenced by a `DiagnosticAuthenticationPortMapping`, then the `PPortPrototype` shall be typed by a `DiagnosticAuthenticationInterface` **at the time when the diagnostic design is complete**.⌋*()*

4.2.3.6 Diagnostic External Authentication Mapping

[TPS_MANI_01362]{DRAFT} **Convey the Authentication state of a diagnostic client to the diagnostic server instance** ⌈For the purpose of conveying the authentication state of a diagnostic client to the diagnostic server instance of DM of a diagnostic client on the *AUTOSAR adaptive platform*, the relation between a `ProcessDesign` and one `RPortPrototype` is created by using the `DiagnosticExternalAuthenticationPortMapping` that refers to

- a `ProcessDesign` in the role process, and
- an `RPortPrototype` typed by a `DiagnosticExternalAuthenticationInterface` in the role `rPortPrototypeInExecutable`. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding `InstanceSpecifier` that the `PortPrototype` will use at run-time.

There is no model element in the context of the `Diagnostic Extract` to which the `DiagnosticExternalAuthenticationPortMapping` could reliably refer to (external authentication is not bound to the existence of UDS service 0x29).

This is the reason why the `DiagnosticExternalAuthenticationPortMapping` is restricted to references to `RPortPrototype` and `ProcessDesign`.⌋*(RS_MANI_00005, RS_MANI_00061)*

[constr_10094]{DRAFT} **Restriction for the applicability of `DiagnosticExternalAuthenticationPortMapping`** ⌈If an `RPortPrototype` is referenced by a `DiagnosticExternalAuthenticationPortMapping`, then the `RPortPrototype` shall be typed by a `DiagnosticExternalAuthenticationInterface` **at the time when the diagnostic design is complete**.⌋*()*

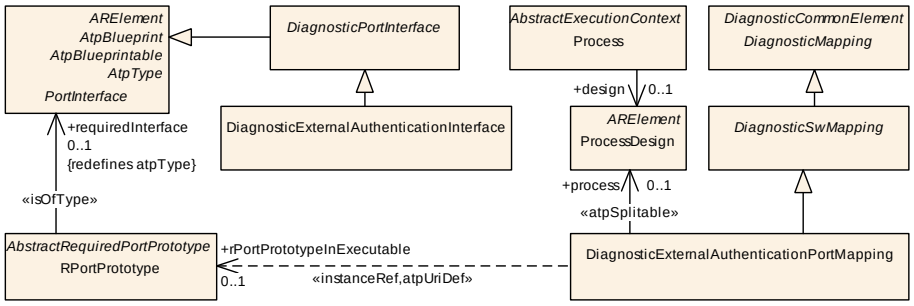


Figure 4.20: Modeling of `DiagnosticExternalAuthenticationPortMapping` to transmit the response to an authentication request back to the DM

Class	**DiagnosticExternalAuthenticationPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the external authentication. **Tags:** atp.recommendedPackage=DiagnosticPortMapppings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process
rPortPrototype InExecutable	RPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic ClientAuthenticationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 4.56: DiagnosticExternalAuthenticationPortMapping

4.2.4 SOVD-related Diagnostic Mapping

This chapter contains a documentation of some `DiagnosticMapping`s that are specific for the context of Service-Oriented Vehicle Diagnostics (SOVD).

4.2.4.1 Diagnostic SOVD Proximity Challenge Mapping

[TPS_MANI_01424]{DRAFT} **Semantics of `DiagnosticSovdProximityChallengePortMapping`** ⌈On the *AUTOSAR adaptive platform*, the relation between a `PPortPrototype` and a `ProcessDesign` is created by using the `DiagnosticSovdProximityChallengePortMapping` that refers to

- a `ProcessDesign` in the role `process`, and
- an `PPortPrototype` typed by a `DiagnosticSovdProximityChallengeInterface` in the role `pPortPrototypeInExecutable`. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding `InstanceSpecifier` that the `PortPrototype` on the application layer will use at run-time.

⌋*(RS_MANI_00005, RS_MANI_00061)*

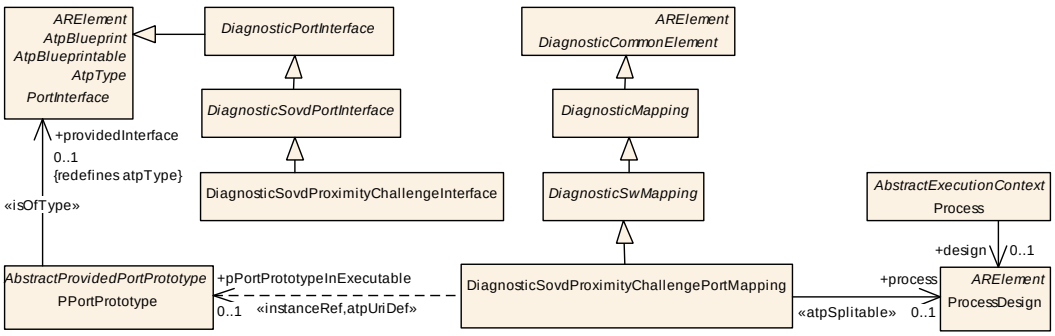


Figure 4.21: Modeling of the `DiagnosticSovdProximityChallengePortMapping`

[constr_10442]{DRAFT} **Restriction for the applicability of `DiagnosticSovdProximityChallengePortMapping`** ⌈If a `PPortPrototype` is referenced by a `DiagnosticSovdProximityChallengePortMapping` in the role `pPortPrototypeInExecutable`, then the `PPortPrototype` shall be typed by a `DiagnosticSovdProximityChallengeInterface` **at the time when the diagnostic design is complete**.⌋*()*

[constr_10443]{DRAFT} **Restriction for the applicability of `DiagnosticSovdProximityChallengeInterface`** ⌈If a `PPortPrototype` is typed by a `DiagnosticSovdProximityChallengeInterface`, then the `PortPrototype` shall only be referenced in the role `pPortPrototypeInExecutable` by a `DiagnosticSovdProximityChallengePortMapping` **at the time when the diagnostic design is complete**.⌋*()*

Only one implementation of an SOVD proximity challenge is allowed in the context of a `DiagnosticContributionSet`. This restriction is formalized by means of [constr_10444].

[constr_10444]{DRAFT} **Existence of `DiagnosticSovdProximityChallengePortMapping`** ⌈Each `DiagnosticContributionSet` shall only reference at most

one DiagnosticSovdProximityChallengePortMapping in the role element **at the time when the diagnostic design is complete.**⌋()

Class	**DiagnosticSovdProximityChallengePortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticSovdMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the SOVD proximity challenge. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdProximityChallengePortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=candidate

Table 4.57: DiagnosticSovdProximityChallengePortMapping

[constr_10445]{DRAFT} **Existence of the reference in the role DiagnosticSovdProximityChallengePortMapping.pPortPrototypeInExecutable** ⌈For each DiagnosticSovdProximityChallengePortMapping, the reference in the role DiagnosticSovdProximityChallengePortMapping.pPortPrototypeInExecutable shall exist **at the time when the diagnostic design is complete.**⌋()

[constr_10446]{DRAFT} **Existence of the reference in the role DiagnosticSovdProximityChallengePortMapping.process** ⌈For each DiagnosticSovdProximityChallengePortMapping, the reference in the role DiagnosticSovdProximityChallengePortMapping.process shall exist **at the time when the diagnostic design is complete.**⌋()

4.2.4.2 Diagnostic SOVD Authorization Port Mapping

[TPS_MANI_01425]{DRAFT} **Semantics of DiagnosticSovdAuthorizationPortMapping** ⌈On the *AUTOSAR adaptive platform*, the relation between a PPortPrototype and a ProcessDesign is created by using the DiagnosticSovdAuthorizationPortMapping that refers to

- a ProcessDesign in the role process, and

- an PPortPrototype typed by a DiagnosticSovdAuthorizationInterface in the role pPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype on the application layer will use at run-time.

⌋(*RS_MANI_00005, RS_MANI_00061*)

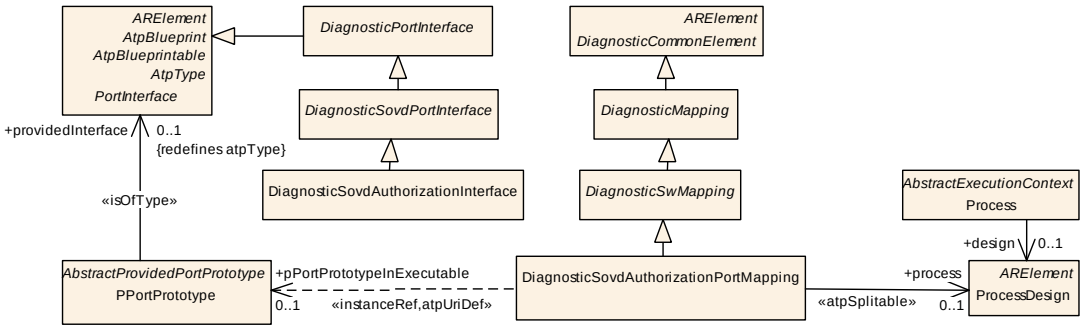


Figure 4.22: Modeling of the DiagnosticSovdAuthorizationPortMapping

[constr_10447]{DRAFT} **Restriction for the applicability of DiagnosticSovdAuthorizationPortMapping** ⌈If a PPortPrototype is referenced in the role pPortPrototypeInExecutable by a DiagnosticSovdAuthorizationPortMapping, then the PPortPrototype shall be typed by a DiagnosticSovdAuthorizationInterface **at the time when the diagnostic design is complete**.⌋*()*

[constr_10448]{DRAFT} **Restriction for the applicability of DiagnosticSovdAuthorizationInterface** ⌈If a PPortPrototype is typed by a DiagnosticSovdAuthorizationInterface, then the PortPrototype shall only be referenced in the role pPortPrototypeInExecutable by a DiagnosticSovdAuthorizationPortMapping **at the time when the diagnostic design is complete**.⌋*()*

Only one implementation of an SOVD authorization is allowed in the context of a DiagnosticContributionSet. This restriction is formalized by means of [constr_10444].

[constr_10449]{DRAFT} **Existence of DiagnosticSovdAuthorizationPortMapping** ⌈Each DiagnosticContributionSet shall only reference in the role element at most one DiagnosticSovdAuthorizationPortMapping **at the time when the diagnostic design is complete**.⌋*()*

Class	**DiagnosticSovdAuthorizationPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticSovdMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the SOVD authorization. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdAuthorizationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=candidate

Table 4.58: DiagnosticSovdAuthorizationPortMapping

[constr_10450]{DRAFT} **Existence of the reference in the role `DiagnosticSovdAuthorizationPortMapping.pPortPrototypeInExecutable`** ⌈For each `DiagnosticSovdAuthorizationPortMapping`, the reference in the role `DiagnosticSovdAuthorizationPortMapping.pPortPrototypeInExecutable` shall exist **`at the time when the diagnostic design is complete`**.⌋*()*

[constr_10451]{DRAFT} **Existence of the reference in the role `DiagnosticSovdAuthorizationPortMapping.process`** ⌈For each `DiagnosticSovdAuthorizationPortMapping`, the reference in the role `DiagnosticSovdAuthorizationPortMapping.process` shall exist **`at the time when the diagnostic design is complete`**.⌋*()*

4.2.4.3 Diagnostic SOVD Bulk Data Port Mapping

[TPS_MANI_01426]{DRAFT} **Mapping of `DiagnosticSovdBulkData` to `PortPrototype` on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticSovdBulkData` and one `PPortPrototype`is created by using the `DiagnosticSovdBulkDataPortMapping` that refers to

- a `DiagnosticSovdBulkData` in the role `serviceInstance`,
- a `ProcessDesign` in the role `process`, and

- an PPortPrototype typed by a DiagnosticSovdBulkDataInterface in the role pPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype on the application layer will use at run-time.

⌋(*RS_MANI_00005*, *RS_MANI_00061*)

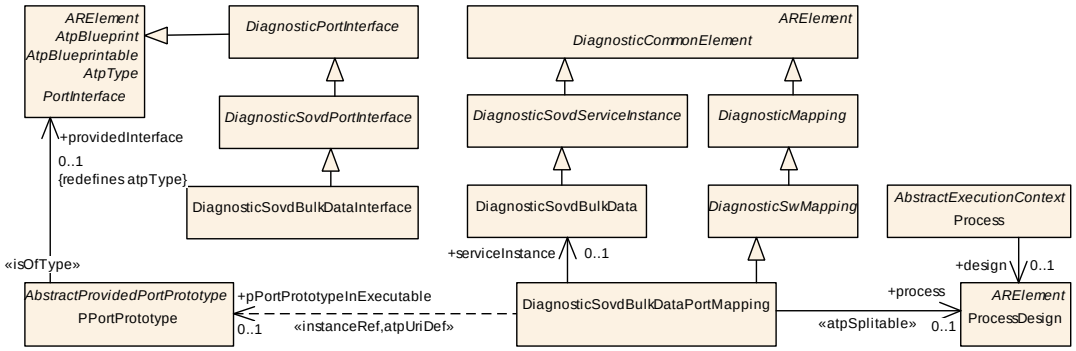


Figure 4.23: Modeling of the DiagnosticSovdBulkDataPortMapping

[constr_10452]{DRAFT} **Restriction for the applicability of DiagnosticSovdBulkDataPortMapping** ⌈If a PPortPrototype is referenced n the role pPortPrototypeInExecutable by a DiagnosticSovdBulkDataPortMapping, then the PPortPrototype shall be typed by a DiagnosticSovdBulkDataInterface **at the time when the diagnostic design is complete**.⌋()

[constr_10453]{DRAFT} **Restriction for the applicability of DiagnosticSovdBulkDataInterface** ⌈If a PPortPrototype is typed by a DiagnosticSovdBulkDataInterface, then the PortPrototype shall only be referenced in the role pPortPrototypeInExecutable by a DiagnosticSovdBulkDataPortMapping **at the time when the diagnostic design is complete**.⌋()

[constr_10454]{DRAFT} **Uniqueness of reference from DiagnosticSovdBulkDataPortMapping to DiagnosticSovdBulkData** ⌈Each instance DiagnosticSovdBulkData shall only be referenced in the role serviceInstance by exactly one DiagnosticSovdBulkDataPortMapping **at the time when the diagnostic design is complete**.⌋()

Class	**DiagnosticSovdBulkDataPortMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticSovdMapping
Note	This mapping associates a PPortPrototype typed by a DiagnosticSovdBulkDataInterface to to the corresponding SOVD service instance that is modeled as DiagnosticSovdBulkData. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticMappings
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*
Aggregated by	ARPackage.element

▽

△

Class	**DiagnosticSovdBulkDataPortMapping**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdBulkDataPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=candidate
serviceInstance	DiagnosticSovdBulk Data	0..1	ref	This reference identifies the applicable diagnostic SOVD service instance. **Tags:** atp.Status=candidate

Table 4.59: DiagnosticSovdBulkDataPortMapping

[constr_10455]{DRAFT} **Existence of the reference in the role `DiagnosticSovdBulkDataPortMapping.serviceInstance`** ⌈For each `DiagnosticSovdBulkDataPortMapping`, the reference in the role `DiagnosticSovdBulkDataPortMapping.serviceInstance` shall exist **`at the time when the diagnostic design is complete`**.⌋*()*

[constr_10456]{DRAFT} **Existence of the reference in the role `DiagnosticSovdBulkDataPortMapping.pPortPrototypeInExecutable`** ⌈For each `DiagnosticSovdBulkDataPortMapping`, the reference in the role `DiagnosticSovdBulkDataPortMapping.pPortPrototypeInExecutable` shall exist **`at the time when the diagnostic design is complete`**.⌋*()*

[constr_10457]{DRAFT} **Existence of the reference in the role `DiagnosticSovdBulkDataPortMapping.process`** ⌈For each `DiagnosticSovdBulkDataPortMapping`, the reference in the role `DiagnosticSovdBulkDataPortMapping.process` shall exist **`at the time when the diagnostic design is complete`**.⌋*()*

4.2.4.4 Diagnostic SOVD Update Port Mapping

[TPS_MANI_01427]{DRAFT} **Mapping of `DiagnosticSovdUpdate` to `PortPrototype` on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticSovdUpdate` and one `PPortPrototype`is created by using the `DiagnosticSovdUpdatePortMapping` that refers to

- a `DiagnosticSovdUpdate` in the role `serviceInstance`,
- a `ProcessDesign` in the role `process`, and

- an PPortPrototype typed by a DiagnosticSovdUpdateInterface in the role pPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype on the application layer will use at run-time.

⌋(*RS_MANI_00005, RS_MANI_00061*)

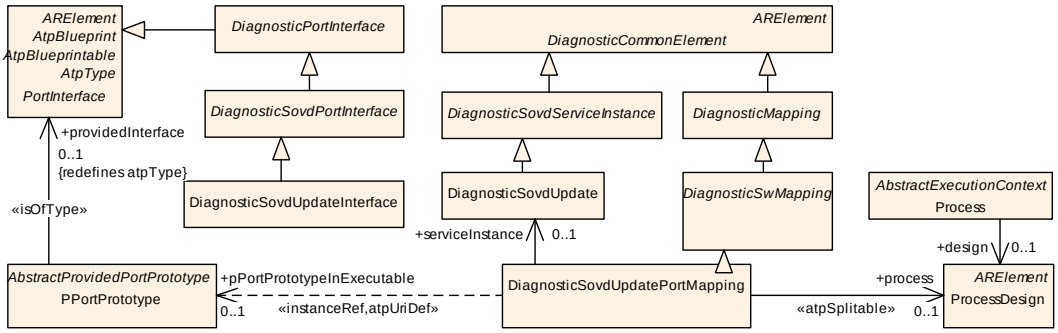


Figure 4.24: Modeling of the DiagnosticSovdUpdatePortMapping

[constr_10458]{DRAFT} **Restriction for the applicability of DiagnosticSovdUpdatePortMapping** ⌈If a PPortPrototype is referenced in the role pPortPrototypeInExecutable by a DiagnosticSovdUpdatePortMapping, then the PPortPrototype shall be typed by a DiagnosticSovdUpdateInterface **at the time when the diagnostic design is complete**.⌋()

[constr_10459]{DRAFT} **Restriction for the applicability of DiagnosticSovdUpdateInterface** ⌈If a PPortPrototype is typed by a DiagnosticSovdUpdateInterface, then the PortPrototype shall only be referenced in the role pPortPrototypeInExecutable by a DiagnosticSovdUpdatePortMapping **at the time when the diagnostic design is complete**.⌋()

[constr_10460]{DRAFT} **Uniqueness of reference from DiagnosticSovdUpdatePortMapping to DiagnosticSovdUpdate** ⌈Each instance of DiagnosticSovdUpdate shall only be referenced in the role serviceInstance by exactly one DiagnosticSovdUpdatePortMapping **at the time when the diagnostic design is complete**.⌋()

Class	**DiagnosticSovdUpdatePortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticSovdMapping			
Note	This mapping associates a PPortPrototype typed by an DiagnosticSovdUpdateInterface with the corresponding SOVD service instance that is modeled as a DiagnosticSovdUpdate. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**DiagnosticSovdUpdatePortMapping**			
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdUpdatePortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=candidate
serviceInstance	DiagnosticSovdUpdate	0..1	ref	This reference identifies the applicable diagnostic SOVD service instance. **Tags:** atp.Status=candidate

Table 4.60: DiagnosticSovdUpdatePortMapping

[constr_10461]{DRAFT} **Existence of the reference in the role `DiagnosticSovdUpdatePortMapping.serviceInstance`** ⌈For each `DiagnosticSovdUpdatePortMapping`, the reference in the role `DiagnosticSovdUpdatePortMapping.serviceInstance` shall exist **`at the time when the diagnostic design is complete`**.⌋*()*

[constr_10462]{DRAFT} **Existence of the reference in the role `DiagnosticSovdUpdatePortMapping.pPortPrototypeInExecutable`** ⌈For each `DiagnosticSovdUpdatePortMapping`, the reference in the role `DiagnosticSovdUpdatePortMapping.pPortPrototypeInExecutable` shall exist **`at the time when the diagnostic design is complete`**.⌋*()*

[constr_10463]{DRAFT} **Existence of the reference in the role `DiagnosticSovdUpdatePortMapping.process`** ⌈For each `DiagnosticSovdUpdatePortMapping`, the reference in the role `DiagnosticSovdUpdatePortMapping.process` shall exist **`at the time when the diagnostic design is complete`**.⌋*()*

4.2.4.5 Diagnostic SOVD Service Validation Port Mapping

[TPS_MANI_01428]{DRAFT} **Mapping of `DiagnosticSovdServiceInstance` to `PortPrototype` on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `PPortPrototype` and a `ProcessDesign` is created by using the `DiagnosticSovdServiceValidationPortMapping` that refers to

- a `ProcessDesign` in the role `process`, and

- an PPortPrototype typed by a DiagnosticSovdServiceValidationInterface in the role pPortPrototypeInExecutable. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding InstanceSpecifier that the PortPrototype on the application layer will use at run-time.

⌋(*RS_MANI_00005, RS_MANI_00061*)

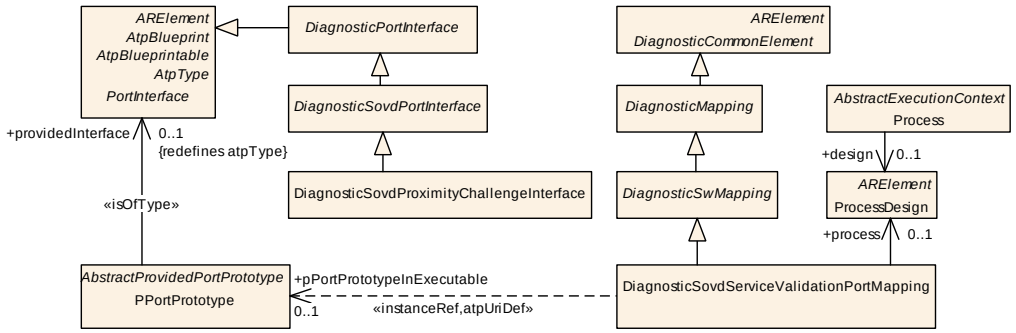


Figure 4.25: Modeling of the DiagnosticSovdServiceValidationPortMapping

[constr_10464]{DRAFT} **Restriction for the applicability of DiagnosticSovdServiceValidationPortMapping** ⌈If a PPortPrototype is referenced in the role pPortPrototypeInExecutable by a DiagnosticSovdServiceValidationPortMapping, then the PPortPrototype shall be typed by a DiagnosticSovdServiceValidationInterface **at the time when the diagnostic design is complete**.⌋()

[constr_10465]{DRAFT} **Restriction for the applicability of DiagnosticSovdServiceValidationInterface** ⌈If a PPortPrototype is typed by a DiagnosticSovdServiceValidationInterface, then the PortPrototype shall only be referenced in the role pPortPrototypeInExecutable by a DiagnosticSovdServiceValidationPortMapping **at the time when the diagnostic design is complete**.⌋()

Class	**DiagnosticSovdServiceValidationPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticSovdMapping			
Note	This mapping class identifies the PortPrototype in the application software that handles the SOVD service validation. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	DiagnosticSovdServiceValidationPortMapping			
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdValidationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Tags:** atp.Status=candidate

Table 4.61: DiagnosticSovdServiceValidationPortMapping

[constr_10466]{DRAFT} **Existence of the reference in the role `DiagnosticSovdServiceValidationPortMapping.pPortPrototypeInExecutable`** ⌈For each `DiagnosticSovdServiceValidationPortMapping`, the reference in the role `DiagnosticSovdServiceValidationPortMapping.pPortPrototypeInExecutable` shall exist **at the time when the diagnostic design is complete**.⌋*()*

[constr_10467]{DRAFT} **Existence of the reference in the role `DiagnosticSovdServiceValidationPortMapping.process`** ⌈For each `DiagnosticSovdServiceValidationPortMapping`, the reference in the role `DiagnosticSovdServiceValidationPortMapping.process` shall exist **at the time when the diagnostic design is complete**.⌋*()*

4.2.4.6 Diagnostic SOVD Configuration Port Mapping

[TPS_MANI_01429]{DRAFT} **Mapping of `DiagnosticSovdConfiguration` to `PortPrototype` on the *AUTOSAR adaptive platform*** ⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticSovdConfiguration` and one `PPortPrototype`is created by using the `DiagnosticSovdConfigurationPortMapping` that refers to

- a `DiagnosticSovdConfiguration` in the role `serviceInstance`,
- a `ProcessDesign` in the role `process`, and
- an `PPortPrototype` typed by a `DiagnosticSovdConfigurationInterface` in the role `pPortPrototypeInExecutable`. The target of this reference does not necessarily have to exist at the time of configuration of the DM. The reference itself, however, is required for constructing the corresponding `InstanceSpecifier` that the `PortPrototype` on the application layer will use at run-time.

⌋*(RS_MANI_00005, RS_MANI_00061)*

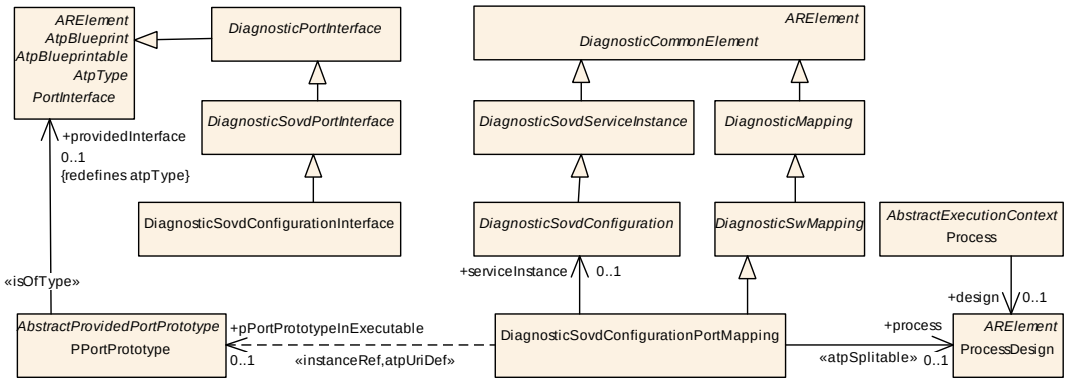


Figure 4.26: Modeling of the `DiagnosticSovdConfigurationPortMapping`

[constr_10468]{DRAFT} **Restriction for the applicability of `DiagnosticSovdConfigurationPortMapping`** ⌈If a `PPortPrototype` is referenced in the role `pPortPrototypeInExecutable` by a `DiagnosticSovdConfigurationPortMapping`, then the `PPortPrototype` shall be typed by a `DiagnosticSovdConfigurationInterface` **at the time when the diagnostic design is complete**.⌋*()*

[constr_10469]{DRAFT} **Restriction for the applicability of `DiagnosticSovdConfigurationInterface`** ⌈If a `PPortPrototype` is typed by a `DiagnosticSovdConfigurationInterface`, then the `PortPrototype` shall only be referenced in the role `pPortPrototypeInExecutable` by a `DiagnosticSovdConfigurationPortMapping` **at the time when the diagnostic design is complete**.⌋*()*

[constr_10470]{DRAFT} **Uniqueness of reference from `DiagnosticSovdConfigurationPortMapping` to `DiagnosticSovdConfiguration`** ⌈Each instance of a sub-class of `DiagnosticSovdConfiguration` shall only be referenced in the role `serviceInstance` by exactly one `DiagnosticSovdConfigurationPortMapping` **at the time when the diagnostic design is complete**.⌋*()*

Class	**DiagnosticSovdConfigurationPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticSovdMapping			
Note	This mapping associates a PPortPrototype typed by a DiagnosticSovdConfigurationInterface to to the corresponding SOVD service instance that is modeled as DiagnosticSovdConfiguration. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *DiagnosticSwMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**DiagnosticSovdConfigurationPortMapping**			
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This aggregation allows for the usage of the Diagnostic SovdConfigurationPortMapping on the AUTOSAR adaptive platform. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	Reference to the representation of a Process that is required because the mapping could be different for different Processes referring to a specific Executable. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process atp.Status=candidate
serviceInstance	DiagnosticSovd Configuration	0..1	ref	This reference identifies the applicable diagnostic SOVD service instance. **Tags:** atp.Status=candidate

Table 4.62: DiagnosticSovdConfigurationPortMapping

[constr_10471]{DRAFT} **Existence of the reference in the role `DiagnosticSovdConfigurationPortMapping.serviceInstance`** ⌈For each `DiagnosticSovdConfigurationPortMapping`, the reference in the role `DiagnosticSovdConfigurationPortMapping.serviceInstance` shall exist **at the time when the diagnostic design is complete**.⌋*()*

[constr_10472]{DRAFT} **Existence of the reference in the role `DiagnosticSovdConfigurationPortMapping.pPortPrototypeInExecutable`** ⌈For each `DiagnosticSovdConfigurationPortMapping`, the reference in the role `DiagnosticSovdConfigurationPortMapping.pPortPrototypeInExecutable` shall exist **at the time when the diagnostic design is complete**.⌋*()*

[constr_10473]{DRAFT} **Existence of the reference in the role `DiagnosticSovdConfigurationPortMapping.process`** ⌈For each `DiagnosticSovdConfigurationPortMapping`, the reference in the role `DiagnosticSovdConfigurationPortMapping.process` shall exist **at the time when the diagnostic design is complete**.⌋*()*

4.2.4.7 Diagnostic SOVD Configuration to Data Identifier Mapping

[TPS_MANI_01430]{DRAFT} **Mapping of `DiagnosticSovdConfigurationParameter` to `DiagnosticDataIdentifier` on the *AUTOSAR adaptive platform***
⌈On the *AUTOSAR adaptive platform*, the relation between a `DiagnosticSovdConfigurationParameter` and one `DiagnosticDataIdentifier` is created by using the `DiagnosticSovdConfigurationDataIdentifierMapping` that refers to

- a `DiagnosticSovdConfigurationParameter` in the role `serviceInstance` and

- a DiagnosticDataIdentifier in the role dataIdentifier

⌋(*RS_MANI_00005*, *RS_MANI_00061*)

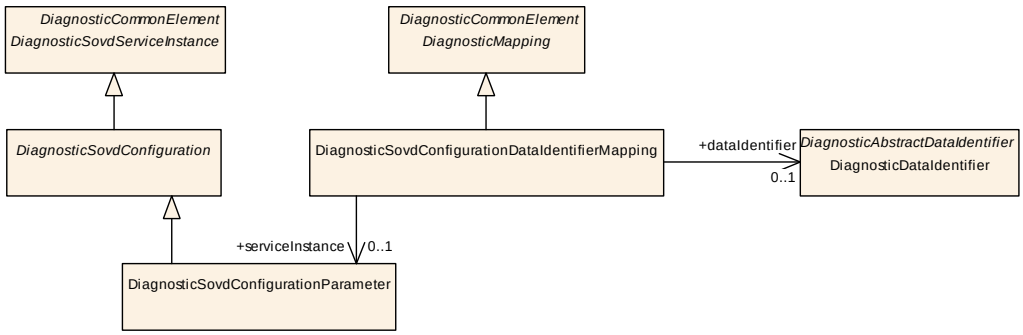


Figure 4.27: Modeling of the DiagnosticSovdConfigurationDataIdentifierMapping

[constr_10474]{DRAFT} **Uniqueness of reference from DiagnosticSovdConfigurationDataIdentifierMapping to DiagnosticSovdConfigurationParameter** ⌈Each instance of a sub-class of DiagnosticSovdConfigurationParameter shall only be referenced by exactly one DiagnosticSovdConfigurationDataIdentifierMapping **at the time when the diagnostic design is complete**.⌋()

Class	**DiagnosticSovdConfigurationDataIdentifierMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticSovdMapping			
Note	This mapping associates a SOVD configuration service instance to a DiagnosticDataIdentifier that carries the configuration payload. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataIdentifier	DiagnosticDataIdentifier	0..1	ref	This reference identifies the applicable data identifier, **Tags:** atp.Status=candidate
serviceInstance	DiagnosticSovd ConfigurationParameter	0..1	ref	This reference identifies the applicable service instance. **Tags:** atp.Status=candidate

Table 4.63: DiagnosticSovdConfigurationDataIdentifierMapping

[constr_10475]{DRAFT} **Existence of the reference in the role DiagnosticSovdConfigurationDataIdentifierMapping.serviceInstance** ⌈For each DiagnosticSovdConfigurationDataIdentifierMapping, the reference in the role DiagnosticSovdConfigurationDataIdentifierMapping.serviceInstance shall exist **at the time when the diagnostic design is complete**.⌋()

[constr_10476]{DRAFT} **Existence of the reference in the role DiagnosticSovdConfigurationDataIdentifierMapping.dataIdentifier** ⌈For each

DiagnosticSovdConfigurationDataIdentifierMapping, the reference in the role DiagnosticSovdConfigurationDataIdentifierMapping.dataIdentifier shall exist **at the time when the diagnostic design is complete**.
()

4.3 Diagnostic Clear Condition

On the *AUTOSAR adaptive platform*, a new model element similar DiagnosticEnableCondition is introduced: DiagnosticClearCondition.

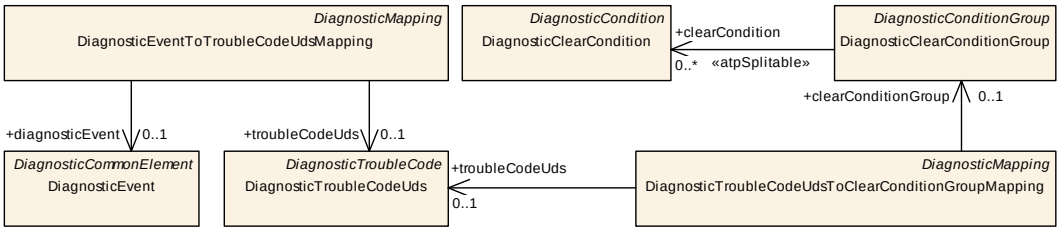


Figure 4.28: Modeling of the diagnostic clear condition

In contrast to DiagnosticEnableCondition, DiagnosticClearCondition is not mapped to a DiagnosticEvent but (via the aggregation by DiagnosticClearConditionGroup) to a DiagnosticTroubleCodeUds.

For this purpose, meta-class DiagnosticTroubleCodeUdsToClearConditionGroupMapping has been defined.

Class	**DiagnosticClearCondition**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition			
Note	This meta-class describes a clear condition for diagnostic purposes. **Tags:** atp.recommendedPackage=DiagnosticConditions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticCondition*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.64: DiagnosticClearCondition

Class	**DiagnosticClearConditionGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition			
Note	Clear condition group which includes one or several clear conditions. **Tags:** atp.recommendedPackage=DiagnosticClearConditionGroups			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticConditionGroup*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**DiagnosticClearConditionGroup**			
clearCondition	DiagnosticClear Condition	*	ref	This aggregation represents the collection of Diagnostic ClearConditions that belong to the DiagnosticClear ConditionGroup. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=clearCondition

Table 4.65: DiagnosticClearConditionGroup

Class	**DiagnosticTroubleCodeUdsToClearConditionGroupMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticClearCondition			
Note	This meta-class provides the ability to map a DiagnosticClearConditionGroup to a collection of Diagnostic TroubleCodeUds. **Tags:** atp.recommendedPackage=DiagnosticMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clearCondition Group	DiagnosticClear ConditionGroup	0..1	ref	Thi reference identifies the applicable DiagnosticClear ConditionGroup.
troubleCodeUds	DiagnosticTroubleCode Uds	0..1	ref	This reference identifies the DiagnosticTroubleCodeUds that are relevant for the mapping.

Table 4.66: DiagnosticTroubleCodeUdsToClearConditionGroupMapping

[constr_1658]{DRAFT} **Number of `DiagnosticTroubleCodeUdsToClearConditionGroupMapping` elements per `DiagnosticTroubleCodeUds`** ⌈The mapping element `DiagnosticTroubleCodeUdsToClearConditionGroupMapping` shall be created no more than once per `DiagnosticTroubleCodeUds`.

If several `DiagnosticTroubleCodeUdsToClearConditionGroupMapping` elements referring to the same `DiagnosticTroubleCodeUds` are defined, then the `Clear Condition Group` mapping shall be regarded as defective

This rule shall be imposed **`at the time when the diagnostic design is complete`**.⌋*()*

4.4 Security Access

the implementation of the diagnostics manager on the adaptive platform requires a refined modeling of meta-class `DiagnosticSecurityAccessClass`.

A new attribute named `sharedTimer` is introduced that controls whether a single timer is used for all security access levels or whether the individual levels utilize separate timers respectively.

Class	**DiagnosticSecurityAccessClass**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::SecurityAccess			
Note	This meta-class contains attributes shared by all instances of the "Security Access" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticSecurityAccesss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticServiceClass*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sharedTimer	Boolean	0..1	attr	Switch between separate or single shared timer instance and timer value. • true: use shared timer instance and timer value for all security access levels combined. • false: use separate timer instance and timer values for each security level.

Table 4.67: DiagnosticSecurityAccessClass

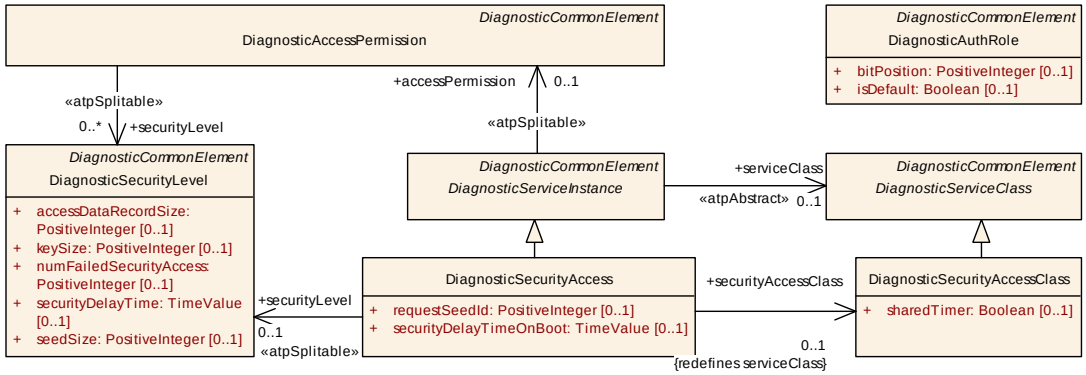


Figure 4.29: Refined modeling of the diagnostic security access

4.5 DiagnosticProvidedDataMapping

[TPS_MANI_01230]{DRAFT} **Semantics of `DiagnosticProvidedDataMapping`** ⌈The meta-class `DiagnosticProvidedDataMapping` does not seem to fulfill the condition for representing a mapping class because it only has one reference to a `DiagnosticDataElement` in the role `dataElement`.

However, the specific nature of this mapping is that the second element (the `DiagnosticProvidedDataMapping.dataProvider`) that is supposed to take place in the mapping cannot precisely be modeled as a single meta-class.

Therefore, there is no better way than to model the `DiagnosticProvidedDataMapping.dataProvider` by a `NameToken`.⌋*(RS_MANI_00005)*

Please note that the list of standardized values of attribute `DiagnosticProvidedDataMapping.dataProvider` is defined in the SWS Diagnostics [19].

Figure 4.30: Modeling of `DiagnosticProvidedDataMapping`

Class	**DiagnosticProvidedDataMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::DiagnosticProvidedDataMapping			
Note	This represents the ability to define the nature of a data access for a DiagnosticDataElement based on a data provider that cannot be modeled explicitly. **Tags:** atp.recommendedPackage=DataMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	DiagnosticDataElement	0..1	ref	This represents the DiagnosticDataElement for which the access is further qualified by the DiagnosticProvidedDataMapping.dataProvider.
dataProvider	NameToken	0..1	attr	This represents the ability to further specify the data provider.

Table 4.68: DiagnosticProvidedDataMapping

[constr_10166]{DRAFT} **Multiplicity of attribute `DiagnosticProvidedDataMapping.dataProvider`** ⌈For each `DiagnosticProvidedDataMapping`, the attribute `dataProvider` shall exist **`at the time when the diagnostic design is complete`**.⌋*()*

4.6 Usage of DiagnosticEnvDataElementCondition on the adaptive Platform

The definition of environmental conditions is explained in the document TPS Diagnostic Extract [2].

Specifically for the *AUTOSAR adaptive platform*, it is possible to obtain data from application software (fetched from a `PPortPrototype` typed by a `DiagnosticDataElementInterface`) as a reference value for the computation of the environmental condition.

[TPS_MANI_01372]{DRAFT} **Properties of data obtained from the application software** ⌈The properties of the data obtained from the application software can be configured in the context of `DiagnosticEnvDataElementCondition`, namely by means of the aggregation of `SwDataDefProps` in the role `swDataDefProps`.⌋*(RS_MANI_00005)*

[constr_10114]{DRAFT} **Existence of attributes of `DiagnosticEnvDataElementCondition` if the reference in the role `pPortPrototype` exists** ⌈If the reference in the role `DiagnosticEnvDataElementCondition.pPortPrototype`

exists, then the aggregations in the roles compareValue, process, and swDataDefProps shall exist **at the time when the diagnostic design is complete**.⌋()

Please note that the reference DiagnosticEnvDataElementCondition.pPortPrototype is decorated with stereotype ≪atpUriDef≫ because the model of the application software does not necessarily have to exist when the diagnostic design is created.

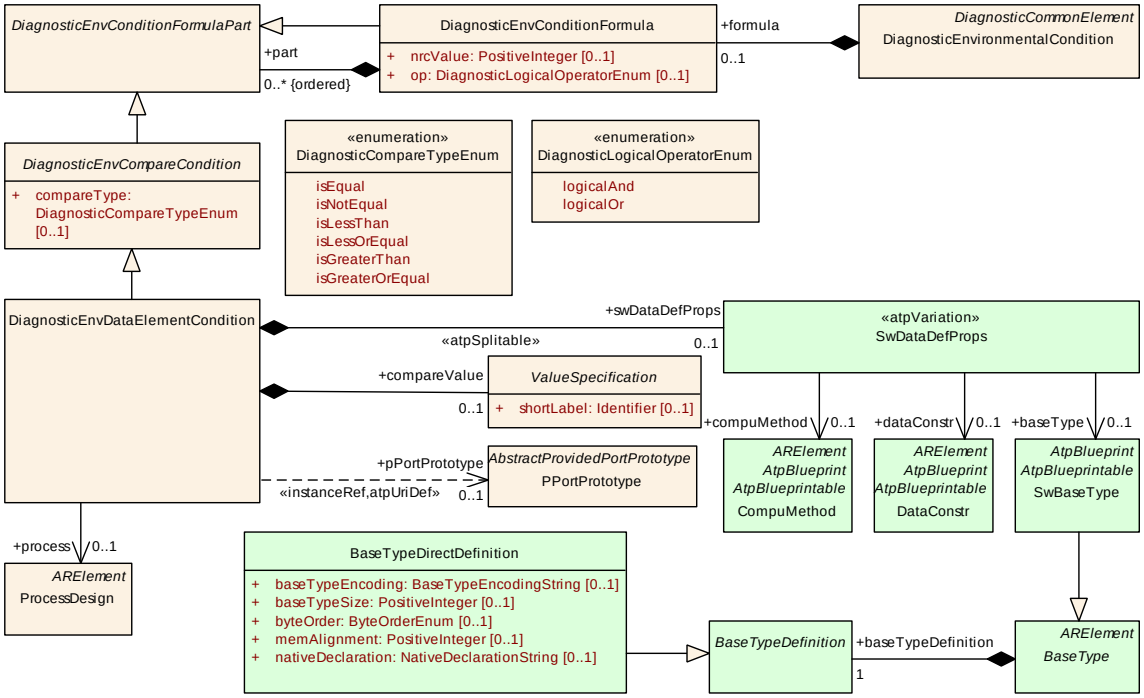


Figure 4.31: Leverage of DiagnosticEnvironmentalCondition on the *AUTOSAR adaptive platform*

Class	**DiagnosticEnvironmentalCondition**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
Note	The meta-class DiagnosticEnvironmentalCondition formalizes the idea of a condition which is evaluated during runtime of the ECU by looking at "environmental" states (e.g. one such condition is that the vehicle is not driving, i.e. vehicle speed == 0). **Tags:** atp.recommendedPackage=DiagnosticEnvironmentalConditions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
formula	DiagnosticEnvCondition Formula	0..1	aggr	This attribute represents the formula part of the DiagnosticEnvironmentalCondition.
modeElement	DiagnosticEnvMode Element	*	aggr	This aggregation contains a representation of Mode Declarations in the context of a DiagnosticEnvironmental Condition.

Table 4.69: DiagnosticEnvironmentalCondition

Class	**DiagnosticEnvDataElementCondition**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
Note	This meta-class represents the ability to formulate a diagnostic environment condition based on the value of a data element owned by the application software.			
Base	*ARObject*, *DiagnosticEnvCompareCondition*, *DiagnosticEnvConditionFormulaPart*			
Aggregated by	DiagnosticEnvConditionFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compareValue	ValueSpecification	0..1	aggr	This aggregation represents the definition of the compare value against which the value taken from the application software shall be compared.
dataPrototype	DataPrototype	0..1	iref	This instanceRef represent the ability to access a data element owned by the application software on the AUTOSAR classic platform. **InstanceRef implemented by:** DataPrototypeInSystem InstanceRef
pPortPrototype	PPortPrototype	0..1	iref	This instanceRef identifies the PortPrototype from which the relevant information for the environment condition can be obtained. This InstanceRef is only relevant for the adaptive platform. **Stereotypes:** atpUriDef **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	ProcessDesign	0..1	ref	This reference identifies the applicable ProcessDesign.
swDataDef Props	SwDataDefProps	0..1	aggr	Via this aggregation it is possible to describe the properties of the data that is obtained from the application for the environmental condition. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=swDataDefProps

Table 4.70: DiagnosticEnvDataElementCondition

Class	***DiagnosticEnvCompareCondition*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
Note	DiagnosticCompareConditions are atomic conditions. They are based on the idea of a comparison at runtime of some variable data with something constant. The type of the comparison (==, !=, <, <=, ...) is specified in DiagnosticCompareCondition.compareType.			
Base	*ARObject*, *DiagnosticEnvConditionFormulaPart*			
Subclasses	DiagnosticEnvDataCondition, DiagnosticEnvDataElementCondition, DiagnosticEnvModeCondition			
Aggregated by	DiagnosticEnvConditionFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
compareType	DiagnosticCompare TypeEnum	0..1	attr	This attributes represents the concrete type of the comparison.

Table 4.71: DiagnosticEnvCompareCondition

Class	DiagnosticEnvConditionFormula			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
Note	A DiagnosticEnvConditionFormula embodies the computation instruction that is to be evaluated at runtime to determine if the DiagnosticEnvironmentalCondition is currently present (i.e. the formula is evaluated to true) or not (otherwise). The formula itself consists of parts which are combined by the logical operations specified by DiagnosticEnvConditionFormula.op. If a diagnostic functionality cannot be executed because an environmental condition fails then the diagnostic stack shall send a negative response code (NRC) back to the client. The value of the NRC is directly related to the specific formula and is therefore formalized in the attribute DiagnosticEnvCondition Formula.nrcValue.			
Base	*ARObject*, *DiagnosticEnvConditionFormulaPart*			
Aggregated by	DiagnosticEnvConditionFormula.part, DiagnosticEnvironmentalCondition.formula			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
nrcValue	PositiveInteger	0..1	attr	This attribute represents the concrete NRC value that shall be returned if the condition fails.
op	DiagnosticLogical OperatorEnum	0..1	attr	This attribute represents the concrete operator (supported operators: and, or) of the condition formula.
part (ordered)	DiagnosticEnvCondition FormulaPart	*	aggr	This aggregation represents the collection of formula parts that can be combined by logical operators.

Table 4.72: DiagnosticEnvConditionFormula

Class	***DiagnosticEnvConditionFormulaPart*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
Note	A DiagnosticEnvConditionFormulaPart can either be a atomic condition, e.g. a DiagnosticEnvCompare Condition, or a DiagnosticEnvConditionFormula, again, which allows arbitrary nesting.			
Base	*ARObject*			
Subclasses	*DiagnosticEnvCompareCondition*, DiagnosticEnvConditionFormula			
Aggregated by	DiagnosticEnvConditionFormula.part			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.73: DiagnosticEnvConditionFormulaPart

Class	***DiagnosticEnvModeElement*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition			
Note	All ModeDeclarations that are referenced in a DiagnosticEnvModeCondition shall be defined as a DiagnosticEnvModeElement of this DiagnosticEnvironmentalCondition. This concept keeps the ARXML clean: It avoids that the DiagnosticEnvConditionFormula is cluttered by lengthy InstanceRef definitions. Furthermore, it allows that an InstanceRef only needs to be defined once and can be used multiple times in the different DiagnosticEnvModeConditions.			
Base	*ARObject*, *Referrable*			
Subclasses	DiagnosticEnvBswModeElement, DiagnosticEnvSwcModeElement			
Aggregated by	DiagnosticEnvironmentalCondition.modeElement			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.74: DiagnosticEnvModeElement

Enumeration	**DiagnosticCompareTypeEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition
Note	Enumeration for the type of a comparison of values usually expressed by the following operators: ==, !=, <, <=, >, >=
Aggregated by	*DiagnosticEnvCompareCondition*.compareType
Literal	***Description***
isEqual	equal **Tags:** atp.EnumerationLiteralIndex=0
isGreaterOrEqual	greater than or equal **Tags:** atp.EnumerationLiteralIndex=5
isGreaterThan	greater than **Tags:** atp.EnumerationLiteralIndex=4
isLessOrEqual	less than or equal **Tags:** atp.EnumerationLiteralIndex=3
isLessThan	less than **Tags:** atp.EnumerationLiteralIndex=2
isNotEqual	not equal **Tags:** atp.EnumerationLiteralIndex=1

Table 4.75: DiagnosticCompareTypeEnum

Enumeration	**DiagnosticLogicalOperatorEnum**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::EnvironmentalCondition
Note	Logical AND and OR operation (&&, ||)
Aggregated by	DiagnosticEnvConditionFormula.op
Literal	***Description***
logicalAnd	Logical AND **Tags:** atp.EnumerationLiteralIndex=0
logicalOr	Logical OR **Tags:** atp.EnumerationLiteralIndex=1

Table 4.76: DiagnosticLogicalOperatorEnum

4.7 SOVD Configuration Miscellanea

This chapter documents that meaning of a number of meta-classes that are specific to the usage of Service-Oriented Vehicle Diagnostics (SOVD).

4.7.1 SOVD Service Instance

[TPS_MANI_01431]{DRAFT} **Definition of SOVD service instance** ⌈Similar to UDS, SOVD supports the concept of a *service instance*. The SOVD service instance is formalized by means of abstract meta-class `DiagnosticSovdServiceInstance`.⌋ *()*

Concrete SOVD services are modeled as concrete sub-classes of DiagnosticSovdServiceInstance. The “big picture” of the modeling of SOVD service instances is sketched in Figure 4.32.

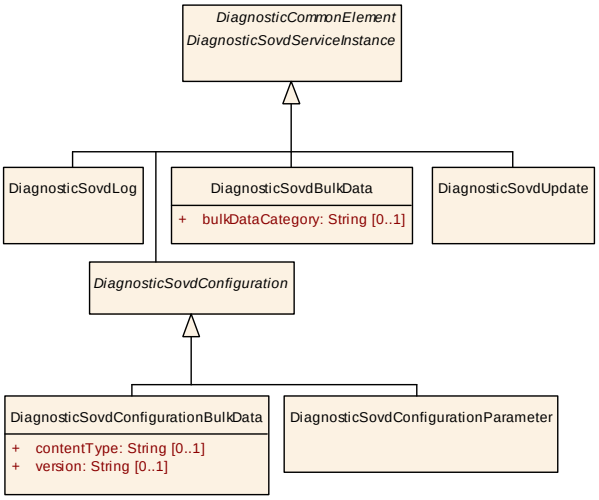


Figure 4.32: Modeling of DiagnosticSovdServiceInstance

Class	***DiagnosticSovdServiceInstance*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::SovdServiceInstance			
Note	This abstract class serves as the base class for the configuration of SOVD service instances. **Tags:** atp.Status=candidate			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticSovdBulkData, *DiagnosticSovdConfiguration*, DiagnosticSovdLog, DiagnosticSovdUpdate			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
method	DiagnosticSovdMethod	0..1	ref	This reference identifies the applicable SOVD method. **Tags:** atp.Status=candidate

Table 4.77: DiagnosticSovdServiceInstance

The SOVD log service is based on the existence of just a single instance of DiagnosticSovdLog for which no matching subclass of DiagnosticSovdPortInterface is defined.

This means that the application software does not have access to the log service[2] and therefore access to SOVD logging is restricted to diagnostic management (DM) is using the logging service internally.

The restriction to just a single instance of DiagnosticSovdLog is formalized by means of [constr_10479].

[constr_10477]{DRAFT} **Existence of DiagnosticSovdLog** ⌈Each DiagnosticContributionSet shall at most reference a single DiagnosticSovdLog in the role element **at the time when the diagnostic design is complete**.⌋*()*

[2]This also explains why there is no mapping class defined for the SOVD logging.

Class	DiagnosticSovdLog			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::SovdServiceInstance			
Note	This meta-class represents a "Log" SOVD service instance. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticSovdServiceInstances			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdService Instance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.78: DiagnosticSovdLog

A similar restriction is in place for the existence of `DiagnosticSovdUpdate`, see [constr_10480].

[constr_10478]{DRAFT} **Existence of `DiagnosticSovdUpdate`** ⌈Each `DiagnosticContributionSet` shall at most reference a single `DiagnosticSovdUpdate` in the role `element` **at the time when the diagnostic design is complete**.⌋*()*

Class	DiagnosticSovdUpdate			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::SovdServiceInstance			
Note	This meta-class represents a "Update" SOVD service instance. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticSovdServiceInstances			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdService Instance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 4.79: DiagnosticSovdUpdate

Class	DiagnosticSovdBulkData			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::SovdServiceInstance			
Note	This meta-class represents a "Bulk Data" SOVD service instance. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticSovdServiceInstances			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdService Instance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
bulkData Category	String	0..1	attr	This attribute provides a categorization of the bulk data. **Tags:** atp.Status=candidate

Table 4.80: DiagnosticSovdBulkData

Class	***DiagnosticSovdConfiguration*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::SovdServiceInstance			
Note	This abstract meta-class represents a "configuration" SOVD service instance. The concrete nature of the service instance is defined by sub-classes. **Tags:** atp.Status=candidate			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticSovdConfigurationBulkData, DiagnosticSovdConfigurationParameter			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.81: DiagnosticSovdConfiguration

Class	**DiagnosticSovdConfigurationParameter**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::SovdServiceInstance			
Note	This meta-class represents a "Configuration Parameter" SOVD service instance. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticSovdServiceInstances			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdConfiguration*, *DiagnosticSovdServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.82: DiagnosticSovdConfigurationParameter

Class	**DiagnosticSovdConfigurationBulkData**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::SovdServiceInstance			
Note	This meta-class represents a "Configuration Bulk Data" SOVD service instance. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticSovdServiceInstances			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticSovdConfiguration*, *DiagnosticSovdServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contentType	String	0..1	attr	This attribute represents the setting of the content (i.e. MIME) type of the bulk data. Please note that the number of supported content types is not restricted by the ASAM SOVD standard and therefore it is unfortunately not possible to constrain the possible settings of this attribute. **Tags:** atp.Status=candidate
version	String	0..1	attr	This attribute represents a version information. Please note that the structure of the version information is not restricted by the ASAM SOVD standard and therefore the structure of the content of this attribute cannot be constrained further in the scope of AUTOSAR. **Tags:** atp.Status=candidate

Table 4.83: DiagnosticSovdConfigurationBulkData

4.7.2 SOVD Method

As sketched in Figure 4.33, the modeling of the behavior of a DiagnosticSovdServiceInstance by means of meta-class DiagnosticSovdMethod follows a pattern similar to the modeling of a diagnostic routine [2] in that primitive actions are defined that represent atomic activities in the execution of the SOVD service.

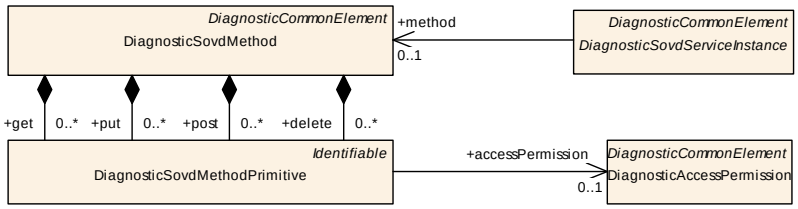


Figure 4.33: Modeling of DiagnosticSovdMethod and the relation to DiagnosticAccessPermission

[TPS_MANI_01432]{DRAFT} **Semantics of DiagnosticSovdMethodPrimitive** ⌈Each of the primitive activities, modeled as DiagnosticSovdMethodPrimitive has a reference to DiagnosticAccessPermission. The basic semantics implemented in the context of a DiagnosticSovdMethod is modeled by mean of the aggregation roles

- get
- put
- post
- delete

The DiagnosticSovdMethodPrimitive are the building blocks out of which the semantics **and access permission**s of each SOVD service can be defined.⌋*()*

[constr_10479]{DRAFT} **Restriction on values of attribute DiagnosticSovdMethodPrimitive.category aggregated in the role put in the context of SOVD Update** ⌈If a DiagnosticSovdMethod is referenced from a DiagnosticSovdUpdate and the attribute put.category exists, then the value of the attribute category within any DiagnosticSovdMethodPrimitive aggregated in the role put is restricted to the following values:

- PREPARE
- EXECUTE
- AUTOMATE

This rule shall be imposed **at the time when the diagnostic design is complete**.⌋*()*

Please find more information about the meaning of the values of DiagnosticSovdMethodPrimitive.category in SWS Diagnostics [19].

[constr_10480]{DRAFT} **Restriction on value of attribute DiagnosticSovdMethodPrimitive.category aggregated in the role get in the context of SOVD Update** ⌈If a DiagnosticSovdMethod is referenced from a DiagnosticSovdUpdate and the attribute get.category exists, then the value of the attribute category within any DiagnosticSovdMethodPrimitive aggregated in the role get is restricted to the value STATUS **at the time when the diagnostic design is complete**.⌋*()*

[constr_10481]{DRAFT} **Restriction on values of attribute DiagnosticSovdMethodPrimitive.category aggregated in the role get in the context of SOVD Log** ⌈If a DiagnosticSovdMethod is referenced from a DiagnosticSovdLog and the attribute get exists, then the value of the attribute category within any DiagnosticSovdMethodPrimitive aggregated in the role get is restricted to the values

- CONFIG
- ENTRIES

This rule shall be imposed **at the time when the diagnostic design is complete**.⌋*()*

[constr_10482]{DRAFT} **Restriction on values of attribute DiagnosticSovdMethodPrimitive.category aggregated in the roles put and delete in the context of SOVD Log** ⌈If a DiagnosticSovdMethod is referenced from a DiagnosticSovdLog and the attributes

- put
- delete

exist, then the value of the attribute category within any DiagnosticSovdMethodPrimitive aggregated in the role

- put
- delete

is restricted to the value CONFIG.

This rule shall be imposed **at the time when the diagnostic design is complete**.⌋*()*

Class	**DiagnosticSovdMethod**
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::SovdServiceInstance
Note	A DiagnosticSovdMethod represents a re-usable complex operation (that consists of primitive operations) in the context of the communication of an SOVD server. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticSovdMethods
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	**DiagnosticSovdMethod**			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
delete	DiagnosticSovdMethod Primitive	*	aggr	This represents the "delete" method primitive. **Tags:** atp.Status=candidate
get	DiagnosticSovdMethod Primitive	*	aggr	This represents the "get" method primitive. **Tags:** atp.Status=candidate
post	DiagnosticSovdMethod Primitive	*	aggr	This represents the "post" method primitive. **Tags:** atp.Status=candidate
put	DiagnosticSovdMethod Primitive	*	aggr	This represents the "delete" method primitive. **Tags:** atp.Status=candidate

Table 4.84: DiagnosticSovdMethod

Class	**DiagnosticSovdMethodPrimitive**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::DiagnosticDesign::SovdServiceInstance			
Note	This meta-class represents a primitive operation inside a DiagnosticSovdMethod. **Tags:** atp.Status=candidate			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DiagnosticSovdMethod.delete, DiagnosticSovdMethod.get, DiagnosticSovdMethod.post, DiagnosticSovdMethod.put			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
access Permission	DiagnosticAccess Permission	0..1	ref	This reference identifes the applicable access permission. **Tags:** atp.Status=candidate

Table 4.85: DiagnosticSovdMethodPrimitive

4.7.3 SOVD Lock

[TPS_MANI_01396]{DRAFT} **Semantics of meta-class `DiagnosticSovdLock`** ⌈SOVD does not integrate with the existing UDS-based session handling. But nevertheless, it is necessary to provide the ability to lock the access for one SOVD client such that no other SOVD or UDS client can access the software that is subject to the diagnostic activity.

For this purpose, meta-class `DiagnosticSovdLock` has been defined. It is referenced by `DiagnosticAccessPermission` as an additional criterion for computing the access permission for a diagnostic client.⌋*(RS_MANI_00005, RS_MANI_00070)*

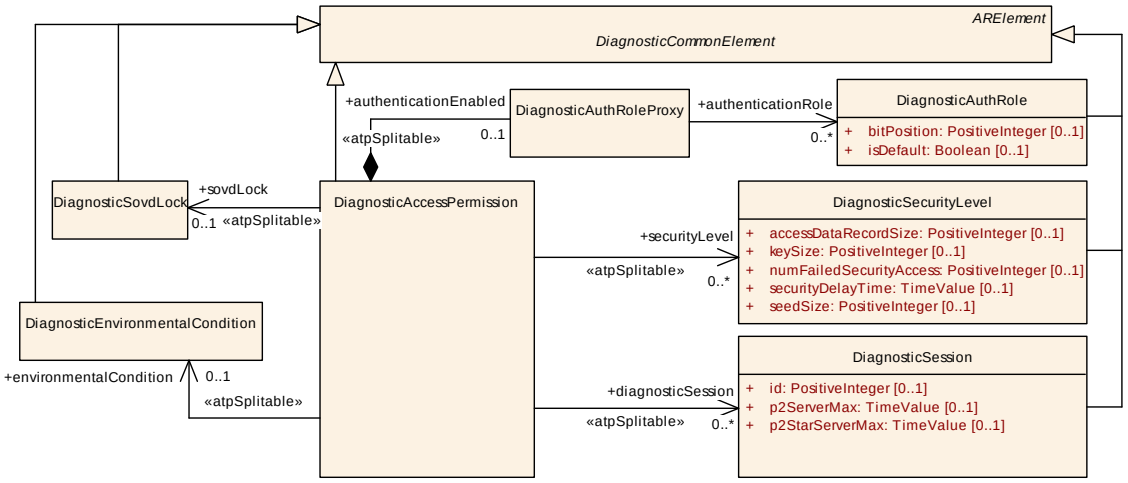


Figure 4.34: Modeling of SovdLock in the context of `DiagnosticAccessPermission`

Class	**DiagnosticSovdLock**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::SOVD			
Note	This meta-class represents the ability to define an SOVD lock to be considered for the computation of a diagnostic access permission. **Tags:** atp.Status=candidate atp.recommendedPackage=DiagnosticSovdLocks			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 4.86: DiagnosticSovdLock

4.7.4 SOVD Categorization of Diagnostic Data Identifiers

[TPS_MANI_01397]{DRAFT} **Standardized values of `DiagnosticDataIdentifier.category` of the usage in SOVD context** ⌈The following value of attribute `DiagnosticDataIdentifier.category` is standardized by AUTOSAR:

- `SOVD_DATA`

⌋*(RS_MANI_00005, RS_MANI_00070)*

4.7.5 SOVD Groups of Diagnostic Data Identifiers

For the support of SOVD, it may be necessary to define a collection of `DiagnosticDataIdentifier`s with the semantics of becoming an *SOVD data group*.

[TPS_MANI_01398]{DRAFT} **Definition of an *SOVD data group*** ⌈The definition of an *SOVD data group* shall be done by defining a `Collection` with the following properties:

- Attribute `collectionSemantics` shall be set to the value `SOVD_DATAGROUP`.
- Attribute `category` shall be set to `SET`.
- The references in the role `element` shall refer to the `DiagnosticDataIdentifier`s that shall be members of the collection.
- Attribute `elementRole` shall be set to `PART_OF_SUBSET`.

⌋*(RS_MANI_00005, RS_MANI_00070)*

4.8 Diagnostic Miscellanea

[TPS_MANI_01358]{DRAFT} **Restriction for the configuration of diagnostic debouncing** ⌈The restriction for the configuration of the diagnostic debouncing, as formulated in [TPS_DEXT_03004], shall **not** apply on *the AUTOSAR adaptive platform.*⌋ *(RS_MANI_00005)*

[constr_10483]{DRAFT} **Applicable values of `baseTypeEncoding` in the context of the definition of a `DiagnosticDataElement`** ⌈Any `SwBaseType` referenced in the role `DiagnosticDataElement.swDataDefProps.baseType` shall only set one of the following values for attribute `SwBaseType.baseTypeDefinition.baseTypeEncoding`:

- **`BOOLEAN`** (for `baseTypeSize` = 8)
- **`NONE`** (for `baseTypeSize` = 8, 16, or 32)
- **`2C`** (for `baseTypeSize` = 8, 16, or 32)
- **`BCD-P`** (for `baseTypeSize` = 8)
- **`BCD-UP`** (for `baseTypeSize` = 8)
- **`IEEE754`** (for `baseTypeSize` = 32)
- **`UTF-8`** (for `baseTypeSize` = 8)
- **`UTF-16`** (for `baseTypeSize` = 16)
- **`WINDOWS-1252`** (for `baseTypeSize` = 8)

⌋*()*

A description of the standardized values mentioned in [constr_10483] can be found in the specification of the TPS Software Component Template [1].

The relation between the possible values of `baseTypeEncoding` and the minimum value of `baseTypeSize` is documented in [TPS_SWCT_01876].

5 Machine Design

On the *AUTOSAR classic platform*, the design of communication-related aspects is attached to the existence of meta-class `EcuInstance`. A similar approach to support the definition of communication-related aspects early in the workflow (and definitely before the manifest configuration starts) is also needed on the *AUTOSAR adaptive platform*.

Therefore, `MachineDesign` has been introduced in order to allow the communication system designer to define a placeholder for an adaptive ECU in the scope of the `System` (the `MachineDesign` corresponds to the `EcuInstance` of AUTOSAR classic).

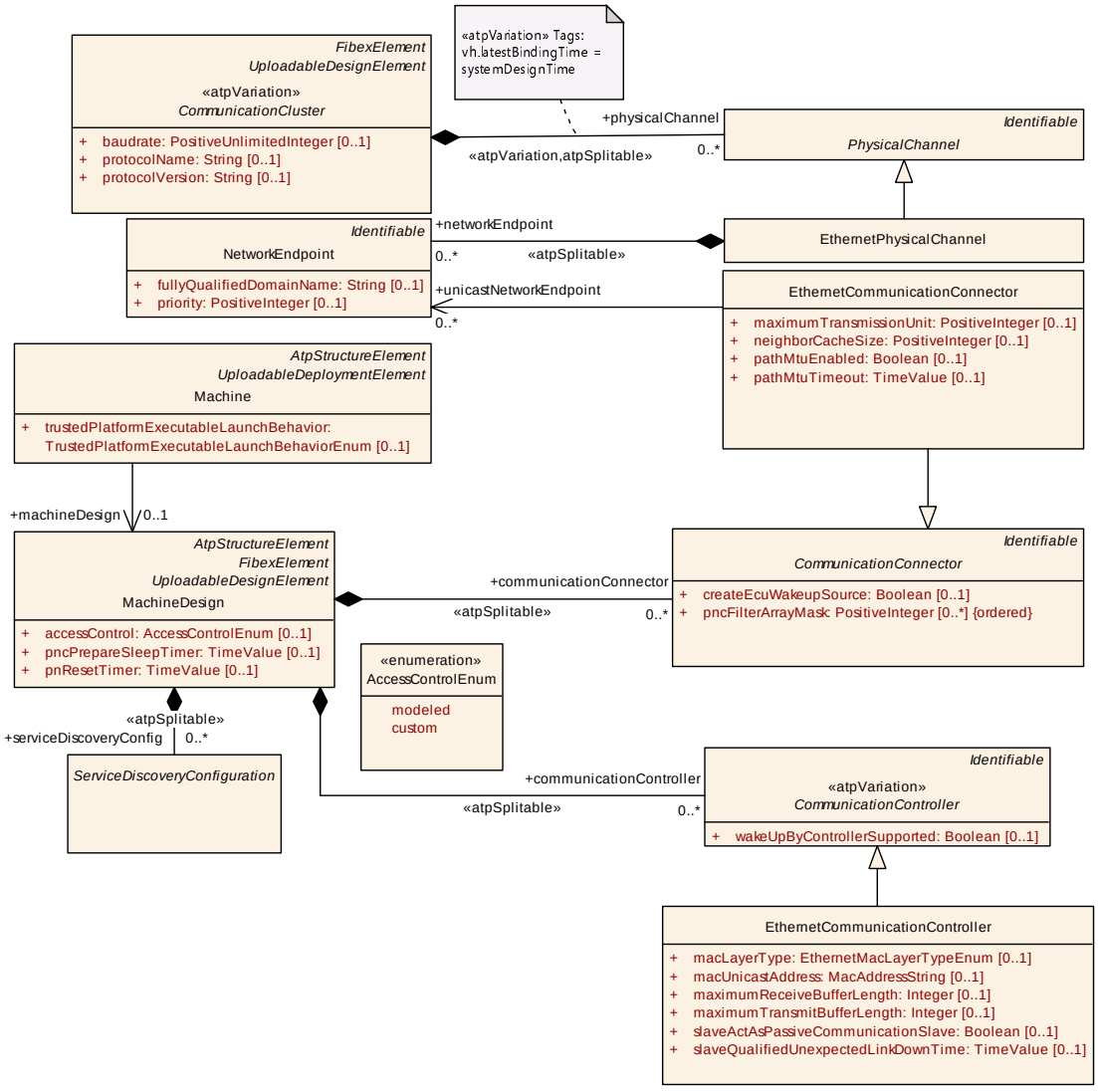


Figure 5.1: MachineDesign

As the *AUTOSAR adaptive platform* focuses on the usage of Ethernet for communication, the task of creating the communication design in the context of the `MachineDesign` boils down to the specification of IP addresses (see section 5.1) and the configuration of service discovery 5.2.

5.1 Network Design aspects of Machine Design

The basic definition of the connectivity of a MachineDesign is created by aggregating the abstract base-class CommunicationConnector in the role communication-Connector. The specific subclass of CommunicationConnector that is used in this context is the EthernetCommunicationConnector.

The EthernetCommunicationConnector is used to connect the MachineDesign with a VLAN that is represented in AUTOSAR by a EthernetPhysicalChannel that is part of an EthernetCluster. Please find more information about the details of EthernetCluster in section 6.2.1.

Class	**MachineDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This meta-class represents the ability to define requirements on a Machine in the context of designing a system. **Tags:** atp.recommendedPackage=MachineDesigns			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Fibex Element*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesign Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
accessControl	AccessControlEnum	0..1	attr	This attribute defines how the access restriction to the Service Instance is defined.
communication Connector	Communication Connector	*	aggr	This aggregation defines the network connection of the machine. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=communicationConnector.shortName
communication Controller	Communication Controller	*	aggr	CommunicationControllers of the Machine that are used for description of 10-Base-T1S topologies **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=communicationController.shortName
ethIpProps	EthIpProps	*	ref	Machine specific IP attributes.
pncPrepare SleepTimer	TimeValue	0..1	attr	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.
pnResetTimer	TimeValue	0..1	attr	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests.
service Discovery Config	ServiceDiscovery Configuration	*	aggr	Set of service discovery configuration settings that are defined on the machine for individual Communication Connectors. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=serviceDiscoveryConfig
tcpIpIcmpProps	EthTcpIpIcmpProps	*	ref	Machine specific ICMP (Internet Control Message Protocol) attributes
tcpIpProps	EthTcpIpProps	*	ref	Machine specific TcpIp Stack attributes.

Table 5.1: MachineDesign

Class	<<atpVariation>> ***CommunicationController*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	The communication controller is a dedicated hardware device by means of which hosts are sending frames to and receiving frames from the communication medium. **Tags:** vh.latestBindingTime=postBuild			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractCanCommunicationController*, EthernetCommunicationController, FlexrayCommunication Controller, UserDefinedCommunicationController			
Aggregated by	EcuInstance.commController, MachineDesign.communicationController			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
wakeUpBy Controller Supported	Boolean	0..1	attr	Defines whether the ECU shall be woken up by this CommunicationController. TRUE: wake up is possible FALSE: wake up is not supported Note: If wakeUpByControllerSupported is set to TRUE the feature shall be supported by both hardware and basic software.

Table 5.2: CommunicationController

[constr_3723]{DRAFT} **Upper multiplicity of reference in the role `MachineDesign.tcpIpProps`** ⌈In the context of `MachineDesign`, the reference in the role `tcpIpProps` shall exist at most once **`at the time when the system design is complete`**.⌋*()*

[constr_3724]{DRAFT} **Upper multiplicity of reference in the role `MachineDesign.tcpIpIcmpProps`** ⌈In the context of `MachineDesign`, the reference in the role `tcpIpIcmpProps` shall exist at most once **`at the time when the system design is complete`**.⌋*()*

[constr_3725]{DRAFT} **Upper multiplicity of reference in the role `MachineDesign.ethIpProps`** ⌈In the context of `MachineDesign`, the reference in the role `ethIpProps` shall exist at most once **`at the time when the system design is complete`**.⌋*()*

Class	***PhysicalChannel*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	This element represents a physical connection (in case of CAN, FlexRay, LIN) or a logical connection (VLAN in case of Ethernet) between communicating devices.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractCanPhysicalChannel*, EthernetPhysicalChannel, FlexrayPhysicalChannel, UserDefinedPhysical Channel			
Aggregated by	*CommunicationCluster*.physicalChannel			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 5.3: PhysicalChannel

Class	<<atpVariation>> **EthernetCluster**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Ethernet-specific cluster attributes. **Tags:** atp.recommendedPackage=CommunicationClusters			
Base	*ARElement*, *ARObject*, *CollectableElement*, *CommunicationCluster*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
couplingPort Connection	CouplingPort Connection	*	aggr	Specification of connections between CouplingElements and EcuInstances. Note: This atpSplitable property has no atp.Splitkey due to atpVariation (PropertySetPattern). **Stereotypes:** atpSplitable; atpVariation **Tags:** vh.latestBindingTime=postBuild
couplingPort StartupActive Time	TimeValue	0..1	attr	The attribute specifies the time in second a coupling port is switched on to enable the host ECU (ECU that maintains an Ethernet switch) to listen to the network for potential network management requests.
couplingPort SwitchoffDelay	TimeValue	0..1	attr	Switch off delay for CouplingPorts in seconds. It denotes the delay of switching off couplingPorts after the request to switch off a couplingPort was issued. (e.g. switch off of Ethernet switch ports).
macMulticast Group	MacMulticastGroup	*	aggr	MacMulticastGroup that is defined for the Subnet (EthernetCluster).

Table 5.4: EthernetCluster

Class	**MacMulticastGroup**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Per EthernetCluster globally defined MacMulticastGroup. One sender can handle many receivers simultaneously if the receivers have all the same macMulticastAddress. The addresses need to be unique for the particular EthernetCluster.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EthernetCluster.macMulticastGroup			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
macMulticast Address	MacAddressString	0..1	attr	A multicast MAC address (Media Access Control address) is a identifier for a group of hosts in a network.

Table 5.5: MacMulticastGroup

[TPS_MANI_03209]{DRAFT} **The meaning of `MachineDesign.accessControl`**
⌈The `MachineDesign.accessControl` defines whether the access control is defined by AUTOSAR means in the Application Design with `receiverIntent` (see [TPS_MANI_01106]) and `senderIntent` (see [TPS_MANI_01107]) or by a custom lists that are created by a non-AUTOSAR process.⌋ *(RS_MANI_00034)*

Enumeration	**AccessControlEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment
Note	This enumeration describes the options for the definition of access restriction to resources.
Aggregated by	MachineDesign.accessControl

▽

△

Enumeration	**AccessControlEnum**
Literal	***Description***
custom	The access restriction to the resource is defined by a non-AUTOSAR process. **Tags:** atp.EnumerationLiteralIndex=1
modeled	The access restriction to the resource is modeled in the AUTOSAR Application Design model or the AUTOSAR Deployment model. **Tags:** atp.EnumerationLiteralIndex=0

Table 5.6: AccessControlEnum

[constr_3320]{DRAFT} **Aggregation of `CommunicationConnector` by `MachineDesign`** ⌈Meta-Class `MachineDesign` shall only aggregate `EthernetCommunicationConnector`s in the role `communicationConnector`. No other subclass of `CommunicationConnector` shall appear in this aggregation.

This rule shall be imposed **`at the time when the machine design is complete`**.⌋*()*

[constr_5324]{DRAFT} **`MachineDesign.communicationController` aggregation restriction** ⌈`MachineDesign` is only allowed to aggregate an `EthernetCommunicationController` in the role `communicationController` **`at the time when the machine design is complete`**.⌋*()*

5.2 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::MachineManifest			
Note	Service Discovery configuration settings for the middleware transport layer.			
Base	*ARObject*			
Subclasses	SomeipServiceDiscovery			
Aggregated by	MachineDesign.serviceDiscoveryConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 5.7: ServiceDiscoveryConfiguration

5.2.1 SOME/IP Service Discovery Configuration

[TPS_MANI_03064]{DRAFT} **SOME/IP Service Discovery message exchange configuration** ⌈`ProvidedServiceInstance`s are announced in SOME/IP by the

server with multicast addressing on a VLAN to a specifically designated IP multicast address (SomeipServiceDiscovery.multicastSdIpAddress) at a specific UDP port number (SomeipServiceDiscovery.someipServiceDiscoveryPort).⌋(*RS_MANI_00019*)

[constr_5045]{DRAFT} **Only one SomeipServiceDiscovery configuration per VLAN is allowed** ⌈Only a single NetworkEndpoint on an EthernetPhysicalChannel (VLAN) is allowed to be referenced by a SomeipServiceDiscovery element in the role multicastSdIpAddress **at the time when the system design is complete**.⌋()

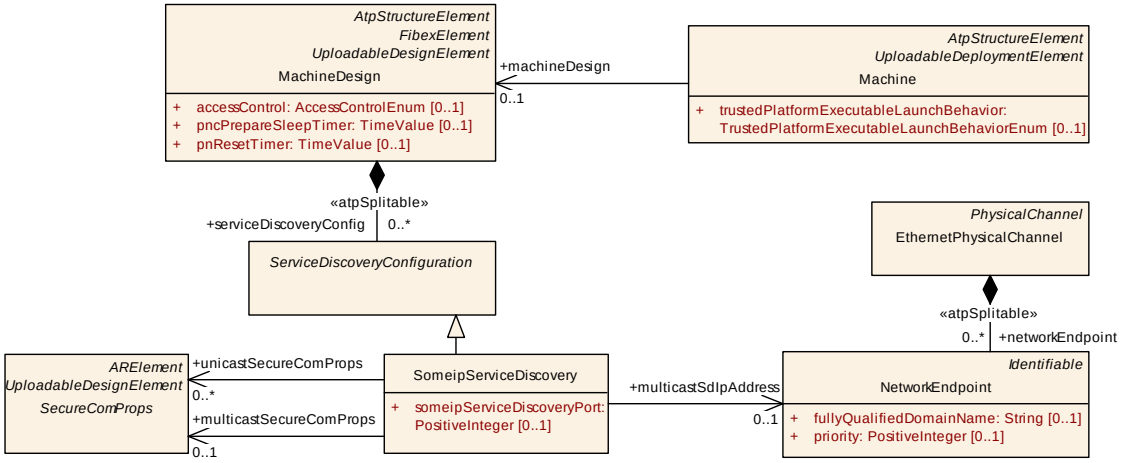


Figure 5.2: SOME/IP Service Discovery Configuration

The SomeipServiceDiscovery is able to reference SecureComProps to define and to configure a security protocol that will 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.

Class	**SomeipServiceDiscovery**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	This meta-class represents a specialization of the generic service discovery for the SOME/IP case.			
Base	*ARObject*, *ServiceDiscoveryConfiguration*			
Aggregated by	MachineDesign.serviceDiscoveryConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
multicastSdIp Address	NetworkEndpoint	0..1	ref	This reference identifies the multicast IP address used for service discovery.

▽

△

Class	**SomeipServiceDiscovery**			
multicastSecure ComProps	SecureComProps	0..1	ref	Reference to a communication security protocol and its configuration settings that will provide communication security for Service Discovery messages that are transmitted using multicast, e.g. FindService message.
someipService DiscoveryPort	PositiveInteger	0..1	attr	This attribute represents the port number reserved for service discovery.
unicastSecure ComProps	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. .

Table 5.8: SomeipServiceDiscovery

[constr_10167]{DRAFT} **Multiplicity of attribute `SomeipServiceDiscovery.someipServiceDiscoveryPort`** ⌈For each `SomeipServiceDiscovery`, the attribute `someipServiceDiscoveryPort` shall exist **at the time when the system design is complete**.⌋*()*

5.3 TcpIp stack configuration properties

The `MachineDesign` references the following elements and allows to set Machine specific TcpIp stack configuration options in the System Design:

- `EthIpProps` - used to configure IPv4 and IPv6
- `EthTcpIpProps` - used to configure TCP and UDP
- `EthTcpIpIcmpProps` - used to configure ICMP

Please note that the System Template [18] defines constraints for the usage of `EthIpProps`, `EthTcpIpProps`, `EthTcpIpIcmpProps`. These constraints are also valid if the `MachineDesign` references these elements.

6 System Design

6.1 Overview

A typical vehicle will most likely be equipped with ECUs developed on the AUTOSAR classic platform and Machines developed on the AUTOSAR adaptive platform. The system design for the entire vehicle has therefore to cover both platform deployment types.

AUTOSAR supports System design descriptions with the possibility to describe Software Components of both AUTOSAR Platforms that will be used in a System and even allows to indicate the service oriented communication between them if possible.

Especially when it comes to the description of the communication behavior of AUTOSAR classic and adaptive ECUs in a harmonized way, the notion of a System Design becomes a special focus point.

All the system design aspects have in common is that they have to cope with both AUTOSAR classic ECUs and adaptive Machines. The basic design aspects of such inter-disciplinary systems have to be already available in the AUTOSAR classic modeling approach otherwise they would not be available to both platforms.

Thus, it is straight forward to take the existing meta-class System as the starting point for the modeling of such mixed systems.

Class	**System**			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The top level element of the System Description. **Tags:** atp.recommendedPackage=Systems			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
fibexElement	FibexElement	*	ref	Reference to ASAM FIBEX elements specifying Communication and Topology. All Fibex Elements used within a System Description shall be referenced from the System Element. atpVariation: In order to describe a product-line, all Fibex Elements can be optional. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=fibexElement.fibexElement, fibex Element.variationPoint.shortLabel vh.latestBindingTime=postBuild
interpolation Routine MappingSet	InterpolationRoutine MappingSet	*	ref	This reference identifies the InterpolationRoutineMapping Sets that are relevant in the context of the enclosing System.

▽

△

Class	System			
mapping	SystemMapping	*	aggr	Aggregation of all mapping aspects relevant in the System Description. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=mapping.shortName, mapping.variation Point.shortLabel vh.latestBindingTime=postBuild
pncVector Length	PositiveInteger	0..1	attr	Length of the partial networking request release information vector (in bytes).
pncVectorOffset	PositiveInteger	0..1	attr	Absolute offset (with respect to the NM-PDU) of the partial networking request release information vector that is defined in bytes as an index starting with 0.
rootSoftware Composition	RootSwComposition Prototype	0..1	aggr	Aggregation of the root software composition, containing all software components in the System in a hierarchical structure. This element is not required when the System description is used for a network-only use-case. atpVariation: The RootSwCompositionPrototype can vary. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=rootSoftwareComposition.shortName, root SoftwareComposition.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime
systemVersion	RevisionLabelString	0..1	attr	Version number of the System Description.

Table 6.1: System

There are use cases to exchange parts of such a SYSTEM_DESIGN_DESCRIPTION in different methodology stages and between different development parties, therefore further system categories are supported by AUTOSAR.

A common approach is for example that the OEM provides a basis for designing an ECU, which is later advanced by the supplier. Therefore Classic AUTOSAR supports System categories like ECU_EXTRACT or ECU_SYSTEM_DESCRIPTION that have only a single ECU in scope. Classic Platform System.categorys are defined in [TPS_SYST_01003].

The AUTOSAR adaptive platform is using the same approach. If an OEM wants to provide design artifacts that are relevant for the configuration of a single Machine all unnecessary information is stripped from the System with category SYSTEM_DESIGN_DESCRIPTION and a definition of the subsystem is provided.

[TPS_MANI_01274]{DRAFT} **Standardized System Category Definitions for the Adaptive Platform** ⌈

Category	Meaning
SYSTEM_DESIGN_DESCRIPTION	The System description that contains design artifacts that are relevant for the Adaptive Platform and Classic Platform.

▽

△

MACHINE_DESIGN_EXTRACT	The System description that contains design artifacts that are relevant for a single Adaptive Platform Machine.

⌋*(RS_MANI_00026)*

[constr_3421]{DRAFT} **Fibex elements applicable for a System of category MACHINE_DESIGN_EXTRACT** ⌈A System with the category MACHINE_DESIGN_EXTRACT is allowed to reference the following fibexElements:

- CommunicationCluster
- MachineDesign
- GlobalTimeDomain
- NmConfig
- SystemMapping that is allowed to contain only a PncMapping

This rule shall be imposed **at the time when the system design is complete**.⌋*()*

6.2 Specification of Communication System Structure

When the communication interaction is designed for a vehicle system the focus is put on the network and the connected ECUs. Whether a specific ECU connected to the network is implemented using AUTOSAR classic or AUTOSAR adaptive does not influence the major communication design.

But of course, it is essential from a car manufacturer point of view whether a specific ECU will be implemented using AUTOSAR classic or adaptive. Thus, already on system design level there is a need to specify the AUTOSAR Platform kind which shall be used to implement an ECU.

In AUTOSAR classic the element EcuInstance is used to define one ECU in the system design.

In AUTOSAR adaptive the element Machine is an entity which already represents a specific ECU Implementation with dedicated configurations for e.g. Processors, ProcessorCores.

The Machine is a model entity which is not in the focus of communication designers and should not be used during system design.

Therefore, the MachineDesign has been introduced, see section 5.

6.2.1 Network connection

The canonical way to specify an IP address is the modeling of a `NetworkEndpoint`, referenced from an `EthernetCommunicationConnector` that is aggregated by `MachineDesign` in the role `communicationConnector`.

In addition to the IP address, the `NetworkEndpoint` may have a *Fully Qualified Domain Name* and a priority.

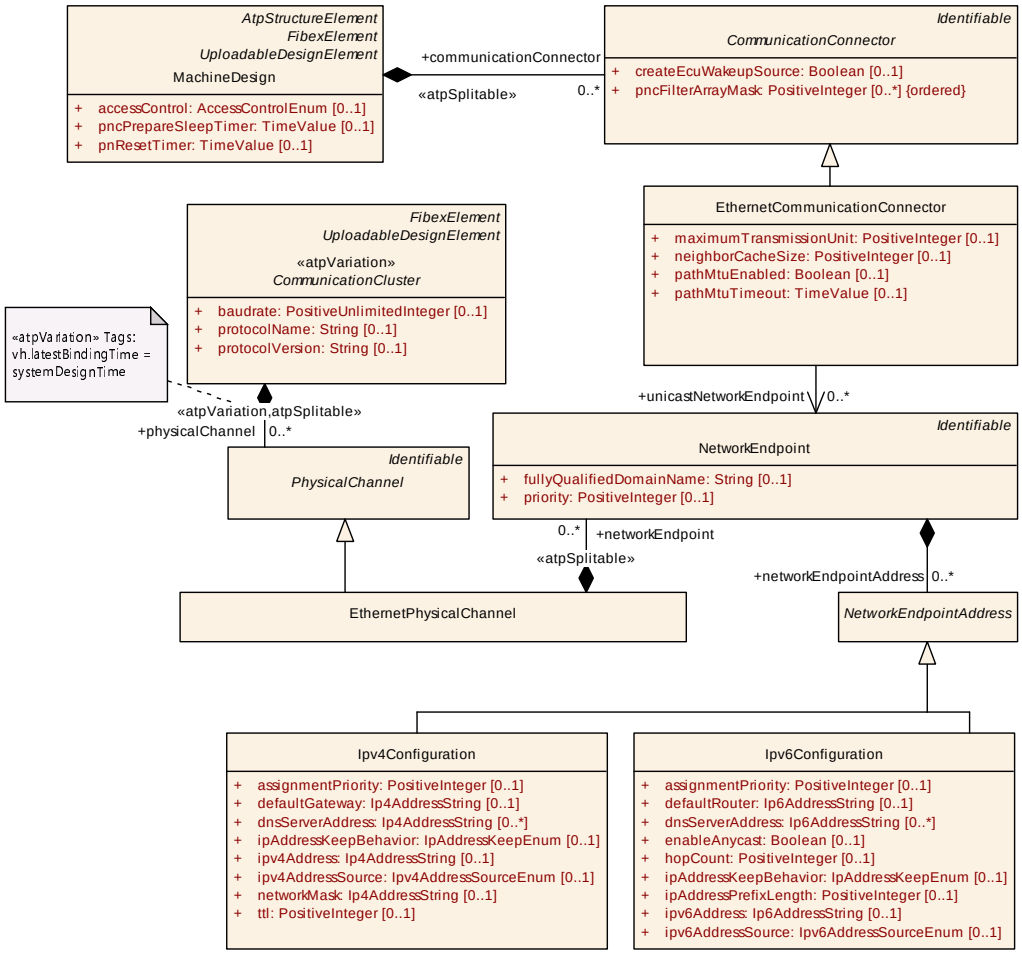


Figure 6.1: Network connection of a MachineDesign

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*			
Aggregated by	EthernetPhysicalChannel.networkEndpoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
fullyQualified DomainName	String	0..1	attr	Defines the fully qualified domain name (FQDN) e.g. some.example.host.
ipSecConfig	IPSecConfig	0..1	aggr	Optional IPSec configuration that provides security services for IP packets.

△

Class	**NetworkEndpoint**			
network Endpoint Address	NetworkEndpoint Address	*	aggr	Definition of a Network Address. **Tags:** xml.name Plural=NETWORK-ENDPOINT-ADDRESSES
priority	PositiveInteger	0..1	attr	Defines the frame priority where values from 0 (best effort) to 7 (highest) are allowed.

Table 6.2: NetworkEndpoint

More precisely, the particular IP address is configured by means of the aggregation of `Ipv4Configuration` or `Ipv6Configuration` in the role `networkEndpointAddress`.

The `NetworkEndpoint` is aggregated by the `EthernetPhysicalChannel` that in turn is aggregated by the `EthernetCluster`.

[TPS_MANI_03052]{DRAFT} **Static IPv4 configuration** ⌈If the value of attribute `ipv4AddressSource` of meta-class `Ipv4Configuration` is set to `Ipv4AddressSourceEnum.fixed` then the `ipv4Address` defines the static IPv4 Address.⌋ *(RS_MANI_00018)*

Class	**Ipv4Configuration**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Internet Protocol version 4 (IPv4) configuration.			
Base	*ARObject*, *NetworkEndpointAddress*			
Aggregated by	NetworkEndpoint.networkEndpointAddress			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
assignment Priority	PositiveInteger	0..1	attr	Priority of assignment (1 is highest). If a new address from an assignment method with a higher priority is available, it overwrites the IP address previously assigned by an assignment method with a lower priority.
defaultGateway	Ip4AddressString	0..1	attr	IP address of the default gateway.
dnsServer Address	Ip4AddressString	*	attr	IP addresses of preconfigured DNS servers. **Tags:** xml.namePlural=DNS-SERVER-ADDRESSES
ipAddressKeep Behavior	IpAddressKeepEnum	0..1	attr	Defines the lifetime of a dynamically fetched IP address.
ipv4Address	Ip4AddressString	0..1	attr	IPv4 Address. Notation: 255.255.255.255. The IP Address shall be declared in case the ipv4AddressSource is FIXED and thus no auto-configuration mechanism is used.
ipv4Address Source	Ipv4AddressSource Enum	0..1	attr	Defines how the node obtains its IP address.
networkMask	Ip4AddressString	0..1	attr	Network mask. Notation 255.255.255.255
ttl	PositiveInteger	0..1	attr	Lifespan of data (0..255). The purpose of the TimeToLive field is to avoid a situation in which an undeliverable datagram keeps circulating on a system.

Table 6.3: Ipv4Configuration

Primitive	**Ip4AddressString**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This is used to specify an IP4 address. Notation: 255.255.255.255 **Tags:** xml.xsd.customType=IP4-ADDRESS-STRING xml.xsd.pattern=(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)|ANY xml.xsd.type=string

Table 6.4: Ip4AddressString

Enumeration	**Ipv4AddressSourceEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology
Note	Defines how the node obtains its IPv4-Address.
Aggregated by	Ipv4Configuration.ipv4AddressSource
Literal	*Description*
autoIp	AutoIP is used to dynamically assign IP addresses at device startup. **Tags:** atp.EnumerationLiteralIndex=0
autoIp_doip	Linklocal IPv4 Address Assignment using DoIP Parameters **Tags:** atp.EnumerationLiteralIndex=2 xml.name=AUTO-IP–DOIP
dhcpv4	DHCP is a service for the automatic IP configuration of a client. **Tags:** atp.EnumerationLiteralIndex=3
fixed	The IP Address shall be declared manually. **Tags:** atp.EnumerationLiteralIndex=4

Table 6.5: Ipv4AddressSourceEnum

[TPS_MANI_03053]{DRAFT} **Static IPv6 configuration** ⌈If the value of attribute `ipv6AddressSource` of meta-class `Ipv6Configuration` is set to `Ipv6AddressSourceEnum.fixed` then the `ipv6Address` defines the static IPv6 Address.⌋ *(RS_MANI_00018)*

Class	**Ipv6Configuration**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Internet Protocol version 6 (IPv6) configuration.			
Base	*ARObject*, *NetworkEndpointAddress*			
Aggregated by	NetworkEndpoint.networkEndpointAddress			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
assignment Priority	PositiveInteger	0..1	attr	Priority of assignment (1 is highest). If a new address from an assignment method with a higher priority is available, it overwrites the IP address previously assigned by an assignment method with a lower priority.
defaultRouter	Ip6AddressString	0..1	attr	IP address of the default router.
dnsServer Address	Ip6AddressString	*	attr	IP addresses of pre configured DNS servers. **Tags:** xml.namePlural=DNS-SERVER-ADDRESSES
enableAnycast	Boolean	0..1	attr	This attribute is used to enable anycast addressing (i.e. to one of multiple receivers).

▽

△

Class	**Ipv6Configuration**			
hopCount	PositiveInteger	0..1	attr	The distance between two hosts. The hop count n means that n gateways separate the source host from the destination host (Range 0..255)
ipAddressKeep Behavior	IpAddressKeepEnum	0..1	attr	Defines the lifetime of a dynamically fetched IP address.
ipAddressPrefix Length	PositiveInteger	0..1	attr	IPv6 prefix length defines the part of the IPv6 address that is the network prefix.
ipv6Address	Ip6AddressString	0..1	attr	IPv6 Address. Notation: FFFF:...:FFFF. The IP Address shall be declared in case the ipv6AddressSource is FIXED and thus no auto-configuration mechanism is used.
ipv6Address Source	Ipv6AddressSource Enum	0..1	attr	Defines how the node obtains its IP address.

Table 6.6: Ipv6Configuration

Primitive	**Ip6AddressString**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This is used to specify an IP6 address. Notation: FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF Alternative notations, short-cuts with duplicate colons like ::, etc. or mixtures using colons and dots, are not allowed. **Tags:** xml.xsd.customType=IP6-ADDRESS-STRING xml.xsd.pattern=[0-9A-Fa-f]{1,4}(:[0-9A-Fa-f]{1,4}){7,7}|ANY xml.xsd.type=string

Table 6.7: Ip6AddressString

Enumeration	**Ipv6AddressSourceEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology
Note	Defines how the node obtains its IPv6-Address.
Aggregated by	Ipv6Configuration.ipv6AddressSource
Literal	*Description*
dhcpv6	DHCP is a service for the automatic IP configuration of a client. **Tags:** atp.EnumerationLiteralIndex=0
fixed	The IP Address shall be declared manually. **Tags:** atp.EnumerationLiteralIndex=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.EnumerationLiteralIndex=2
linkLocal_doip	Linklocal IPv6 Address Assignment using DoIP Parameters **Tags:** atp.EnumerationLiteralIndex=3 xml.name=LINK-LOCAL–DOIP
router Advertisement	IPv6 Stateless Autoconfiguration. **Tags:** atp.EnumerationLiteralIndex=4

Table 6.8: Ipv6AddressSourceEnum

Enumeration	**IpAddressKeepEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology
Note	Defines the behavior after a dynamic IP address has been assigned.
Aggregated by	Ipv4Configuration.ipAddressKeepBehavior, Ipv6Configuration.ipAddressKeepBehavior
Literal	***Description***
forget	After a dynamic IP address has been assigned just use it for this session. **Tags:** atp.EnumerationLiteralIndex=0
storePersistently	After a dynamic IP address has been assigned store the address persistently. **Tags:** atp.EnumerationLiteralIndex=1

Table 6.9: IpAddressKeepEnum

Please note that there is also the possibility to describe a `MacMulticastConfiguration` as `NetworkEndpointAddress` in addition to `Ipv4Configuration` and `Ipv6Configuration` in a system topology description. This may be useful for description of endpoints that are not based on IP, e.g. for streaming protocols like AVB (Audio Video Bridging). But please note that there is no foundation software or ara::com support for such `MacMulticastConfiguration` `NetworkEndpoint`s in Adaptive Autosar. For SOME/IP communication such `NetworkEndpoint`s are excluded by [constr_3288].

6.2.1.1 Support of 10BASE-T1S Network Topologies

Please note that 10BASE-T1S network topology description is supported in the System Design with a `CouplingPortConnection` that points with the `nodePort` reference to `CouplingPort`s that represent the 10Base-T1S PHYs connected to the network.

More details about the modeling of 10BASE-T1S networks can be found in the System Template [18]. Since the same modeling approach is used in the Classic Platform and the Adaptive Platform the detailed description of the meta-model available in the System Template is not repeated in this specification.

6.2.1.2 Support of Ethernet tunneling through CAN XL

On AUTOSAR AP, CAN XL supports only the “mapped tunneling” of Ethernet frames through the physical CAN XL network (i.e. SDU Type 0x05h as specified by the *CiA* standardization of CAN XL). Configuration of this tunneling is done in a similar way as in CP by using the category `CAN_XL` for `EthernetCommunicationConnector`.

[TPS_MANI_03668]{DRAFT} **Supported type of Ethernet tunneling through CAN XL on Adaptive Platform** ⌈On the Adaptive Platform, CAN XL only supports SDU Type 0x05h (Ethernet mapped tunneling). If an AP machine receives a CAN XL frame containing an SDU Type other than 0x05h, it shall discard this frame without processing its content.⌋ *()*

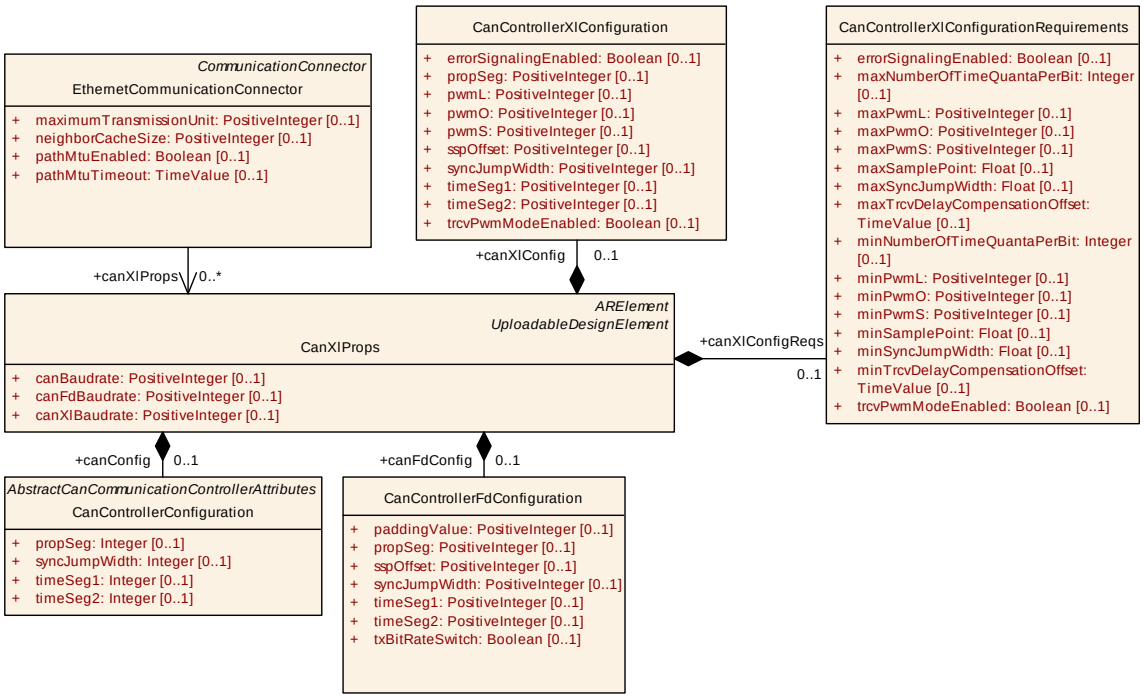


Figure 6.2: Modeling of `CanXlProps`

In other words, CAN XL frames containing (native or tunneled) Classical CAN, CAN FD or CAN XL payload cannot be meaningfully processed by an AP machine as it does not contain a CAN communication stack.

[TPS_MANI_03669]{DRAFT} **Ethernet tunneling through CAN XL** ⌈If an Ethernet connection of a `MachineDesign` shall be realized by Ethernet frames tunneled through a CAN XL network, the `EthernetCommunicationConnector` aggregated by the `MachineDesign` shall set the value of attribute `category` to `CAN_XL` and additionally have a reference in the role `canXlProps` to meta-class `CanXlProps` which contains the required CAN XL specific configuration parameters, i.e. `CanControllerXlConfiguration` and `CanControllerXlConfigurationRequirements`.⌋*()*

Class	**CanXlProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This meta-class is used to configure Machine specific CAN XL attributes. **Tags:** atp.recommendedPackage=CanXlProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
canBaudrate	PositiveInteger	0..1	attr	Specifies the data segment CAN 2.0 baud rate of the CAN XL controller in bits/s.
canConfig	CanController Configuration	0..1	aggr	CAN 2.0 configuration parameters for the CAN XL controller.
canFdBaudrate	PositiveInteger	0..1	attr	Specifies the data segment CAN FD baud rate of the CAN XL controller in bits/s.

▽

△

Class	**CanXlProps**			
canFdConfig	CanControllerFd Configuration	0..1	aggr	CAN FD configuration parameters for the CAN XL controller.
canXlBaudrate	PositiveInteger	0..1	attr	Specifies the data segment CAN XL baud rate of the CAN XL controller in bits/s.
canXlConfig	CanControllerXl Configuration	0..1	aggr	CAN XL configuration parameters for the CAN XL controller.
canXlConfig Reqs	CanControllerXl Configuration Requirements	0..1	aggr	CAN XL configuration parameter requirements for the CAN XL controller.

Table 6.10: CanXlProps

[constr_3693]{DRAFT} **`EthernetCommunicationConnector.category` is set to `CAN_XL`** ⌈If a `EthernetCommunicationConnector` is aggregated by the `MachineDesign` where attribute `category` is set to `CAN_XL`, then a reference from the `EthernetCommunicationConnector` to a `CanXlProps` in the role `canXlProps` shall exist **`at the time when the system design is complete`**.⌋*()*

[constr_3694]{DRAFT} **Existence of `canXlConfig` vs. `canXlConfigReqs`** ⌈For each `CanXlProps`, one of

- `canXlConfig` or
- `canXlConfigReqs`

shall exist **`at the time when the system design is complete`**.⌋*()*

6.2.1.3 IP configuration properties

This chapter introduces meta-classes and attributes that can be used to configure the TCP/IP Stack in the Adaptive Platform.

The Internet Protocol (IP) is the main protocol of the TCP/IP stack and is responsible for delivering datagrams from a source host identified by the source address to one or multiple destination hosts identified by the destination address.

AUTOSAR supports the configuration of

- Internet Protocol of version 4 with `Ipv4Props`. This includes the configuration of
  - ARP (Address Resolution Protocol) with `Ipv4ArpProps`
  - Auto IP with `Ipv4AutoIpProps`
  - Packet fragmentation and reassembly with `Ipv4FragmentationProps`
- Internet Protocol of version 6 with `Ipv6Props`. This includes the configuration of
  - NDP (Neighbor Discovery Protocol) with `Ipv6NdpProps`
  - DHCP (Dynamic Host Configuration Protocol) with `Dhcpv6Props`

– Packet fragmentation and reassembly with `Ipv6FragmentationProps`

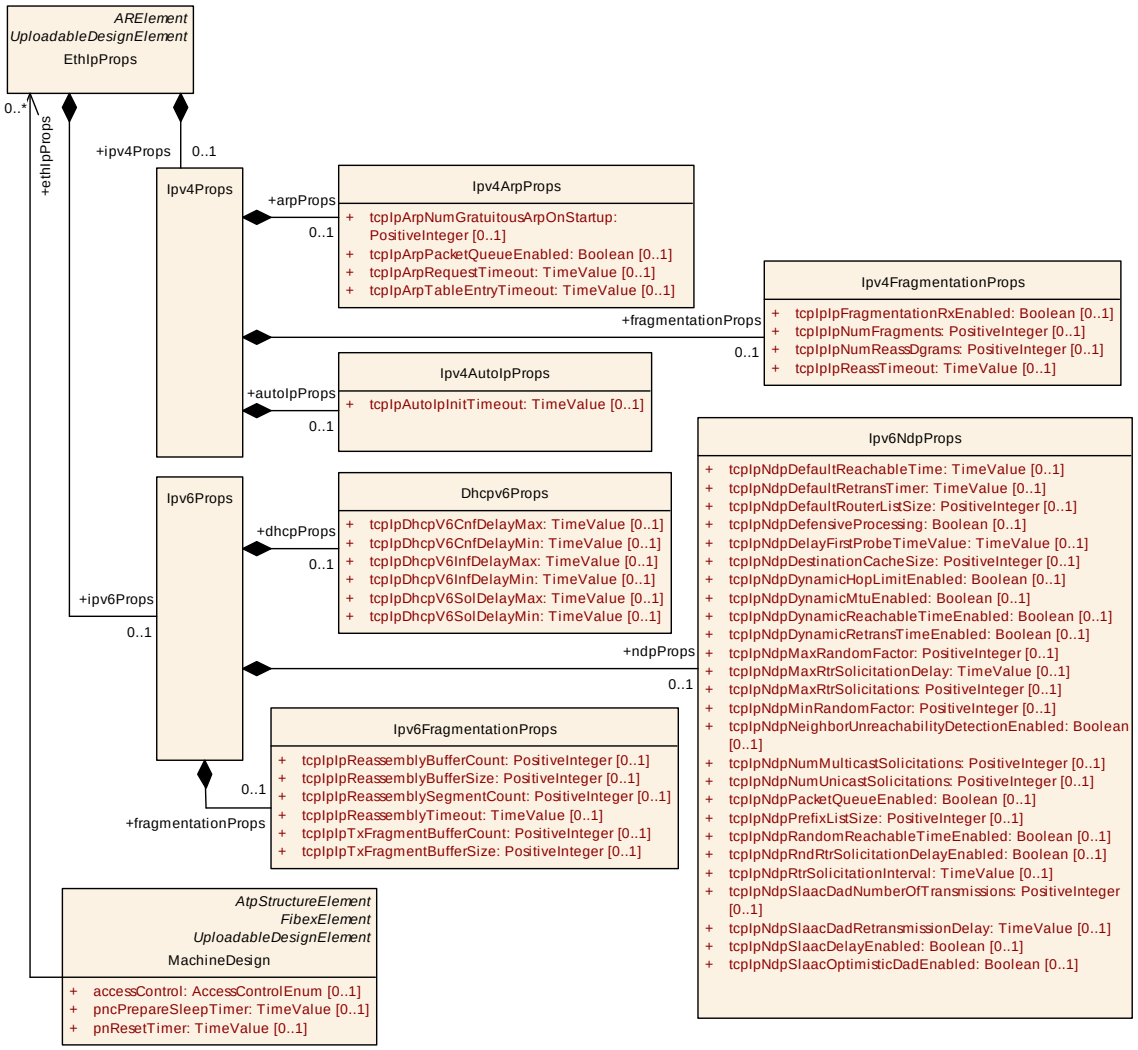


Figure 6.3: Machine specific IP configuration options

Please note that for the configuration of the Internet Protocol the same modeling approach is used as in the Classic Platform (System Template [18]). Therefore, the following constraints defined in the System Template [18] are valid for this specification as well:

- [constr_5126]
- [constr_5127]
- [constr_5128]
- [constr_5129]
- [constr_5130]
- [constr_5131]
- [constr_5132]

- [constr_5133]
- [constr_5134]
- [constr_5135]
- [constr_5136]
- [constr_5137]
- [constr_5138]
- [constr_5139]
- [constr_5140]
- [constr_5141]
- [constr_5142]
- [constr_5143]
- [constr_5144]
- [constr_5145]
- [constr_5146]
- [constr_5147]
- [constr_5148]
- [constr_5149]
- [constr_5151]
- [constr_5152]
- [constr_5153]

Note that in the context of the *AUTOSAR adaptive platform*, the imposition time for these constraints shall be **at the time when the system design is complete**.

Class	**EthIpProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class is used to configure the Machine specific IP attributes. **Tags:** atp.recommendedPackage=EthIpProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ipv4Props	Ipv4Props	0..1	aggr	Configuration options for IPv4.
ipv6Props	Ipv6Props	0..1	aggr	Configuration options for IPv6.

Table 6.11: EthIpProps

Class	Ipv4Props			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for IPv4.			
Base	*ARObject*			
Aggregated by	EthIpProps.ipv4Props			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
arpProps	Ipv4ArpProps	0..1	aggr	Configuration properties for the ARP (Address Resolution Protocol).
autoIpProps	Ipv4AutoIpProps	0..1	aggr	Configuration options for Auto-IP (automatic private IP addressing).
fragmentation Props	Ipv4Fragmentation Props	0..1	aggr	Configuration options for IPv4 packet fragmentation/ reassembly.

Table 6.12: Ipv4Props

Class	Ipv4ArpProps			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Specifies the configuration options for the ARP (Address Resolution Protocol).			
Base	*ARObject*			
Aggregated by	Ipv4Props.arpProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
tcpIpArpNum GratuitousArp OnStartup	PositiveInteger	0..1	attr	This attribute specifies the number of gratuitous ARP replies which shall be sent on assignment of a new IP address.
tcpIpArpPacket QueueEnabled	Boolean	0..1	attr	This attribute enables (TRUE) or disables (FALSE) support of the ARP Packet Queue according to IETF RFC 1122, section 2.3.2.2.
tcpIpArp Request Timeout	TimeValue	0..1	attr	This attribute specifies a timeout in seconds for the validity of ARP requests. After the transmission of an ARP request the TcpIp shall skip the transmission of any further ARP requests to the same destination within a duration of tcpIpArpRequestTimeout seconds. (IETF RFC 1122, section 2.3.2.1).
tcpIpArpTable EntryTimeout	TimeValue	0..1	attr	This attribute specifies the timeout in seconds after which an unused ARP entry is removed.

Table 6.13: Ipv4ArpProps

Class	Ipv4AutoIpProps			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Specifies the configuration options for Auto-IP (automatic private IP addressing).			
Base	*ARObject*			
Aggregated by	Ipv4Props.autoIpProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
tcpIpAutoIpInit Timeout	TimeValue	0..1	attr	This attribute specifies the time in seconds Auto-IP waits at startup, before beginning with ARP probing. This delay is used to give DHCP time to acquire a lease in case a DHCP server is present.

Table 6.14: Ipv4AutoIpProps

Class	**Ipv4FragmentationProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Specifies the configuration options for IPv4 packet fragmentation/reassembly.			
Base	*ARObject*			
Aggregated by	Ipv4Props.fragmentationProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpIp FragmentationRxEnabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support for reassembling of incoming datagrams that are fragmented according to IETF RFC 815 (IP Datagram Reassembly Algorithms).
tcpIpIpNumFragments	PositiveInteger	0..1	attr	Specifies the maximum number of IP fragments per datagram.
tcpIpIpNumReassDgrams	PositiveInteger	0..1	attr	Specifies the maximum number of fragmented IP datagrams that can be reassembled in parallel.
tcpIpIpReassTimeout	TimeValue	0..1	attr	Specifies the timeout in [s] after which an incomplete datagram gets discarded.

Table 6.15: Ipv4FragmentationProps

Class	**Ipv6Props**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for IPv6.			
Base	*ARObject*			
Aggregated by	EthIpProps.ipv6Props			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dhcpProps	Dhcpv6Props	0..1	aggr	Configuration properties for DHCPv6.
fragmentationProps	Ipv6FragmentationProps	0..1	aggr	Configuration properties for IPv6 packet fragmentation/reassembly.
ndpProps	Ipv6NdpProps	0..1	aggr	Configuration properties for the Neighbor Discovery Protocol for IPv6.

Table 6.16: Ipv6Props

Class	**Ipv6FragmentationProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for IPv6 packet fragmentation/reassembly.			
Base	*ARObject*			
Aggregated by	Ipv6Props.fragmentationProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpIp ReassemblyBufferCount	PositiveInteger	0..1	attr	Number of buffers that can be used for fragment reassembly. In case of a reassembly error or if not all fragments are received in time this buffer will be blocked until the specified "Fragment Reassembly Timeout" has been exceeded. A value of 0 disables fragment reassembly.
tcpIpIp ReassemblyBufferSize	PositiveInteger	0..1	attr	Size of each fragment tx buffer in bytes.

▽

△

Class	**Ipv6FragmentationProps**			
tcpIpIp Reassembly SegmentCount	PositiveInteger	0..1	attr	Specifies the maximum number of consecutive data segments that can be managed in each reassembly buffer. If all fragments are received in order, only one segment will be needed. To deal with fragments received out of order this value should be configured bigger than 1.
tcpIpIp Reassembly Timeout	TimeValue	0..1	attr	Specifies the timeout in seconds after which an incomplete datagram gets discarded.
tcpIpIpTx FragmentBuffer Count	PositiveInteger	0..1	attr	These buffers will be used if the IpV6 receives packets from the upper layer that do not fit into the MTU and thus must be fragmented. A value of 0 disables tx fragmentation.
tcpIpIpTx FragmentBuffer Size	PositiveInteger	0..1	attr	Size of each fragment tx buffer in bytes.

Table 6.17: Ipv6FragmentationProps

Class	**Dhcpv6Props**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for DHCPv6.			
Base	*ARObject*			
Aggregated by	Ipv6Props.dhcpProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpDhcp V6CnfDelayMax	TimeValue	0..1	attr	Maximum delay in seconds before sending the first Confirm message. If this value is bigger than the previous minimum delay value a random delay will be chosen from the interval.
tcpIpDhcp V6CnfDelayMin	TimeValue	0..1	attr	Minimum delay in seconds before the first Confirm message will be sent.
tcpIpDhcpV6Inf DelayMax	TimeValue	0..1	attr	Maximum delay in seconds before sending the first Information Request message. If this value is bigger than the previous minimum delay value a random delay will be chosen from the interval.
tcpIpDhcpV6Inf DelayMin	TimeValue	0..1	attr	Minimum delay (s) before the first Information Request message will be sent.
tcpIpDhcpV6Sol DelayMax	TimeValue	0..1	attr	Maximum delay in seconds before sending the first Solicit message. If this value is bigger than the previous minimum delay value a random delay will be chosen from the interval.
tcpIpDhcpV6Sol DelayMin	TimeValue	0..1	attr	Minimum delay (s) before the first Solicit message will be sent.

Table 6.18: Dhcpv6Props

Class	**Ipv6NdpProps**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology
Note	This meta-class specifies the configuration options for the Neighbor Discovery Protocol for IPv6.
Base	*ARObject*
Aggregated by	Ipv6Props.ndpProps

▽

△

Class	**Ipv6NdpProps**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpNdpDefault ReachableTime	TimeValue	0..1	attr	Configuration of the ReachableTime (s) specified in [RFC4861 6.3.2. Host Variables].
tcpIpNdpDefault RetransTimer	TimeValue	0..1	attr	Configures the default value (s) for the RetransTimer variable specified in [RFC4861 6.3.2. Host Variables].
tcpIpNdpDefault RouterListSize	PositiveInteger	0..1	attr	Maximum number of default router entries.
tcpIpNdp Defensive Processing	Boolean	0..1	attr	If enabled the NDP shall only process Neighbor Advertisements which are received in reaction to a previously transmitted Neighbor Solicitation as well as skipping updates to the Neighbor Cache based on received Neighbor Solicitations. If disabled all Neighbor Advertisements and Solicitations shall be processed as specified in RFC4861.
tcpIpNdpDelay FirstProbeTime Value	TimeValue	0..1	attr	Delay before sending the first NUD probe in (s).
tcpIpNdp Destination CacheSize	PositiveInteger	0..1	attr	Maximum number of entries in the destination cache.
tcpIpNdp DynamicHop LimitEnabled	Boolean	0..1	attr	If enabled the default hop limit may be reconfigured based on received Router Advertisements.
tcpIpNdp DynamicMtu Enabled	Boolean	0..1	attr	Allow dynamic reconfiguration of link MTU via Router Advertisements.
tcpIpNdp Dynamic ReachableTime Enabled	Boolean	0..1	attr	If enabled the default Reachable Time value may be reconfigured based on received Router Advertisements.
tcpIpNdp Dynamic RetransTime Enabled	Boolean	0..1	attr	If enabled the default Retransmit Timer value may be reconfigured based on received Router Advertisements.
tcpIpNdpMax RandomFactor	PositiveInteger	0..1	attr	Maximum random factor used for randomization
tcpIpNdpMaxRtr Solicitation Delay	TimeValue	0..1	attr	Maximum delay before the first Router Solicitation will be sent after interface initialization in (s).
tcpIpNdpMaxRtr Solicitations	PositiveInteger	0..1	attr	Maximum number of Router Solicitations that will be sent before the first Router Advertisement has been received.
tcpIpNdpMin RandomFactor	PositiveInteger	0..1	attr	Minimum random factor used for randomization
tcpIpNdp Neighbor Unreachability Detection Enabled	Boolean	0..1	attr	Neighbor Unreachability Detection is used to remove unused entries from the neighbor cache. This feature is a basic feature of NDP and should be turned on.
tcpIpNdpNum Multicast Solicitations	PositiveInteger	0..1	attr	Maximum number of multicast solicitations that will be sent when performing address resolution.
tcpIpNdpNum Unicast Solicitations	PositiveInteger	0..1	attr	Maximum number of unicast solicitations that will be sent when performig Neighbor Unreachability Detection.
tcpIpNdpPacket QueueEnabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of a NDP Packet Queue according to IETF RFC 4861, section 7.2.2.

▽

△

Class	**Ipv6NdpProps**			
tcpIpNdpPrefix ListSize	PositiveInteger	0..1	attr	Maximum number of entries in the on-link prefix list.
tcpIpNdp Random ReachableTime Enabled	Boolean	0..1	attr	If enabled the value of ReachableTime will be multiplied with a random value between MIN_RANDOM_FACTOR and MAX_RANDOM_FACTOR in order to prevent multiple nodes from transmitting at exactly the same time.
tcpIpNdpRndRtr Solicitation DelayEnabled	Boolean	0..1	attr	If enabled the first router solicitation will be delayed randomly from [0...MAX_RTR_SOLICITATION_DELAY]. Otherwise the first router solicitation will be sent after exactly MAX_RTR_SOLICITATION_DELAY milliseconds.
tcpIpNdpRtr Solicitation Interval	TimeValue	0..1	attr	Interval between consecutive Router Solicitations in (s).
tcpIpNdpSlaac DadNumberOf Transmissions	PositiveInteger	0..1	attr	Number of Neighbor Solicitations that have to be unanswered in order to set an autoconfigurated address to PREFERRED (usable) state.
tcpIpNdpSlaac Dad Retransmission Delay	TimeValue	0..1	attr	Sets the maximum value for the address configuration delay (s).
tcpIpNdpSlaac DelayEnabled	Boolean	0..1	attr	If enabled transmission of the first DAD Neighbor Solicitation will be delayed by a random value from [0...MAX_DAD_DELAY].
tcpIpNdpSlaac OptimisticDad Enabled	Boolean	0..1	attr	Enable Optimistic Duplicate Address Detection (DAD) according to RFC4429.

Table 6.19: Ipv6NdpProps

6.2.1.4 TCP and UDP configuration properties

The Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are transport protocols that are based on the Internet Protocol (IP) that is located on the network layer in the OSI model.

The Transport Protocol configuration is supported with the `EthTcpIpProps` meta-class and the two aggregated classes:

- `UdpProps` - used for configuration of Udp
- `TcpProps` - used for configuration of Tcp

Please note that for the configuration of TCP and UDP the same modeling approach is used as in the Classic Platform (System Template [18]). Therefore, the following constraints defined in the System Template [18] are valid for this specification as well:

- [constr_5118]
- [constr_5119]
- [constr_5120]
- [constr_5121]

- [constr_5122]
- [constr_5123]
- [constr_5124]

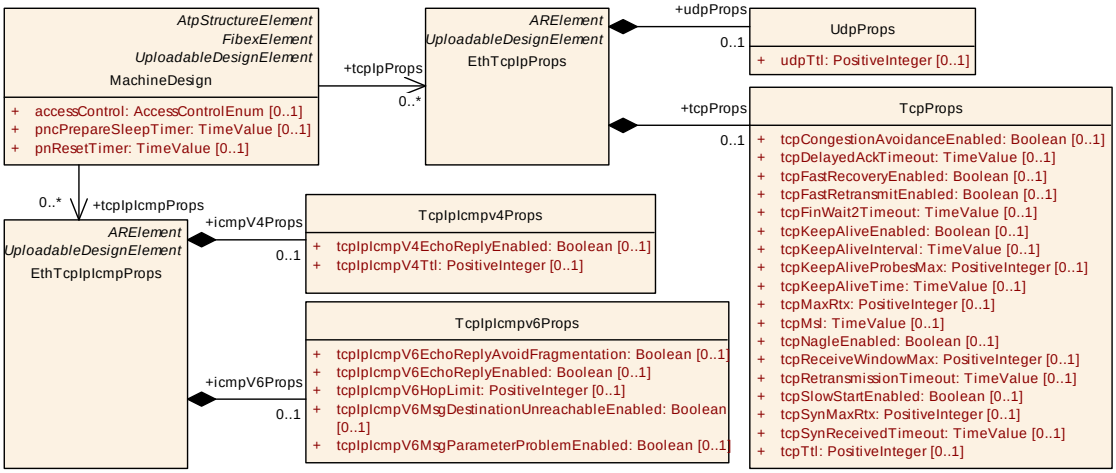


Figure 6.4: Machine specific TCP/UDP and ICMP configuration options

Note that in the context of the *AUTOSAR adaptive platform*, the imposition time for these constraints shall be **at the time when the system design is complete**.

Class	**EthTcpIpProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class is used to configure the Machine specific TcpIp Stack attributes. **Tags:** atp.recommendedPackage=EthTcpIpProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpProps	TcpProps	0..1	aggr	TCP configuration properties
udpProps	UdpProps	0..1	aggr	UDP configuration properties

Table 6.20: EthTcpIpProps

Class	**UdpProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for UDP (User Datagram Protocol).			
Base	*ARObject*			
Aggregated by	EthTcpIpProps.udpProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
udpTtl	PositiveInteger	0..1	attr	Default Time-to-live value of outgoing UDP packets.

Table 6.21: UdpProps

Class	**TcpProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for TCP (Transmission Control Protocol).			
Base	*ARObject*			
Aggregated by	EthTcpIpProps.tcpProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpCongestion Avoidance Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of TCP congestion avoidance algorithm according to IETF RFC 5681.
tcpDelayedAck Timeout	TimeValue	0..1	attr	The maximal time an acknowledgement is delayed for transmission in seconds.
tcpFast Recovery Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of TCP Fast Recovery according to IETF RFC 5681.
tcpFast Retransmit Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of TCP Fast Retransmission according to IETF RFC 5681.
tcpFin Wait2Timeout	TimeValue	0..1	attr	Timeout in [s] to receive a FIN from the remote node (after this node has initiated connection termination), i.e. maximum time waiting in FINWAIT-2 for a connection termination request from the remote TCP.
tcpKeepAlive Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) TCP Keep Alive Probes according to IETF RFC 1122 chapter 4.2.3.6.
tcpKeepAlive Interval	TimeValue	0..1	attr	Specifies the interval in seconds between subsequent keepalive probes.
tcpKeepAlive ProbesMax	PositiveInteger	0..1	attr	Maximum number of times that a TCP Keep Alive is retransmitted before the connection is closed.
tcpKeepAlive Time	TimeValue	0..1	attr	Specifies the time in [s] between the last data packet sent (simple ACKs are not considered data) and the first keepalive probe.
tcpMaxRtx	PositiveInteger	0..1	attr	Maximum number of times that a TCP segment is retransmitted before the TCP connection is closed. This parameter is only valid if tcpRetransmissionTimeout is configured. Note: This parameter also applies for FIN retransmissions.
tcpMsl	TimeValue	0..1	attr	Maximum segment lifetime in [s].
tcpNagle Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of Nagle's algorithm according to IETF RFC 1122 (chapter 4.2.3.4 When to Send Data). If enabled the Nagle's algorithm is activated per default for all TCP sockets, but can be deactivated per Socket (with the attribute TcpTp.nagle Algorithm).
tcpReceive WindowMax	PositiveInteger	0..1	attr	Default value of maximum receive window in bytes.
tcp Retransmission Timeout	TimeValue	0..1	attr	Timeout in [s] before an unacknowledged TCP segment is sent again. If the timeout is disabled, no TCP segments shall be retransmitted.
tcpSlowStart Enabled	Boolean	0..1	attr	Enables (TRUE) or disables (FALSE) support of TCP slow start algorithm according to IETF RFC 5681.
tcpSynMaxRtx	PositiveInteger	0..1	attr	Maximum number of times that a TCP SYN is retransmitted.
tcpSynReceived Timeout	TimeValue	0..1	attr	Timeout in [s] to complete a remotely initiated TCP connection establishment, i.e. maximum time waiting in SYN-RECEIVED for a confirming connection request acknowledgement after having both received and sent a connection request.
tcpTtl	PositiveInteger	0..1	attr	Default Time-to-live value of outgoing TCP packets.

Table 6.22: TcpProps

6.2.1.5 ICMP configuration properties

Internet Control Message Protocol (ICMP) is located in the network layer in the OSI model and is used by network devices to diagnose network communication issues, e.g. it is used to determine whether the data is reaching its intended destination in a timely manner.

AUTOSAR supports the configuration of

- the Internet Control Message Protocol Version 4 for the Internet Protocol Version 4 (IPv4) with `TcpIpIcmpv4Props`
- the Internet Control Message Protocol Version 6 for the Internet Protocol Version 6 (IPv6) with `TcpIpIcmpv6Props`

Please note that for the configuration of ICMP the same modeling approach is used as in the Classic Platform (System Template [18]). Therefore the following constraints defined in the System Template [18] are valid for this specification as well:

- [constr_5125]
- [constr_5154]

Note that in the context of the *AUTOSAR adaptive platform*, the imposition time for these constraints shall be **at the time when the system design is complete**.

Class	**EthTcpIpIcmpProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class is used to configure the Machine specific ICMP (Internet Control Message Protocol) attributes **Tags:** atp.recommendedPackage=EthTcpIcmpProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
icmpV4Props	TcpIpIcmpv4Props	0..1	aggr	ICMPv4 configuration properties
icmpV6Props	TcpIpIcmpv6Props	0..1	aggr	ICMPv6 configuration properties

Table 6.23: EthTcpIpIcmpProps

Class	**TcpIpIcmpv4Props**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for ICMPv4 (Internet Control Message Protocol).			
Base	*ARObject*			
Aggregated by	EthTcpIpIcmpProps.icmpV4Props			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**TcpIpIcmpv4Props**			
tcpIpIcmp V4EchoReply Enabled	Boolean	0..1	attr	This attribute enables or disables transmission of ICMP echo reply message in case of a ICMP echo reception.
tcpIpIcmpV4Ttl	PositiveInteger	0..1	attr	This attribute is only relevant in case that ICMP (Internet Control Message Protocol) is used. It specifies the default Time-to-live value of outgoing ICMP packets.

Table 6.24: TcpIpIcmpv4Props

Class	**TcpIpIcmpv6Props**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	This meta-class specifies the configuration options for ICMPv6 (Internet Control Message Protocol).			
Base	*ARObject*			
Aggregated by	EthTcpIpIcmpProps.icmpV6Props			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpIpIcmp V6EchoReply Avoid Fragmentation	Boolean	0..1	attr	This attribute defines whether the echo reply is only transmitted in case that the incoming ICMPv6 Echo Request (Pings) fits the MTU of the respective interface, i.e. can be transmitted without IPv6 fragmentation.
tcpIpIcmp V6EchoReply Enabled	Boolean	0..1	attr	This attribute enables or disables transmission of ICMP echo reply message in case of a ICMP echo reception.
tcpIpIcmp V6HopLimit	PositiveInteger	0..1	attr	Default Hop-Limit value of outgoing ICMPv6 packets.
tcpIpIcmp V6Msg Destination Unreachable Enabled	Boolean	0..1	attr	This attribute Enables/Disables the transmission of Destination Unreachable Messages.
tcpIpIcmp V6Msg Parameter Problem Enabled	Boolean	0..1	attr	If enabled an ICMPv6 parameter problem message will be sent if a received packet has been dropped due to unknown options or headers that are found in the packet.

Table 6.25: TcpIpIcmpv6Props

6.2.2 Securing Communication with IPsec

IPsec is a protocol suite that provides cryptographic protection for IP datagrams in IPv4 and IPv6 network packets.

IPsec uses a security association to specify security properties that are shared between the communicating parties. The security association defines a relationship between two or more parties and determines which security services will be used to communicate securely. In other words the security association serves as a “contract” between the different devices.

A single security association protects data in one communication direction. Two security associations shall be present to secure traffic in both directions. Each security association can provide encryption, data integrity and data authentication.

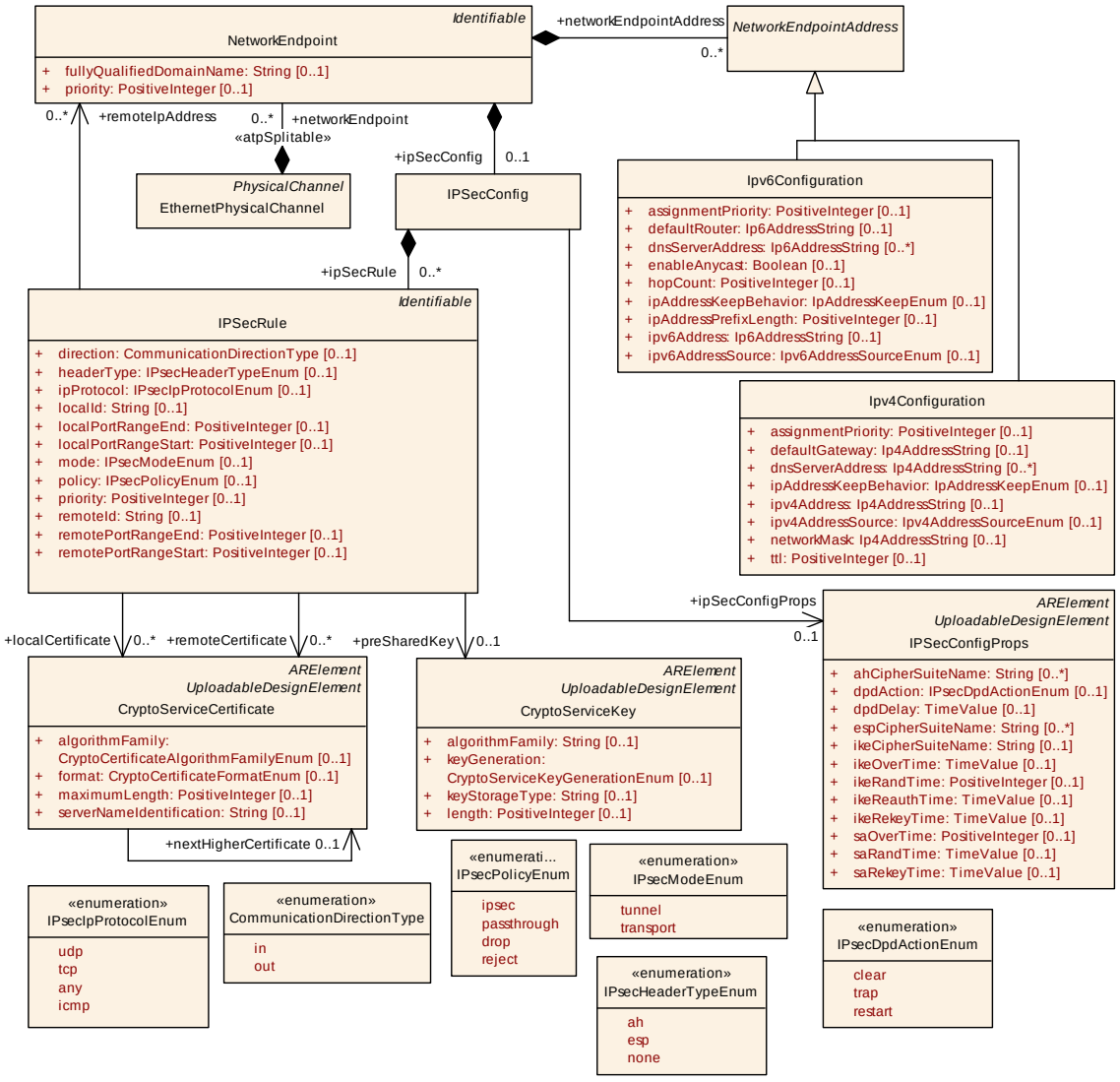


Figure 6.5: IPsec configuration model

In addition, the senders and receivers of IP datagrams can determine the required protection for an IP packet according to IPsec security policies. These are rules that define how datagrams are processed that are received by a device. For example, security policies are used to decide if a particular packet needs to be dropped or needs to be processed by IPsec.

[TPS_MANI_03203]{DRAFT} **Configuration of IPsec** ⌈The `IPSecConfig` meta-class that is aggregated by a `NetworkEndpoint` in the role `ipSecConfig` provides the ability to define IPsec settings that are necessary to configure IPsec security associations and IPsec security policies.⌋ *(RS_MANI_00036)*

[TPS_MANI_03204]{DRAFT} **Definition of `IPSecRule`s** ⌈The `IPSecConfig` meta-class may contain one or several `IPSecRule`s. Each `IPSecRule` defines the network connection that is monitored by IPsec by defining the local endpoint and the remote

endpoint. Each endpoint is defined by the IP Address and the Tcp/Udp Port. The communication direction for which the `IPSecRule` is valid is defined by the `direction` attribute.⌋ *(RS_MANI_00036)*

Class	**IPSecConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	IPsec is a protocol that is designed to provide "end-to-end" cryptographically-based security for IP network connections.			
Base	*ARObject*			
Aggregated by	NetworkEndpoint.ipSecConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ipSecConfig Props	IPSecConfigProps	0..1	ref	Global IPsec configuration settings that are valid for all IPSecRules that are defined on the NetworkEndpoint.
ipSecRule	IPSecRule	*	aggr	IPSec rules and filters that are defined in the IPSecConfig for a specific NetworkEndpoint.

Table 6.26: IPSecConfig

Class	**IPSecRule**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This element defines an IPsec rule that describes communication traffic that is monitored, protected and filtered.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	IPSecConfig.ipSecRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
direction	Communication DirectionType	0..1	attr	This attribute defines the direction in which the traffic is monitored. If this attribute is not set a bidirectional traffic monitoring is assumed.
headerType	IPsecHeaderTypeEnum	0..1	attr	Header type specifying the IPsec security mechanism.
ipProtocol	IPsecIpProtocolEnum	0..1	attr	This attribute defines the relevant IP protocol used in the Security Policy Database (SPD) entry.
localCertificate	CryptoService Certificate	*	ref	This reference identifies the applicable certificate used for a local authentication.
localId	String	0..1	attr	This attribute defines how the local participant should be identified for authentication.
localPortRange End	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines an end value for the local port range. If this attribute is not set then this rule shall be effective for all local ports. Please note that port ranges are currently not supported in the AUTOSAR AP's operating system backend. If AP systems are involved, each IPsec rule may only contain a single port.
localPortRange Start	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines a start value for the local port range. If this attribute is not set then this rule shall be effective for all local ports. Please note that port ranges are currently not supported in the AUTOSAR AP's operating system backend. If AP systems are involved, each IPsec rule may only contain a single port.
mode	IPsecModeEnum	0..1	attr	This attribute defines the type of the connection.

▽

△

Class	**IPSecRule**			
policy	IPsecPolicyEnum	0..1	attr	An IPsec policy defines the rules that determine which type of IP traffic needs to be secured using IPsec and how that traffic is secured.
preSharedKey	CryptoServiceKey	0..1	ref	This reference identifies the applicable cryptograhic key used for authentication.
priority	PositiveInteger	0..1	attr	This attribute defines the priority of the IPSecRule (SPD entry). The processing of entries is based on priority, starting with the highest priority "0".
remote Certificate	CryptoService Certificate	*	ref	This reference identifies the applicable certificate used for a remote authentication.
remoteId	String	0..1	attr	This attribute defines how the remote participant should be identified for authentication.
remoteIp Address	NetworkEndpoint	*	ref	Definition of the remote NetworkEndpoint. With this reference the connection between the local Network Endpoint and the remote NetworkEndpoint is described on which the traffic is monitored.
remotePort RangeEnd	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines an end value for the remote port range. If this attribute is not set then this rule shall be effective for all local ports. Please note that port ranges are currently not supported in the AUTOSAR AP's operating system backend. If AP systems are involved, each IPsec rule may only contain a single port.
remotePort RangeStart	PositiveInteger	0..1	attr	This attribute restricts the traffic monitoring and defines a start value for the remote port range. If this attribute is not set then this rule shall be effective for all local ports. Please note that port ranges are currently not supported in the AUTOSAR AP's operating system backend. If AP systems are involved, each IPsec rule may only contain a single port.

Table 6.27: IPSecRule

Class	**IPSecConfigProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This element holds all the attributes for configuration of IPsec that are independent of specific IPsec rules. **Tags:** atp.recommendedPackage=IPSecConfigProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ahCipherSuite Name	String	*	attr	AH (Authentication Header) algorithm to be used for the connection, e.g. HMAC/SHA2-256
dpdAction	IPsecDpdActionEnum	0..1	attr	This attribute defines what to do if the peer is considered dead. If not configured "restart" shall be assumed.

▽

△

Class	**IPSecConfigProps**			
dpdDelay	TimeValue	0..1	attr	This attribute describes the interval to check the liveness of a peer actively using IKEv2 INFORMATIONAL exchanges. Active DPD checking is only enforced if no IKE or ESP/AH packet has been received for the configured DPD delay. In not configured the value "5 minutes" shall be assumed.
espCipherSuite Name	String	*	attr	ESP (Encapsulating Security Payload) algorithm that provides encryption and optional authentication for the connection, e.g. AES-128+SHA2-256.
ikeCipherSuite Name	String	0..1	attr	IKE encryption/authentication algorithms to be used for the connection.
ikeOverTime	TimeValue	0..1	attr	This attribute describes the hard deadline when an SA becomes invalid in percentage. Example: ikeOverTime of max(ikeReauthTime, ikeRekey Time). Default: 10 %
ikeRandTime	PositiveInteger	0..1	attr	This attribute defines in percentage by how long before the expiration of ikeReauthTime and ikeRekeyTime will be rekeyed/reauthenticated. Default: 10%
ikeReauthTime	TimeValue	0..1	attr	This attribute defines the absolute time after which an IKE SA will be reauthenticated. 0 means reauthentication is disabled.
ikeRekeyTime	TimeValue	0..1	attr	This attribute defines the absolute time after which an IKE SA will be rekeyed. 0 means rekey is disabled.
saOverTime	PositiveInteger	0..1	attr	This attribute describes the hard deadline when an IPsec SA becomes invalid in percentage. Example: saOverTime * saRekeyTime. Default: 110%
saRandTime	TimeValue	0..1	attr	This attribute defines by how long before the expiration of saRekeyTime will be rekeyed.
saRekeyTime	TimeValue	0..1	attr	This attribute defines the absolute time after which an IPsec SA will be rekeyed. 0 means rekey is disabled.

Table 6.28: IPSecConfigProps

Enumeration	**IPsecIpProtocolEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	Definition of supported TcpIp protocols that are supported in Security Policy Database (SPD) entries in IPSec configurations.
Aggregated by	IPSecRule.ipProtocol
Literal	*Description*
any	ANY protocol **Tags:** atp.EnumerationLiteralIndex=3
icmp	Internet Control Message Protocol (ICMP) **Tags:** atp.EnumerationLiteralIndex=2

▽

△

Enumeration	**IPsecIpProtocolEnum**
tcp	TCP Protocol **Tags:** atp.EnumerationLiteralIndex=1
udp	UDP Protocol **Tags:** atp.EnumerationLiteralIndex=0

Table 6.29: IPsecIpProtocolEnum

[constr_5102]{DRAFT} **Usage of remote port ranges in `IPSecRule` is not allowed** ⌈`IPSecRule.remotePortRangeStart` and `IPSecRule.remotePortRangeEnd` shall always be set to the same value **`at the time when the system design is complete`**.⌋*()*

[constr_5103]{DRAFT} **Usage of local port ranges in `IPSecRule` is not allowed** ⌈`IPSecRule.localPortRangeStart` and `IPSecRule.localPortRangeEnd` shall always be set to the same value **`at the time when the system design is complete`**.⌋*()*

The reason for [constr_5102] and [constr_5103] is that port ranges are currently not supported by the AUTOSAR Adaptive Platform operating system backend and each `IPSecRule` is allowed to define only a single local Port and a single remote Port.

[TPS_MANI_03232]{DRAFT} **Definition of general IPsec configuration settings** ⌈General configuration properties that are independent of particular `IPSecRule`s are collected in the `IPSecConfigProps` element that is referenced from the `IPSecConfig` in the role `ipSecConfigProps`.⌋*(RS_MANI_00036)*

[TPS_MANI_03205]{DRAFT} **IPsec policy** ⌈The `IPSecRule.policy` attribute defines how IP packets are handled that are going over the network connection defined by the `IPSecRule`. In detail, it defines whether the IP packet is processed by IPsec or not.⌋*(RS_MANI_00036)*

Enumeration	**IPsecPolicyEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	Defines the filter actions that are supported by IPsec.
Aggregated by	IPSecRule.policy
Literal	***Description***
drop	Signifying that packets should be discarded **Tags:** atp.EnumerationLiteralIndex=3
ipsec	Signifying that packets should be protected. **Tags:** atp.EnumerationLiteralIndex=1
passthrough	Signifying that no IPsec processing should be done at all. **Tags:** atp.EnumerationLiteralIndex=2
reject	Signifying that packets should be discarded and a diagnostic ICMP returned. **Tags:** atp.EnumerationLiteralIndex=4

Table 6.30: IPsecPolicyEnum

IPsec can be configured to operate in two different modes, Tunnel and Transport mode. With tunnel mode, the entire IP packet is protected by IPsec. IPsec wraps the original packet, encrypts it and adds a new IP header to it.

The tunnel mode is most commonly used between VPN gateways and the IP addresses of the newly added outer IP header are that of the VPN Gateways. In other words the traffic between the two VPN Gateways is protected and each gateway acts as a proxy for the hosts behind it.

The transport mode provides the protection of the Data Payload of the IP datagram with an AH or ESP header. The IP Header remains the same and IPsec inserts its header between the IP header and the upper level headers.

The IPsec transport mode can be used when securing traffic between two hosts or between a host and a VPN gateway.

[TPS_MANI_03233]{DRAFT} **IPsec mode** ⌈The `IPSecRule.mode` attribute defines whether the IP packet is processed in the `transport` or `tunnel` mode.⌋*(RS_MANI_00036)*

Please note that AUTOSAR currently supports only the `transport mode` as configuration option.

Enumeration	**IPsecModeEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This enumeration describes the supported IPSec modes.
Aggregated by	IPSecRule.mode
Literal	***Description***
transport	Signifying that the IPSec transport mode is used. With the transport mode the original IP header is retained and only the IP payload and ESP trailer is encrypted. **Tags:** atp.EnumerationLiteralIndex=1
tunnel	Signifying that the IPSec tunnel mode is used. With tunnel mode, the entire original IP packet is protected by IPSec. This means IPSec wraps the original packet, encrypts it, adds a new IP header and sends it to the other side. **Tags:** atp.EnumerationLiteralIndex=0

Table 6.31: IPsecModeEnum

IPsec uses two protocols:

- AH - Authentication Header
- ESP - Encapsulating Security Payload

The AH protocol provides a mechanism for authentication only and authenticates the entire IP packet, including the outer IP header.

The ESP protocol provides data confidentiality (encryption) and/or authentication (data integrity, data origin authentication, and replay protection).

When ESP is used in transport mode, the IP payload is encrypted and the original IP header is moved to the front of the message. The ESP header is inserted after the

IP header and is signed together with the IP payload. The original IP header remains unprotected.

When ESP is used in tunnel mode a new IP Header is created and the ESP header is added in front of the original IP Packet. The entire original IP packet is encrypted and signed in this mode.

[TPS_MANI_03206]{DRAFT} **IPsec AH and ESP protocol configuration** ⌈In the `IPSecRule` it is possible to define the IPsec protocol that shall be used to protect IP packets that are going over the defined network connection. The attribute `headerType` defines whether AH, ESP or neither one is used.⌋*(RS_MANI_00036)*

Enumeration	**IPsecHeaderTypeEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	IPsec Header Type options
Aggregated by	IPSecRule.headerType
Literal	***Description***
ah	Authentication Header (AH) **Tags:** atp.EnumerationLiteralIndex=0
esp	Encapsulating Security Payloads (ESP) **Tags:** atp.EnumerationLiteralIndex=1
none	No header **Tags:** atp.EnumerationLiteralIndex=2

Table 6.32: IPsecHeaderTypeEnum

[TPS_MANI_03234]{DRAFT} **IPsec AH and ESP CipherSuites** ⌈The attributes `ahCipherSuiteName` and `espCipherSuiteName` define the supported AH and ESP algorithms.⌋*(RS_MANI_00036)*

The naming convention for `ahCipherSuiteName`, `espCipherSuiteName` and `IPSecConfigProps.ikeCipherSuiteName` shall follow the naming convention for cryptographic primitives that is defined in [16].

[TPS_MANI_03207]{DRAFT} **IPsec Internet Key Exchange protocol configuration** ⌈In the `IPSecRule` it is possible to define how IKE protocol authenticates the remote party and how the local party authenticates itself to the remote party. In other words both sides use the same method. The usage of the `IPSecRule.preSharedKey` reference defines that the pre-shared key is used. The usage of the `IPSecRule.localCertificate` and `IPSecRule.remoteCertificate` defines that Digital Signature Authentication is used.⌋*(RS_MANI_00036)*

Please note that the supported IKE CipherSuites are configured with the `IPSecConfigProps.ikeCipherSuiteName`. The `IPSecConfigProps` contains additional IKE specific configuration settings.

Enumeration	**IPsecDpdActionEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	Potential Dead Peer Detection (Dpd) Actions
Aggregated by	IPSecConfigProps.dpdAction
Literal	***Description***
clear	Deletes the SA. **Tags:** atp.EnumerationLiteralIndex=0
restart	Immediately tries to establish the connection. **Tags:** atp.EnumerationLiteralIndex=2
trap	tries to establish the connection after traffic is sent to the peer. **Tags:** atp.EnumerationLiteralIndex=1

Table 6.33: IPsecDpdActionEnum

[TPS_MANI_03208]{DRAFT} **Protection of `AdaptivePlatformServiceInstance` by IPsec** ⌈To describe the protection of an `AdaptivePlatformServiceInstance` by IPsec the `AdaptivePlatformServiceInstance` needs to be mapped by a `ServiceInstanceToMachineMapping` to an `EthernetCommunicationConnector` that points with the `unicastNetworkEndpoint` to a `NetworkEndpoint` that aggregates the `IPSecConfig` that in turn describes IPsec Security Associations.⌋ *(RS_MANI_00036)*

Please note that IP Multicast protection by IPsec is not supported. It is by intention not possible to model the IPsec protection of IP Multicast communication since the IP Multicast address is defined in the `SomeipProvidedEventGroup` by the two attributes `ipv4MulticastIpAddress` and `ipv6MulticastIpAddress`. The `NetworkEndpoint` element is used for description of IP Unicast Endpoints only. This means that only the IP Unicast communication of an `AdaptivePlatformServiceInstance` that is described according to [TPS_MANI_03208] will be protected by IPsec.

6.2.3 MACsec

MACsec (Media Access Control security) is defined by IEEE standard 802.1AE and operates at the medium access control layer and defines connectionless data confidentiality and integrity for media access independent protocols.

The MACsec standard specifies a set of protocols to meet the security requirements for protecting data traversing Ethernet LANs.

MACsec allows unauthorized LAN connections to be identified and excluded from communication within the network. In common with IPsec and TLS, MACsec defines a security infrastructure to provide data confidentiality, data integrity and data origin authentication.

With security on the MAC layer, protocols not based on IP as well as protocols using multicast can be protected by MACsec.

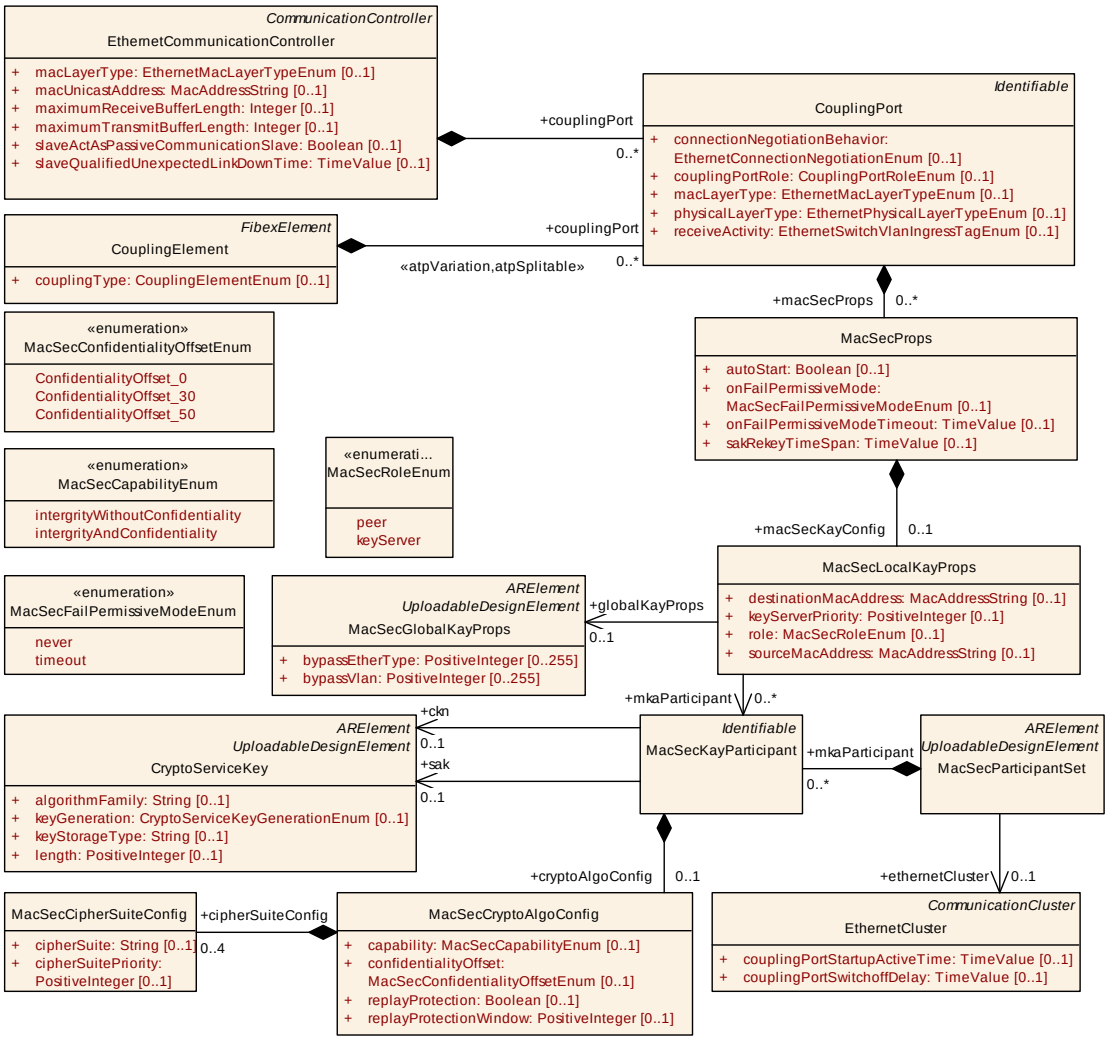


Figure 6.6: MACsec configuration

The MACsec standard relies on the usage of authentic partners, which are called MAC Security Entities (SecYs). The standard organizes this by MACsec Protocol Data Units (MPDU), which resides in the MAC layer (ISO/OSI layer 2). The MAC Security Key Agreement Entity (KaY) will take care of properly configuring and managing of the MAC Security Entities to enable the secure communication channel.

In the System Design model the configuration of MACsec is supported on `CouplingPort`s that in turn are aggregated either by a `CouplingElement` or by an `EthernetCommunicationController`. This allows the configuration of MACsec on a switch port or on an `EthernetCommunicationController` of a `MachineDesign`. Please note that the same modeling approach is used in the Classic Platform System Template as well.

[TPS_MANI_03317]{DRAFT} **MACsec configuration** ⌈A `CouplingPort` that aggregates the `MacSecProps` in the role `macSecProps` defines a MAC Security Entity.⌋ *(RS_MANI_00036)*

Please note that constraint [constr_5361] from the System Template is also valid in the System Design model of the Adaptive Platform.

Please note that constraint [constr_3726] from the System Template is also valid in the System Design model of the Adaptive Platform.

[TPS_MANI_03318]{DRAFT} **MAC Security Key Agreement Entity configuration** ⌈The `MacSecLocalKayProps` element that is referenced by the `MacSecProps` in the role `macSecKayConfig` in combination with `MacSecGlobalKayProps` that is referenced by `MacSecLocalKayProps` in the role `globalKayProps` defines the configuration settings for the MACsec Key Agreement that is responsible for the peer discovery and key negotiation to secure the Ethernet link.⌋*(RS_MANI_00036)*

Class	**MacSecProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class allows to configure MACsec (Media access control security) and the MKA (MACsec Key Agreement) for the CouplingPort (PHY). **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	CouplingPort.macSecProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
autoStart	Boolean	0..1	attr	This attribute defines how the Port Access Entity (PAE) is started: • true := Autostart • false := Manual Start **Tags:** atp.Status=candidate
macSecKay Config	MacSecLocalKayProps	0..1	aggr	Properties to configure the MKA instance (KaY) for a controlled CouplingPort (PaE). **Tags:** atp.Status=candidate
onFail Permissive Mode	MacSecFailPermissive ModeEnum	0..1	attr	This attribute sets the behavior of the Port Access Entity in case MACsec does not succeed. **Tags:** atp.Status=candidate
onFail Permissive ModeTimeout	TimeValue	0..1	attr	Timeout in seconds to enable the controlled port in case onFailPermissiveMode is set to Timeout. **Tags:** atp.Status=candidate
sakRekeyTime Span	TimeValue	0..1	attr	Time in seconds to trigger the rekey of an in use SAK (Static Secure Association key). If set to 0, the rekey will not be triggered after a time span. **Tags:** atp.Status=candidate

Table 6.34: MacSecProps

Class	**MacSecLocalKayProps**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	Configuration of the MAC Security Key Agreement Entity (KaY). **Tags:** atp.Status=candidate
Base	*ARObject*
Aggregated by	MacSecProps.macSecKayConfig

▽

△

Class	**MacSecLocalKayProps**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destinationMac Address	MacAddressString	0..1	attr	This attribute defines the destination MAC Address that is used to calculate the ICV (Integrity Check Value). **Tags:** atp.Status=candidate
globalKayProps	MacSecGlobalKay Props	0..1	ref	Reference to properties that are shared between MAC Security Key Agreement Entities. **Tags:** atp.Status=candidate
keyServer Priority	PositiveInteger	0..1	attr	This attribute defines the key-server priority. **Tags:** atp.Status=candidate
mkaParticipant	MacSecKayParticipant	*	ref	Reference to MKA participant settings supported on the CouplingPort. **Tags:** atp.Status=candidate
role	MacSecRoleEnum	0..1	attr	Role of the MAC Security Key Agreement Entity **Tags:** atp.Status=candidate
sourceMac Address	MacAddressString	0..1	attr	This attribute defines the source MAC Address that is used to calculate the ICV (Integrity Check Value). **Tags:** atp.Status=candidate

Table 6.35: MacSecLocalKayProps

Class	**MacSecGlobalKayProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	Configuration of the MAC Security Key Agreement Entity properties that are shared by different KaY configurations. **Tags:** atp.Status=candidate atp.recommendedPackage=MacSecGlobalKayProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bypassEther Type	PositiveInteger	0..255	attr	This attribute is used to define EtherTypes that are bypassed by MACsec. The providedEtherType will not be MACsec protected. **Tags:** atp.Status=candidate
bypassVlan	PositiveInteger	0..255	attr	This attribute is used to define VLAN-IDs that are bypassed by MACsec. The provided VLAN-IDs will not be MACsec protected. (VLAN-ID 0 is interpreted as no-VLAN –> Bypass untagged traffic) **Tags:** atp.Status=candidate

Table 6.36: MacSecGlobalKayProps

Class	MacSecParticipantSet			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	Collection of MACsec Kay Participants on an Ethernet Link. **Tags:** atp.Status=candidate atp.recommendedPackage=MacSecKayParticipantSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ethernetCluster	EthernetCluster	0..1	ref	Reference to the EthernetCluster (Link) on which the KaY participants are located **Tags:** atp.Status=candidate
mkaParticipant	MacSecKayParticipant	*	aggr	Configuration of a MKA Participant. **Tags:** atp.Status=candidate

Table 6.37: MacSecParticipantSet

Class	MacSecKayParticipant			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class configures a MKA participant. **Tags:** atp.Status=candidate atp.recommendedPackage=MacSecKayParticipants			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	MacSecParticipantSet.mkaParticipant			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ckn	CryptoServiceKey	0..1	ref	Reference to the key where the ckn (Connectivity Association key) is stored. **Tags:** atp.Status=candidate
cryptoAlgo Config	MacSecCryptoAlgo Config	0..1	aggr	Cryptography that is used by the MKA Participant. **Tags:** atp.Status=candidate
sak	CryptoServiceKey	0..1	ref	Reference to the key where SAK shall be stored. **Tags:** atp.Status=candidate

Table 6.38: MacSecKayParticipant

Class	MacSecCryptoAlgoConfig			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class defines the cryptography configuration for MACsec. **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	MacSecKayParticipant.cryptoAlgoConfig			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
capability	MacSecCapabilityEnum	0..1	attr	This attribute defines the MACsec capability. **Tags:** atp.Status=candidate
cipherSuite Config	MacSecCipherSuite Config	0..4	aggr	Cipher suite configuration to use with MACsec. **Tags:** atp.Status=candidate

△

Class	**MacSecCryptoAlgoConfig**			
confidentiality Offset	MacSecConfidentiality OffsetEnum	0..1	attr	The MACsec confidentiality offset specifies the number of bytes starting from the frame header. MACsec encrypts only the bytes after the offset in a frame. **Tags:** atp.Status=candidate
replayProtection	Boolean	0..1	attr	This attribute is used to configure the MACsec replay protection. **Tags:** atp.Status=candidate
replayProtection Window	PositiveInteger	0..1	attr	In case replay protection is active, this attribute defines the replay protection window. **Tags:** atp.Status=candidate

Table 6.39: MacSecCryptoAlgoConfig

Class	**MacSecCipherSuiteConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class defines the cipher suite configuration to use with MACsec. cipherSuitePriority is present in case the MKA instance acts as a Key Server to select the cipher suite to use for MACsec. **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	MacSecCryptoAlgoConfig.cipherSuiteConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cipherSuite	String	0..1	attr	Cipher Suite to use for MACsec. **Tags:** atp.Status=candidate
cipherSuite Priority	PositiveInteger	0..1	attr	In case the MKA instance acts as a Key Server, the priority is used to select the Cipher Suite to use with MACsec from the supported Ciphers. **Tags:** atp.Status=candidate

Table 6.40: MacSecCipherSuiteConfig

[TPS_MANI_03319]{DRAFT} **Standardized values for the attribute `cipherSuite` of meta-class `MacSecCipherSuiteConfig`** ⌈The following values for `MacSecCipherSuiteConfig.cipherSuite` are reserved by the AUTOSAR standard:

- GCM-AES-128
- GCM-AES-256
- GCM-AES-XPN-128
- GCM-AES-XPN-256

⌋*(RS_MANI_00036)*

[TPS_MANI_03320]{DRAFT} **Semantics of `MacSecCipherSuiteConfig.cipherSuitePriority`** ⌈The `MacSecCryptoAlgoConfig` can define up to four `MacSecCipherSuiteConfig`s. If more then one `MacSecCipherSuiteConfig` is defined then the `cipherSuitePriority` decides about the chosen cipher. The `cipherSuitePriority` of value 1 means the highest priority and 4 means the lowest priority.⌋*(RS_MANI_00036)*

Enumeration	**MacSecConfidentialityOffsetEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This enum defines the MACsec capability options. **Tags:** atp.Status=candidate
Aggregated by	MacSecCryptoAlgoConfig.confidentialityOffset
Literal	***Description***
Confidentiality Offset_0	confidentiality offset of 0. **Tags:** atp.EnumerationLiteralIndex=0 xml.name=CONFIDENTIALITY-OFFSET–0
Confidentiality Offset_30	confidentiality offset of 30. **Tags:** atp.EnumerationLiteralIndex=1 xml.name=CONFIDENTIALITY-OFFSET–30
Confidentiality Offset_50	confidentiality offset of 50. **Tags:** atp.EnumerationLiteralIndex=2 xml.name=CONFIDENTIALITY-OFFSET–50

Table 6.41: MacSecConfidentialityOffsetEnum

Enumeration	**MacSecCapabilityEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This enum defines the MACsec capability options. **Tags:** atp.Status=candidate
Aggregated by	MacSecCryptoAlgoConfig.capability
Literal	***Description***
intergrityAnd Confidentiality	Option that ensures confidentiality and integrity **Tags:** atp.EnumerationLiteralIndex=1
intergrityWithout Confidentiality	Option that ensures integrity without confidentiality **Tags:** atp.EnumerationLiteralIndex=0

Table 6.42: MacSecCapabilityEnum

Enumeration	**MacSecRoleEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This enum defines the MACsec Role options. **Tags:** atp.Status=candidate
Aggregated by	MacSecLocalKayProps.role
Literal	***Description***
keyServer	Port acts in the KeyServer role **Tags:** atp.EnumerationLiteralIndex=1
peer	Port acts in the peer role **Tags:** atp.EnumerationLiteralIndex=0

Table 6.43: MacSecRoleEnum

Enumeration	**MacSecFailPermissiveModeEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	Behavior options of the Port Access Entity in case MACsec does not succeed. **Tags:** atp.Status=candidate
Aggregated by	MacSecProps.onFailPermissiveMode
Literal	***Description***
never	The controlled port will never be set to enabled if the participants cannot establish and successfully use a MACsec Secure Channel. **Tags:** atp.EnumerationLiteralIndex=0
timeout	The controlled port will be set to enabled and MACsec will not be used in the port if the timeout value (onFailPermissiveModeTimeout) is reached and the following conditions apply: - A participant belonging to the same CA was recognized and authenticated. - A secure channel could be established. - Both participants can transmit and receive MACsec protected traffic through the SC. **Tags:** atp.EnumerationLiteralIndex=1

Table 6.44: MacSecFailPermissiveModeEnum

For MACsec to start up, a key distribution based on MAC Security Key Agreement runs first. The pre-shared key is identified by the `ckn`. While one might choose only to communicate encrypted, the key distribution itself needs to be unencrypted but integrity protected.

In addition, user may choose to allow additional Ethernet frames unprotected, e.g., for installing keys into ECUs. These frames can be selected via so-called bypass rules based on e.g., EtherType or VLAN-ID defined in `MacSecGlobalKayProps`.

Each `MacSecKayParticipant` that is referenced by `MacSecLocalKayProps` in the role `mkaParticipant` defines the Connectivity Association Key and the `MacSecCryptoAlgoConfig`. The `MacSecKayParticipant` settings in the connectivity association need to match on both `CouplingPort`s of the link to enable MACsec.

For better control and predictability, the MAC Security Key Agreement participant that will get the KeyServer role (the node generating the MACsec Key for this link) can be configured by `role`.

6.2.4 Partial Network

AUTOSAR supports power saving during vehicle operation time with the partial networking mechanism. This mechanism allows shutting down and starting up the bus communication interfaces of groups of ECUs (Partial Network Cluster) during normal bus communication.

On the VFB Level Partial Networks are represented by Virtual Function Clusters and are described with `PortGroup`s. The Virtual Function Cluster groups the communication necessary to realize one or more vehicle functions that can become activated/deactivated during normal vehicle operation. The Virtual Function Clusters are mapped onto Partial Network Clusters.

[TPS_MANI_03224]{DRAFT} **Modeling of a Partial Network Cluster** ⌈A Partial Network Cluster is modeled with the `PncMapping` element and is identified by the `pncIdentifier`. The `PncMapping` defines the collection of `AdaptivePlatformServiceInstance`s that are participating in the partial network with the `PncMapping`.`serviceInstance` reference.⌋*(RS_MANI_00062)*

[TPS_MANI_03225]{DRAFT} **References to VLANs in `PncMapping`** ⌈There are two ways for a `PncMapping` to relate to a VLAN or an untagged channel:

- via the reference from `PncMapping` to `PhysicalChannel` in the role `physicalChannel`,
- via a `ServiceInstanceToMachineMapping` that maps the `AdaptivePlatformServiceInstance` and that also refers an `EthernetCommunicationConnector` that in turn is connected to an `EthernetPhysicalChannel` via the `unicastNetworkEndpoint`.

The two options are not mutually exclusive, they can exist at the same time.⌋*(RS_MANI_00062)*

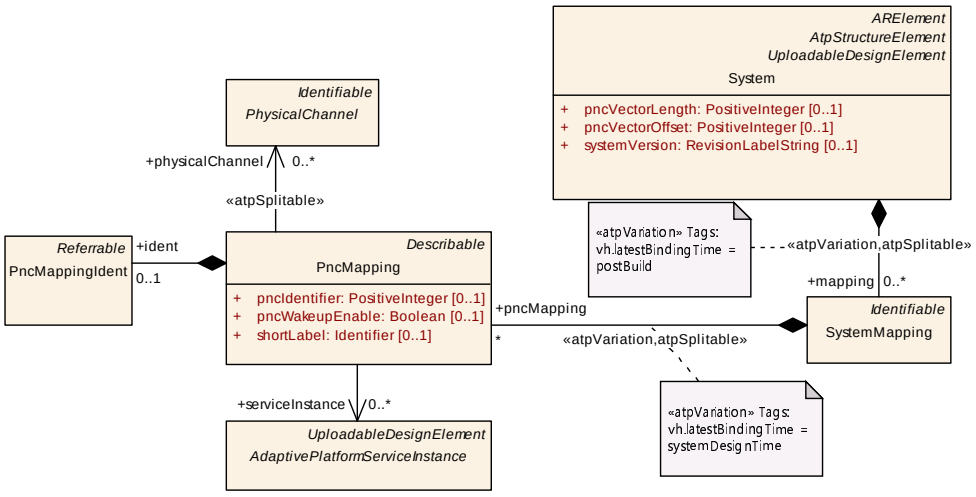


Figure 6.7: PncMapping with collection of ServiceInstances that are participating in the Partial Network Cluster

Class	**SystemMapping**			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The system mapping aggregates all mapping aspects that are relevant in the System Description.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	System.mapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**SystemMapping**			
ddsISignalTo TopicMapping	DdsCpISignalToDds TopicMapping	*	aggr	Collection of DdsISignalToDdsTopicMappings. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=ddsISignalToTopicMapping, ddsISignalTo TopicMapping.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=postBuild
pncMapping	PncMapping	*	aggr	Mappings between Virtual Function Clusters and Partial Network Clusters. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=pncMapping, pncMapping.variation Point.shortLabel vh.latestBindingTime=systemDesignTime

Table 6.45: SystemMapping

Class	**PncMapping**			
Package	M2::AUTOSARTemplates::SystemTemplate::PncMapping			
Note	Describes a mapping between one or several Virtual Function Clusters onto Partial Network Clusters. A Virtual Function Cluster is realized by a PortGroup. A Partial Network Cluster is realized by one or more ServiceInstances.			
Base	*ARObject*, *Describable*			
Aggregated by	SystemMapping.pncMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ident	PncMappingIdent	0..1	aggr	This adds the ability to become referrable to PncMapping.
physical Channel	PhysicalChannel	*	ref	This reference maps the partial network to a communication channel. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=physicalChannel
pncConsumed Provided ServiceInstance Group	ConsumedProvided ServiceInstanceGroup	*	ref	ConsumedProvidedServiceInstanceGroup used in a Partial Network Cluster. This reference is optional, since this could be used for starting and stopping Consumed ProvidedServiceInstanceGroup according the requested partial network, but is not necessarily needed. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=pncConsumedProvidedServiceInstance Group.consumedProvidedServiceInstanceGroup, pnc ConsumedProvidedServiceInstanceGroup.variation Point.shortLabel vh.latestBindingTime=postBuild
pncIdentifier	PositiveInteger	0..1	attr	Identifer of the Partial Network Cluster. This number represents the absolute bit position of this Partial Network Cluster in the NM Pdu.
pncWakeup Enable	Boolean	0..1	attr	If this parameter is available and set to true then this PNC will be woken up as soon as a channel wakeup occurs on a channel where this PNC is assigned to. This is ensured by adding this PNC to the corresponding channel wakeup sources during upstream mapping.
serviceInstance	AdaptivePlatform ServiceInstance	*	ref	Reference to ServiceInstances that are participating in a Partial Network Cluster.
shortLabel	Identifier	0..1	attr	This attribute specifies an identifying shortName for the PncMapping. It shall be unique in the System scope.

▽

△

Class	PncMapping			
vfc	PortGroup	*	iref	Virtual Function Cluster to be mapped onto a Partial Network Cluster. This reference is optional in case that the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy systems. **InstanceRef implemented by:** PortGroupInSystem InstanceRef

Table 6.46: PncMapping

6.3 Specification of Application Software System Structure

The root element of a System Design model is the `System` element that is already known from the AUTOSAR classic platform. The `System` aggregates the `RootSwCompositionPrototype` that represents the top-level-composition of all software components that are available in a given system.

[TPS_MANI_03110]{DRAFT} **Allowed components in system description with `category SYSTEM_DESIGN_DESCRIPTION`.** ⌈`SwComponentPrototype`s nested inside the `CompositionSwComponentType` that is referenced by the `RootSwCompositionPrototype` of a `System` with `category SYSTEM_DESIGN_DESCRIPTION` are allowed to be of any `SwComponentType` that is supported by Classic or by Adaptive AUTOSAR.⌋ *(RS_MANI_00026)*

Class	RootSwCompositionPrototype			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The RootSwCompositionPrototype represents the top-level-composition of software components within a given System. According to the use case of the System, this may for example be 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 Port Prototypes, PortInterfaces, VariableDataPrototypes, SwcInternalBehavior etc.), and their ports are interconnected using SwConnectorPrototypes.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, System.rootSoftwareComposition			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
software Composition	CompositionSw ComponentType	0..1	tref	We assume that there is exactly one top-level composition that includes all Component instances of the system. **Stereotypes:** isOfType

Table 6.47: RootSwCompositionPrototype

If a Software Component communicates over the service oriented communication and provides or requires a ServiceInterface the opposite communication end is not always known upfront. In the System with category SYSTEM_DESIGN_DESCRIPTION a System Designer may want to indicate the service oriented communication between endpoints if it is already known at the System Design time.

[TPS_MANI_03114]{DRAFT} **Usage of AssemblySwConnectors in the System Design model** ⌈In the System with category SYSTEM_DESIGN_DESCRIPTION it is allowed to indicate the service oriented communication between two communication endpoints by AssemblySwConnectors if the required RPortPrototype is searching for a specific service instance, i.e. if the RPortPrototypeProps.searchIntention is set to searchForSpecificInstance.

If the searchIntention is set to searchForAllInstances, the AssemblySwConnector shall not be used to connect this RPortPrototype.⌋*(RS_MANI_00026)*

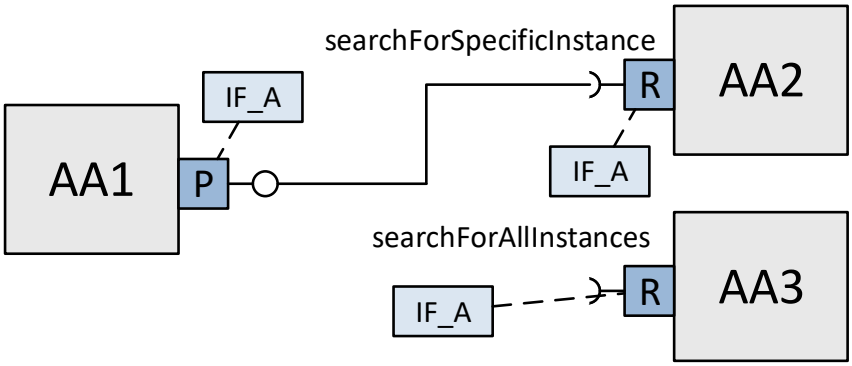


Figure 6.8: Example for Assembly connectors in System Design model

6.4 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	**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.recommendedPackage=InterfaceMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventMapping	EventMapping	*	aggr	Mapping of a VariableDataPrototype in a SenderReceiver Interface to an Event in a ServiceInterface.
fieldMapping	FieldMapping	*	aggr	Mapping of a Field in a ServiceInterface to ClientServer Operations that represent the getter and setter methods and to a VariableDataPrototype that represents the notifier in the Field.
fireAndForget MethodMapping	FireAndForgetMethod Mapping	*	aggr	Mapping of a Fire&Forget Method that is located in a ServiceInterface to a VariableDataPrototype in a Sender ReceiverInterface or to a Trigger in a TriggerInterface.
methodMapping	MethodMapping	*	aggr	Mapping of a ClientServerOperation in a ClientServer Interface to a Method in a ServiceInterface.

Table 6.48: InterfaceMapping

[constr_3370]{DRAFT} **`InterfaceMapping` shall map all elements of a single `ServiceInterface`** ⌈The mappings that are included in an `InterfaceMapping` shall map all elements of a single `ServiceInterface` (i.e. `field`s, `event`s, `method`s) to `PortInterface` elements of the classic platform **`at the time when the system design is complete`**.⌋*()*

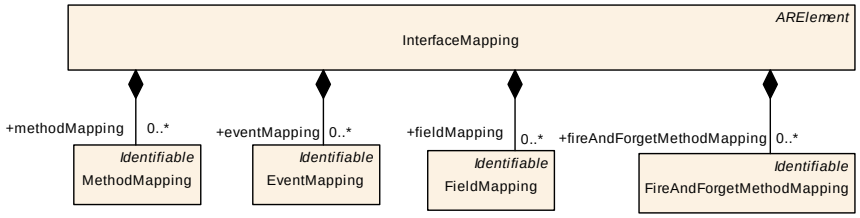


Figure 6.9: InterfaceMapping Overview

Figure 6.10 shows a possible System Design modeling approach where an a piece of application software is communicating in a service oriented way over SOME/IP with classic Software Components. SWC_1 requires a `ClientServerInterface` IF_Y with a `ClientServerOperation` and a `SenderReceiverInterface` IF_X with a `VariableDataPrototype`. SWC_2 requires a `SenderReceiverInterface` IF_X with a `VariableDataPrototype`.

The two `PortInterface`s IF_X and IF_Y are mapped to a single `ServiceInterface` IF_A using an `InterfaceMapping`.

On the other side the application software AA1 provides the `ServiceInterface` IF_A.

Note that this is a mapping on `PortInterface` level. If each `PortInterface` is only used once in a network the actual communication can be directly derived out of the

InterfaceMapping. If PortInterfaces are used several times on a network there is the need to take the network configuration into account in order to be able to emulate how the service discovery will behave on the network. From this information the actual communication relations on software level can be deduced.

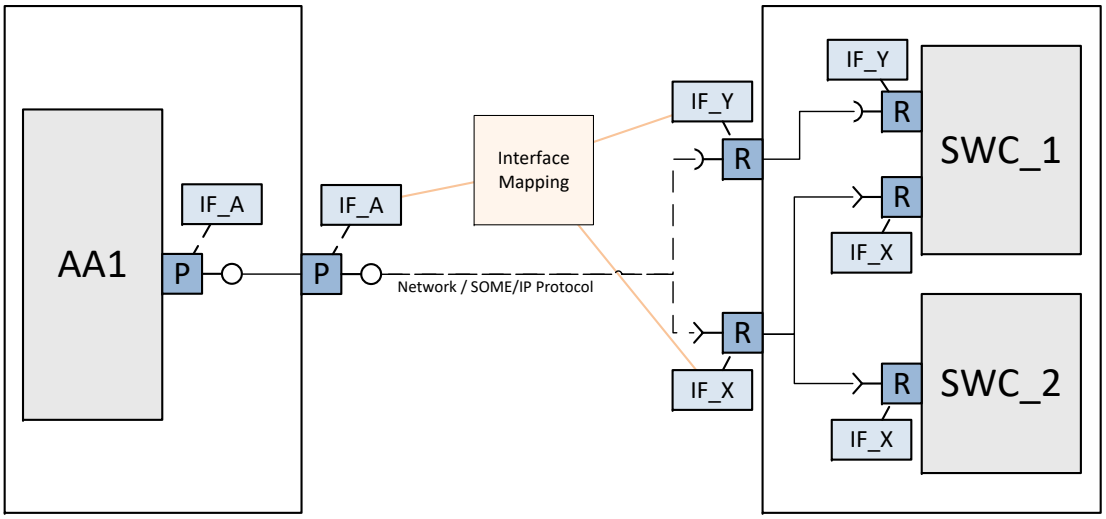


Figure 6.10: Example for a modeling of Service Oriented communication between application software on the *AUTOSAR adaptive platform* and Software Components of the Classic Platform

6.4.1 MethodMapping

[TPS_MANI_03111]{DRAFT} **Mapping between method and operationlocated in a ClientServerInterface** ⌈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)*

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.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	InterfaceMapping.methodMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientServer Operation	ClientServerOperation	0..1	ref	Reference to a ClientSeverOperation that is located in a ClientSeverInterface.
method	ClientServerOperation	0..1	ref	Reference to a Method that is located in a Service Interface.

Table 6.49: MethodMapping

[advisory_01000]{DRAFT} **Existence of the reference `MethodMapping.clientServerOperation`** ⌈For each `MethodMapping`, the reference in the role `MethodMapping.clientServerOperation` should exist **`at the time when the system design is complete`**.⌋*()*

[advisory_01001]{DRAFT} **Existence of reference `MethodMapping.method`** ⌈For each `MethodMapping`, the reference in the role `MethodMapping.method` should exist **`at the time when the system design is complete`**.⌋*()*

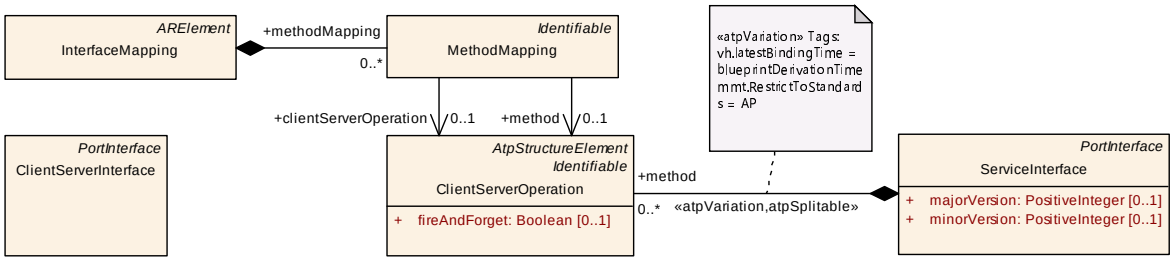


Figure 6.11: Mapping of a Method to a ClientServerOperation

6.4.2 EventMapping

[TPS_MANI_03112]{DRAFT} **Mapping between an `event` and a `dataElement`** ⌈The mapping between an `event` located in a `ServiceInterface` and a `dataElement` located in a `SenderReceiverInterface` is provided by the class `EventMapping`.⌋*(RS_MANI_00026)*

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.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	InterfaceMapping.eventMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	VariableDataPrototype	0..1	ref	Reference to a VariableDataPrototype that is located in a SenderReceiverInterface.
event	VariableDataPrototype	0..1	ref	Reference to an Event that is located in a Service Interface.

Table 6.50: EventMapping

[advisory_01002]{DRAFT} **Existence of reference `EventMapping.dataElement`** ⌈For each `EventMapping`, the reference in the role `EventMapping.dataElement` should exist **`at the time when the system design is complete`**.⌋*()*

[advisory_01003]{DRAFT} **Existence of reference `EventMapping.event`** ⌈For each `EventMapping`, the reference in the role `EventMapping.event` should exist **`at the time when the system design is complete`**.⌋*()*

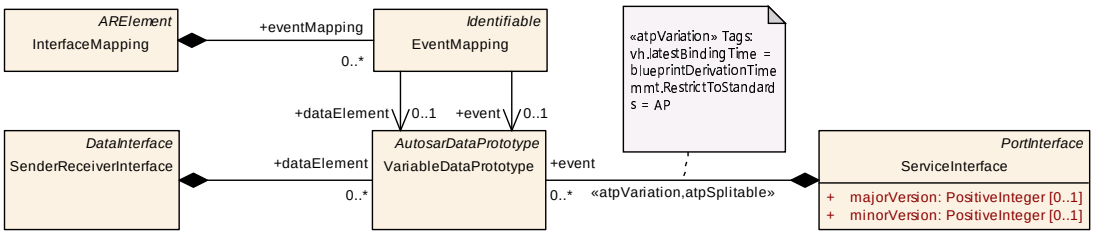


Figure 6.12: Mapping between an `event` and a `dataElement`

6.4.3 FieldMapping

[TPS_MANI_03113]{DRAFT} **Mapping between a `field` and elements of Classic Platform `PortInterface`s** ⌈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]{DRAFT} **`FieldMapping.notifierDataElement` reference** ⌈The `FieldMapping` shall only contain the `notifierDataElement` reference if the `hasNotifier` attribute in the referenced `field` is set to true **at the time when the system design is complete**.⌋*()*

[constr_3368]{DRAFT} **`FieldMapping.getterOperation` reference** ⌈The `FieldMapping` shall only contain the `getterOperation` reference if the `hasGetter` attribute in the referenced `field` is set to true **at the time when the system design is complete**.⌋*()*

[constr_3369]{DRAFT} **`FieldMapping.setterOperation` reference** ⌈The `FieldMapping` shall only contain the `setterOperation` reference if the `hasSetter` attribute in the referenced `field` is set to true **at the time when the system design is complete**.⌋*()*

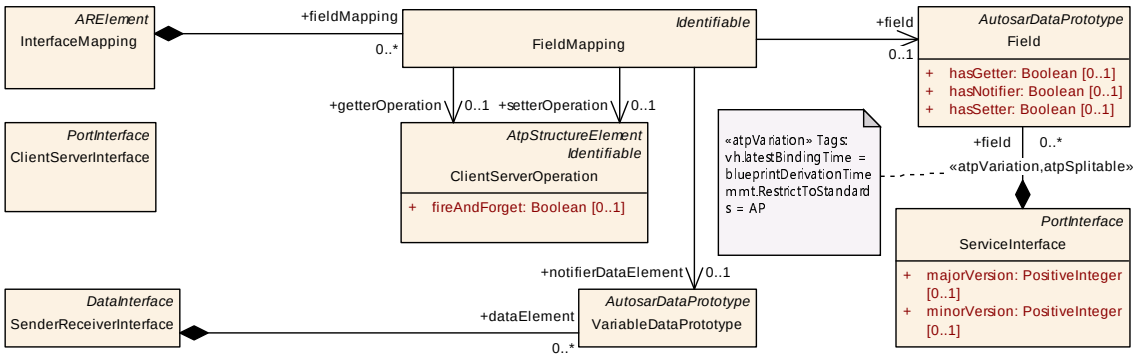


Figure 6.13: Mapping between a `field` and elements of Classic Platform `PortInterface`s

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.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	InterfaceMapping.fieldMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
field	Field	0..1	ref	Reference to a field that is located in a ServiceInterface.
getterOperation	ClientServerOperation	0..1	ref	Reference to a ClientServerOperation that represents the getter Method in the Field.
notifierData Element	VariableDataPrototype	0..1	ref	Reference to a VariableDataPrototype that represents the notifier in the Field.
setterOperation	ClientServerOperation	0..1	ref	Reference to a ClientServerOperation that represents the setter Method in the Field.

Table 6.51: FieldMapping

[advisory_01004]{DRAFT} **Existence of reference `FieldMapping.field`** ⌈For each `FieldMapping`, the reference in the role `FieldMapping.field` should exist **`at the time when the system design is complete`**.⌋*()*

[advisory_01005]{DRAFT} **Existence of references `FieldMapping.getterOperation`, `setterOperation`, and `notifierDataElement`** ⌈For each `FieldMapping`, at least one of the references in the role

- `FieldMapping.getterOperation`
- `FieldMapping.setterOperation`
- `FieldMapping.notifierDataElement`

should exist **`at the time when the system design is complete`**.⌋*()*

6.4.4 FireAndForgetMethodMapping

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 3.3.1.5.1.

On the *AUTOSAR adaptive platform*, 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].

On the *AUTOSAR classic platform*, 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 ArgumentDataPrototypes within the ClientServerOperation.

This representation ensures that the SOME/IP serialization results in the same byte stream as in the Adaptive Platform where all arguments which have the direction in are serialized according to the order of the ArgumentDataPrototypes within the ClientServerOperation.

If the fire and forget method is without any parameters a Trigger is used to describe such a method on the *AUTOSAR classic platform*.

It is important that the SOME/IP MessageType is set to REQUEST_NO_RETURN if a fire and forget method is transmitted over SOME/IP.

[TPS_MANI_03115]{DRAFT} **Mapping between a fire and forget method and elements of Classic Platform PortInterfaces** ⌈The mapping between a method for which the value of attribute method.fireAndForget is set to true and elements of Classic Platform PortInterfaces is provided by the class FireAndForgetMethodMapping.

If the fire and forget method is represented on the *AUTOSAR 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 on the *AUTOSAR classic platform* by a Trigger then this trigger is mapped to a method located in a ServiceInterface.⌋*(RS_MANI_00026)*

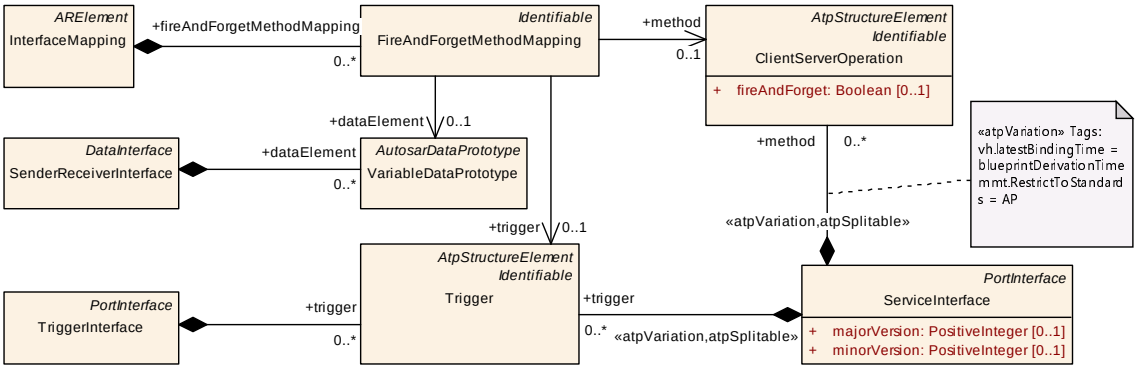


Figure 6.14: Mapping between a fire and forget method and elements of Classic Platform PortInterfaces

[constr_3371]{DRAFT} **Mutually exclusive existence of FireAndForgetMethodMapping.dataElement reference and FireAndForgetMethodMapping.trigger reference** ⌈A FireAndForgetMethodMapping shall never reference a dataElement and a trigger at the same time **at the time when the system design is complete**.⌋*()*

[constr_3376]{DRAFT} **FireAndForgetMethodMapping shall reference only fire and forget methods** ⌈A FireAndForgetMethodMapping is only allowed to reference a ClientServerOperation in role method for which the value of attribute

method.fireAndForget is set to true **at the time when the system design is complete.**⌋()

Class	**FireAndForgetMethodMapping**			
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.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	InterfaceMapping.fireAndForgetMethodMapping			
Attribute	***Type***	***Mult.***	***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.
method	ClientServerOperation	0..1	ref	Reference to a Fire&Forget Method that is located in a ServiceInterface.
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.

Table 6.52: FireAndForgetMethodMapping

[advisory_01006]{DRAFT} **Existence of reference FireAndForgetMethodMapping.method** ⌈For each FireAndForgetMethodMapping, the reference in the role FireAndForgetMethodMapping.method should exist **at the time when the system design is complete.**⌋()

[advisory_01007]{DRAFT} **Existence of references FireAndForgetMethodMapping.dataElement and trigger** ⌈For each FireAndForgetMethodMapping, at least one of the references in the role

- FireAndForgetMethodMapping.dataElement
- FireAndForgetMethodMapping.trigger

should exist **at the time when the system design is complete.**⌋()

7 Sub-System Design

7.1 Overview

The nature of the *AUTOSAR adaptive platform* as a platform for deploying software units in the field implies that the software units that can be installed in the field need some design-support upfront.

More specifically, the software units that can be deployed in the field typically represent some sort of more or less self-contained driving function.

In other words, the design support for this purpose need to be tailored to facilitate the design of application-level software that communicates with other application level software.

It is assumed that one of the first steps in such a design is the definition of services that are provided and services that are required by the driving function under development.

Such a definition of required and provided services can be used as an input into the design of other such driving functions and, over time, a view of the communication on the level of driving functions is rendered.

It is further assumed that the communication view of the driving functions is mostly of interest for an OEM and the individual driving functions may be sub-contracted to tier-1 suppliers.

This means that for the tier-1 supplier the list of provided and required services of the driving function represents a technical contract against which the function shall be developed.

On design level, meta-class `SoftwareClusterDesign` is used for the formalization of software that might represent such a driving function. In other words, it is assumed that a workflow exists where the design of a certain functionality on the AUTOSAR adaptive platform starts with the creation of a `SoftwareClusterDesign`.

In this case, it is further assumed that the definition of the required and provided service instances for the respective functionality is a good starting point for the development.

Please note that `SoftwareClusterDesign` supports an arbitrary complexity of software and is therefore not bound to the design of, e.g. a single driving function.

7.2 Software Cluster Design

The creation of a `SoftwareCluster` (see section 15.3) represents a complex workflow that has to be done in the deployment phase of the overall methodology for the *AUTOSAR adaptive platform*.

In order to reduce the complexity of this step in the workflow, AUTOSAR supports a front-loading approach for the definition of a `SoftwareCluster`, the modeling of

SoftwareClusterDesign. In this context, the term “front-loading” means that model elements defined in the context of a SoftwareClusterDesign can be (directly) taken over for the creation of a corresponding SoftwareCluster.

The membership of model elements to a given SoftwareClusterDesign is expressed by means of references to the respective model elements. The referenced model elements are “claimed” by the referencing SoftwareClusterDesign.

The pattern established by these references is very similar to the pattern established by means of the modeling of System.fibexElement or DiagnosticContributionSet.element.

Please note that the modeling of SoftwareClusterDesign prioritizes the usage of references over the usage of aggregations because the migration from SoftwareClusterDesign to SoftwareCluster is much facilitated no actual copying of model elements is required, only the content of references have to be taken over in the migration.

[TPS_MANI_01112]{DRAFT} **Semantics of SoftwareClusterDesign** ⌈The existence of a SoftwareClusterDesign represents the formalized response to requirements that have initially been formulated by an OEM and that may be enriched as the development of the software progresses.

Finally, the SoftwareClusterDesign shall be taken by the integration as a further input to the definition of the result of the integration step: the definition of the SoftwareCluster.⌋*(RS_MANI_00035)*

Just to be sure, the SoftwareClusterDesign is not intended to be uploaded to the target platform. It is just an early form of the final SoftwareCluster that indeed gets uploaded. The existence of the SoftwareClusterDesign is motivated from the methodological point of view.

[constr_1560]{DRAFT} **Usage of SoftwareClusterDesign.requiredARElement** ⌈The reference SoftwareClusterDesign.requiredARElement shall not be used to refer to another SoftwareClusterDesign or even SoftwareCluster **at the time when the sub-system design is complete**.⌋*()*

[TPS_MANI_01369]{DRAFT} **Semantics of reference SoftwareClusterDesign.requiredARElement** ⌈The reference SoftwareClusterDesign.requiredARElement is used to identify all model elements that are typed by a sub-class of meta-class ARElement. This way, it is possible to express which model elements belong to the enclosing SoftwareClusterDesign.⌋*(RS_MANI_00035)*

[TPS_MANI_01211]{DRAFT} **Specification of executable software within SoftwareClusterDesign** ⌈One of the most prominent contents of an uploadable software package is the reference to the executable software.

Within the definition of a SoftwareClusterDesign, this reference is implicitly given by means of the reference SoftwareClusterDesign.containedProcess.

The target of SoftwareClusterDesign.containedProcess is a ProcessDesign that represents the design-level representation of an instance (formalized as Process) of the corresponding executable program (the software image), formalized as Executable (*RS_MANI_00035*)

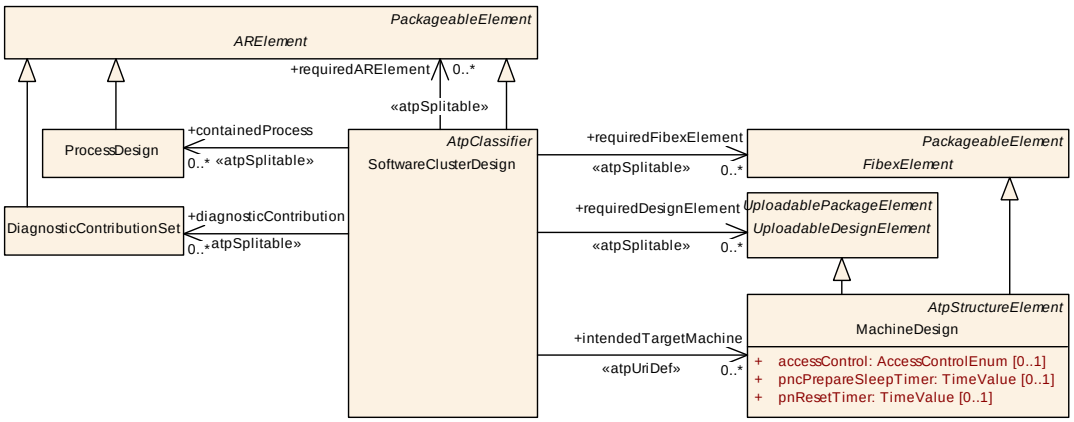


Figure 7.1: Modeling of SoftwareClusterDesign

Class	**SoftwareClusterDesign**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign			
Note	This meta-class represents the ability for the OEM to design the grouping of software uploadable to a specific target Machine. **Tags:** atp.recommendedPackage=SoftwareClusterDesigns			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contained Process	ProcessDesign	*	ref	This reference represent the ProcessDesigns contained in the enclosing SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=containedProcess
diagnostic Contribution	DiagnosticContribution Set	*	ref	This reference identifies the corresponding collection of DiagnosticContributionSet. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=diagnosticContribution
intendedTarget Machine	MachineDesign	*	ref	This reference can be taken to identify the Machine Design for which the final SoftwareCluster shall be developed. **Stereotypes:** atpUriDef
required ARElement	ARElement	*	ref	This reference represents the collection of ARElements that are required for the completeness of the definition of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=requiredARElement
requiredDesign Element	UploadableDesign Element	*	ref	This reference points to uploadable design elements that have been identified as relevant in the context of the enclosing SoftwareClusterDesign. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=requiredDesignElement

△

Class	**SoftwareClusterDesign**			
requiredFibex Element	FibexElement	*	ref	This reference represents the collection of fibexElements that are required for the completeness of the definition of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=requiredFibexElement
root Composition	RootSwClusterDesign ComponentPrototype	0..1	aggr	This aggregation represents the design of the software inside the SwClusterDesign terms of the communication endpoints.

Table 7.1: SoftwareClusterDesign

[constr_3727]{DRAFT} **Upper multiplicity of reference in the role `SoftwareClusterDesign.intendedTargetMachine`** ⌈In the context of `SoftwareClusterDesign`, the reference in the role `intendedTargetMachine` shall exist at most once **`at the time when the sub-system design is complete`**.⌋*()*

[TPS_MANI_01117]{DRAFT} **Semantics of `SoftwareClusterDesign.intendedTargetMachine`** ⌈The specification of `SoftwareClusterDesign.intendedTargetMachine` allows for focusing the specification of an uploadable software package to a specific `MachineDesign` from early phases of a development project.⌋*(RS_MANI_00035)*

Please note that `SoftwareCluster` doesn't have a dedicated reference to the target `Machine`.

This relation is expressed by means of a reference to `Process` that in turn can be mapped to a dedicated `Machine` by means of a `ProcessToMachineMapping`. In this context, [constr_1536] applies.

[TPS_MANI_01118]{DRAFT} **Relation between `SoftwareClusterDesign` and `DiagnosticContributionSet`** ⌈An important aspect of the definition of a `SoftwareClusterDesign` is the question what diagnostic extract shall be associated with the `SoftwareClusterDesign`.

For this purpose, a reference from `SoftwareClusterDesign` to `DiagnosticContributionSet` in the role `diagnosticContribution` is provided.

In an early stage of the development process, it is intentionally made possible to reference multiple `DiagnosticContributionSet`s in order to support the decentralized (e.g. partly done by OEM and partly done by supplier) configuration of the diagnostics stack.⌋*(RS_MANI_00035)*

[TPS_MANI_01189]{DRAFT} **Software Cluster and `DiagnosticContributionSet.category`** ⌈A `DiagnosticContributionSet` used in the context of a `SoftwareCluster` shall set the value of attribute `category` to `DIAGNOSTICS_SWCL_EXTRACT`.⌋*(RS_MANI_00035)*

Please mind the intentionally introduced difference between `SoftwareCluster` and `SoftwareClusterDesign` in terms of the relation to `DiagnosticContributionSet`.

In other words, the multiplicity of the references to DiagnosticContributionSet intentionally differ.

As already explained, the SoftwareClusterDesign shall support the decentralized configuration of the DiagnosticContributionSet while the SoftwareCluster requires the existence of a final (merged) DiagnosticContributionSet.

[TPS_MANI_01119]{DRAFT} **Reference to model elements from SoftwareClusterDesign** ⌈SoftwareClusterDesign has the ability to define the following references to model elements relevant for the definition of an uploadable software package:

- references to meta-classes derived from UploadableDesignElement are formalized by way of SoftwareClusterDesign.requiredDesignElement.
- references to meta-classes derived from ARElement are formalized by way of SoftwareClusterDesign.requiredARElement.
- references to meta-classes derived from FibexElement are formalized by way of SoftwareClusterDesign.requiredFibexElement.

⌋*(RS_MANI_00035)*

Please note that the conversion of a SoftwareClusterDesign to a SoftwareCluster is not formalized by AUTOSAR. This step can be done by a tool at the discretion of the integrator.

In other words, in some cases it may be applicable to do this conversion relatively early in the development project while other projects may require to keep the SoftwareClusterDesign around for a longer period in time.

7.3 Provided and required Services of Software Cluster Design

In order to support the definition of required and provided services early in the design of a SoftwareCluster[1], AUTOSAR supports the definition of a RootSwClusterDesignComponentPrototype in the context of a given SoftwareClusterDesign.

The RootSwClusterDesignComponentPrototype itself refers to a SwComponentType that in turn exposes PortPrototypes to the outside world.

Note that for the specific case of the RootSwClusterDesignComponentPrototype it is expected that the referenced SwComponentType represents a CompositionSwComponentType without any further detailing. A detailing is obviously unnecessary because the only purpose is the exposure of PortPrototypes to which AdaptivePlatformServiceInstances can be mapped.

A dedicated mapping class, ServiceInstanceToSwClusterDesignPortPrototypeMapping, is defined to support the creation of the described relation between PortPrototype and AdaptivePlatformServiceInstance.

[1] For more information, please refer to section 15.3.

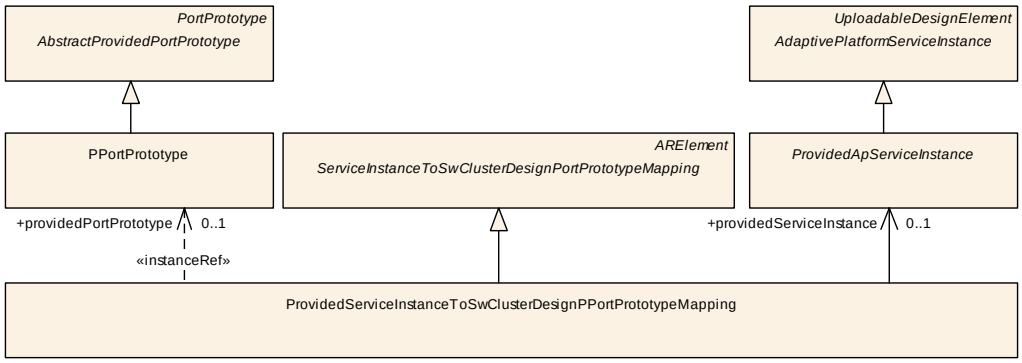


Figure 7.2: Modeling of the ProvidedServiceInstanceToSwClusterDesignPPort-PrototypeMapping

[TPS_MANI_01275]{DRAFT} **Semantics of meta-class ServiceInstance-ToSwClusterDesignPortPrototypeMapping** ⌈The software-component used to type the RootSwClusterDesignComponentPrototype typically exposes a set of PortPrototypes to the outside world.

These PortPrototypes could be used for the specification of required and provided service instances. For this purpose, meta-class ServiceInstanceToSwCluster-DesignPortPrototypeMapping is used.⌋*(RS_MANI_00011)*

In Figure 7.4, the ServiceInstanceToSwClusterDesignPortPrototypeMap-ping is represented by a block labeled “mapping” with a circled 1. The block labeled “mapping” with a circled 2 represents the CompositionPortToExecutable-PortMapping, as described in section 7.4.

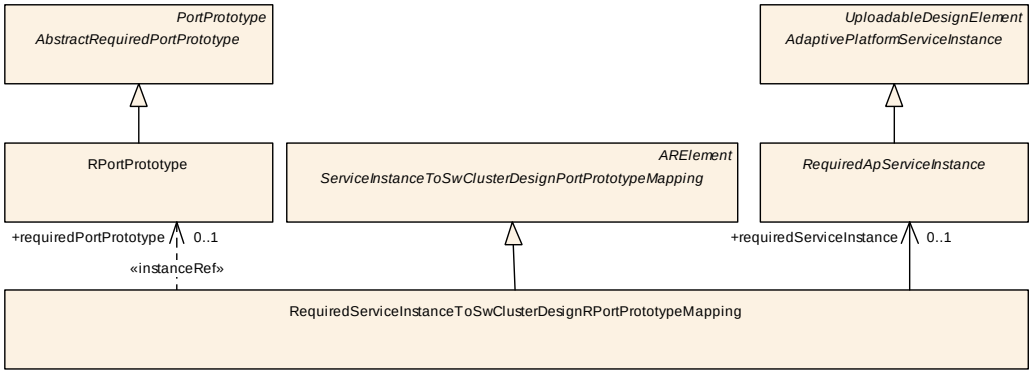


Figure 7.3: Modeling of the RequiredServiceInstanceToSwClusterDesignRPort-PrototypeMapping

Class	**RootSwClusterDesignComponentPrototype**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define the service endpoints in the scope of a SwClusterDesign.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, SoftwareClusterDesign.rootComposition			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**RootSwClusterDesignComponentPrototype**			
applicationType	SwComponentType	0..1	tref	This SwComponentType acts as the Type of the RootSw ClusterDesignComponentPrototype. **Stereotypes:** isOfType

Table 7.2: RootSwClusterDesignComponentPrototype

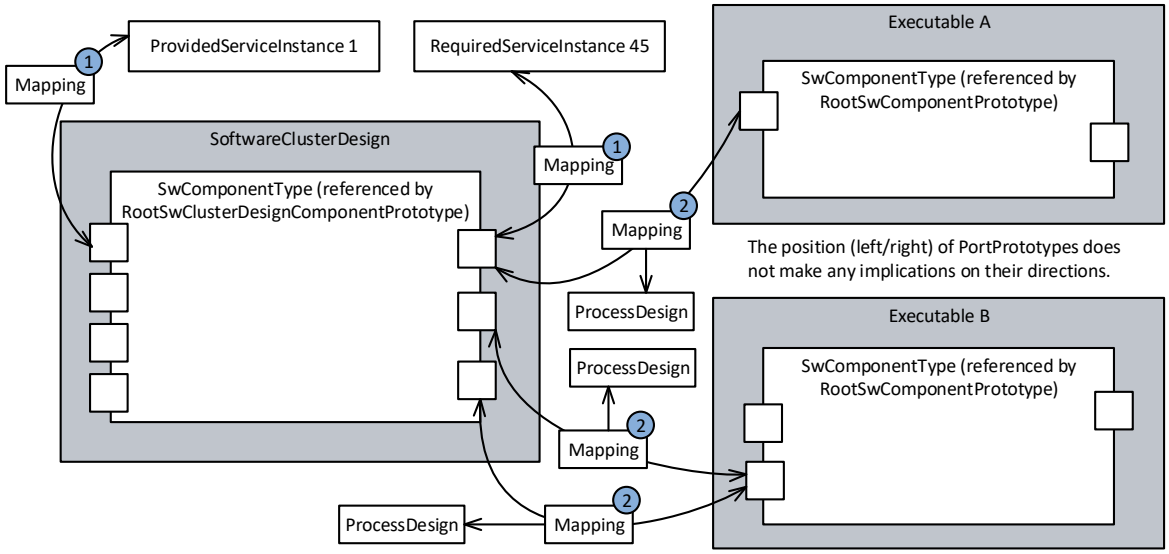


Figure 7.4: Modeling of mappings in the context of `SoftwareClusterDesign`

[advisory_01008]{DRAFT} **Multiplicity of reference in the role `RootSwClusterDesignComponentPrototype.applicationType`** ⌈For each `RootSwClusterDesignComponentPrototype`, the reference in the role `applicationType` should exist **`at the time when the sub-system design is complete`**.⌋*()*

Class	***ServiceInstanceToSwClusterDesignPortPrototypeMapping*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow			
Note	This abstract meta-class represents the ability to assign a transport-layer-dependent ServiceInstance to a PortPrototype in the context of the SoftwareClusterDesign. With this mapping it is possible to define the list of provided and required AdaptivePlatformServiceInstances in the scope of the SoftwareCluster Design.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping, RequiredServiceInstanceToSw ClusterDesignRPortPrototypeMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 7.3: ServiceInstanceToSwClusterDesignPortPrototypeMapping

Class	RequiredServiceInstanceToSwClusterDesignRPortPrototypeMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow			
Note	This concrete meta-class represents the ability to assign a transport-layer-dependent RequiredService Instance to an RPortPrototype in the context of the SoftwareClusterDesign. With this mapping it is possible to define the list of provided and required AdaptivePlatformServiceInstances in the scope of the SoftwareClusterDesign. **Tags:** atp.recommendedPackage=ServiceInstanceToSwClusterDesignPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInstanceToSwClusterDesignPortPrototypeMapping*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
requiredPort Prototype	RPortPrototype	0..1	iref	This reference identifies the applicable PortPrototype in the scope of the SwClusterDesign. **InstanceRef implemented by:** RPortPrototypeIn SoftwareClusterDesignInstanceRef
requiredService Instance	RequiredApService Instance	0..1	ref	Reference to a RequiredServiceInstance mapped to a given RPortPrototype in the scope of the SwCluster Design.

Table 7.4: RequiredServiceInstanceToSwClusterDesignRPortPrototypeMapping

Class	ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow			
Note	This concrete meta-class represents the ability to assign a transport-layer-dependent ProvidedService Instance to a PPortPrototype in the context of the SoftwareClusterDesign. With this mapping it is possible to define the list of provided and required AdaptivePlatformServiceInstances in the scope of the Software ClusterDesign. **Tags:** atp.recommendedPackage=ServiceInstanceToSwClusterDesignPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInstanceToSwClusterDesignPortPrototypeMapping*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
providedPort Prototype	PPortPrototype	0..1	iref	This reference identifies the applicable PortPrototype in the scope of the SwClusterDesign. **InstanceRef implemented by:** PPortPrototypeIn SoftwareClusterDesignInstanceRef
providedService Instance	ProvidedApService Instance	0..1	ref	Reference to a ProvidedServiceInstance mapped to a given PPortPrototype in the scope of the SwCluster Design.

Table 7.5: ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping

7.4 Mapping of Services to Executables

A typical next step in the design workflow could be to decide about the modeling of `Executable`s inside the `SoftwareClusterDesign`. The `PortPrototype`s used in the modeling of an `Executable` actually implement the endpoints to which required and provided service instances shall be mapped.

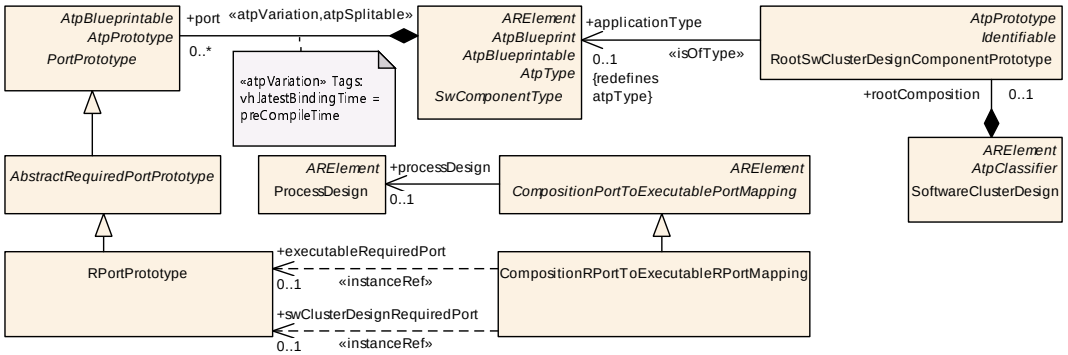


Figure 7.5: Modeling of the `RootSwClusterDesignComponentPrototype` and the `CompositionRPortToExecutableRPortMapping`

[TPS_MANI_01276]{DRAFT} **Semantics of `CompositionRPortToExecutableRPortMapping` and `CompositionPPortToExecutablePPortMapping`** ⌈In the context of the creation of an `SoftwareClusterDesign`, it is not possible to already define the actual mapping of `PortPrototype`s to `AdaptivePlatformServiceInstance`s.

To counter this issue, and as an additional guidance for the later creation of the actual `ServiceInstanceToPortPrototypeMapping`s it is possible to create another mapping inside the scope of the `SoftwareClusterDesign` that maps the `PortPrototype`s defined in the context of the `RootSwClusterDesignComponentPrototype` to the refined `PortPrototype`s defined in the context of `Executable`s.⌋ *(RS_MANI_00011)*

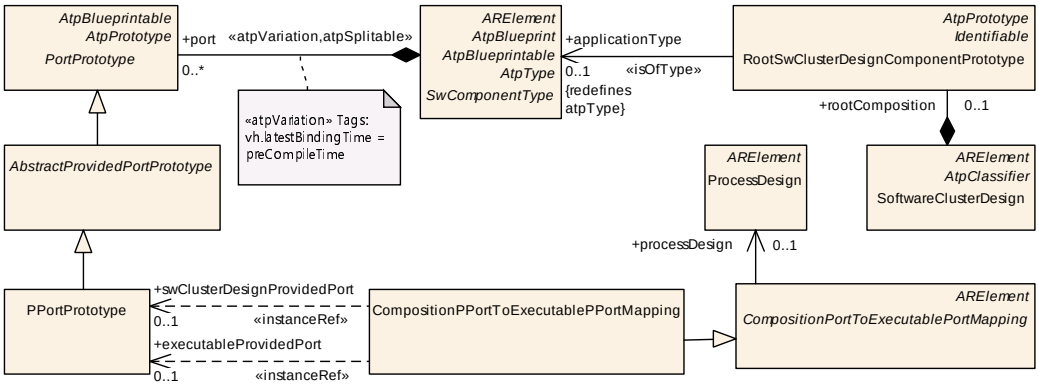


Figure 7.6: Modeling of the `RootSwClusterDesignComponentPrototype` and the `CompositionPPortToExecutablePPortMapping`

This way, it is possible to retrace the design decisions on the level of the `RootSwClusterDesignComponentPrototype` one level deeper and provide a guidance for the creation of the `ServiceInstanceToPortPrototypeMapping`, as described in section 11.3.

[TPS_MANI_01282]{DRAFT} **Semantics of reference `CompositionPortToExecutablePortMapping.processDesign`** ⌈The reference `CompositionPortToExecutablePortMapping.processDesign` identifies the applicable `ProcessDesign` for the mapping. This reference therefore disambiguates the existence of multiple `CompositionPortToExecutablePortMapping` that refer to the exact same `PortPrototype` in the context of an `Executable`.⌋*(RS_MANI_00011)*

The statement made by [TPS_MANI_01282] is further explained in Figure 7.4. Two `CompositionPortToExecutablePortMapping` refer to the same `PortPrototype` on the surface of the `Executable` B.

It is important to understand that each of these `CompositionPortToExecutablePortMapping`s refer to a different `ProcessDesign`.

This means that, at run-time, the two `CompositionPortToExecutablePortMapping`s apply to different instances of the `Executable` B launched as different `Process`es (that each, in turn, refer to one of the `ProcessDesign`s referenced by the `Executable` B).

Class	***CompositionPortToExecutablePortMapping*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow			
Note	This abstract meta-class acts as a base class for the specification of a mapping between a PortPrototype owned by a RootSwClusterDesignComponentPrototype to a PortPrototype owned by a Component Prototype inside an Executable.rootSwComponentType.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	CompositionPPortToExecutablePPortMapping, CompositionRPortToExecutableRPortMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
processDesign	ProcessDesign	0..1	ref	This reference identifies the impacted ProcessDesign for this mapping. This allows for mapping multiple services to the same PortPrototype on an Executable by also referencing different ProcessDesigns.

Table 7.6: CompositionPortToExecutablePortMapping

Class	**CompositionRPortToExecutableRPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow			
Note	This meta-class has the ability to associate an RPortPrototype defined in the context of a SwCluster Design to an RPortPrototype in the context of an Executable. **Tags:** atp.recommendedPackage=CompositionPortToExecutablePortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *CompositionPortToExecutablePortMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
executable RequiredPort	RPortPrototype	0..1	iref	This reference identifies the applicable PortPrototype in the context on an Executable. **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

▽

△

Class	**CompositionRPortToExecutableRPortMapping**			
swCluster DesignRequired Port	RPortPrototype	0..1	iref	This reference identifies the applicable RPortPrototype in the context of the SwClusterDesign. **InstanceRef implemented by:** RPortPrototypeIn SoftwareClusterDesignInstanceRef

Table 7.7: CompositionRPortToExecutableRPortMapping

Class	**CompositionPPortToExecutablePPortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::DesignWorkflow			
Note	This meta-class has the ability to associate a PPortPrototype defined in the context of a SwClusterDesign to a PPortPrototype in the context of an Executable. **Tags:** atp.recommendedPackage=CompositionPortToExecutablePortMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *CompositionPortToExecutablePortMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
executable ProvidedPort	PPortPrototype	0..1	iref	This reference identifies the applicable PortPrototype in the context on an Executable. **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
swCluster DesignProvided Port	PPortPrototype	0..1	iref	This reference identifies the applicable PPortPrototype in the context of the SwClusterDesign. **InstanceRef implemented by:** PPortPrototypeIn SoftwareClusterDesignInstanceRef

Table 7.8: CompositionPPortToExecutablePPortMapping

8 Execution Manifest

8.1 Overview

The purpose of the execution manifest is to provide information that is needed for the actual deployment of an application (formally modeled as an SwComponentType) onto the AUTOSAR adaptive platform.

One aspect of the deployment information is the provision of information that could in principle be provided as part of the application software code but which would make the application software code become very much bound to specific usage scenarios.

The general idea is to keep the application software code as independent as possible from the deployment scenario in order to increase the odds that the application software can be reused in different deployment scenarios.

In particular, the usage of PortPrototypes as a means to express communication with the "outside" of the application software allows for abstracting away the details (the concrete service instance identification) of the service configuration. As far as the model is concerned, the API between the application and the middleware is represented by the PortPrototype.

The application code does not use specific service instances but takes the PortPrototype as a symbolic replacement for this information. The specifics of this modeling aspect are described in section 11.

8.2 Process

The top-level element of the Execution Manifest definition is the Process, in reference to the fact that the unit of deployment on the *AUTOSAR adaptive platform* is a binary that, at runtime, makes a POSIX process.

[TPS_MANI_01308]{DRAFT} **Process is not designed for re-usability** ⌈Meta-class Process has **not** been created with the goal of reusing it on different Machines.

However, there is *some* potential for reusing configuration aspects in the definition of the Process.stateDependentStartupConfig.startupConfig.⌋*(RS_MANI_00006)*

Class	**Process**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest
Note	This meta-class provides information required to execute the referenced Executable. **Tags:** atp.recommendedPackage=Processes
Base	*ARElement*, *ARObject*, *AbstractExecutionContext*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *Uploadable PackageElement*

▽

△

Class	**Process**			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ProcessDesign	0..1	ref	This reference represents the identification of the design-time representation for the Process that owns the reference.
executable	Executable	*	ref	Reference to executable that is executed in the process. **Stereotypes:** atpUriDef
functionCluster Affiliation	String	0..1	attr	This attribute specifies which functional cluster the Process is affiliated with.
numberOf RestartAttempts	PositiveInteger	0..1	attr	This attribute defines how often a process shall be restarted if the start fails. numberOfRestartAttempts = "0" OR Attribute not existing, start once numberOfRestartAttempts = "1", start a second time
preMapping	Boolean	0..1	attr	This attribute describes whether the executable is preloaded into the memory.
processState Machine	ModeDeclarationGroup Prototype	0..1	aggr	Set of Process States that are defined for the process.
securityEvent	SecurityEventDefinition	*	ref	The reference identifies the collection of SecurityEvents that can be reported by the Process. **Stereotypes:** atpSplitable; atpUriDef **Tags:** atp.Splitkey=securityEvent atp.Status=candidate
stateDependent StartupConfig	StateDependentStartup Config	*	aggr	Applicable startup configurations.

Table 8.1: Process

[constr_3732]{DRAFT} **Upper multiplicity of reference in the role `Process.executable`** ⌈In the context of `Process`, the reference in the role `executable` shall exist at most once **`at the time when the creation of the manifest is finished`**.⌋*()*

Class	***AbstractExecutionContext*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class acts as a base class for entities that execute code on different levels, e.g. container, process, thread, fiber.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	Process			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 8.2: AbstractExecutionContext

[TPS_MANI_01337]{DRAFT} **Standardized values for attribute `Process.functionClusterAffiliation`** ⌈The following values of attribute `Process.functionClusterAffiliation` are standardized by AUTOSAR:

- **STATE_MANAGEMENT**
- **PLATFORM_HEALTH_MANAGEMENT**

⌋(*RS_MANI_00006*)

Please note that it is possible to use values other than from the standardized set in attribute `Process.functionClusterAffiliation`.

However, it is important that proprietary values of this attribute are formulated in a way that a potential clash with future standardized values can be avoided.

Clash-avoidance could be implemented by using a company-specific or project-specific prefix, infix, or suffix.

The `preMapping` approach of a `Process` is described in more detail in the SWS Execution Management [11], specifically in the context of [SWS_EM_02109].

8.2.1 Process relates to Executable

[TPS_MANI_01011]{DRAFT} **Connection between application design and application deployment** ⌈The connection between the *application design* and the *application deployment* is implemented by means of a reference from meta-class `Process` to meta-class `Executable` in the role `executable`.

By modeling the reference in this direction it is possible to keep the design level independent of the deployment level and, at the same time, bind the deployment to a specific design.⌋(*RS_MANI_00006*)

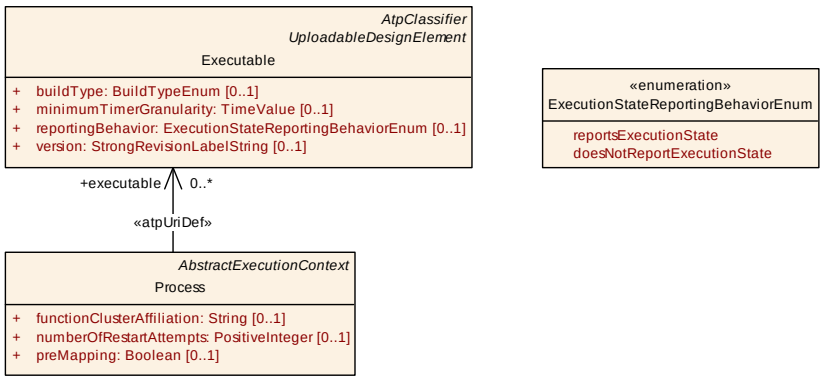


Figure 8.1: Relation of meta-classes `Executable` and `Process`

Please note that the meta-model, as depicted in Figure 8.1 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 8.2

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).

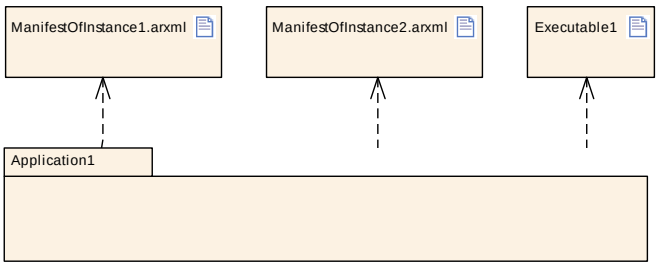


Figure 8.2: Example deployment where one Executable is bundled with two ARXML files that each contain the description of one Process

Therefore, it is necessary to allow for the definition of startup configurations on a per-Process-basis. This aspect is described in section 8.3.

8.2.2 Process States

A Process is a state-full entity. The state machine associated with a Process is modeled based on the aggregation of a ModeDeclarationGroupPrototype in the role processStateMachine, as sketched by Figure 8.3.

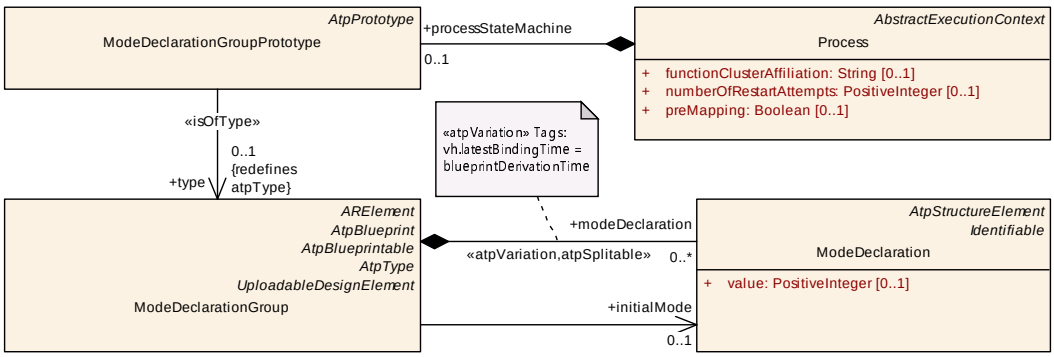


Figure 8.3: Modeling of process states

The supported process states that are defined in the Process.processStateMachine are described in more detail in [11].

It is possible to define a so-called *execution dependency*, such that one Process can only launch if a referenced other Process is in a given process state. The details are explained in section 8.3.5.

8.3 Startup Configuration

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

[TPS_MANI_01012]{DRAFT} **Formal modeling of application startup behavior** ⌈The formal modeling of application startup behavior is implemented by means of the aggregation of meta-class `StateDependentStartupConfig` in the role `Process.stateDependentStartupConfig`.⌋*(RS_MANI_00007)*

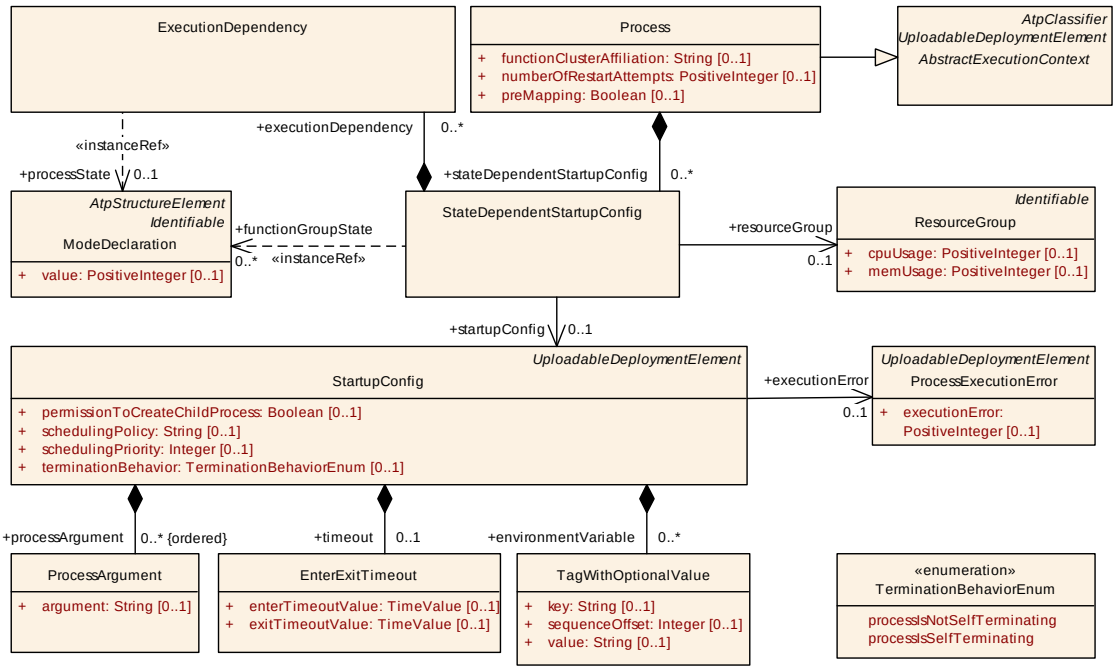


Figure 8.4: Content of a `Process`

8.3.1 State-dependent Startup Configuration

[TPS_MANI_01013]{DRAFT} **Semantics of meta-class `StateDependentStartupConfig`** ⌈The purpose of meta-class `StateDependentStartupConfig` is to qualify the startup configuration represented by meta-class `StartupConfig` for specific `ModeDeclaration`s.

In other words, the intention is to express that the `StartupConfig` is applicable if the state machines that control the startup are in the states represented by the `ModeDeclaration` referenced in the role `StateDependentStartupConfig.functionGroupState`.⌋*(RS_MANI_00007)*

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

[constr_3423]{DRAFT} **`StateDependentStartupConfig` of a `Process` shall reference a `functionGroupState`** ⌈Each `StateDependentStartupConfig` of a `Process` shall reference at least one `ModeDeclaration` in the role `functionGroupState` **at the time when the creation of the manifest is finished**.⌋*()*

However, the references to Function Group States within the context of one Process shall only refer to Function Group States **of the same Function Group**. This aspect is formalized by [constr_1688].

[constr_1688]{DRAFT} **StateDependentStartupConfig shall only refer to Function Group States of the same Function Group** ⌈For all StateDependentStartupConfigs aggregated in the role Process.stateDependentStartupConfig, references in the role functionGroupState to ModeDeclaration shall only refer to ModeDeclarations aggregated by the same ModeDeclarationGroup in the context of the same ModeDeclarationGroupPrototype (that represents the actual Function Group) **at the time when the creation of the manifest is finished**.⌋*()*

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

[constr_3396]{DRAFT} **Number of Process.stateDependentStartupConfig that refer to the same functionGroupState** ⌈Within the context of a given Process, no two StateDependentStartupConfigs shall refer to the same ModeDeclaration in the role functionGroupState **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_01046]{DRAFT} **Semantics of StateDependentStartupConfig.functionGroupState** ⌈The ModeDeclarations referenced in the role StateDependentStartupConfig.functionGroupState shall be considered in a way such that the StateDependentStartupConfig applies if **any** of the referenced ModeDeclarations is active.

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

[constr_3424]{DRAFT} **StateDependentStartupConfig shall never reference the functionGroupState Off** ⌈A StateDependentStartupConfig shall never reference the ModeDeclaration that has the shortName Off in the role functionGroupState. Please note that the Off ModeDeclaration is a special state in a Function Group as defined by [TPS_MANI_03195].

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

[constr_1618]{DRAFT} **Ability to shut down** ⌈In the context of one Machine, at least one Process shall have a stateDependentStartupConfig.functionGroupState that has the shortName Shutdown **at the time when the creation of the manifest is finished**.⌋*()*

[constr_1619]{DRAFT} **Ability to restart** ⌈In the context of one Machine, at least one Process shall have a stateDependentStartupConfig.functionGroupState that has the shortName Restart **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_01209]{DRAFT} **Definition of environment variables in process scope** ⌈It is possible to define environment variables in the scope of any given `Process`.

For this purpose the aggregation of `TagWithOptionalValue` in the role `StartupConfig.environmentVariable` exists.

The name of the environment variable shall be specified by means of the attribute `TagWithOptionalValue.key`, the value can be modeled by means of `TagWithOptionalValue.value`.

This encloses the ability to define environment variables with empty values. For this purpose, the attribute `TagWithOptionalValue.value` shall simply be omitted.⌋ *(RS_MANI_00007)*

Class	**StateDependentStartupConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class defines the startup configuration for the process depending on a collection of machine states.			
Base	*ARObject*			
Aggregated by	Process.stateDependentStartupConfig			
Attribute	***Type***	***Mult.***	***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 ProcessState before the referencing process is started.
functionGroupState	ModeDeclaration	*	iref	This represent the applicable functionGroupMode. **InstanceRef implemented by:** FunctionGroupStateInFunctionGroupSetInstanceRef
resourceConsumption	ResourceConsumption	0..1	aggr	This aggregation provides the ability to define resource consumption boundaries on a per-process-startup-config basis.
resourceGroup	ResourceGroup	0..1	ref	Reference to an applicable resource group.
startupConfig	StartupConfig	0..1	ref	Reference to a reusable startup configuration with startup parameters.

Table 8.3: StateDependentStartupConfig

[constr_10175]{DRAFT} **Multiplicity of attribute `StateDependentStartupConfig.resourceGroup`** ⌈For each `StateDependentStartupConfig`, the attribute `resourceGroup` shall exist **at the time when the creation of the manifest is finished**.⌋ *()*

[constr_10176]{DRAFT} **Multiplicity of attribute `StateDependentStartupConfig.startupConfig`** ⌈For each `StateDependentStartupConfig`, the attribute `startupConfig` shall exist **at the time when the creation of the manifest is finished**.⌋ *()*

Class	**StartupConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class represents a reusable startup configuration for processes.. **Tags:** atp.recommendedPackage=StartupConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
environment Variable	TagWithOptionalValue	*	aggr	This aggregation represents the collection of environment variables that shall be added to the respective Process's environment prior to launch.
executionError	ProcessExecutionError	0..1	ref	this reference is used to identify the applicable execution error
permissionTo CreateChild Process	Boolean	0..1	attr	This attribute defines if Process is permitted to create child Processes. When setting this parameter to true two things should be kept in mind: 1) safety and security implication of this configuration, 2) the fact that Process will assume management responsibilities for child Processes (i.e. it will be responsible for terminating Processes that it creates).
process Argument (ordered)	ProcessArgument	*	aggr	This aggregation represents the collection of command-line arguments applicable to the enclosing StartupConfig.
scheduling Policy	String	0..1	attr	This attribute represents the ability to define the scheduling policy for the initial thread of the application.
scheduling Priority	Integer	0..1	attr	This is the scheduling priority requested by the application itself.
termination Behavior	TerminationBehavior Enum	0..1	attr	This attribute defines the termination behavior of the Process.
timeout	EnterExitTimeout	0..1	aggr	This aggregation can be used to specify the timeouts for launching and terminating the process depending on the StartupConfig.

Table 8.4: StartupConfig

[TPS_MANI_01277]{DRAFT} **Definition of a start-up timeout for a `StartupConfig` of a `Process`** ⌈Meta-class `StartupConfig` provides the ability to define a start-up timeout for a `Process` by means of the attribute `enterTimeoutValue` that is aggregated by meta-class `EnterExitTimeout` that is aggregated by the `StartupConfig` in the role `timeout`.⌋ *(RS_MANI_00007)*

[TPS_MANI_01278]{DRAFT} **Definition of a termination timeout for a `StartupConfig` of a `Process`** ⌈Meta-class `StartupConfig` provides the ability to define a termination timeout for a `Process` by means of the attribute `exitTimeoutValue` that is aggregated by meta-class `EnterExitTimeout` that is aggregated by the `StartupConfig` in the role `timeout`.⌋ *(RS_MANI_00007)*

8.3.2 Scheduling

[TPS_MANI_01061]{DRAFT} **Requirements on scheduling** ⌈The attributes StartupConfig.schedulingPolicy and StartupConfig.schedulingPriority make requirements on the scheduling of the main thread of a process that is created out of launching the corresponding Executable.⌋*(RS_MANI_00007)*

[TPS_MANI_01328]{DRAFT} **Standardized values for attribute StartupConfig.schedulingPolicy** ⌈The following values are standardized for attribute StartupConfig.schedulingPolicy:

- **SCHED_RR**
- **SCHED_FIFO**
- **SCHED_OTHER**

⌋*(RS_MANI_00007)*

It is possible to use a custom, non-standardized value for the attribute StartupConfig.schedulingPolicy but this option comes with the obligation to use a value that is guaranteed to not clash with possible future extensions of the collection of standardized values.

[TPS_MANI_01188]{DRAFT} **Semantics of attribute schedulingPriority** ⌈The value of attribute StartupConfig.schedulingPriority shall be interpreted such that the higher values represent a higher scheduling priority.⌋*(RS_MANI_00007)*

[constr_1692]{DRAFT} **Value of schedulingPriority** ⌈The value of attribute StartupConfig.schedulingPriority shall be set to a positive integer value **at the time when the creation of the manifest is finished**.⌋*()*

8.3.3 Process Arguments

Please find more information about the interpretation of ProcessArgument in the SWS Execution Manifest [11].

[constr_1769]{DRAFT} **Existence of ProcessArgument.argument** ⌈For each ProcessArgument, attribute argument shall exist **at the time when the creation of the manifest is finished**.⌋*()*

Class	**ProcessArgument**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class has the ability to define command line arguments for processing by the Main function.			
Base	*ARObject*			
Aggregated by	StartupConfig.processArgument			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument	String	0..1	attr	This represents one command-line argument to be processed by the executable software.

Table 8.5: ProcessArgument

8.3.4 Association with Resource Group

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

[TPS_MANI_01017]{DRAFT} **Relation of startup configuration to resource group** ⌈The modeling of a resource group is possible by means of meta-class ResourceGroup in the OsModuleInstantiation of the Machine and the assignment of a Process to a ResourceGroup is supported by the association from StateDependentStartupConfig to ResourceGroup in the role resourceGroup.⌋*(RS_MANI_00007)*

[constr_3413]{DRAFT} **StateDependentStartupConfig of a Process is mapped to exactly one ResourceGroup** ⌈Each StateDependentStartupConfig of a Process shall be assigned to exactly one ResourceGroup that is defined in the Machine Manifest **at the time when the creation of the manifest is finished**.⌋*()*

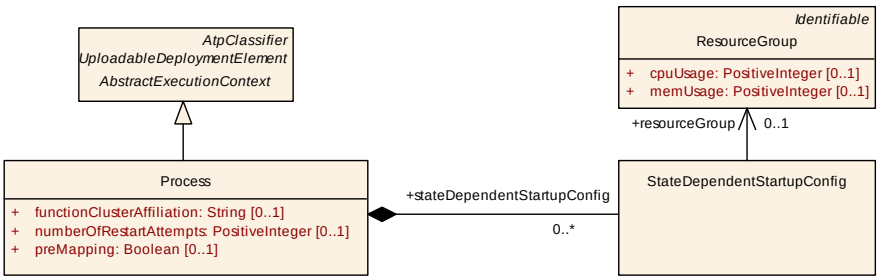


Figure 8.5: Modeling of how Process relates to ResourceGroup

8.3.5 Execution Dependency

The modeling of an execution dependency makes two Processes become associated to each other by means of the definition of an ExecutionDependency.

But since the reference that defines the execution dependency is modeled as an ≪instanceRef≫, the referenced Process needs to be extracted from the context references in the ≪instanceRef≫.

Once the two Processes are identified it is necessary for the validity of the startup dependency that they refer to the identical Function Group.

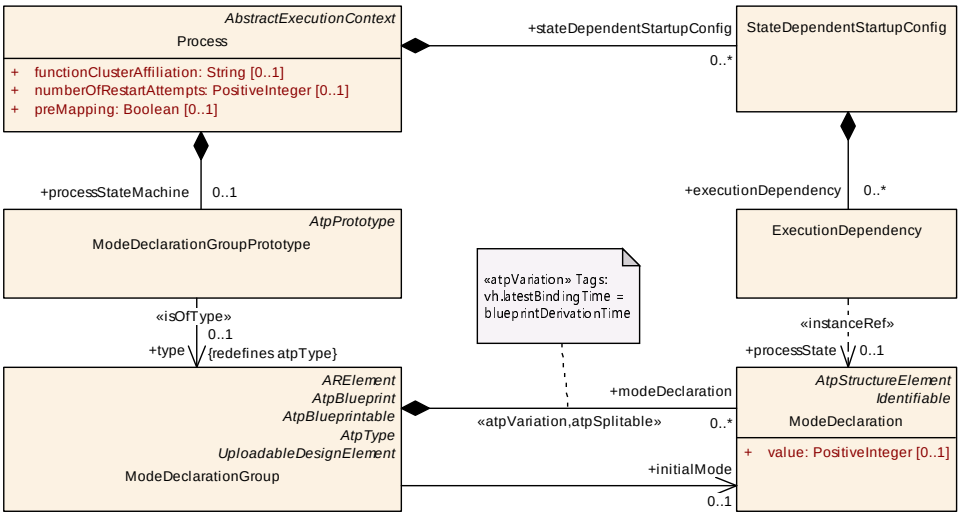


Figure 8.6: Modeling of how `Process` relates to `ModeDeclaration` owned by another `Process`

[TPS_MANI_01041]{DRAFT} **Startup configuration supports the definition of a launch sequence dependency** ⌈The modeling of startup configuration also supports the definition of a launch sequence dependency, formalized by the meta-class `ExecutionDependency` that is aggregated by `StateDependentStartupConfig` in the role `executionDependency`.

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

[constr_1689]{DRAFT} **Modeling of a startup dependency between different `Process`es** ⌈The existence of attribute `Process.stateDependentStartupConfig.executionDependency` is only valid if

- the owner of the `stateDependentStartupConfig.executionDependency` (in other words: the **referencing** `Process`) and
- the owner of the `ModeDeclarationGroupPrototype` referenced in the role `contextModeDeclarationGroupPrototype` within the reference `stateDependentStartupConfig.executionDependency.processState` (i.e. the **referenced** `Process`)

refer to the **identical `Function Group State`** formalized as `ModeDeclaration` **`at the time when the creation of the manifest is finished`**.⌋ *()*

Figure 8.7 provides an exemplary explanation of [constr_1689]. In this example, `Process` “B” (the referencing `Process` as of [constr_1689]) defines an `executionDependency` to `Process` “A”.

This `executionDependency` is only valid if both `Process` “A” and `Process` “B” aggregate a `StateDependentStartupConfig` that refers to the same `Function Group State` “MD” within `Function Group` “FG”.

Process "A" can be found by following the ExecutionDependency (specifically the contextModeDeclarationGroupPrototype) and the ≪instanceRef≫ that goes from the ExecutionDependency to the Process State "PS".

The **owner** of "PS" is Process "B", and if "B" refers to Function Group State "MD" within Function Group "FG" and if "A" refers to Function Group State"MD" within "FG" then the constraint [constr_1689] is fulfilled.

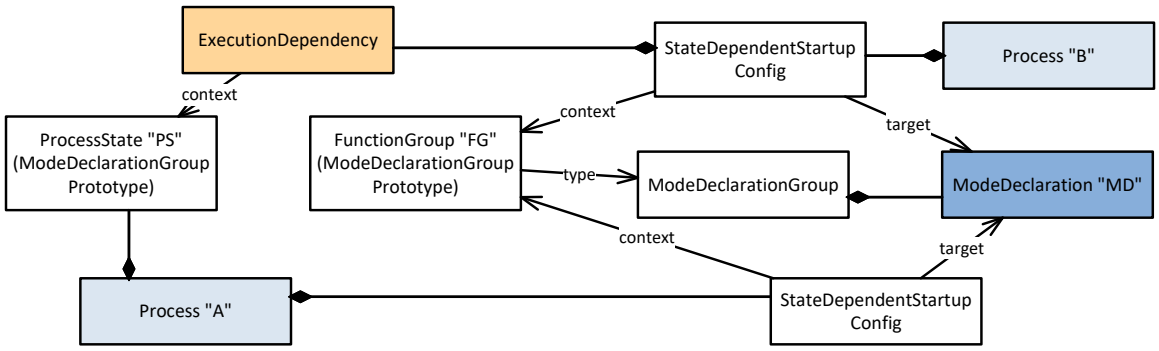


Figure 8.7: Explanation of dependencies from one Process to another

[constr_10411]{DRAFT} **Existence of ExecutionDependency and references to Function Group States** ⌈Each StateDependentStartupConfig that aggregates at least one ExecutionDependency in the role executionDependency shall reference at most one ModeDeclaration in the role functionGroupState.

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

There are scenarios where potential references to more than one functionGroupState in the presence of an ExecutionDependency lead to a conflict at runtime (specifically if the Process referenced in the ExecutionDependency is self-terminating). The prevention of such a scenario is the reason for the existence of [constr_10411].

For the exploration of such a scenario, Figure 8.8 sketches a scenario that is not allowed in the presence of [constr_10411].

In the sketched scenario in Figure 8.8, the execution of Process "B" depends on the launch and sub-sequent termination of Process "A". Process "A" owns two StateDependentStartupConfigs, one that references the Function Group State "FGS1" and another that references Function Group State "FGS2".

At runtime, this means that in the event of switching the Function Group to Function Group State "FGS1" (or "FGS2") both Processes "A" and "B" want to be executed.

Thanks to the existence of the ExecutionDependency, a POSIX-process that represents Process "A" is launched first, then a POSIX-process that represents Process "B".

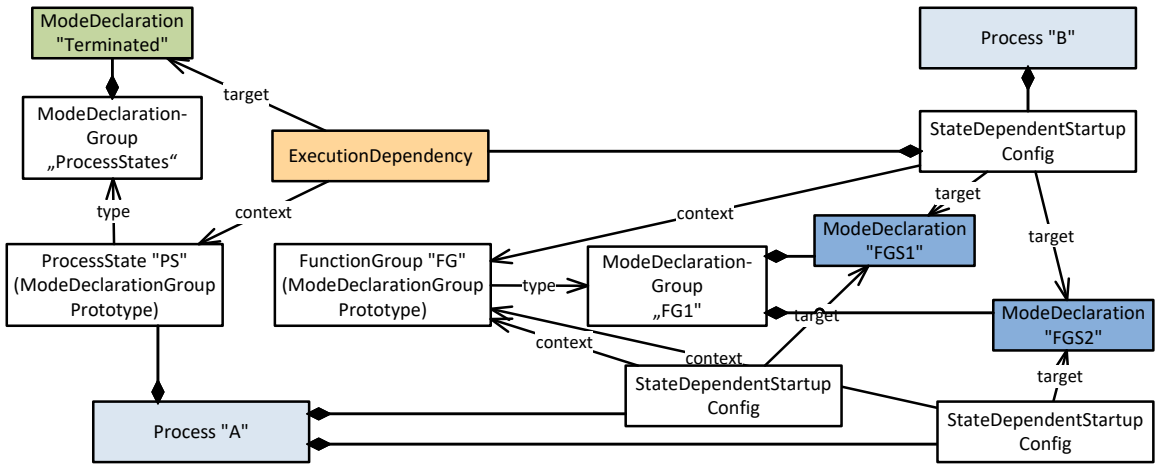


Figure 8.8: `StateDependentStartupConfig` references more than one `Function Group State` (invalid configuration)

Now, if the `Function Group State` switched from “FGS1” to “FGS2” (or vice versa), it is (with respect to the modeling of the example) expected that the same sequence of execution will be repeated. But instead, the following sequence will occur:

- a new POSIX-process that represents `Process` “A” will be launched and
- the POSIX-process that represents `Process` “B” will **not** be terminated and re-launched because `Process` “B” aggregates a `StateDependentStartupConfig` that references both `Function Group State`s “FGS1” and “FGS2”.

As a result, the order of execution (“A” launches and terminates, then “B” launches) as defined by the `ExecutionDependency` cannot be implemented in the event of a switch of `Function Group State`s between the two `Function Group State`s referenced by the `StateDependentStartupConfig` aggregated by `Process` “B”.

In order to prevent this contradiction between modeled semantics and real-world behavior, [constr_10411] has been created.

On the other hand, the scenario depicted in Figure 8.9 is cleared according to [constr_10411].

In the example depicted in Figure 8.9 it is expected that, upon `Function Group State` “FGS1” becoming active, a POSIX-process representing `Process` “A” is launched and, at some time later, terminates.

Then, a POSIX-process representing `Process` “B” is executed. Of course, the same sequence would occur upon the `Function Group State` being switched from any other `Function Group State` except “FGS1” to “FGS2”.

When the `Function Group` is switched to “FGS2”, the POSIX-process that represents `Process` “B” terminates as the exit from “FGS1” happens.

As “FGS2” is entered, a POSIX-process representing Process “A” is launched. After it terminates, a POSIX-process that represents Process “B” starts being executed, which is exactly the expected sequence.

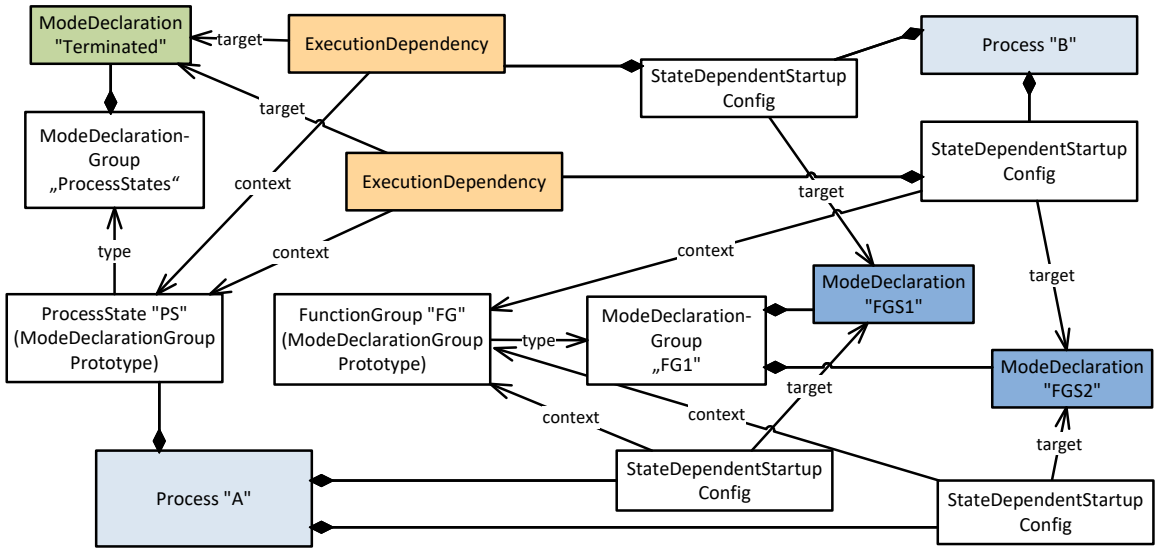


Figure 8.9: Two StateDependentStartupConfigs that each reference just one Function Group State

To summarize the example behavior, the intended execution dependency is realized when

- one of the referenced Function Group States becomes active or
- a switch from one referenced Function Group State to the other referenced Function Group State occurs.

Class	**ExecutionDependency**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This element defines a ProcessState in which a dependent process needs to be before the process that aggregates the ExecutionDependency element can be started.			
Base	*ARObject*			
Aggregated by	StateDependentStartupConfig.executionDependency			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
processState	ModeDeclaration	0..1	iref	This represent the applicable modeDeclaration that represents an ProcessState. **InstanceRef implemented by:** ModeInProcessInstanceRef

Table 8.6: ExecutionDependency

[constr_1606]{DRAFT} **Processes with mutual ExecutionDependencys** ⌈A Process.stateDependentStartupConfig.executionDependency shall not refer to any ModeDeclaration owned by a second Process that in turn refers via stateDependentStartupConfig.executionDependency to any ModeDeclaration

owned by the first Process **at the time when the creation of the manifest is finished**.⌋*()*

8.3.6 Assignment of Processes to Function Group states

There are use cases where starting and terminating of individual groups of processes is necessary. This is supported in AUTOSAR by Function Groups that group processes together.

A Function Group may have a number of Function Group States, e.g. Running, Idle, Terminating. The StateDependentStartupConfig of a Process can be assigned to a Function Group State and the start-up of the Process will then depend on this assignment.

The modeling of a Function Group and its Function Group States is described in section 8.4 in more detail. The usage of Function Groups is described in more detail in [11].

[TPS_MANI_03152]{DRAFT} **Assignment of a StateDependentStartupConfig to a Function Group State** ⌈The StateDependentStartupConfig is assigned to a Function Group State with the functionGroupState reference.⌋*(RS_MANI_00041)*

8.3.7 Resource Consumption Boundaries

[TPS_MANI_01269]{DRAFT} **Specification of boundaries for resource consumption** ⌈It is possible to specify boundaries for resource consumption, specifically in terms of memory consumption for memory of a given startup configuration of a Process: the formalization of process memory usage is represented by meta-class MemoryUsage, aggregated via meta-class ResourceConsumption at StateDependentStartupConfig.

The actual value of the memory usage is computed out of the sum of all aggregated ResourceConsumption.memoryUsage.⌋*(RS_MANI_00020)*

Please note the difference between the ability of defining resource consumption boundaries for a single Process, as opposed to the ability to associate a Process with a ResourceGroup that has the ability to also define resource consumption boundaries, albeit on a more coarse-grained level.

In contrast to that, the StateDependentStartupConfig.resourceConsumption allows for a fine-grained definition that can even observe the differences in resource consumption with respect to different startup configurations.

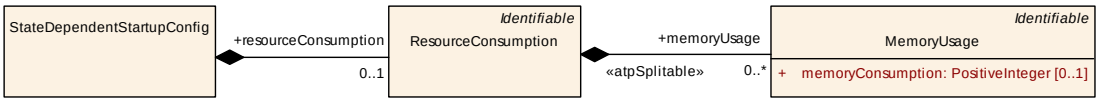


Figure 8.10: Modeling of resource consumption boundaries for a given `Process`

Class	**ResourceConsumption**			
Package	M2::AUTOSARTemplates::CommonStructure::ResourceConsumption			
Note	Description of consumed resources by one implementation of a software.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcuResourceEstimation.bswResourceEstimation, EcuResourceEstimation.rteResourceEstimation, *Implementation*.resourceConsumption, StateDependentStartupConfig.resourceConsumption			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
memoryUsage	MemoryUsage	*	aggr	Collection of the memory allocated by the owner. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=memoryUsage.shortName

Table 8.7: ResourceConsumption

Class	**MemoryUsage**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class is used to describe the memory consumption.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ResourceConsumption.memoryUsage			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
memory Consumption	PositiveInteger	0..1	attr	Provides a formal worst case system usage. The unit is byte.

Table 8.8: MemoryUsage

8.3.8 Error and Termination Behavior

[TPS_MANI_01334]{DRAFT} **Semantics of `StartupConfig.terminationBehavior`** ⌈The attribute `StartupConfig.terminationBehavior` defines the termination behavior of the `Process` in terms of whether (or not) the `Process` that references the enclosing `StartupConfig` in the role `stateDependentStartupConfig.startupConfig` is configured to self-terminate.⌋ *(RS_MANI_00007)*

Enumeration	**TerminationBehaviorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest
Note	This enumeration provides options for controlling of how a Process terminates.
Aggregated by	StartupConfig.terminationBehavior
Literal	***Description***
processIsNotSelf Terminating	The Process terminates only on request from Execution Management. **Tags:** atp.EnumerationLiteralIndex=0
processIsSelf Terminating	The Process is allowed to terminate without request from Execution Management. **Tags:** atp.EnumerationLiteralIndex=1

Table 8.9: TerminationBehaviorEnum

[constr_10007]{DRAFT} **Existence of `ProcessExecutionError.executionError`** ⌈For each `ProcessExecutionError`, attribute `executionError` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10008]{DRAFT} **Value of `ProcessExecutionError.executionError`** ⌈The value of attribute `ProcessExecutionError.executionError` shall at least be set to 1 (or higher) **`at the time when the creation of the manifest is finished`**.⌋*()*

Class	**ProcessExecutionError**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class has the ability to describe the value of a execution error along with a documentation of its semantics. **Tags:** atp.recommendedPackage=ProcessExecutionErrors			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
executionError	PositiveInteger	0..1	attr	This attribute defines the numeric value which Execution Management and Platform Health Management reports to State Management if the Process terminates unexpectedly or violates its supervision. It shall give further error information for error recovery.

Table 8.10: ProcessExecutionError

8.4 Function Groups

8.4.1 Semantics of Function Group

`Function Group`s with `Function Group State`s individually control groups of Application processes. The `Process` state may depend on a mode that is defined in the `Function Group` in case that the `StateDependentStartupConfig` refers to the `Function Group State` with the `functionGroupState` reference.

The usage of `Function Group`s is described in more detail in [11].

[TPS_MANI_03145]{DRAFT} **Description of a `Function Group`** ⌈By defining a `ModeDeclarationGroupPrototype` aggregated in the role `FunctionGroupSet`. `functionGroup` it is possible to define a `Function Group` that has a `shortName` and a set of Modes (States).

The `ModeDeclarationGroupPrototype` points to a reusable `ModeDeclarationGroup` in the role `type` that contains the different modes as `ModeDeclaration`s and a designated `initialMode`.⌋*(RS_MANI_00041)*

[TPS_MANI_03194]{DRAFT} **`Function Group State`** ⌈A `Function Group State` is described by a `ModeDeclaration` within a `ModeDeclarationGroup` that is referenced in the role `type` by a `ModeDeclarationGroupPrototype` aggregated as `functionGroup` by a `FunctionGroupSet`. The `Function Group State` is identified by its `shortName`.⌋*(RS_MANI_00041)*

The modeling described in [TPS_MANI_03145] and [TPS_MANI_03194] is depicted in Figure 8.11.

[TPS_MANI_03195]{DRAFT} **Off state in `Function Group`** ⌈Each `functionGroup` shall define a `ModeDeclaration` with the `shortName` `Off`. This `ModeDeclaration` shall also be referenced in the role `initialMode` by `ModeDeclarationGroup` that types the respective `functionGroup`.⌋*(RS_MANI_00041)*

[constr_1786]{DRAFT} **Restriction to use `functionGroup` in terms of `SoftwareCluster`** ⌈Each `functionGroup` shall only be referenced in the role `claimedFunctionGroup` by **at most one** `SoftwareCluster` **at the time when the creation of the manifest is finished**.⌋*()*

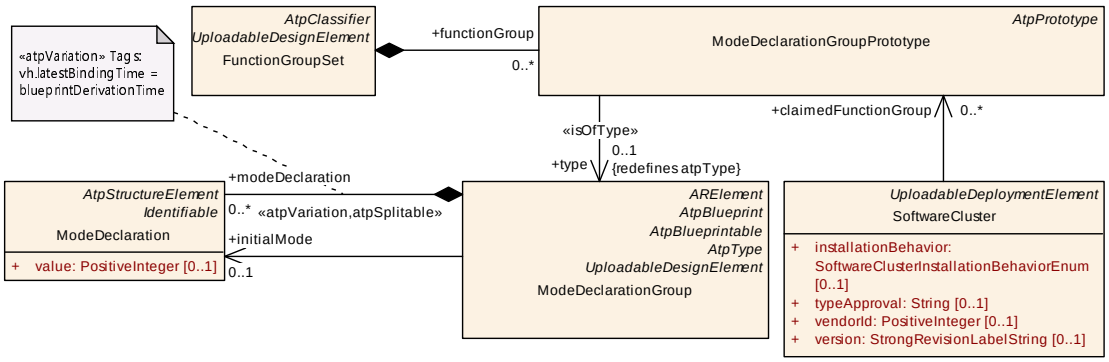


Figure 8.11: Configuration of `Function Group`s

[constr_1787]{DRAFT} **Restricted use of `Function Group`s in the context of a `SoftwareCluster`** ⌈All `Process`es referenced by a `SoftwareCluster` in the role `containedProcess` shall only aggregate `StateDependentStartupConfig`s where the reference `functionGroupState` refers to a `ModeDeclarationGroupPrototype` (as context) that is also referenced by the same `SoftwareCluster` in the role `claimedFunctionGroup` **at the time when the creation of the manifest is finished**.⌋*()*

The description of `SoftwareCluster` can be found in section 15.

[constr_10023]{DRAFT} **Mandatory content of any `functionGroup`** ⌈All `ModeDeclarationGroupPrototype`s aggregated by a `FunctionGroupSet` in the role `functionGroup` shall refer to a `ModeDeclarationGroup` that contains one `ModeDeclaration` with the `shortName` **`Verify at the time when the creation of the manifest is finished`**.⌋*()*

Class	**FunctionGroupSet**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class provides the ability to create arbitrary collections of function groups. **Tags:** atp.recommendedPackage=FunctionGroupSets			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
functionGroup	ModeDeclarationGroup Prototype	*	aggr	This aggregation represents the collection of function groups.

Table 8.11: FunctionGroupSet

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*			
Aggregated by	*AtpClassifier*.atpFeature, BswModuleDescription.providedModeGroup, BswModuleDescription.requiredModeGroup, FirewallStateSwitchInterface.firewallStateMachine, FunctionGroupSet.functionGroup, ModeSwitchInterface.modeGroup, Process.processStateMachine, StateManagementStateNotification.stateMachine			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
type	ModeDeclarationGroup	0..1	tref	The "collection of ModeDeclarations" (= ModeDeclarationGroup) supported by a component **Stereotypes:** isOfType

Table 8.12: 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*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
initialMode	ModeDeclaration	0..1	ref	The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred.

▽

△

Class	ModeDeclarationGroup			
mode Declaration	ModeDeclaration	*	aggr	The ModeDeclarations collected in this ModeDeclaration Group. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeDeclaration.shortName, mode Declaration.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime

Table 8.13: 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*			
Aggregated by	*AtpClassifier*.atpFeature, ModeDeclarationGroup.modeDeclaration			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
value	PositiveInteger	0..1	attr	The RTE shall take the value of this attribute for generating the source code representation of this Mode Declaration.

Table 8.14: ModeDeclaration

8.5 Reporting of Security Events

It is possible to report so-called security events (formalized by meta-class `SecurityEventDefinition`) from the context of a `Process`.

This approach works for application-level software as well as for functional clusters with the exception of the Execution Manager (because the Execution Manager is itself not modeled as a `Process`).

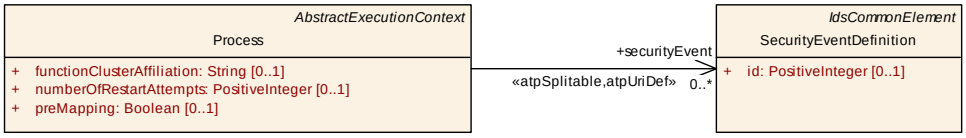


Figure 8.12: Modeling of support for the reporting of `SecurityEventDefinition` on deployment level

Please find more information about the semantics and usage of security events in the TPS Security Extract Template [20].

Class	**SecurityEventDefinition**			
Package	M2::AUTOSARTemplates::SecurityExtractTemplate			
Note	This meta-class defines a security-related event as part of the intrusion detection system. **Tags:** atp.Status=candidate atp.recommendedPackage=SecurityEventDefinitions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *IdsCommonElement*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventSymbol Name	SymbolProps	0..1	aggr	This aggregation defines optionally an alternative Event Name for the SecurityEventDefinition in case there is a collision of shortNames. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=eventSymbolName.shortName atp.Status=candidate
id	PositiveInteger	0..1	attr	This attribute represents the numerical identification of the defined security event. The identification shall be unique within the scope of the IDS. **Tags:** atp.Status=candidate

Table 8.15: SecurityEventDefinition

9 Machine Manifest

9.1 Machine

The Machine meta-class defines the entity on which one *Adaptive AUTOSAR Software Stack* is running with an operating system. The Machine may be physical or virtual.

Some aspects of the actual Machine are already available from the System Design (see chapter 6.2) at the MachineDesign.

The information defined at the MachineDesign is available to the Machine as well since Machine has a reference to the MachineDesign in the role machineDesign (see figure 5.1).

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

An overview of the Machine meta-class is sketched in Figure 9.1.

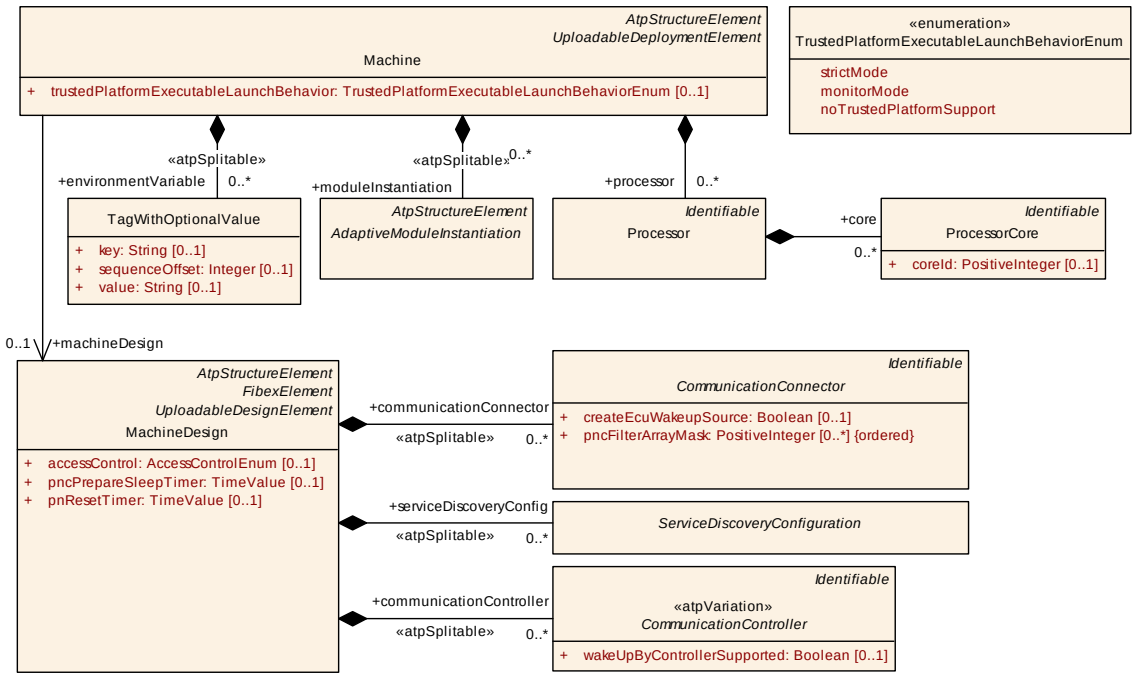


Figure 9.1: Overview about the content of the Machine configuration

[TPS_MANI_01273]{DRAFT} **Support for trusted Platform** ⌈If attribute Machine.trustedPlatformExecutableLaunchBehavior is set to a value that is different from noTrustedPlatformSupport then features of the "trusted platform" are activated, depending on the concrete value of Machine.trustedPlatformExecutableLaunchBehavior.⌋ *(RS_MANI_00022)*

Class	**Machine**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	Machine that represents an Adaptive Autosar Software Stack. **Tags:** atp.recommendedPackage=Machines			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
default Application Timeout	EnterExitTimeout	0..1	aggr	This aggregation defines a default timeout in the context of a given Machine with respect to the launching and termination of applications.
environment Variable	TagWithOptionalValue	*	aggr	This aggregation represents the collection of environment variables that shall be added to the environment defined on the level of the enclosing Machine. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=environmentVariable
machineDesign	MachineDesign	0..1	ref	Reference to the MachineDesign this Machine is implementing.
module Instantiation	AdaptiveModule Instantiation	*	aggr	Configuration of Adaptive Autosar module instances that are running on the machine. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=moduleInstantiation.shortName
processor	Processor	*	aggr	This represents the collection of processors owned by the enclosing machine.
secure Communication Deployment	SecureCommunication Deployment	*	aggr	Deployment of secure communication protocol configuration settings to crypto module entities. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=secureCommunication Deployment.shortName
trustedPlatform Executable LaunchBehavior	TrustedPlatform ExecutableLaunch BehaviorEnum	0..1	attr	This attribute controls the behavior of how authentication affects the ability to launch for each Executable.

Table 9.1: Machine

Class	***SecureCommunicationDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	The meta-class represents the ability to define a deployment of secure communication protocol configuration settings to crypto module entities.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	SecOcDeployment, TlsDeployment			
Aggregated by	Machine.secureCommunicationDeployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 9.2: SecureCommunicationDeployment

[constr_10169]{DRAFT} **Multiplicity of reference in the role `Machine.machineDesign`** ⌈For each `Machine`, the reference in the role `machineDesign` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10170]{DRAFT} **Multiplicity of attribute Machine.trustedPlatformExecutableLaunchBehavior** ⌈For each Machine, the attribute trustedPlatformExecutableLaunchBehavior shall exist **at the time when the creation of the manifest is finished.**⌋()

[constr_10171]{DRAFT} **Multiplicity of attribute Machine.processor** ⌈For each Machine, at least one aggregation on the role processor shall exist **at the time when the creation of the manifest is finished.**⌋()

Enumeration	**TrustedPlatformExecutableLaunchBehaviorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest
Note	This enumeration provides options for controlling the behavior of how authentication affects the ability to launch an Executable.
Aggregated by	Machine.trustedPlatformExecutableLaunchBehavior
Literal	***Description***
monitorMode	An Executable shall always launch, even if the corresponding authentication fails **Tags:** atp.EnumerationLiteralIndex=1
noTrustedPlatform Support	This value shall be used if there is no TrustedPlatform support on the Machine **Tags:** atp.EnumerationLiteralIndex=2
strictMode	An Executable shall not launch if the corresponding authentication fails. **Tags:** atp.EnumerationLiteralIndex=0

Table 9.3: TrustedPlatformExecutableLaunchBehaviorEnum

9.2 Processor

The Machine is able to aggregate one or several Processors, as depicted by Figure 9.1. Each Processor consists of one or several ProcessorCores.

Meta-class ProcessorCore provides attribute coreId that can be used e.g. in a bit-mask to better control the utilization of processing resources.

[constr_1549]{DRAFT} **Value of ProcessorCore.coreId** ⌈The value of ProcessorCore.coreId shall be unique in the context of the enclosing Processor **at the time when the creation of the manifest is finished.**⌋()

Class	**Processor**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	This represents a processor for the execution of an AUTOSAR adaptive platform			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	Machine.processor			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
core	ProcessorCore	*	aggr	This represents the collection of cores owned by the enclosing processor.

Table 9.4: Processor

[constr_10172]{DRAFT} **Multiplicity of attribute `Processor.core`** ⌈For each `Processor`, the attribute `core` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

Class	ProcessorCore			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	This meta-class represents the ability to model a processor core for the execution of an AUTOSAR adaptive platform.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	Processor.core			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
coreId	PositiveInteger	0..1	attr	This attribute represents a numerical value assigned to the specific core. The value can be taken e.g. for use in a bitmask.

Table 9.5: ProcessorCore

[constr_10173]{DRAFT} **Multiplicity of attribute `ProcessorCore.coreId`** ⌈For each `ProcessorCore`, the attribute `coreId` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

9.3 Environment Variable

[TPS_MANI_01208]{DRAFT} **Definition of environment variables in the scope of a `Machine`** ⌈It is possible to define environment variables in the scope of the entire `Machine`.

For this purpose the aggregation of `TagWithOptionalValue` in the role `Machine.environmentVariable` exists.

The name of the environment variable shall be specified by means of the attribute `TagWithOptionalValue.key`, the value can be modeled by means of `TagWithOptionalValue.value`.

This encloses the ability to define environment variables with empty values. For this purpose, the attribute `TagWithOptionalValue.value` shall simply be omitted.⌋ *(RS_MANI_00022, RS_MANI_00023)*

Please note that the aggregation `Machine.environmentVariable` has been defined with the stereotype ≪`atpSplitable`≫. The consequence of this modeling is that it is possible to contribute to the definition of environment variables from **different sources**.

Appendix B.8.1 contains a description of a scenario for the definition of environment variables in different partial models.

9.4 Process To Machine Mapping

9.4.1 General Modeling Approach

[TPS_MANI_03147]{DRAFT} **Mapping of a `Process` to a `Machine`** ⌈The meta-class `ProcessToMachineMapping` provides the ability to map a `Process` to a `Machine`.⌋ *(RS_MANI_00006)*

[constr_1553]{DRAFT} **Restriction for `ProcessToMachineMapping`** ⌈The following restrictions apply for the usage of `ProcessToMachineMapping`:

1. Each combination of `Process` and `Machine` shall only be referenced by one `ProcessToMachineMapping` in the role `process` or `machine`.
2. Each `Process` shall only be referenced by a single `ProcessToMachineMapping` in the role `process`.

This rule shall be imposed **`at the time when the creation of the manifest is finished`**.⌋ *()*

Please note that [constr_1553] does not imply that a given `Machine` shall only be referenced by a single `ProcessToMachineMapping`. It only says that one `Process` shall only be mapped once, to exactly one `Machine`.

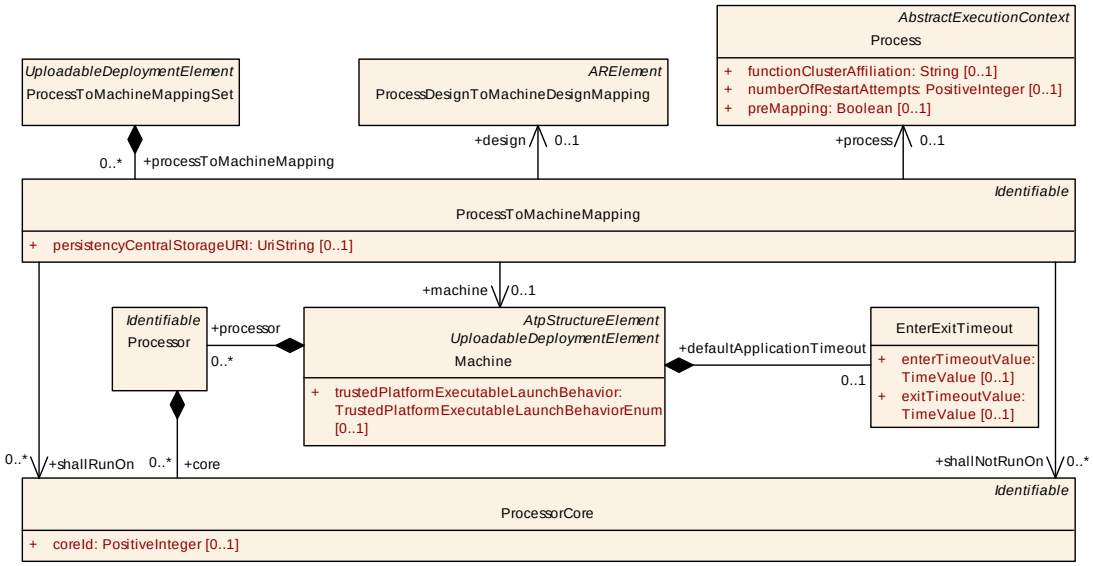


Figure 9.2: Mapping of a `Process` to a `Machine`

[constr_5004]{DRAFT} **Mapping of a `Process` to a `Machine` is mandatory in the Execution Manifest** ⌈Each `Process` shall be mapped by a `ProcessToMachineMapping` to one `Machine` **`at the time when the creation of the manifest is finished`**.⌋ *()*

[constr_5004] means that a formal description of the assignment of a `Process` to a `Machine` shall be provided in the Execution Manifest, even though the Manifest will be uploaded to the `Machine` in combination with other artifacts to which the Manifest

applies. The formal ProcessToMachineMapping was introduced because it is useful in the processing of the model in many cases.

Please note that according to the AUTOSAR Methodology the Execution Manifest is created on the basis of an existing Machine Manifest and therefore the link to the Machine can always be created in the Execution Manifest.

[constr_10090]{DRAFT} **Existence of ProcessToMachineMapping.persistencyCentralStorageURI** ⌈Attribute ProcessToMachineMapping.persistencyCentralStorageURI shall exist if the Process referenced in the role ProcessToMachineMapping.process is also referenced by at least one of

- PersistencyPortPrototypeToDeploymentMapping in the role process
- FunctionalClusterInteractsWithPersistencyDeploymentMapping in the role process

at the time when the creation of the manifest is finished.⌋()

Class	**ProcessToMachineMappingSet**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	This meta-class acts as a bucket for collecting ProcessToMachineMappings. **Tags:** atp.recommendedPackage=ProcessToMachineMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
processTo Machine Mapping	ProcessToMachine Mapping	*	aggr	This represents the collection of ProcessToMachine Mappings of the enclosing ProcessToMachineMapping Set.

Table 9.6: ProcessToMachineMappingSet

Class	**ProcessToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	This meta-class has the ability to associate a Process with a Machine. This relation involves the definition of further properties, e.g. timeouts.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ProcessToMachineMappingSet.processToMachineMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ProcessDesignTo MachineDesignMapping	0..1	ref	This reference represents the identification of the design-time representation for the ProcessToMachine Mapping that owns the reference.
machine	Machine	0..1	ref	This reference identifies the Machine in the context of the ProcessToMachineMapping.
nonOsModule Instantiation	NonOsModule Instantiation	0..1	ref	This supports the optional case that the process represents a platform module.
persistency CentralStorage URI	UriString	0..1	attr	This attribute identifies a central place for the mapped Process to store the list of available storages and version information.

▽

△

Class	**ProcessToMachineMapping**			
process	Process	0..1	ref	This reference identifies the Process in the context of the ProcessToMachineMapping.
shallNotRunOn	ProcessorCore	*	ref	This reference indicates a collection of cores onto which the mapped process shall not be executing.
shallRunOn	ProcessorCore	*	ref	This reference indicates a collection of cores onto which the mapped process shall be executing.

Table 9.7: ProcessToMachineMapping

Primitive	**UriString**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	A Uniform Resource Identifier (URI), is a compact string of characters used to identify or name a resource. **Tags:** xml.xsd.customType=URI-STRING xml.xsd.type=string

Table 9.8: UriString

[constr_10174]{DRAFT} **Multiplicity of the reference in the role `ProcessToMachineMapping.process`** ⌈For each `ProcessToMachineMapping`, the reference in the role `process` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

9.4.2 Core Affinity

[TPS_MANI_03148]{DRAFT} **Description of Core affinity** ⌈The meta-class `ProcessToMachineMapping` provides the ability to restrict the assignment of processes to selected `ProcessorCore`s with the two references `shallRunOn` and `shallNotRunOn`.⌋*(RS_MANI_00020)*

[constr_3393]{DRAFT} **Usage of `shallRunOn` and `shallNotRunOn` references** ⌈The `ProcessorCore` that is referenced by a `ProcessToMachineMapping` in the role `shallRunOn` or `shallNotRunOn` shall be aggregated by the `Machine` that is referenced in the role `machine` by the same `ProcessToMachineMapping` **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_1676]{DRAFT} **Consistency of references `shallRunOn` and `shallNotRunOn`** ⌈Within the context of one `ProcessToMachineMapping`, all `ProcessorCore`s referenced in the role `shallRunOn` or `shallNotRunOn` shall be aggregated by the same `Processor` **`at the time when the creation of the manifest is finished`**.⌋*()*

If a model defines that a given `Process` shall run on a select set of `ProcessorCore`s then there is hardly a use case to (in addition) also specify the opposite, i.e. that the `Process` shall not run on another set of `ProcessorCore`s, and vice versa.

In other words, either there is a motivation to identify the ProcessorCores on which a Process is supposed to run or there is a motivation to do the exact opposite and specify the ProcessorCores where the Process is not supposed to run.

This conclusion provides the motivation for the existence of [constr_1677].

[constr_1677]{DRAFT} **Mutual exclusive existence of references shallRunOn and shallNotRunOn** ⌈For any given ProcessToMachineMapping, either the reference in the role shallRunOn or the reference in the role shallNotRunOn may exist **at the time when the creation of the manifest is finished**.⌋*()*

9.4.3 Default Start-up and Termination Timeout

[TPS_MANI_03151]{DRAFT} **Default value for termination timeout** ⌈The meta-class Machine provides the ability to define a default value for termination timeout of applications in the context of the Machine with the attribute exitTimeoutValue that is available in the EnterExitTimeout meta-class that is aggregated by the Machine in the role defaultApplicationTimeout.⌋*(RS_MANI_00007)*

[TPS_MANI_01412]{DRAFT} **Default value for startup timeout** ⌈The meta-class Machine provides the ability to define a default value for startup timeout of applications in the context of the Machine with the attribute enterTimeoutValue that is available in the EnterExitTimeout meta-class that is aggregated by the Machine in the role defaultApplicationTimeout.⌋*(RS_MANI_00007)*

[constr_10430]{DRAFT} **Existence of attribute Machine.defaultApplicationTimeout** ⌈For each Machine, the attribute defaultApplicationTimeout shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10431]{DRAFT} **Existence of attribute Machine.defaultApplicationTimeout.enterTimeoutValue** ⌈For each Machine, the attribute defaultApplicationTimeout.enterTimeoutValue shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10432]{DRAFT} **Existence of attribute Machine.defaultApplicationTimeout.exitTimeoutValue** ⌈For each Machine, the attribute defaultApplicationTimeout.exitTimeoutValue shall exist **at the time when the creation of the manifest is finished**.⌋*()*

Class	**EnterExitTimeout**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	This meta-class represents the ability to specify a pair of timeouts, one for entering, and one for exiting.			
Base	*ARObject*			
Aggregated by	Machine.defaultApplicationTimeout, StartupConfig.timeout			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**EnterExitTimeout**			
enterTimeout Value	TimeValue	0..1	attr	This attribute represents the value of the enter timeout in seconds.
exitTimeout Value	TimeValue	0..1	attr	This attribute represents the value of the exit timeout in seconds.

Table 9.9: EnterExitTimeout

10 Platform Module Development

The model of platform modules and their instantiation has two major use-cases:

- provide dedicated attributes to configure the platform modules
- define the potential start of the module's executable as process.

The two use-cases are combined in one modeling approach: the Machine.moduleInstantiation, which collects sub-classes of AdaptiveModuleInstantiation. This modeling approach boils down to the variety of platform module models found in this chapter.

The OsModuleInstantiation defines several attributes to be configured for the Os, however the OsModuleInstantiation is the only AdaptiveModuleInstantiation where it is not possible to map it to a Process model element.

Of course there will be processes running the Os on the Machine anyway, however, these processes are not modeled.

Then there is the scenario where dedicated sub-classes of NonOsModuleInstantiation exist. Here the specific attributes are provided individually per sub-class, e.g. NmInstantiation or LogAndTraceInstantiation.

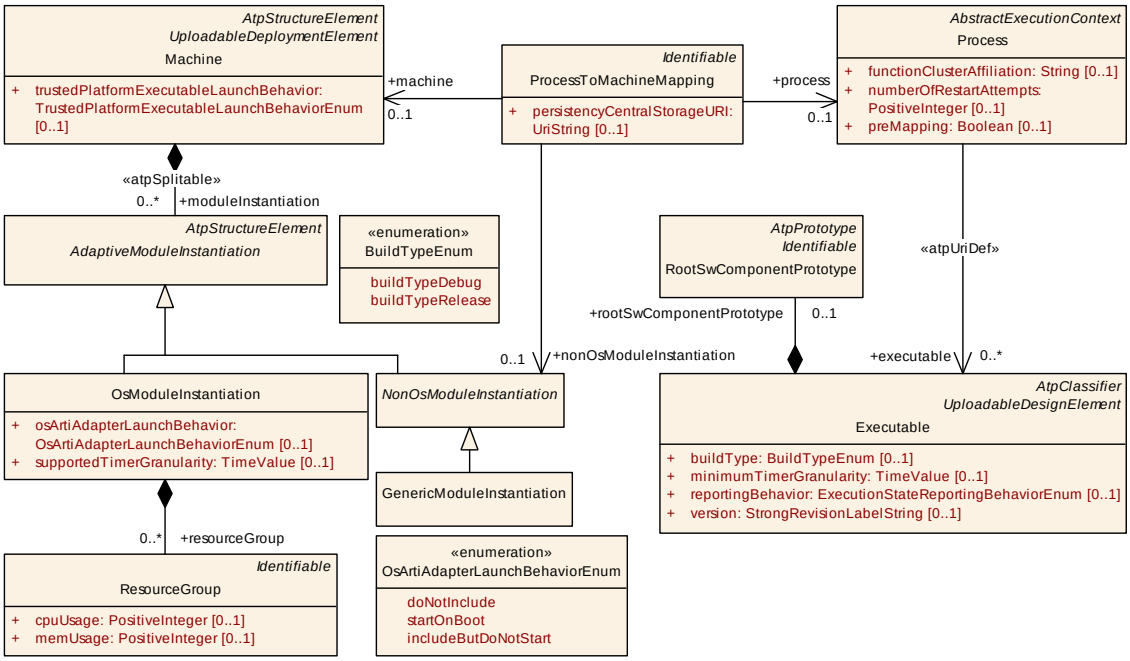


Figure 10.1: Adaptive Autosar Module Configuration

Those NonOsModuleInstantiations are independent from the startup behavior implementation. If a stack implementation decides to implement a specific functional cluster in a dedicated Process, then the specific NonOsModuleInstantiation will also be part of a ProcessToMachineMapping.

But, if the stack implementation decides to implement a specific functional cluster as a library (or make the functionality part of another functional cluster), then the specific NonOsModuleInstantiation just defines the configuration values for that functionality and does not explicitly take part in a ProcessToMachineMapping.

Another scenario is a rather distributed nature of a functional cluster, where there is no need to provide centralized configuration means. This is applicable for example to PersistencyDeployment or PlatformHealthManagementContribution.

The functional behavior of the functional cluster is determined by the sum of several contributions. There is no single configuration entity provided.

Nevertheless, if a stack implementation decides to implement such a distributed functional cluster as a single Executable, the GenericModuleInstantiation can be used to define the startup behavior for a specific machine.

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

Class	***AdaptiveModuleInstantiation*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the abstract attributes for the configuration of an adaptive autosar module instance on a specific machine.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*NonOsModuleInstantiation*, OsModuleInstantiation			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.1: AdaptiveModuleInstantiation

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

[constr_1490]{DRAFT} **Allowed value for Executable.category if ProcessToMachineMapping references a NonOsModuleInstantiation** ⌈If a ProcessToMachineMapping references a NonOsModuleInstantiation, then the Process referenced in the role ProcessToMachineMapping.process shall only refer (in the role Process.executable) to an Executable where attribute Executable.category is set to PLATFORM_LEVEL (see [constr_1605]) **at the time when the creation of the manifest is finished**.⌋*()*

Please note that the model relation described in [constr_1490] is sketched in Figure 10.1.

The meta-class GenericModuleInstantiation can be used to define configuration settings of generic modules and modules that are not standardized by AUTOSAR. Different modules are distinguishable by the category attribute.

Please note that both elements are `Identifiable` and therefore are able to describe special data (`sdg`), by which means it is possible to define generic custom settings that are not represented by the standard model. For more information, please refer to the AUTOSAR Generic Structure Template [6].

[TPS_MANI_03096]{DRAFT} **`Machine`-specific configuration settings for a generic module** ⌈The `Machine`-specific configuration settings for a generic module are collected in `GenericModuleInstantiation` where the value of attribute `category` value denotes the module.⌋*(RS_MANI_00023)*

Class	**GenericModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
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.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.2: GenericModuleInstantiation

10.1 OS Module configuration

[TPS_MANI_03098]{DRAFT} **`Machine`-specific configuration settings for the OS module** ⌈The `Machine`-specific configuration settings for the OS module are collected in `OsModuleInstantiation`.⌋*(RS_MANI_00023)*

Class	**OsModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
Note	This meta-class defines the attributes for the OS configuration on a specific machine.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
osArtiAdapter LaunchBehavior	OsArtiAdapterLaunch BehaviorEnum	0..1	attr	This attribute controls the startup behavior of the OS/ ARTI Adapter. **Tags:** atp.Status=candidate
resourceGroup	ResourceGroup	*	aggr	This represents the collection of ResourceGroups owned by the enclosing OsModuleImplementation.
supportedTimer Granularity	TimeValue	0..1	attr	This attribute describes the supported timer granularity (TimeValue of one tick).

Table 10.3: OsModuleInstantiation

[TPS_MANI_01433]{DRAFT} **Semantics of attribute `OsModuleInstantiation.osArtiAdapterLaunchBehavior`** ⌈Attribute `OsModuleInstantiation.osArtiAdapterLaunchBehavior` defines launch behavior of the OS/ARTI Adapter that is used in the system for tracing the states of the operating system, as described in SWS Operating System Interface [21].⌋ *()*

Enumeration	**OsArtiAdapterLaunchBehaviorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::LogAndTrace
Note	This enumeration provides options to start the OS/ARTI Adapter when starting the machine. **Tags:** atp.Status=candidate
Aggregated by	OsModuleInstantiation.osArtiAdapterLaunchBehavior
Literal	***Description***
doNotInclude	Do not include the OS/ARTI Adapter when building the system. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=candidate
includeButDoNot Start	Include the OS/ARTI Adapter when building the system but do not start it automatically. **Tags:** atp.EnumerationLiteralIndex=2 atp.Status=candidate
startOnBoot	Include the OS/ARTI Adapter when building the system and start it when booting the system. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=candidate

Table 10.4: OsArtiAdapterLaunchBehaviorEnum

Class	***NonOsModuleInstantiation*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the abstract attributes for the configuration of an adaptive autosar module other than the OS module.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AdaptiveFirewallModuleInstantiation, CryptoModuleInstantiation, DoIpInstantiation, GenericModule Instantiation, IamModuleInstantiation, *IdsPlatformInstantiation*, LogAndTraceInstantiation, Nm Instantiation, *SovdModuleInstantiation*, StateManagementModuleInstantiation, TimeSyncModule Instantiation, *UcmModuleInstantiation*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.5: NonOsModuleInstantiation

AUTOSAR supports the configuration of `ResourceGroup`s in the `OsModuleInstantiation` of the `Machine` that correspond for example to cgroups (aka control groups) in Linux. `ResourceGroup`s provide a mechanism to manage system resources by partitioning constraints like `cpuUsage` and `memUsage` into groups that limit the resource usage for a collection of processes (see also [TPS_MANI_01017]).

[constr_1661]{DRAFT} **Multiplicity of `OsModuleInstantiation.resourceGroup`** ⌈Any given `OsModuleInstantiation` shall always define at least

one resourceGroup **at the time when the creation of the manifest is finished**.⌋()

The rationale for [constr_1661] is that the StateDependentStartupConfig requires a reference to a ResourceGroup.

More information about the semantics of meta-class ResourceGroup can be found in [SWS_OSI_02001].

Class	**ResourceGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
Note	This meta-class represents a resource group that limits the resource usage of a collection of processes.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	OsModuleInstantiation.resourceGroup			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cpuUsage	PositiveInteger	0..1	attr	CPU resource limit in percentage of the total CPU capacity on the machine.
memUsage	PositiveInteger	0..1	attr	Memory limit in bytes.

Table 10.6: ResourceGroup

10.2 Persistency Deployment

10.2.1 Overview

This chapter explains the part of the support for persistent storage in terms of mapping of concrete storage models to the corresponding parts of the application software.

[TPS_MANI_01205]{DRAFT} **Semantics of meta-class PersistencyDeployment** ⌈Abstract meta-class PersistencyDeployment provides shared attributes to more specific specializations.⌋*(RS_MANI_00027)*

[TPS_MANI_01380]{DRAFT} **Usage of PersistencyDeployment** ⌈Meta-class PersistencyDeployment can be used in different scenarios:

- Persistency is used in combination with the application software by means of PersistencyPortPrototypeToDeploymentMapping.
- Persistency is used in combination with another functional cluster by means of FunctionalClusterInteractsWithPersistencyDeploymentMapping.

⌋*(RS_MANI_00027)*

Class	***PersistencyDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This abstract meta-class serves as a base class for concrete classes representing different aspects of persistency.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusivePackageElement*, *UploadablePackageElement*			
Subclasses	PersistencyFileStorage, PersistencyKeyValueStorage			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
deploymentUri (ordered)	PersistencyDeployment Uri	*	aggr	This aggregation represents the collection of URIs relevant for the enclosing PersistencyDeployment.
maximum AllowedSize	PositiveUnlimitedInteger	0..1	attr	The value of this attribute represents the maximum size (unit: bytes) allowed at deployment time for the enclosing PersistencyDeployment.
minimum SustainedSize	PositiveInteger	0..1	attr	The value of this attribute represents the minimum size (unit: bytes) guaranteed at deployment time for the enclosing PersistencyDeployment.
redundancy Handling	PersistencyRedundancy Handling	*	aggr	This aggregation represents the chosen approaches to handle redundancy.
updateStrategy	PersistencyCollection LevelUpdateStrategy Enum	0..1	attr	This attribute shall be used to specify the update strategy of the respective PersistencyDeployment as a whole.
version	StrongRevisionLabel String	0..1	attr	The attribute represents the version of the `PersistencyFileStorage` or `PersistencyKeyValueStorage`.

Table 10.7: PersistencyDeployment

[constr_10177]{DRAFT} **Multiplicity of attribute `PersistencyDeployment.updateStrategy`** ⌈For each `PersistencyDeployment`, the attribute `updateStrategy` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10365]{DRAFT} **Existence of `PersistencyDeployment.deploymentUri`** ⌈For each concrete sub-class of `PersistencyDeployment`, attribute `deploymentUri` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10509]{DRAFT} **Existence of attribute `PersistencyDeployment.maximumAllowedSize`** ⌈For each concrete sub-class of `PersistencyDeployment`, attribute `maximumAllowedSize` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10510]{DRAFT} **Existence of attribute `PersistencyDeployment.minimumSustainedSize`** ⌈For each concrete sub-class of `PersistencyDeployment`, attribute `minimumSustainedSize` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

Class	PersistencyDeploymentUri			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to contain URIs relevant for the persistency deployment.			
Base	*ARObject*			
Aggregated by	*PersistencyDeployment*.deploymentUri			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
uri	UriString	0..1	attr	This attribute holds the storage location for the concrete subclass of PersistencyDeployment, e.g. file on the file system.

Table 10.8: PersistencyDeploymentUri

The possibility that attribute `deploymentUri` has a multiplicity greater than 1 is only foreseen to be utilized if redundancy handling on the basis of m-out-of-n is activated, i.e. the aggregation of `PersistencyRedundancyMOutOfN` is aggregated in the role `PersistencyDeployment.redundancyHandling`.

In this case, the actual multiplicity of `PersistencyDeployment.deploymentUri` in an AUTOSAR model shall be one of

- 1: in this case all copies are stored in one location.
- 2: in this case, the "main" copy is stored in the location `deploymentUri`[0] and all other copies are stored at the location `deploymentUri`[1]
- the value of `PersistencyRedundancyMOutOfN.n`: each copy is stored in the location `deploymentUri`[i], where $0 <= i < n$

Please find more details about the storage strategy in the SWS Persistency [13].

[constr_10366]{DRAFT} **Possible multiplicities of `PersistencyDeployment.deploymentUri`** ⌈Possible multiplicities of `PersistencyDeployment.deploymentUri` shall be one of

- 1
- 2
- value of attribute `PersistencyRedundancyMOutOfN.n`

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

[constr_10367]{DRAFT} **Condition for the multiplicity of attribute `PersistencyDeployment.deploymentUri`** ⌈The multiplicity of attribute `PersistencyDeployment.deploymentUri` shall only be greater than 1 if meta-class `PersistencyRedundancyMOutOfN` is aggregated in the role `PersistencyDeployment.redundancyHandling` and attribute `PersistencyDeployment.redundancyHandling.scope` is set to the value `PersistencyRedundancyHandlingScopeEnum.persistencyRedundancyHandlingScopeStorage`.

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

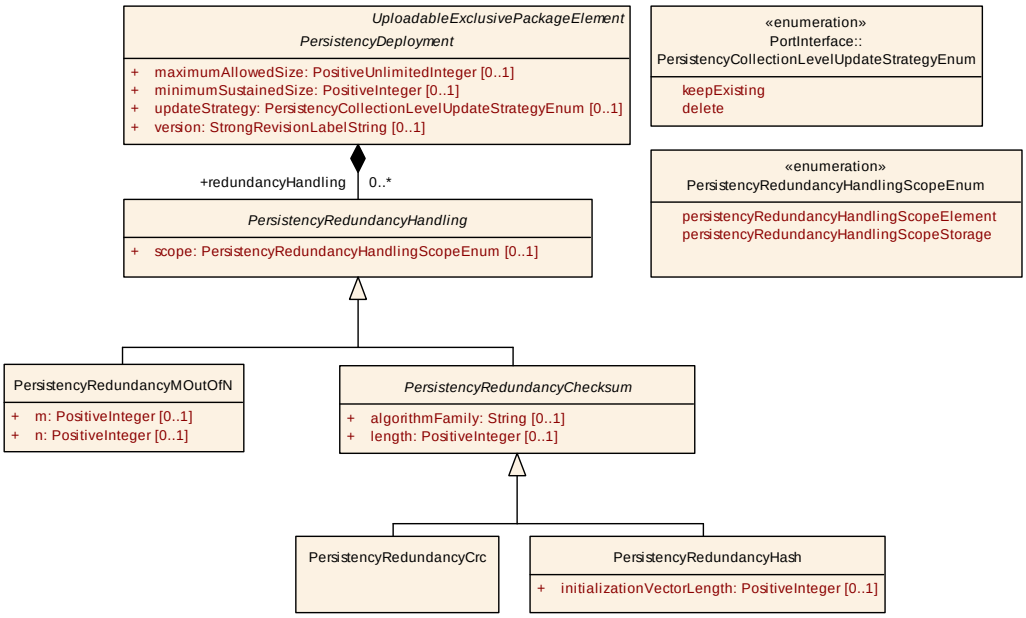


Figure 10.2: Modeling of the abstract base class `PersistencyDeployment`

[constr_10035]{DRAFT} **Completeness of the `PersistencyDeployment.version`** ⌈The `PersistencyDeployment.version` shall contain all the following parts:

- Major version
- Minor version
- Patch version
- Additional labels for pre-release version and build metadata

`at the time when the creation of the manifest is finished`.⌋*()*

The `version` is used to determine whether an update or rollback of persistent data is required. When an application opens the `PersistencyFileStorage` or `PersistencyKeyValueStorage`, the Persistency module will check the `PersistencyDeployment.version` in the manifest against the stored version information.

If the stored version is lower than the manifest version, the Persistency module will update persistent data after creating a backup of the data (see [SWS_PER_00386] for more details).

If the stored version is higher than the manifest version, a rollback of the persistent data from the backup is required (see [SWS_PER_00396] for more details).

[TPS_MANI_01321]{DRAFT} **Semantics of meta-class `PersistencyDeploymentElement`** ⌈Meta-class `PersistencyDeploymentElement` represents an abstract base class for the modeling of different aspects of persistency on element level.⌋ *(RS_MANI_00027)*

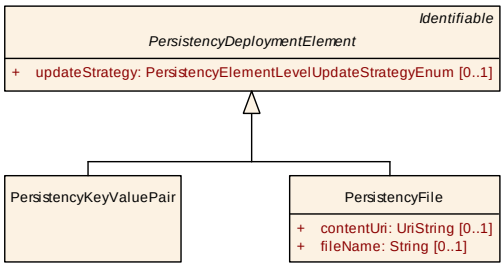


Figure 10.3: Modeling of the abstract base class `PersistencyDeploymentElement`

Class	***PersistencyDeploymentElement*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This abstract meta-class serves as a base class for concrete classes representing different aspects of elements of a `PersistencyDeployment`.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	PersistencyFile, PersistencyKeyValuePair			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
updateStrategy	PersistencyElement LevelUpdateStrategy Enum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyDeploymentElement.

Table 10.9: PersistencyDeploymentElement

[constr_10426]{DRAFT} **Multiplicity of attribute `PersistencyDeploymentElement.updateStrategy`** ⌈For each `PersistencyDeploymentElement`, the attribute `updateStrategy` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[TPS_MANI_01322]{DRAFT} **Semantics of meta-class `PersistencyPortPrototypeToDeploymentMapping`** ⌈Meta-class `PersistencyPortPrototypeToDeploymentMapping` represents an abstract base class for the modeling of the mapping of concrete persistency cases (key-value storage, file storage) to a `PortPrototype` and a `Process`.⌋*(RS_MANI_00027)*

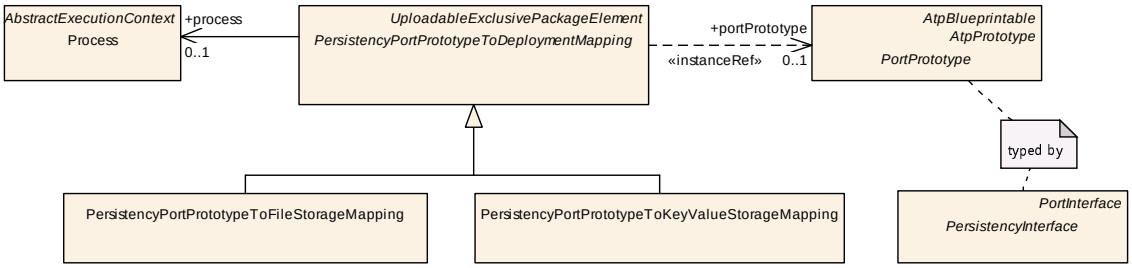


Figure 10.4: Modeling of the abstract base class `PersistencyPortPrototypeToDeploymentMapping`

Class	*PersistencyPortPrototypeToDeploymentMapping* (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This abstract bas class implements the shared functionality of all mapping between a PortPrototype, a Process, and a specific subclass of PersistencyDeployment.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusivePackageElement*, *UploadablePackageElement*			
Subclasses	PersistencyPortPrototypeToFileStorageMapping, PersistencyPortPrototypeToKeyValueStorageMapping			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
portPrototype	PortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **InstanceRef implemented by:** PortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference represents the process required as context for the mapping.

Table 10.10: PersistencyPortPrototypeToDeploymentMapping

[constr_10178]{DRAFT} **Multiplicity of the reference in the role PersistencyPortPrototypeToDeploymentMapping.process** ⌈For each PersistencyPortPrototypeToDeploymentMapping, the reference in the role process shall exist **at the time when the creation of the manifest is finished**.⌋*()*

10.2.1.1 Redundancy Handling

[TPS_MANI_01206]{DRAFT} **Modeling of redundancy in the context of PersistencyDeployment** ⌈The deployment level provides the ability to provide a more detailed definition of redundant behavior for both key-value storage and file storage.

This modeling is attached to the abstract base class PersistencyDeployment in order to let both aspects of persistency on the *AUTOSAR adaptive platform* benefit from the existence of meta-class PersistencyRedundancyHandling.⌋*(RS_MANI_00027)*

[constr_1710]{DRAFT} **Consistency of values of attributes PersistencyInterface.redundancy and PersistencyRedundancyHandling.scope** ⌈If attribute PersistencyInterface.redundancy is set to value PersistencyRedundancyEnum.redundantPerElement then attribute PersistencyRedundancyHandling.scope shall be set to PersistencyRedundancyHandlingScopeEnum.persistencyRedundancyHandlingScopeElement for at least one PersistencyRedundancyHandling aggregated by the corresponding PersistencyDeployment **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_01371]{DRAFT} **PersistencyDeployment.redundancyHandling does not exist** ⌈If no aggregation in the role redundancyHandling exists for a specific PersistencyDeployment, then the configuration of the redundancy handling shall be taken from the value of PersistencyInterface.redundancyHandling, if this aggregation exists.⌋*(RS_MANI_00027)*

10.2.1.2 Update Handling

[TPS_MANI_01155]{DRAFT} **PersistencyDeployment.updateStrategy overrides PersistencyInterface.updateStrategy** ⌈The value of attribute PersistencyDeployment.updateStrategy shall overrule the value of PersistencyInterface.updateStrategy for any combination of PersistencyInterface mapped to a PersistencyDeployment by means of a PersistencyPortPrototypeToDeploymentMapping.⌋*(RS_MANI_00027)*

[TPS_MANI_01147]{DRAFT} **Semantics of PersistencyDeployment.updateStrategy** ⌈The attribute PersistencyDeployment.updateStrategy shall be used to specify the strategy for updating the actual persistent elements.

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

[TPS_MANI_01157]{DRAFT} **Semantics of updateStrategy on collection level** ⌈

updateStrategy	Use Case: Installation	Use Case: Update
delete	irrelevant	delete all elements not contained in current manifest
keepExisting	irrelevant	keep all elements not contained in current manifest

⌋*(RS_MANI_00027)*

[TPS_MANI_01313]{DRAFT} **Definition of updateStrategy on element level** ⌈The definition of the update strategy on element level is modeled by means of the abstract base class PersistencyDeploymentElement (and its attribute updateStrategy) from which the concrete sub-classes for persistency elements are derived.⌋ *(RS_MANI_00027)*

[TPS_MANI_01159]{DRAFT} **Semantics of updateStrategy on element level** ⌈

updateStrategy	Use Case: Installation	Use Case: Update
delete	don't create	remove
keepExisting	create	do nothing
overwrite	create	replace

⌋*(RS_MANI_00027)*

[TPS_MANI_01148]{DRAFT} **Semantics of PersistencyDeploymentElement.updateStrategy** ⌈The attribute PersistencyDeploymentElement.updateStrategy can be used to specify the strategy for updating the actual persistent element that corresponds to PersistencyDeploymentElement.⌋*(RS_MANI_00027)*

[TPS_MANI_01156]{DRAFT} **PersistencyDeploymentElement.updateStrategy overrides PersistencyDeployment.updateStrategy** ⌈The value specified for PersistencyDeploymentElement.updateStrategy overrides the value of PersistencyDeployment.updateStrategy for this specific PersistencyDeploymentElement.⌋*(RS_MANI_00027)*

[TPS_MANI_01182]{DRAFT} **Value of PersistencyDeploymentElement.updateStrategy overrides PersistencyInterfaceElement.updateStrategy** ⌈The value of attribute PersistencyDeploymentElement.updateStrategy overrides the value of attribute PersistencyInterfaceElement.updateStrategy⌋*(RS_MANI_00027)*

This means that the integrator of the software gets the authority to either agree to the designer's point of view or else overrule the designer's decision based on superior knowledge regarding the integration strategy.

10.2.1.3 Size Handling

[TPS_MANI_01196]{DRAFT} **Semantics of PersistencyDeployment.minimumSustainedSize** ⌈Attribute PersistencyDeployment.minimumSustainedSize can be used for the definition of a **minimum amount of storage** that the PersistencyDeployment will need to allocate from an integrator's point of view.

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

[TPS_MANI_01197]{DRAFT} **Semantics of PersistencyDeployment.maximumAllowedSize** ⌈Attribute PersistencyDeployment.maximumAllowedSize can be used for the definition of the **maximum amount of storage** that the PersistencyDeployment may allocate at runtime from an integrator's point of view.

The existence of PersistencyDeployment.maximumAllowedSize does not constitute a binding requirement to the platform that this amount of storage shall be available at any time.⌋*(RS_MANI_00027)*

For explanation, the amount of storage available shall be at least the sum of the values of minimumSustainedSize.

That said, it is consequently plausible that storage might be exceeded if more than the minimum amount of storage (let alone the maximum amount) is allocated by all the key-value storage at the same time.

10.2.1.4 Security Handling

The encryption and/or authentication of data stored in a Key-Value Storage or File Storage is described in the manifest by PersistencyDeploymentToCryptoKeySlotMapping or PersistencyDeploymentElementToCryptoKeySlotMapping that are described in more detail in chapter 14.4 and chapter 14.5.

If the PersistencyDeploymentToCryptoKeySlotMapping.keySlotUsage or PersistencyDeploymentElementToCryptoKeySlotMapping.keySlotUsage is set to encryption, the Persistency cluster shall encrypt the data before storing

it to the persistent memory or shall decrypt the data after reading it from persistent memory.

If the PersistencyDeploymentToCryptoKeySlotMapping.keySlotUsage or PersistencyDeploymentElementToCryptoKeySlotMapping.keySlotUsage is set to verification, the Persistency cluster shall sign the data before storing it to the persistent memory or verify the signature of the data after reading it from persistent memory.

Please note that the PersistencyDeploymentToCryptoKeySlotMapping is able to define a verificationHash that shall by used by the PersistencyCluster to verify the data. The same is true for the PersistencyDeploymentElementToCryptoKeySlotMapping.verificationHash.

10.2.1.5 Preference Rules

In the context of the modeling of persistency design (see section 3.3.5) and deployment (section 10.2), information can be available on the *design* and *deployment* level, and on *storage* and *element* level. In general, the following rules apply regarding the preference of this information:

- if a configuration item exists in the context of the *design element level*, then the value of this configuration item shall be preferred over the value of a corresponding configuration item available on the *design storage level*.
- if a configuration item exists in the context of the *deployment element level*, then the value of this configuration item shall be preferred over the value of a corresponding configuration item available on the *deployment storage level*.
- if a configuration item exists in the context of the *deployment storage level*, then the value of this configuration item shall be preferred over the value of a corresponding configuration item available on the *design storage level*.
- if a configuration item exists in the context of the *deployment element level*, then the value of this configuration item shall be preferred over the value of a corresponding configuration item available on the *design element level*.

In other words, there is a growing priority of the value of a configuration item starting from *design storage level*, to *design element level*, to *deployment storage level*, and finally to *deployment element level*.

On the other hand, models may exist that leave out certain information on the deployment level in favor of implicitly taking over the corresponding information from design level. This approach is not encouraged because (where possible) model semantics should preferably be defined explicitly rather than implied by rules.

Nevertheless, the following rules apply:

- if a configuration item exists on the *deployment storage level* but is missing on the *deployment element level*, then the *deployment element level* shall implicitly take over the value from the *deployment storage level*.
- if a configuration item exists on the *design storage level* but is missing on the *deployment storage level*, then the *deployment storage level* shall implicitly take over the value from the *design storage level*.
- if a configuration item exists on the *design element level* but is missing on the *deployment element level*, then the *deployment element level* shall implicitly take over the value from the *design element level*.
- if a configuration item exists on the *design storage level* but is missing on the *design element level*, then the *design element level* implicitly takes over the value from the *design storage level*.

10.2.2 Deployment of Persistent Key-Value Storage

[TPS_MANI_01079]{DRAFT} **Semantics of meta-class `PersistencyKeyValueStorage`** ⌈Meta-class `PersistencyKeyValueStorage` represents an actual key-value storage used for persistently storing data.⌋ *(RS_MANI_00027)*

Class	**PersistencyKeyValueStorage**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to model a key-value storage on deployment level. **Tags:** atp.recommendedPackage=PersistencyKeyValueStorages			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PersistencyDeployment*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusive PackageElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
keyValuePair	PersistencyKeyValue Pair	*	aggr	This aggregation represents the key-value-pairs owned by the enclosing `PersistencyKeyValueStorage`.

Table 10.11: PersistencyKeyValueStorage

[TPS_MANI_01144]{DRAFT} **Semantics of `PersistencyKeyValuePair`** ⌈Meta-class `PersistencyKeyValuePair` represents an **entry** to a key-value storage (formalized by `PersistencyKeyValueStorage`) used for persistently storing data.⌋ *(RS_MANI_00027)*

[constr_10102]{DRAFT} **Existence of initial values for `PersistencyKeyValuePair`** ⌈For each `PersistencyKeyValuePair`, if the value of attribute `updateStrategy` is set to either of the values

- `keepExisting` or
- `overwrite`,

then attribute PersistencyKeyValuePair.initValue shall exist **at the time when the creation of the manifest is finished**.⌋()

Class	PersistencyKeyValuePair			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to formally model a key-value pair in the context of the deployment of persistency.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PersistencyDeploymentElement*, *Referrable*			
Aggregated by	PersistencyKeyValueStorage.keyValuePair			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
initValue	ValueSpecification	0..1	aggr	This aggregation represents the ability to define an initial value for the value side of the key-value pair. Please note that it does not make sense to configure an initial value if the PersistencyDeploymentElement.updateStrategy is set to the value delete.
valueDataType	AbstractImplementationDataType	0..1	ref	This reference represents the data type applicable for the value of the key-value pair.

Table 10.12: PersistencyKeyValuePair

[constr_10179]{DRAFT} **Multiplicity of attribute PersistencyKeyValuePair.valueDataType** ⌈For each PersistencyKeyValuePair, the attribute valueDataType shall exist **at the time when the creation of the manifest is finished**.⌋()

[constr_10083]{DRAFT} **Existence of initial values for PersistencyKeyValuePair** ⌈For each PersistencyKeyValuePair, if the value of attribute updateStrategy is set to the value delete, then attribute PersistencyKeyValuePair.initValue shall not exist **at the time when the creation of the manifest is finished**.⌋()

The modeling of PersistencyKeyValuePair aggregated in the role PersistencyKeyValueStorage.keyValuePair is optional. It would be possible to use persistency functionality regardless of the existence of keyValuePair.

However, the presence of keyValuePair gives more freedom and ways for the customization of behavior.

[TPS_MANI_01078]{DRAFT} **Semantics of PersistencyPortPrototypeToKeyValueStorageMapping** ⌈Meta-class PersistencyPortPrototypeToKeyValueStorageMapping has the ability to map a specific PortPrototype referenced in the role portPrototype to a PersistencyKeyValueStorage referenced in the role keyValueStorage.

The mapping also comprises a reference to meta-class process in order to accommodate for the fact that identical combinations of keyValueStorage and portPrototype may or may not apply for a given Process that represents the enclosing Executable at runtime.⌋(*RS_MANI_00027*)

[constr_1555]{DRAFT} **Restriction applicable for PersistencyPortPrototypeToKeyValueStorageMapping.portPrototype** ⌈The reference PersistencyPortPrototypeToKeyValueStorageMapping.portPrototype shall only be

used for a PortPrototype typed by a PersistencyKeyValueStorageInterface **at the time when the creation of the manifest is finished**.⌋ ()

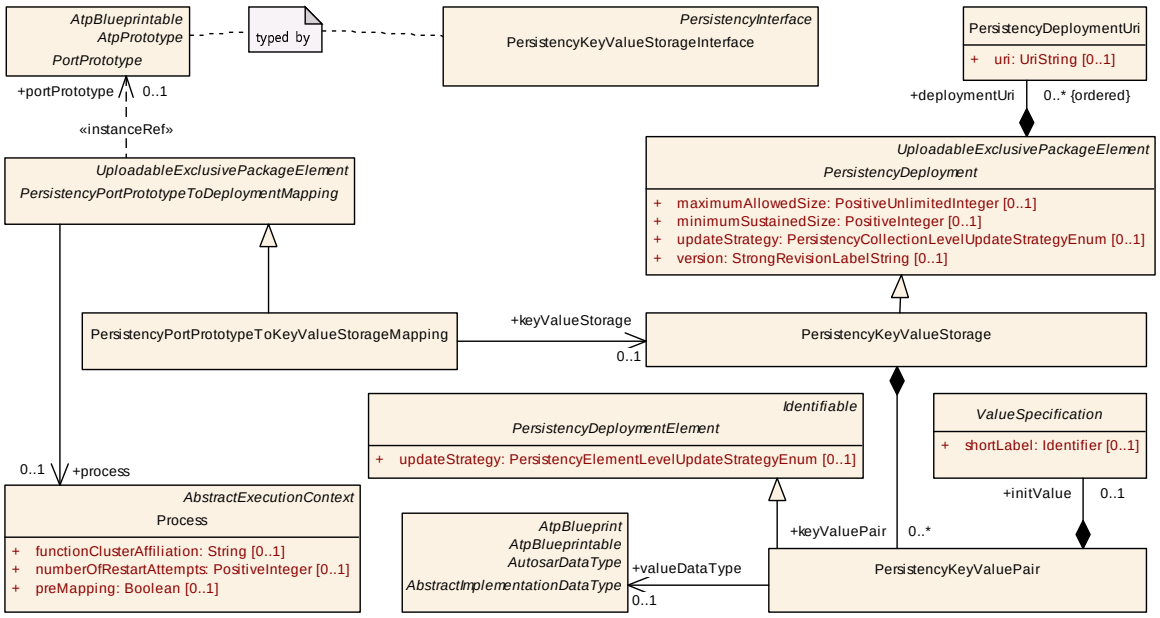


Figure 10.5: Connect a specific PortPrototype to a PersistencyKeyValueStorage

Class	**PersistencyPortPrototypeToKeyValueStorageMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to define a mapping between a PortPrototype and a key-value storage. **Tags:** atp.recommendedPackage=PersistencyPortPrototypeToKeyValueStorageMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PersistencyPortPrototypeToDeploymentMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusivePackageElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
keyValue Storage	PersistencyKeyValue Storage	0..1	ref	This reference represents the mapped key-value storage.

Table 10.13: PersistencyPortPrototypeToKeyValueStorageMapping

[constr_10180]{DRAFT} **Multiplicity of the reference in the role PersistencyPortPrototypeToKeyValueStorageMapping.keyValueStorage** ⌈For each PersistencyPortPrototypeToKeyValueStorageMapping, the reference in the role keyValueStorage shall exist **at the time when the creation of the manifest is finished**.⌋ ()

Please note that typically the existence of PersistencyKeyValueStorage.keyValuePair depends on the existence of PersistencyKeyValueStorageInterface.dataElement.

On the other hand, if a PersistencyKeyValueStorage contains PersistencyKeyValuePairs that do not correspond to any dataElements of the PersistencyKeyValueStorageInterface that is mapped (indirectly) via PersistencyPortPrototypeToKeyValueStorageMapping then those keyValuePairs are created within the PersistencyKeyValueStorage.

[TPS_MANI_01146]{DRAFT} **Initial value for PersistencyKeyValuePair** ⌈It is possible to define an initial value for a given PersistencyKeyValuePair by means of the aggregation of ValueSpecification in the role initValue.⌋*(RS_MANI_00027)*

[constr_1554]{DRAFT} **Restriction regarding attribute PersistencyKeyValuePair.initValue** ⌈The concrete sub-class of ValueSpecification aggregated in the role PersistencyKeyValuePair.initValue shall not (after resolving a possible redirection by means of ConstantReference) be a ReferenceValueSpecification **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_01315]{DRAFT} **PersistencyKeyValuePair.initValue overrides PersistencyDataRequiredComSpec.initValue** ⌈The value of attribute PersistencyKeyValuePair.initValue shall overrule the value of PersistencyDataRequiredComSpec.initValue for any combination of PersistencyKeyValueStorageInterface mapped to a PersistencyKeyValueStorage by means of a PersistencyPortPrototypeToKeyValueStorageMapping.⌋*(RS_MANI_00027)*

This means that the integrator of the software gets the authority to either agree to the designer's point of view or else overrule the designer's decision based on superior knowledge regarding the integration strategy.

[constr_1582]{DRAFT} **PersistencyKeyValuePair.valueDataType shall match to AbstractImplementationDataType for the corresponding PersistencyDataElement** ⌈Each PersistencyKeyValuePair.valueDataType shall match the AbstractImplementationDataType that either directly or indirectly (via the applicable DataTypeMap) types the corresponding (based on identical values of the respective shortName) PersistencyDataElement **at the time when the creation of the manifest is finished**.⌋*()*

[constr_1666]{DRAFT} **References from PersistencyPortPrototypeToKeyValueStorageMapping to PersistencyKeyValueStorage** ⌈Each PersistencyKeyValueStorage shall only be referenced by at most one PersistencyPortPrototypeToKeyValueStorageMapping **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_01323]{DRAFT} **Matching pairs of PersistencyDataElement and PersistencyKeyValuePair** ⌈Matching pairs of PersistencyDataElement and

PersistencyKeyValuePair shall be identified by having the identical value of attribute shortName within the scope of a PersistencyKeyValueStorageInterface (or a PortPrototype typed by the PersistencyKeyValueStorageInterface) mapped to a PersistencyKeyValueStorage by means of a PersistencyPortPrototypeToKeyValueStorageMapping.⌋(*RS_MANI_00027*)

10.2.3 Deployment of File Storage

[TPS_MANI_01150]{DRAFT} **Semantics of PersistencyFileStorage** ⌈A PortPrototype typed by a PersistencyFileStorageInterface actually builds an abstraction for an entire directory of files.

This abstraction is also visible in the deployment by means of the existence of the companion meta-class PersistencyFileStorage.

This approach allows for the dynamic creation and/or deletion of files during runtime while still keeping the structural model of the file interaction static.⌋(*RS_MANI_00027*)

At one point, however, it is necessary to boil down the relation of such a PortPrototype typed by a PersistencyFileStorageInterface 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 PersistencyPortPrototypeToFileStorageMapping, as depicted in Figure 10.6.

Class	**PersistencyFileStorage**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class comes with the ability to define a collection of single files (directory) that creates the deployment-side counterpart to a PortPrototype typed by a PersistencyFileStorageInterface. **Tags:** atp.recommendedPackage=PersistencyFileStorages			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PersistencyDeployment*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusive PackageElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
file	PersistencyFile	*	aggr	This aggregation represents the collection of files aggregated by the PersistencyFileStorage.

Table 10.14: PersistencyFileStorage

[TPS_MANI_01080]{DRAFT} **Semantics of meta-class PersistencyPortPrototypeToFileStorageMapping** ⌈Meta-class PersistencyPortPrototypeToFileStorageMapping creates a mapping between a PortPrototype referenced in the role portPrototype to a PersistencyFileStorage referenced in the role fileStorage under consideration of a Process referenced in the role process.⌋(*RS_MANI_00027*)

Class	**PersistencyPortPrototypeToFileStorageMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to define a mapping between a collection of files on deployment level to a given `PortPrototype`. **Tags:** atp.recommendedPackage=PersistencyPortPrototypeToFileStorageMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyPortPrototypeToDeploymentMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusivePackageElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
fileStorage	PersistencyFileStorage	0..1	ref	This reference represents the mapped file storage.

Table 10.15: PersistencyPortPrototypeToFileStorageMapping

[constr_10182]{DRAFT} **Multiplicity of the reference in the role `PersistencyPortPrototypeToFileStorageMapping.fileStorage`** ⌈For each `PersistencyPortPrototypeToFileStorageMapping`, the reference in the role `fileStorage` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

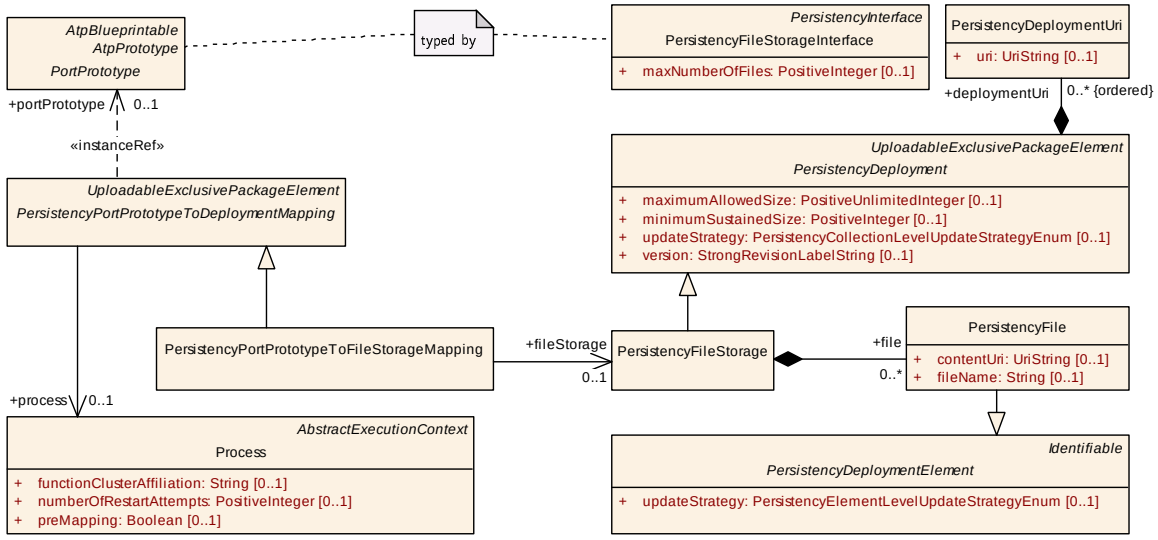


Figure 10.6: Connect a specific `PortPrototype` to a `PersistencyFile`

[TPS_MANI_01149]{DRAFT} **Semantics of `PersistencyFileStorage.file`** ⌈The usage of `PersistencyFileStorage.file` allows for the explicit modeling of elements of the `PersistencyFileStorage`.

The creation of this aggregation is optional. It can be used to define the update strategy and/or initial content of selected files.⌋*(RS_MANI_00027)*

[constr_10103]{DRAFT} **Existence of initial values for `PersistencyFile`** ⌈For each `PersistencyFile`, if the value of attribute `updateStrategy` is set to either of the values

- `keepExisting` or
- `overwrite`,

then attribute PersistencyFile.contentUri shall exist **at the time when the creation of the manifest is finished**.⌋()

[constr_1556]{DRAFT} **Restriction applicable for PersistencyPortPrototypeToFileStorageMapping.portPrototype** ⌈The reference PersistencyPortPrototypeToFileStorageMapping.portPrototype shall only be used for a PortPrototype typed by a PersistencyFileStorageInterface **at the time when the creation of the manifest is finished**.⌋()

Class	**PersistencyFile**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the model of a file as part of the persistency on deployment level. **Tags:** atp.recommendedPackage=PersistencyFiles			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PersistencyDeploymentElement*, *Referrable*			
Aggregated by	PersistencyFileStorage.file			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contentUri	UriString	0..1	attr	This attribute represents the URI that identifies the initial content of the PersistencyFile.
fileName	String	0..1	attr	This attribute holds filename part of the storage location for the PersistencyFile, e.g. file on the file system.

Table 10.16: PersistencyFile

[constr_10183]{DRAFT} **Multiplicity of attribute PersistencyFile.fileName** ⌈For each PersistencyFile, the attribute fileName shall exist **at the time when the creation of the manifest is finished**.⌋()

[constr_10082]{DRAFT} **Existence of initial values for PersistencyFile** ⌈For each PersistencyFile, if the value of attribute updateStrategy is set to the value delete, then attribute PersistencyFile.contentUri shall not exist **at the time when the creation of the manifest is finished**.⌋()

[TPS_MANI_01179]{DRAFT} **Semantics of PersistencyFileElement.contentUri/PersistencyFile.contentUri vs. PersistencyDeployment.deploymentUri and PersistencyFileElement.fileName/PersistencyFile.fileName** ⌈Attributes PersistencyFileElement.contentUri and (after deployment) PersistencyFile.contentUri describe the URI of the file storage that is used to initialize the PersistencyFile (used during install or update).

On the other hand, the combination of PersistencyDeployment.deploymentUri and the PersistencyFileElement.fileName or (after deployment) PersistencyFile.fileName denote the position of the PersistencyFile in the ECU (used at run-time).⌋(*RS_MANI_00027*)

[constr_1589]{DRAFT} **Value of file.fileName** ⌈Within the scope of any given PersistencyFileStorage, the value of all file.fileName shall be unique **at the time when the creation of the manifest is finished**.

A fileName is considered unique if there are no other fileNames with **exactly** the same sequence of characters[1].⌋()

[TPS_MANI_01187]{DRAFT} **Matching pairs of PersistencyFileElement and PersistencyFile** ⌈Matching pairs of PersistencyFileElement and PersistencyFile shall be identified by having the identical value of attribute shortName within the scope of a PersistencyFileStorageInterface (or a PortPrototype typed by the PersistencyFileStorageInterface) mapped to a PersistencyFileStorage by means of a PersistencyPortPrototypeToFileStorageMapping.⌋(*RS_MANI_00027*)

[constr_1613]{DRAFT} **File name of matching pairs of PersistencyFileElement and PersistencyFile** ⌈The value of attributes PersistencyFileElement.fileName and PersistencyFile.fileName shall be identical for matching pairs (as identified by the application of [TPS_MANI_01187]) of PersistencyFileStorage and PersistencyFile **at the time when the creation of the manifest is finished**.⌋()

[constr_1667]{DRAFT} **References from PersistencyPortPrototypeToFileStorageMapping to PersistencyFileStorage** ⌈Each PersistencyFileStorage shall only be referenced by at most one PersistencyPortPrototypeToFileStorageMapping **at the time when the creation of the manifest is finished**.⌋()

10.3 Platform Health Management Deployment

10.3.1 Overview

This chapter explains the interaction of application software with the Platform Health Management [12].

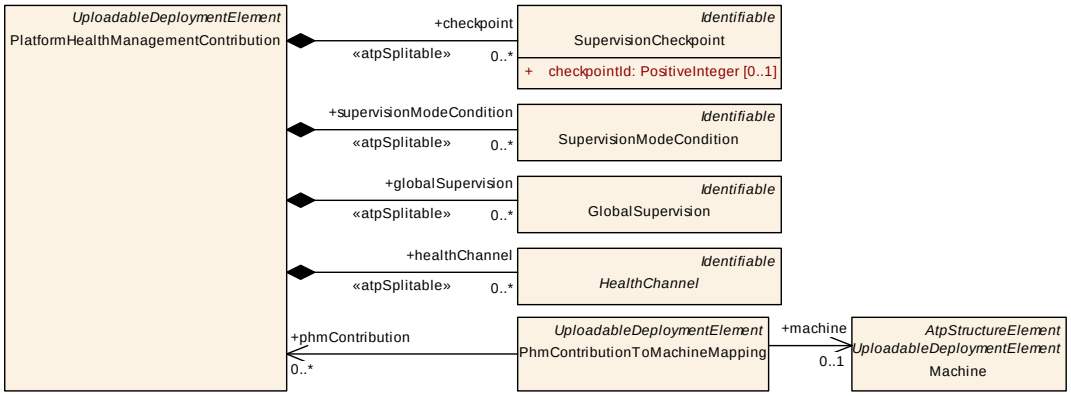


Figure 10.7: Modeling of PlatformHealthManagementContribution

[1]The characters “x” and “X” are not considered as identical characters for this purpose.

The PlatformHealthManagementContribution allows describing aspects for the deployment of configuration how the Platform Health Management shall behave during runtime.

[TPS_MANI_03544]{DRAFT} **Definition of PlatformHealthManagementContribution** ⌈The meta-class PlatformHealthManagementContribution allows to define a set of configuration entities for the Platform Health Management.⌋ *(RS_MANI_00023, RS_MANI_00032)*

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

- Supervision (section 10.3.3)
- Health channels (section 10.3.4)
- Recovery Notification (section 10.3.5)

Class	**PlatformHealthManagementContribution**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a contribution to the Platform Health Management. **Tags:** atp.recommendedPackage=PlatformHealthManagementContributions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checkpoint	SupervisionCheckpoint	*	aggr	Collection of checkpoints in the context of a Platform HealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=checkpoint.shortName xml.sequenceOffset=10
global Supervision	GlobalSupervision	*	aggr	Collection of GlobalSupervisions in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=globalSupervision.shortName xml.sequenceOffset=30
healthChannel	HealthChannel	*	aggr	Collection of HealthChannels in the context of a Platform HealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=healthChannel.shortName xml.sequenceOffset=40
supervision ModeCondition	SupervisionMode Condition	*	aggr	Collection of SupervisionModeConditions in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=supervisionModeCondition.shortName xml.sequenceOffset=20

Table 10.17: PlatformHealthManagementContribution

[TPS_MANI_03502]{DRAFT} **Enabling of `PlatformHealthManagementContribution` on a `Machine`** ⌈To enable an instance of `PlatformHealthManagementContribution` on a specific `Machine` the `PlatformHealthManagementContribution` shall be mapped to the `Machine` via a `PhmContributionToMachineMapping`.⌋*(RS_MANI_00023, RS_MANI_00032)*

[constr_3568]{DRAFT} **No support for cross `PlatformHealthManagementContribution` references** ⌈All references originating on elements aggregated by one `PlatformHealthManagementContribution` shall only refer to elements that are part of the same `PlatformHealthManagementContribution` aggregation chain **at the time when the creation of the manifest is finished**.⌋*()*

Class	**PhmContributionToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element associates one or more PlatformHealthManagementContributions with a Machine. **Tags:** atp.recommendedPackage=PhmContributionToMachineMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
machine	Machine	0..1	ref	This reference identifies the Machine in the context of the PhmContributionToMachineMapping.
phm Contribution	PlatformHealth Management Contribution	*	ref	This reference identifies one or more PlatformHealth ManagementContributions in the context of a Phm ContributionToMachineMapping.

Table 10.18: PhmContributionToMachineMapping

An application software can define the usage of several Platform Health Management supervisions (see chapter 3.3.7.2) and health channels (see chapter 3.3.7.3).

In order to define the interaction between the application software and the Platform Health Management the `PlatformHealthManagementContribution` creates its own representations of the `RPortPrototype`s typed by the `PhmSupervisedEntityInterface` and `PhmHealthChannelInterface` and creates relations to the application software `RPortPrototype`s (see figure 10.8).

In chapter 3.3.7.2 it is explained that the application software just calls methods in the context of the respective `RPortPrototype`s to interact with the Platform Health Management. From the application developer these methods have no addressing information, because the identity of the `RPortPrototype` is the identification in the scope of the application software.

The deployed structure (according to figure 10.7) however requires more information when an `API` at the Platform Health Manager is called, namely:

- `RPortPrototype.shortName` i.e. `InstanceSpecifier`
- `Process` identification during runtime.

These additional arguments have to be injected to the API by the implementation of the interaction between the software component and the Platform Health Management (which implements the relations from figure 10.7). The order of this argument injection is determined by the specification of the Platform Health Management APIs.

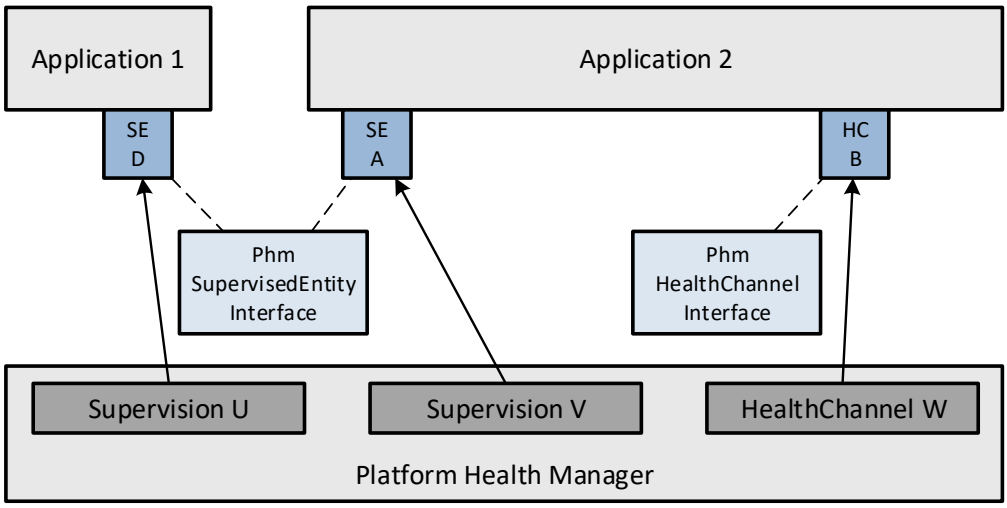


Figure 10.8: Interaction of application software with the platform health manager

10.3.2 Relation between design and deployment

The application design in chapters 3.3.7.2 and 3.3.7.3 uses the declarations provided in the Interface definitions for PhmSupervisedEntityInterface and PhmHealthChannelInterface. Specifically the handling of ids (checkpointId and statusId) requires a synchronized usage with the respective Interface definition.

It is required to establish a contract between the application code and the Phm deployment. The application code shall only use such id values which are declared at the respective Interface definitions (see [TPS_MANI_03624]).

During the configuration of the PHM the PHM artifacts are created. At deployment state there is no access to the design model available, thus the numeric values used for checkpointId and statusId at the Interface definition (PhmSupervisedEntityInterface and PhmHealthChannelInterface) are not available to the Phm.

Therefore the numeric values for checkpointId and statusId are replicated in the deployment model of the PHM:

- SupervisionCheckpoint.checkpointId replicates PhmCheckpoint.checkpointId

- HealthChannelExternalReportedStatus.statusId replicates PhmHealthChannelStatus.statusId

It is a methodological task to make sure that the Ids correspond to each other and match in value. This consistency can be checked using specific tooling on the deployment model and the design model (see also [TPS_MANI_03624]).

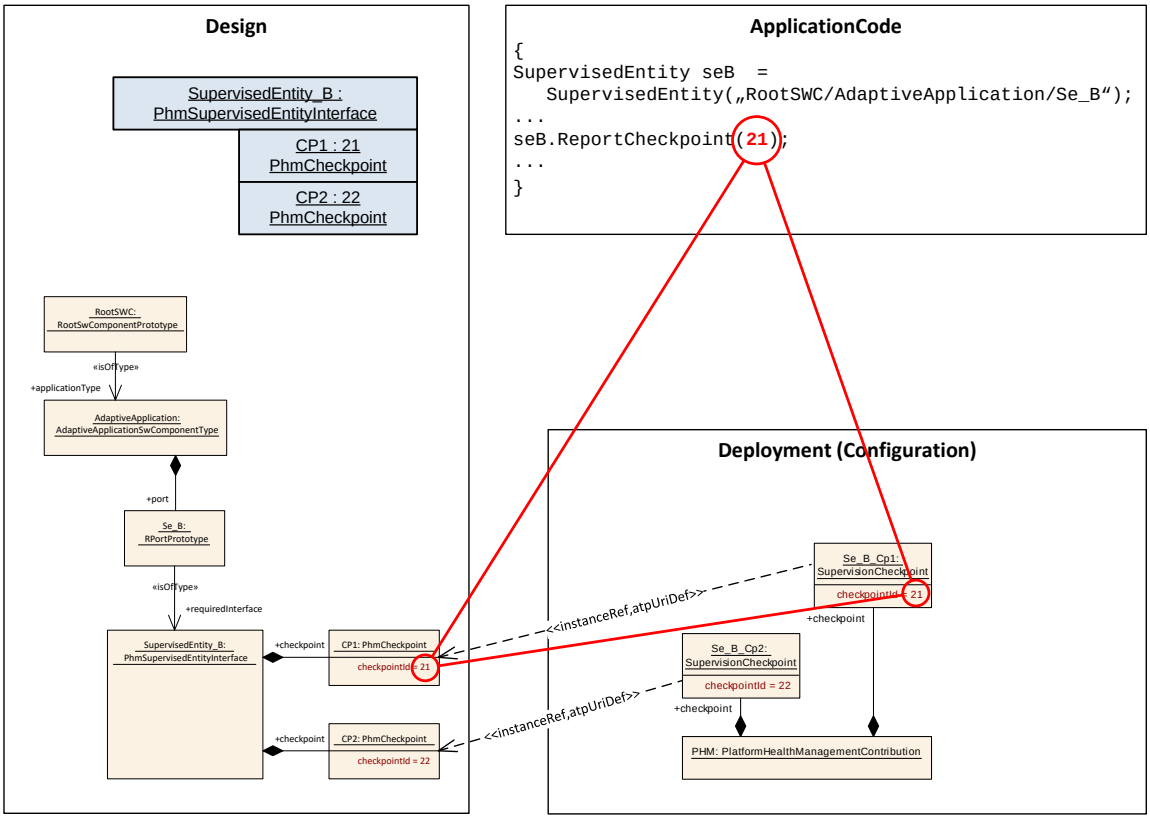


Figure 10.9: Example how IDs have to be in sync

The example in Figure 10.9 illustrates that the used IDs have to in sync in order to let design, code, and configuration work together.

10.3.3 Supervision deployment

In the application design chapter of this document the declaration of supervised entities and checkpoints has been described (see section 3.3.7.2). These declarations provide the view on supervision from the application software code point. Since the application Executable can be started multiple times (via individual Processes), the configuration of the Platform Health Management needs to cope with these individual Executable instances.

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

This means that only if a `PhmCheckpoint` is reported from the context of this `Process` it is considered to be this `SupervisionCheckpoint`.

Note: If the `SupervisionCheckpoint` represents the supervision of the *Execution Management* process, it will not be possible to configure `SupervisionCheckpoint.process` (unless there is a dummy configuration of *Execution Management* process) as the process for Execution Management is not a modeled `Process`.

It is up to the Adaptive Platform Stack Vendor to define the mechanism how to identify the `SupervisionCheckpoint` (thereby the supervisions `AliveSupervision`, `DeadlineSupervision`, and `LogicalSupervision`) corresponding to the *Execution Management*.

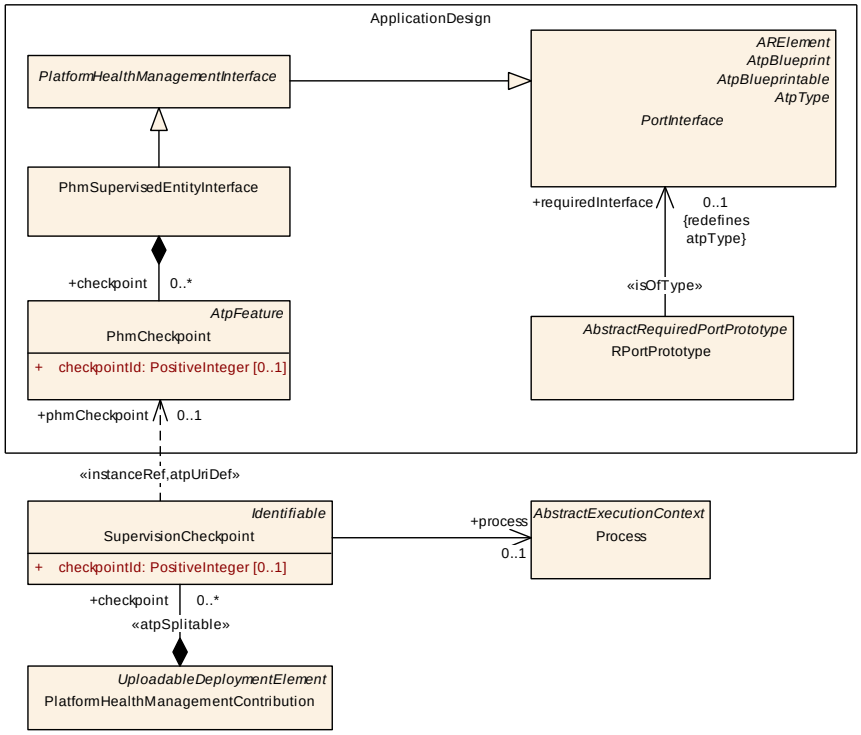


Figure 10.10: Modeling of `SupervisionCheckpoint`

For the Platform Health Management supervision to take effect it is required to define the instance the application is executed in, thus the reference to a `Process` has to be taken into account. In the model the `Process` also defines under which conditions (`StateDependentStartupConfig`) and with which arguments (`ProcessArgument`) the `Executable` will be started.

To also define the conditions for the supervision the state dependency is defined at the supervisions as well, see chapter 10.3.3.2.

For the configuration of the Platform Health Management the definition of SupervisionCheckpoint is used to stand in for the corresponding PhmCheckpoint including the execution context of the respective Process.

The attribute SupervisionCheckpoint.phmCheckpoint replicates the value of the referenced PhmCheckpoint.checkpointId. During deployment the PhmSupervisedEntityInterface and its content is no longer available and therefore needs to be made available to the Phm.

[TPS_MANI_03626]{DRAFT} **Consistency of SupervisionCheckpoint.phmCheckpoint and PhmCheckpoint.checkpointId** ⌈The value of SupervisionCheckpoint.phmCheckpoint shall be identical to the value of PhmCheckpoint.checkpointId which is referenced in SupervisionCheckpoint.phmCheckpoint.⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03505]{DRAFT} **Existence of SupervisionCheckpoint** ⌈For each PhmCheckpoint in the scope of a Process and RPortPrototype typed by a PhmSupervisedEntityInterface in the application definition, a SupervisionCheckpoint shall be defined. The correspondence of the two is defined by the instance reference SupervisionCheckpoint.phmCheckpoint.⌋*(RS_MANI_00023, RS_MANI_00032)*

Since every possibly reported PhmCheckpoint is represented by a corresponding SupervisionCheckpoint, it is possible to explicitly exclude a set of SupervisionCheckpoints from the supervision activities using the NoCheckpointSupervision definition (see section 10.3.3.9).

Class	**SupervisionCheckpoint**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element contains an instance reference to a RPortPrototype representing a checkpoint for Platform Health Management.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	PlatformHealthManagementContribution.checkpoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checkpointId	PositiveInteger	0..1	attr	Defines the numeric value which is used to identify the reporting of this SupervisionCheckpoint to the Phm.
phmCheckpoint	PhmCheckpoint	0..1	iref	Instance reference to the PhmCheckpoint defined in the context of a PortInterface. **Stereotypes:** atpUriDef **InstanceRef implemented by:** PhmCheckpointIn ExecutableInstanceRef
process	Process	0..1	ref	Reference to the Process this checkoint shall be monitored.

Table 10.19: SupervisionCheckpoint

[TPS_MANI_03651]{DRAFT} **Mandatory SupervisionCheckpoint.process reference** ⌈Unless the SupervisionCheckpoint refers to a PhmCheckpoint which is defined in the scope of the *Execution Management*, each SupervisionCheckpoint shall define a process reference.⌋*(RS_MANI_00023, RS_MANI_00032)*

[constr_3649]{DRAFT} **Consistent `SupervisionCheckpoint.process` reference** ⌈If a `SupervisionCheckpoint` refers to a `Process` in the role `process`, then

- the `SupervisionCheckpoint` shall refer to a `PhmCheckpoint` which is defined in a `PhmSupervisedEntityInterface` and
- that `PhmSupervisedEntityInterface` types an `RPortPrototype` of an `AdaptiveApplicationSwComponentType` and
- that `AdaptiveApplicationSwComponentType` is used in the scope of an `Executable` and
- that `Executable` is referenced by the same `Process` as `SupervisionCheckpoint.process`.

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

[constr_1742]{DRAFT} **Multiplicity of reference `SupervisionCheckpoint.phmCheckpoint`** ⌈At the time of deployment of a `SupervisionCheckpoint`, one reference to meta-class `PhmCheckpoint` in the role `phmCheckpoint` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_1764]{DRAFT} **Counterpart of `PhmCheckpoint`** ⌈Each `PhmCheckpoint` shall be referenced once and only once in the role `targetPhmCheckpoint` by a `PhmCheckpointInExecutableInstanceRef` with the same `Executable` and chain of `contextComponentPrototype` and `contextRPortPrototype` that is aggregated by a `SupervisionCheckpoint` in combination with a specific `Process`. This reference shall exist **at the time when the creation of the manifest is finished**.⌋*()*

This means that each `SupervisionCheckpoint` is the unique representation of the corresponding `PhmCheckpoint` instance in the context of a `PhmCheckpointInExecutableInstanceRef` and one `Process`.

Please note that the detailed modeling of the ≪`InstanceRef`≫ from `SupervisionCheckpoint` to `PhmCheckpoint` is documented in section D.6.

10.3.3.1 Global supervision

The `GlobalSupervision` collects `PhmSupervision`s (`AliveSupervision`s, `LogicalSupervision`s, `DeadlineSupervision`s, `NoCheckpointSupervision`s, and `NoSupervision`s) and aggregates the individual states of these Alive, Deadline, and Logical Supervisions into one `GlobalSupervision` status (see also figure 10.11).

[constr_3623]{DRAFT} **SupervisionCheckpoints in the context of a GlobalSupervision** ⌈All SupervisionCheckpoints belonging to the same PhmSupervisedEntityInterface instance (SupervisionCheckpoints with identical PhmCheckpointInExecutableInstanceRef.contextRootSwComponentPrototype, contextComponentPrototype, contextRPortPrototype, and process references) shall only be referenced by PhmSupervisions which are aggregated by the same GlobalSupervision **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3624]{DRAFT} **At least one Supervision defined in the context of a GlobalSupervision** ⌈At least one AliveSupervision, LogicalSupervision, or DeadlineSupervision shall be defined in the scope of a GlobalSupervision **at the time when the creation of the manifest is finished**.⌋*()*

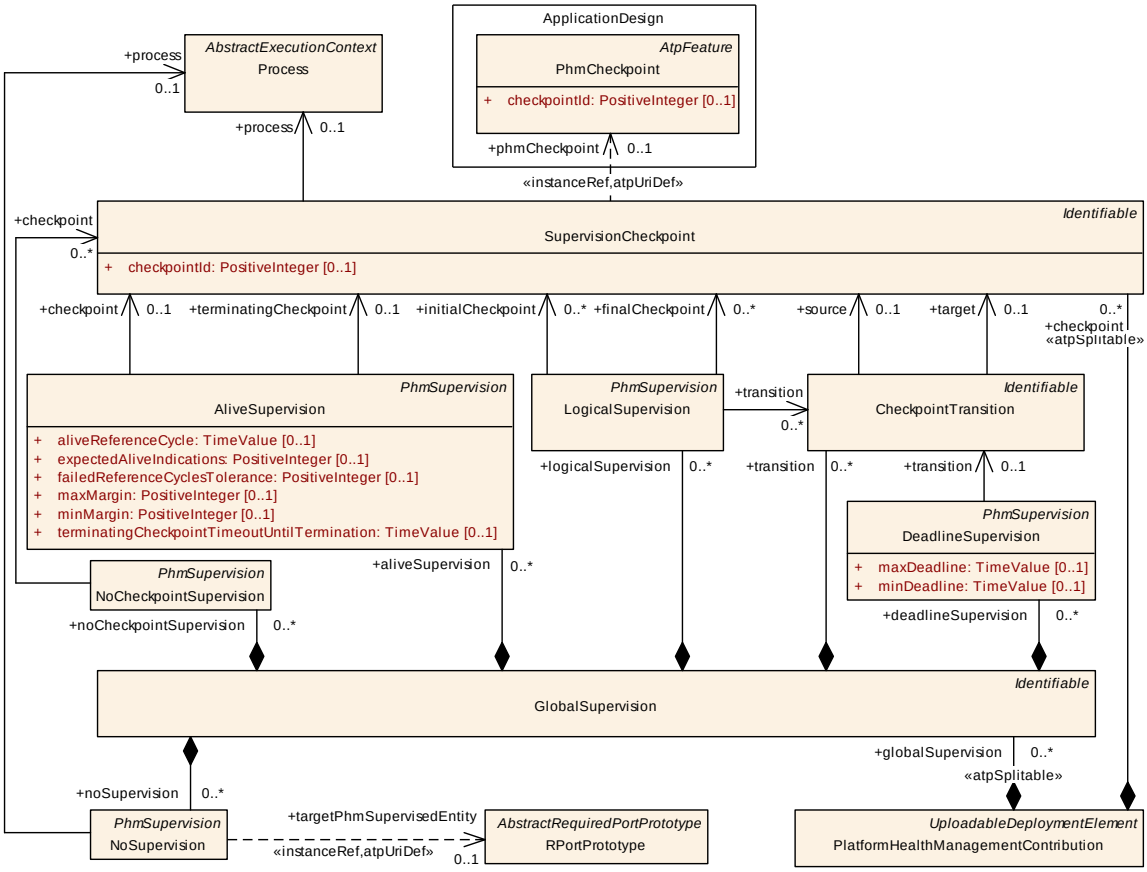


Figure 10.11: Modeling of GlobalSupervision

[TPS_MANI_03512]{DRAFT} **Applicability of global supervision without Process context** ⌈The aggregated AliveSupervisions, LogicalSupervisions, and DeadlineSupervisions contributing to a specific GlobalSupervision may refer to SupervisionCheckpoints where each SupervisionCheckpoint may refer to a different Process.⌋*(RS_MANI_00023, RS_MANI_00032)*

With [TPS_MANI_03512] the GlobalSupervision and the aggregation AliveSupervisions, LogicalSupervisions, and DeadlineSupervisions can be used to establish three use-cases:

- compose the status of one Executable instance (Process) in case all aggregated PhmSupervisions are defined with the same Process context.
- compose the status of several Executable instances (Processes) of a FunctionGroup in case the aggregated PhmSupervisions are defined with (partially) different Process contexts within a FunctionGroup context.
- compose the status of all Executable instances (Processes) of a FunctionGroup in case the aggregated PhmSupervisions are defined with (all) different Process contexts within a FunctionGroup context.

[constr_3625]{DRAFT} **DeadlineSupervision referencing CheckpointTransition in the context of a GlobalSupervision** ⌈DeadlineSupervision aggregated in a GlobalSupervision shall only refer to a CheckpointTransition which is aggregated by the same GlobalSupervision **at the time when the creation of the manifest is finished**.⌋()

[constr_3626]{DRAFT} **LogicalSupervision referencing CheckpointTransition in the context of a GlobalSupervision** ⌈LogicalSupervision aggregated in a GlobalSupervision shall only refer to CheckpointTransitions that are aggregated by the same GlobalSupervision **at the time when the creation of the manifest is finished**.⌋()

Class	**GlobalSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a collection of AliveSupervisions, DeadlineSupervisions, and LogicalSupervisions in order to provide an aggregated supervision state.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	PlatformHealthManagementContribution.globalSupervision			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
alive Supervision	AliveSupervision	*	aggr	Collection of AliveSupervisions in the context of this GlobalSupervision.
deadline Supervision	DeadlineSupervision	*	aggr	Collection of DeadlineSupervisions in the context of this GlobalSupervision.
logical Supervision	LogicalSupervision	*	aggr	Collection of LogicalSupervisions in the context of this GlobalSupervision.
noCheckpoint Supervision	NoCheckpoint Supervision	*	aggr	Definition of No Checkpoint Supervision.
noSupervision	NoSupervision	*	aggr	Collection of NoSupervisions in the context of this Global Supervision.
supervision Mode	SupervisionMode	*	aggr	Collection of SupervisionModes in the context of this GlobalSupervision. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=supervisionMode.shortName
transition	CheckpointTransition	*	aggr	Collection of CheckpointTransitions in the context of this GlobalSupervision.

Table 10.20: GlobalSupervision

10.3.3.2 Supervision Mode

Expected execution (timing or sequence) of the Software can change based on certain conditions. Hence, the value of the Supervision attributes might have to be changed based on conditions as well. `SupervisionMode` defines the Supervision attributes which change based on the referenced `modeCondition`.

The Execution Management may use `Function Group`s and `Function Group State`s to define the startup conditions of application software. The behavior of the application software can change based on the `Function Group State`. Hence the monitoring shall also be configurable based on `Function Group State` changes.

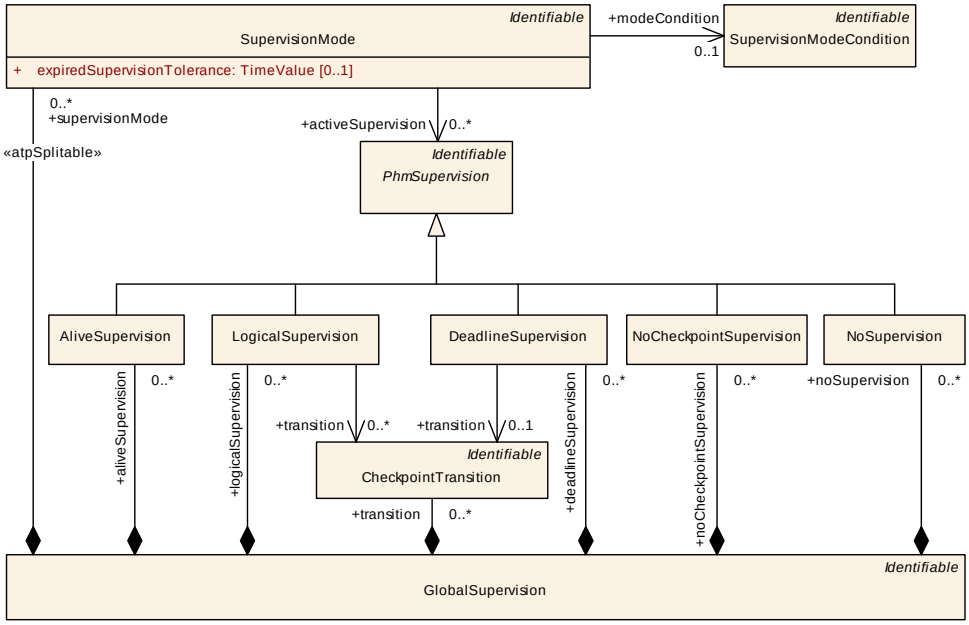


Figure 10.12: Mode dependent supervision

A `SupervisionMode` does define a set of `PhmSupervision`s in the role `activeSupervision` which shall be active while this `SupervisionMode` is active. This modeling allows for one specific `PhmSupervision` to be defined as active in several `SupervisionMode`s.

[TPS_MANI_03513]{DRAFT} **Collection of supervisions into a `GlobalSupervision`** ⌈The `PhmSupervision`s referenced from a `SupervisionMode` in the role `activeSupervision` are taken into account to evaluate the supervision status of the `GlobalSupervision` in that particular `SupervisionMode`.⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03633]{DRAFT} **Semantics of several supervisionMode references for one supervision** ⌈If a `PhmSupervision` is referenced in the role `activeSupervision` then this `PhmSupervision` will be active while the referencing `SupervisionMode` is active.⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**SupervisionMode**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a SupervisionMode.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	GlobalSupervision.supervisionMode			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
active Supervision	PhmSupervision	*	ref	The reference defines which PhmSupervisions shall be active in this specific SupervisionMode.
expired Supervision Tolerance	TimeValue	0..1	attr	Defines in this SupervisionMode the acceptable amount of time with EXPIRED supervision status of the enclosing GlobalSupervision before it is considered STOPPED.
modeCondition	SupervisionMode Condition	0..1	ref	Reference to SupervisionModeCondition (Condition under which the configuration made under this SupervisionMode are to be applied).

Table 10.21: SupervisionMode

[TPS_MANI_03514]{DRAFT} **Expiration tolerance for `SupervisionMode`** ⌈The attribute `SupervisionMode.expiredSupervisionTolerance` defines the time which a `GlobalSupervision` is maintained in the state *expired* before it is considered *stopped* in that particular `SupervisionMode`.⌋*(RS_MANI_00023, RS_MANI_00032)*

[constr_3639]{DRAFT} **Existence of `SupervisionMode.expiredSupervisionTolerance`** ⌈If the `SupervisionMode` refers to a `PhmSupervision` that in turn references a `SupervisionCheckpoint` and that `SupervisionCheckpoint` refers to a `Process` where the `Executable` has the attribute `category` set to APPLICATION_LEVEL,
then the attribute `expiredSupervisionTolerance` shall NOT exist .

For each `SupervisionMode` the attribute `expiredSupervisionTolerance` shall exist.

This rule shall be imposed **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_3640]{DRAFT} **Existence of `SupervisionMode.modeCondition`** ⌈For each `SupervisionMode` the attribute `modeCondition` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

10.3.3.3 Supervision Mode condition

Each `SupervisionMode` refers to exactly one `SupervisionModeCondition` in the role `modeCondition`. The `SupervisionModeCondition` defines an collection of one or several `stateReference`s which are combined to provide a single logical state.

The reference to `Function Group`s and their `Function Group State`s is essential for the definition of `SupervisionModeCondition`s as the supervised `Process`es are started and stopped according to changes in `Function Group State`s.

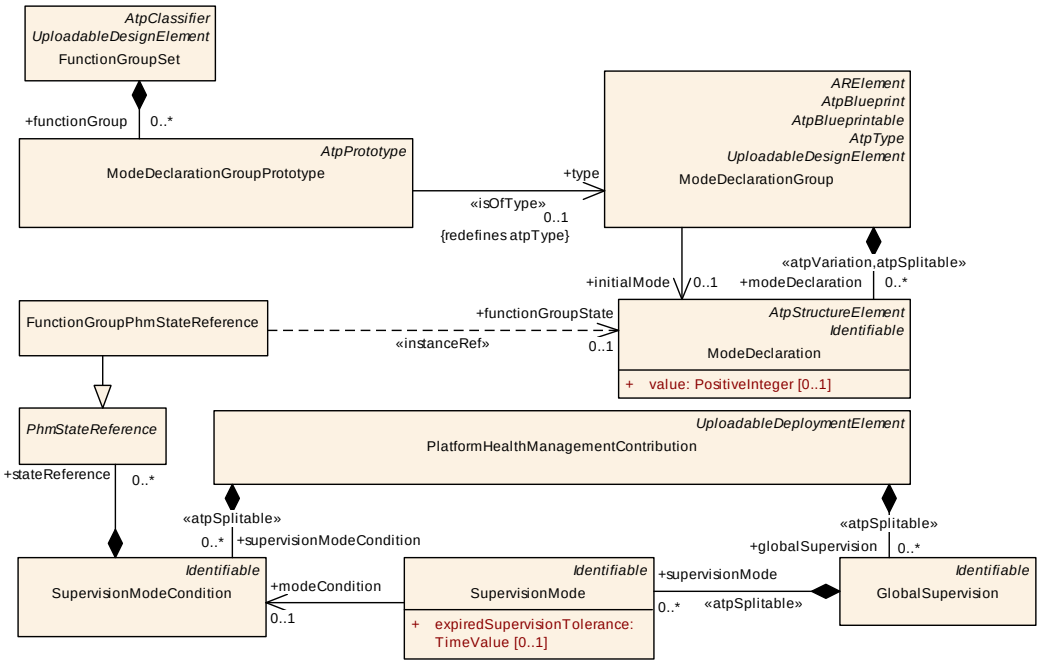


Figure 10.13: Modeling of `SupervisionMode`

Class	**SupervisionModeCondition**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a SupervisionModeCondition in the context of platform health management contribution.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	PlatformHealthManagementContribution.supervisionModeCondition			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
stateReference	PhmStateReference	*	aggr	Collection of stateReferences.

Table 10.22: SupervisionModeCondition

[constr_3627]{DRAFT} **Existence of `SupervisionModeCondition.stateReference`** ⌈At the time of deployment of `SupervisionModeCondition`, at least one aggregation of `PhmStateReference` in the role `SupervisionModeCondition.stateReference` shall exist **`at the time when the creation of the manifest is finished.`**⌋ *()*

Class	**FunctionGroupPhmStateReference**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Function Group state dependency.			
Base	*ARObject*, *PhmStateReference*			
Aggregated by	SupervisionModeCondition.stateReference			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	FunctionGroupPhmStateReference			
functionGroup State	ModeDeclaration	0..1	iref	This represent the applicable functionGroupState. **InstanceRef implemented by:** FunctionGroupStateIn FunctionGroupSetInstanceRef

Table 10.23: FunctionGroupPhmStateReference

[constr_3628]{DRAFT} **Reference to `Function Group State` from a `SupervisionModeCondition`** ⌈If a `Function Group State` is referenced by a `SupervisionModeCondition` in the scope of one `GlobalSupervision`, then that same `Function Group State` shall NOT be referenced by any other `SupervisionModeCondition` in the scope of the same `GlobalSupervision` **at the time when the creation of the manifest is finished**.⌋*()*

Rationale for [constr_3628]: It is guaranteed that only one of the `SupervisionMode`s is active at any point in time. The `SupervisionModeCondition`s referenced under a `GlobalSupervision` in the context of `SupervisionMode` shall be mutually exclusive.

E.g. Following `SupervisionModeCondition`s referenced under the same `GlobalSupervision` in the context of `SupervisionMode` is not allowed:

- SupervisionModeCondition_1:
  (FunctionGroup_X in State 'a') or (FunctionGroup_X in State 'b')
- SupervisionModeCondition_2:
  (FunctionGroup_X in State 'a') or (FunctionGroup_X in State 'c')

In the above example, (FunctionGroup_X in State 'a') is common between the 2 `SupervisionModeCondition`s.

[constr_3629]{DRAFT} **Identical `Function Group` in the scope of a `GlobalSupervision`** ⌈Within the context of one `GlobalSupervision`, all `SupervisionMode.modeCondition` shall only aggregate `FunctionGroupPhmStateReference`s in the role `stateReference` where the reference `FunctionGroupPhmStateReference.functionGroupState.contextModeDeclarationGroupPrototype` refers to the identical `ModeDeclarationGroupPrototype` (that implements the `Function Group`, as far as state management is concerned) **at the time when the creation of the manifest is finished**.⌋*()*

That is, within the context of one `GlobalSupervision`, all `SupervisionMode.modeCondition.stateReference` shall reference states(modes) of the same `Function Group`.

[constr_3630]{DRAFT} **`GlobalSupervision` and `Process` relation** ⌈Within the context of one `GlobalSupervision`, all aggregated `PhmSupervision`s shall refer to `SupervisionCheckpoint`s where the referenced `Process` aggregates a `stateDependentStartupConfig` that in turn refers to a `functionGroupState` where the `contextModeDeclarationGroupPrototype` refers to the identical `ModeDeclarationGroupPrototype` (that implements the `Function Group`, as far as state

management is concerned) **at the time when the creation of the manifest is finished**.⌋()

That is, all PhmSupervisions aggregated in a GlobalSupervision are referencing SupervisionCheckpoints corresponding to Processes that are configured to be executed in state/s of the same Function Group.

[constr_3631]{DRAFT} **Global supervision restricted to one Function Group** ⌈The Function Group (ModeDeclarationGroupPrototype) referenced in [constr_3629] and [constr_3630] shall be identical for one particular GlobalSupervision **at the time when the creation of the manifest is finished**.⌋()

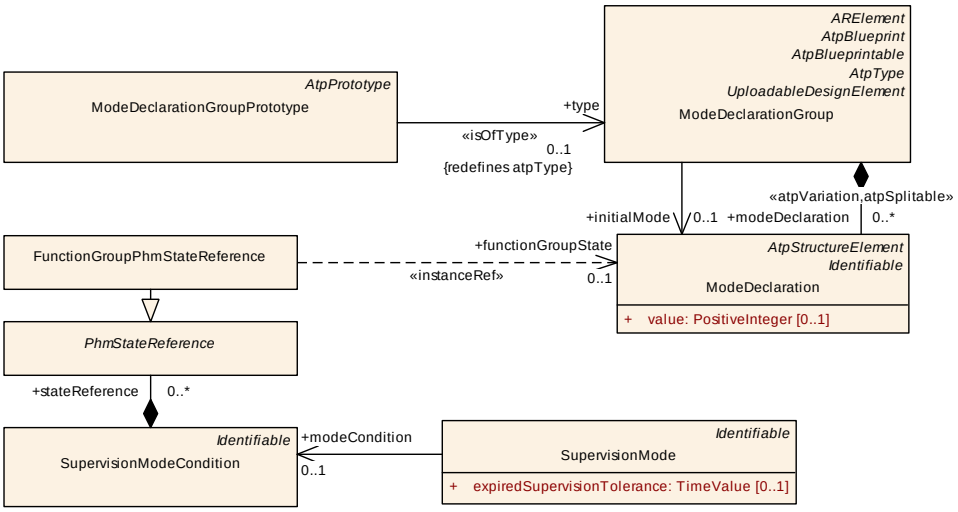


Figure 10.14: Modeling of FunctionGroupPhmStateReference

[constr_3632]{DRAFT} **Supervision of a Supervised Entity Instance in the scope of a Function Group State** ⌈A Supervised Entity Instance shall be configured with checkpoint supervision (all SupervisionCheckpoints of the Supervised Entity Instance are covered by AliveSupervision, DeadlineSupervision, LogicalSupervision, NoCheckpointSupervision) or NoSupervision in all Function Group States in which the corresponding Process is configured to be executed **at the time when the creation of the manifest is finished**.⌋()

Rationale for [constr_3632]: The configuration of supervision for a specific Supervised Entity Instance is not missed unintentionally in a Function Group State.

Note that [constr_3632] requires for the Supervised Entity Instance to be part of the supervision definition in all Function Group States in which the corresponding Process is configured to be executed. In case there is no supervision required in a specific Function Group State (because the Supervised Entity Instance does not need to be supervised in that specific Function Group State), then NoSupervision can be configured for that Supervised Entity Instance.

Note: If a Process survives a Function Group State change running (because the Process is defined to be running in two consecutive Function Group States),

then it is important that the configuration of the activeSupervision for the Supervised Entity Instances corresponding to the Process is identical for the SupervisionModes that corresponds to the affected Function Group States. This relations can not be formally expressed (because the possible Function Group State transitions are not formalized) and therefore this restriction can not be defined as a constraint.

A possible mitigation strategy is to define different Process arguments and therefore force a re-start of the Process, even if the Function Group States have been switched consecutively.

10.3.3.4 NoSupervision definition

In the scope of a GlobalSupervision, a NoSupervision can be defined for a Supervised Entity Instance as a whole. A NoSupervision can be configured to indicate that this Supervised Entity Instance shall not be monitored by the Phm in a specific SupervisionMode.

The Supervised Entity Instance is identified using the NoSupervision.targetPhmSupervisedEntity instance ref and NoSupervision.process reference.

Class	**NoSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines explicitly that NO supervision shall be applied for a specific Supervised Entity instance.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmSupervision*, *Referrable*			
Aggregated by	GlobalSupervision.noSupervision			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
process	Process	0..1	ref	Reference to the Process this NoSupervision applies to.
targetPhm SupervisedEntity	RPortPrototype	0..1	iref	Instance reference to the RPortPrototype which represents the Supervised Entity instance. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 10.24: NoSupervision

[constr_3674]{DRAFT} **Existence of NoSupervision.targetPhmSupervisedEntity** ⌈For each NoSupervision the attribute targetPhmSupervisedEntity shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3675]{DRAFT} **Existence of NoSupervision.process** ⌈For each NoSupervision the attribute process shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3676]{DRAFT} **Exclusive usage of NoSupervision** ⌈For a Supervised Entity Instance, in a given SupervisionMode, either a checkpoint supervision (AliveSupervision, DeadlineSupervision, LogicalSupervision, and/or

NoCheckpointSupervision referencing all SupervisionCheckpoints corresponding to the Supervised Entity Instance) or NoSupervision shall be configured, but not both.

This rule shall be imposed **at the time when the creation of the manifest is finished**⌋*()*

Note: The NoSupervision approach disables the supervision for all the SupervisionCheckpoints of a Supervised Entity Instance.

10.3.3.5 AliveSupervision definition

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

[TPS_MANI_03508]{DRAFT} **Definition of an AliveSupervision for a SupervisionCheckpoint** ⌈An AliveSupervision definition provides attributes to configure the supervision of the referenced SupervisionCheckpoint.

- aliveReferenceCycle defines the time base used to monitor the reporting of this specific SupervisionCheckpoint
- expectedAliveIndications defines the number of indications which shall be observed during the time period defined by aliveReferenceCycle
- minMargin and maxMargin define the acceptable deviation from the expectedAliveIndications within the time period defined by aliveReferenceCycle

⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03575]{DRAFT} **Definition of no minimum alive supervision** ⌈If the value AliveSupervision.minMargin equals 0, this defines that no minimum alive supervision shall be performed.⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03576]{DRAFT} **Definition of no maximum alive supervision** ⌈If the value AliveSupervision.maxMargin equals INF, this defines that no maximum alive supervision shall be performed.⌋*(RS_MANI_00023, RS_MANI_00032)*

[constr_3539]{DRAFT} **Only one AliveSupervision per SupervisionCheckpoint** ⌈A SupervisionCheckpoint shall only be referenced up to once by an AliveSupervision in the role checkpoint in the context of an identical SupervisionMode **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_03515]{DRAFT} **Failure tolerance for `AliveSupervision`** ⌈The attribute `AliveSupervision.failedReferenceCyclesTolerance` defines the acceptable amount of `AliveSupervision.aliveReferenceCycle`s with incorrect/-failed `AliveSupervision`.⌋ *(RS_MANI_00023, RS_MANI_00032)*

Class	**AliveSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines an AliveSupervision for one checkpoint.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmSupervision*, *Referrable*			
Aggregated by	GlobalSupervision.aliveSupervision			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
aliveReference Cycle	TimeValue	0..1	attr	Time period at which the Alive Supervision mechanism compares the amount of received Alive Indications for the SupervisionCheckpoint against the expectedAlive Indications.
checkpoint	SupervisionCheckpoint	0..1	ref	Reference to a checkpoint in the context of Alive Supervision.
expectedAlive Indications	PositiveInteger	0..1	attr	Defines the amount of expected Alive Indications of the SupervisionCheckpoint within the aliveReferenceCycle.
failedReference Cycles Tolerance	PositiveInteger	0..1	attr	This attribute defines the acceptable amount of alive ReferenceCycles with incorrect/failed AliveSupervision.
maxMargin	PositiveInteger	0..1	attr	Defines the amount of Alive Indications of the Supervision Checkpoint that are acceptable to be additional to the expectedAliveIndications within the aliveReferenceCycle.
minMargin	PositiveInteger	0..1	attr	Defines the amount of Alive Indications of the Supervision Checkpoint that are acceptable to be missing to the expectedAliveIndications within the aliveReferenceCycle.
terminating Checkpoint	SupervisionCheckpoint	0..1	ref	Reference to the SupervisionCheckpoint which is defined as the terminating checkpoint of this AliveSupervision.
terminating Checkpoint TimeoutUntil Termination	TimeValue	0..1	attr	Defines the time a process shall terminate after it has announced its start of termination by reporting terminatingCheckpoint.

Table 10.25: AliveSupervision

[constr_3633]{DRAFT} **Mandatory attributes of `AliveSupervision`** ⌈The following attributes of `AliveSupervision` shall be defined **at the time when the integration into a `SoftwareCluster` is finished**:

- `aliveReferenceCycle`
- `checkpoint`
- `expectedAliveIndications`
- `failedReferenceCyclesTolerance`
- `minMargin`
- `maxMargin`

This rule shall be imposed **`at the time when the creation of the manifest is finished`**⌋ *()*

10.3.3.5.1 AliveSupervision of self-terminating processes

If a Process is defined to be self-terminating (according to [TPS_MANI_01334]) and that Process has an AliveSupervision defined, then it is essential for that Process to indicate to the Phm when that Process is about to self terminate. The indication is done using a dedicated SupervisionCheckpoint and is associated with the AliveSupervision. In this case the Phm can stop the AliveSupervision in a defined way when the application reports that AliveSupervision.terminatingCheckpoint has been reached.

[constr_3709]{DRAFT} **AliveSupervision.terminatingCheckpoint required for self terminating Processes** ⌈Only if a Process

- refers to a StartupConfig (via stateDependentStartupConfig), and that StartupConfig has the attribute StartupConfig.terminationBehavior set to the value TerminationBehaviorEnum.processIsSelfTerminating, and
- the StateDependentStartupConfig.functionGroupState is identical to the respective GlobalSupervision.supervisionMode.modeCondition.stateReference.functionGroupState and
- there exists an AliveSupervision which refers to a SupervisionCheckpoint in the role checkpoint, and that SupervisionCheckpoint refers to that Process, then

there shall exist an AliveSupervision.terminatingCheckpoint reference from the AliveSupervision **at the time when the creation of the manifest is finished**.⌋*()*

That is, if an AliveSupervision is defined for a Process in a SupervisionMode that corresponds to a Function Group State in which the Process is configured to be self-terminating, then and only then AliveSupervision.terminatingCheckpoint shall be defined.

[constr_3710]{DRAFT} **Process referenced by AliveSupervision.terminatingCheckpoint** ⌈The SupervisionCheckpoint that is referenced in the role AliveSupervision.terminatingCheckpoint shall refer to the same Process as the SupervisionCheckpoint that is referenced by the AliveSupervision.checkpoint **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3711]{DRAFT} **AliveSupervision.terminatingCheckpointTimeoutUntilTermination** ⌈If an AliveSupervision has the reference AliveSupervision.terminatingCheckpoint defined, then the attribute AliveSupervision.terminatingCheckpointTimeoutUntilTermination shall be defined **at the time when the creation of the manifest is finished**.⌋*()*

10.3.3.6 `CheckpointTransition` definition

For the definition of further supervision strategies the need to first define possible `CheckpointTransition`s between `SupervisionCheckpoint`s arises. Since the application software design does not provide any transition definition between checkpoints, it is essential to define possible `CheckpointTransition`s.

The definition of `CheckpointTransition`s is done in the scope of the `GlobalSupervision` and can be used by the `LogicalSupervision` and `DeadlineSupervision` of the same `GlobalSupervision`.

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

[constr_3634]{DRAFT} **Multiplicity of `CheckpointTransition.source` and `CheckpointTransition.target`** ⌈Each `CheckpointTransition` shall define exactly one `source` reference and one `target` reference **at the time when the creation of the manifest is finished**.⌋*()*

Class	CheckpointTransition			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines one transition between two checkpoints.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	GlobalSupervision.transition			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
source	SupervisionCheckpoint	0..1	ref	Reference to the source checkpoint for this transition.
target	SupervisionCheckpoint	0..1	ref	Reference to the target checkpoint for this transition.

Table 10.26: CheckpointTransition

10.3.3.7 `LogicalSupervision` definition

The `LogicalSupervision` defines a supervision graph of allowed `CheckpointTransition`s which is monitored by the Platform Health Management without any timing considerations, just the order of reported checkpoints is considered for the monitoring.

[constr_3540]{DRAFT} **`SupervisionCheckpoint` in supervision graph** ⌈Each `SupervisionCheckpoint` shall only be part of one supervision graph in the context of an identical `SupervisionMode` **at the time when the creation of the manifest is finished**.⌋*()*

When a `SupervisionCheckpoint` belonging to the supervision graph is reported to the Platform Health Management where there is no `CheckpointTransition` defined from the last reported `SupervisionCheckpoint` as `source` to the current reported

SupervisionCheckpoint as target, this situation violates the LogicalSupervision.

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

Class	**LogicalSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines a LogicalSupervision graph consisting of transitions, initial- and final checkpoints.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmSupervision*, *Referrable*			
Aggregated by	GlobalSupervision.logicalSupervision			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
finalCheckpoint	SupervisionCheckpoint	*	ref	Reference to the final Checkpoint(s) for this Logical Supervision. **Tags:** xml.sequenceOffset=20
initialCheckpoint	SupervisionCheckpoint	*	ref	Reference to the initial Checkpoint(s) for this Logical Supervision. **Tags:** xml.sequenceOffset=10
transition	CheckpointTransition	*	ref	Reference to the transitions for this LogicalSupervision. **Tags:** xml.sequenceOffset=30

Table 10.27: LogicalSupervision

[constr_1736]{DRAFT} **Multiplicity of reference LogicalSupervision.initialCheckpoint** ⌈At the time of deployment of a LogicalSupervision, at least one reference to meta-class SupervisionCheckpoint in the role initialCheckpoint shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_1737]{DRAFT} **Multiplicity of reference LogicalSupervision.finalCheckpoint** ⌈At the time of deployment of a LogicalSupervision, at least one reference to meta-class SupervisionCheckpoint in the role finalCheckpoint shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_1740]{DRAFT} **Multiplicity of reference LogicalSupervision.transition** ⌈At the time of deployment of a LogicalSupervision, at least one reference to meta-class CheckpointTransition in the role LogicalSupervision.transition shall exist **at the time when the creation of the manifest is finished**.⌋*()*

10.3.3.8 DeadlineSupervision definition

The DeadlineSupervision defines timing attributes for one specific CheckpointTransition.

[TPS_MANI_03511]{DRAFT} **Definition of DeadlineSupervision** ⌈A DeadlineSupervision defines timing attributes which are monitored by the Platform Health Management for one specific CheckpointTransition.⌋*(RS_MANI_00023, RS_MANI_00032)*

[constr_3635]{DRAFT} **Mandatory attributes of DeadlineSupervision** ⌈The following attributes of DeadlineSupervision shall be defined:

- transition
- minDeadline
- maxDeadline

This rule shall be imposed **at the time when the creation of the manifest is finished**⌋*()*

[TPS_MANI_03573]{DRAFT} **Definition of no minimum deadline supervision** ⌈If the value DeadlineSupervision.minDeadline equals 0, this defines that no minimum deadline supervision shall be performed.⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03574]{DRAFT} **Definition of no maximum deadline supervision** ⌈If the value DeadlineSupervision.maxDeadline equals INF, this defines that no maximum deadline supervision shall be performed.⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	**DeadlineSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines an DeadlineSupervision for one transition.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmSupervision*, *Referrable*			
Aggregated by	GlobalSupervision.deadlineSupervision			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
maxDeadline	TimeValue	0..1	attr	Defines the longest time span before which the deadline is considered to be met for transition.
minDeadline	TimeValue	0..1	attr	Defines the shortest time span after which the deadline is considered to be met for transition.
transition	CheckpointTransition	0..1	ref	Reference to the transition in the context of a Deadline Supervision.

Table 10.28: DeadlineSupervision

10.3.3.9 NoCheckpointSupervision definition

The NoCheckpointSupervision is used to define a set of SupervisionCheckpoints for which no Alive, Deadline and Logical supervision shall be performed. It is a

means to define a complete coverage of the configuration with respect to the possibly reported SupervisionCheckpoints. If a set of SupervisionCheckpoints shall not be considered by the Phm for Supervision (Alive, Deadline and Logical) evaluation, then these SupervisionCheckpoints can be referenced from the NoCheckpointSupervision in the role NoCheckpointSupervision.checkpoint.

[TPS_MANI_03670]{DRAFT} **Definition of NoCheckpointSupervision** ⌈A NoCheckpointSupervision defines explicitly that no supervision (Alive, Deadline and Logical) shall be applied to the SupervisionCheckpoints which are referenced in the role NoCheckpointSupervision.checkpoint.⌋*(RS_MANI_00023, RS_MANI_00032)*

[constr_3712]{DRAFT} **Exclusive usage of NoCheckpointSupervision** ⌈If a SupervisionCheckpoint is referenced by a NoCheckpointSupervision in the role checkpoint, then that SupervisionCheckpoint shall not be referenced by any other checkpoint supervision (AliveSupervision, DeadlineSupervision (via CheckpointTransition), LogicalSupervision (also via or CheckpointTransition)) in the scope of one SupervisionMode **at the time when the creation of the manifest is finished**.⌋*()*

Class	**NoCheckpointSupervision**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines explicitly that NO supervision shall be applied for a set of SupervisionCheckpoints.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PhmSupervision*, *Referrable*			
Aggregated by	GlobalSupervision.noCheckpointSupervision			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checkpoint	SupervisionCheckpoint	*	ref	Reference to the set of SupervisionCheckpoints which shall not be considered for any kind of supervision.

Table 10.29: NoCheckpointSupervision

10.3.4 Health channel deployment

The HealthChannel is used as an abstraction to the Platform Health Management input for the RecoveryNotification.

[constr_3730]{DRAFT} **Upper multiplicity of reference in the role HealthChannel.recoveryNotification** ⌈In the context of HealthChannel, the reference in the role recoveryNotification shall exist at most once **at the time when the creation of the manifest is finished**.⌋*()*

The specialized use-cases for HealthChannels are described in the following sections. A HealthChannel can either represent

- the status of a GlobalSupervision using the HealthChannelSupervision or
- the status of a reported PhmHealthChannelInterface using the HealthChannelExternalStatus

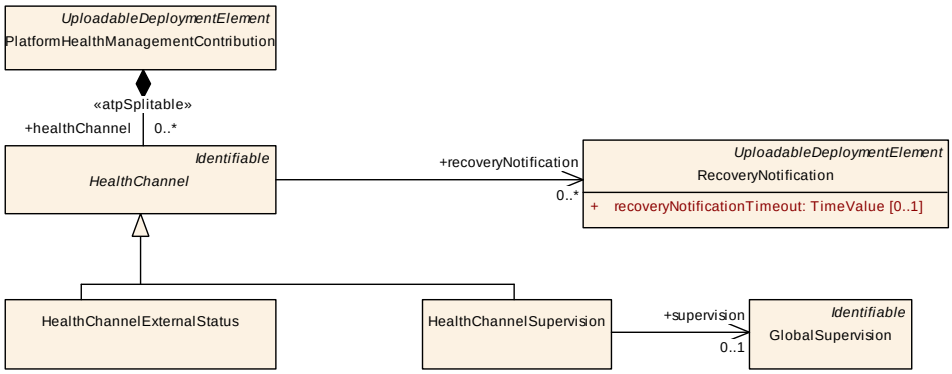


Figure 10.15: Modeling of `HealthChannel`

Class	***HealthChannel*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines the source of a health channel.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	HealthChannelExternalStatus, HealthChannelSupervision			
Aggregated by	PlatformHealthManagementContribution.healthChannel			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
recovery Notification	RecoveryNotification	*	ref	Defines the RecoveryNotification for this HealthChannel.

Table 10.30: HealthChannel

[TPS_MANI_03672]{DRAFT} **Definition of `RecoveryNotification` for an `HealthChannel`** ⌈The reference `HealthChannel.recoveryNotification` defines which `RecoveryNotification` shall be triggered if the `HealthChannel` is violated.⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03673]{DRAFT} **`RecoveryNotification` referenced by several `HealthChannel.recoveryNotification`s** ⌈One `RecoveryNotification` may be referenced by several `HealthChannel.recoveryNotification`s defining that the violation of any of the referencing `HealthChannel`s will trigger that `RecoveryNotification`.⌋*(RS_MANI_00023, RS_MANI_00032)*

[constr_3719]{DRAFT} **`RecoveryNotification` referenced either by `HealthChannelExternalStatus` or `HealthChannelSupervision`** ⌈A `RecoveryNotification` shall either be referenced from up to one `HealthChannelExternalStatus` element or from one or more `HealthChannelSupervision` elements **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_3719] exists because the notification signature differs depending on whether the notification is caused by a `HealthChannelExternalStatus` or `HealthChannelSupervision`. The restriction of only one `HealthChannelExternalStatus` reference exists because the notification uses the statusId of the `HealthChannelExternalStatus` as argument.

10.3.4.1 Supervision health channel deployment

The HealthChannelSupervision is used to take the status of a GlobalSupervision trigger the RecoveryNotification in case a violation is detected by the Phm.

[TPS_MANI_03516]{DRAFT} **Status for HealthChannelSupervision** ⌈The status of the GlobalSupervision which is referenced in the role supervision is taken as the trigger for the RecoveryNotification.⌋*(RS_MANI_00023, RS_MANI_00032)*

Class	HealthChannelSupervision			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a health channel representing the status of a PhmSupervision.			
Base	*ARObject*, *HealthChannel*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	PlatformHealthManagementContribution.healthChannel			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
supervision	GlobalSupervision	0..1	ref	Reference to the GlobalSupervision as source for the health channel.

Table 10.31: HealthChannelSupervision

10.3.4.2 External health channel deployment

The HealthChannelExternalStatus is used to define a list of HealthChannelExternalReportedStatus in the role healthChannel. If for this HealthChannelExternalStatus and the referenced Process a status is reported which is part of the list in healthChannel then this HealthChannelExternalStatus is considered violated and a notification to the respective RecoveryNotification is performed.

[TPS_MANI_03545]{DRAFT} **Existence of HealthChannelExternalStatus** ⌈For each RPortPrototype typed by a PhmHealthChannelInterface there may be a HealthChannelExternalStatus defined.⌋*(RS_MANI_00023, RS_MANI_00032)*

The attribute HealthChannelExternalReportedStatus.statusId replicates the value of the referenced PhmHealthChannelStatus.statusId. During deployment the PhmHealthChannelInterface and its content is no longer available and therefore needs to be made available to the Phm.

[TPS_MANI_03625]{DRAFT} **Consistency of HealthChannelExternalReportedStatus.statusId and PhmHealthChannelStatus.statusId** ⌈The value of HealthChannelExternalReportedStatus.statusId shall be identical to the value of PhmHealthChannelStatus.statusId which is referenced in HealthChannelExternalReportedStatus.status.⌋*(RS_MANI_00023, RS_MANI_00032)*

[TPS_MANI_03546]{DRAFT} **Definition of reported health status RPortPrototype** ⌈The RPortPrototype typed by a PhmHealthChannelInterface is used

to report the status of a health channel by the application software identified as InstanceSpecifier. This specific RPortPrototype is defined as the targetRPortPrototype of the instance reference HealthChannelExternalStatus.healthChannel.⌋(*RS_MANI_00023*, *RS_MANI_00032*)

[TPS_MANI_03517]{DRAFT} **Evaluation of HealthChannelExternalStatus** ⌈The reported value of the HealthChannelExternalStatus according to [TPS_MANI_03546] will be compared to the list of statuses provided in notifiedStatus. If the reported status value matches one of the listed statusIds then this HealthChannelExternalStatus is considered violated and the respective RecoveryNotification is issued.⌋(*RS_MANI_00023*, *RS_MANI_00032*)

[TPS_MANI_03553]{DRAFT} **Applicability of health channel to a specific Process** ⌈The reference HealthChannelExternalStatus.process defines to which specific Process this HealthChannelExternalStatus definition shall be applied to.⌋(*RS_MANI_00023*, *RS_MANI_00032*)

This means that only if a PhmHealthChannelStatus is reported from the context of this Process it is considered to be this HealthChannelExternalStatus.

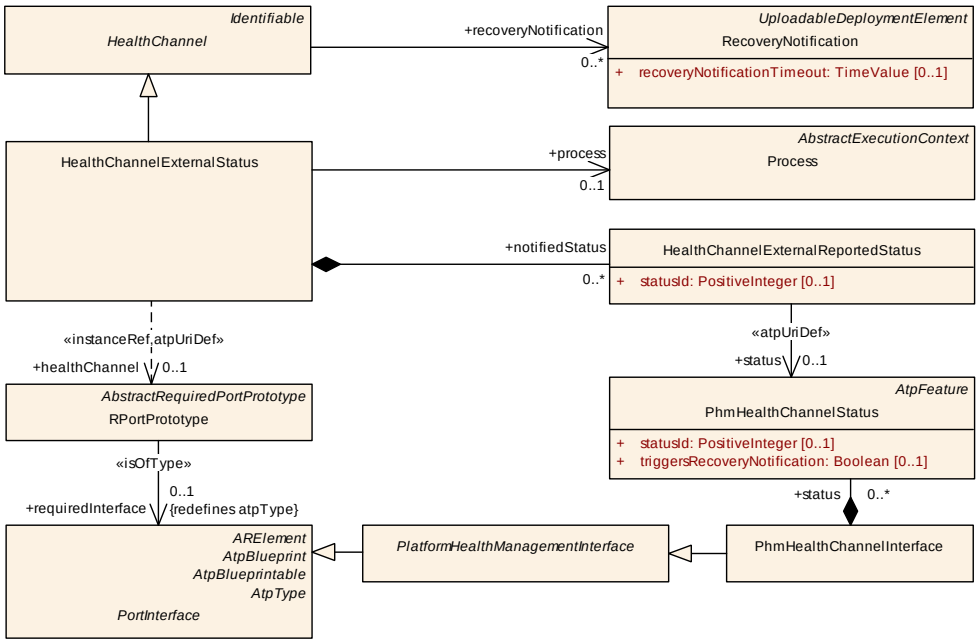


Figure 10.16: Modeling of HealthChannelExternalStatus

Class	**HealthChannelExternalStatus**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a health channel representing the status of an external health channel.			
Base	*ARObject*, *HealthChannel*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	PlatformHealthManagementContribution.healthChannel			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**HealthChannelExternalStatus**			
healthChannel	RPortPrototype	0..1	iref	Refers to the HealthChannel. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef
notifiedStatus	HealthChannelExternal ReportedStatus	*	aggr	This is a list of statuses which shall trigger the Recovery Notification of this HealthChannelExternalStatus.
process	Process	0..1	ref	Defines the Process this Health Channel shall be monitored.

Table 10.32: HealthChannelExternalStatus

Class	**HealthChannelExternalReportedStatus**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This element defines a health channel representing the status of an external health channel.			
Base	*ARObject*			
Aggregated by	HealthChannelExternalStatus.notifiedStatus			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
status	PhmHealthChannel Status	0..1	ref	Reference to one status of a PhmHealthChannel. **Stereotypes:** atpUriDef
statusId	PositiveInteger	0..1	attr	Defines the numeric value which is used to identify the reporting of this HealthChannelExternalReportedStatus to the Phm.

Table 10.33: HealthChannelExternalReportedStatus

10.3.5 Recovery Notification

If Phm detects a Supervision violation or Health Channel violation then the associated `RecoveryNotification` at the `HealthChannel` is activated. This general setup is illustrated in figure 3.43. Via the `RecoveryNotificationToPPortPrototypeMapping` this `RecoveryNotification` is mapped to a dedicated `PPortPrototype` in the context of a dedicated `Process` implementing the State Management functionality.

[constr_3612]{DRAFT} **Multiplicity of references `recoveryNotification`, `recoveryAction`, and `process` at `RecoveryNotificationToPPortPrototypeMapping`** ⌈The references `recoveryNotification`, `recoveryAction`, and `process` shall be defined for each `RecoveryNotificationToPPortPrototypeMapping` **`at the time when the creation of the manifest is finished`**.⌋*()*

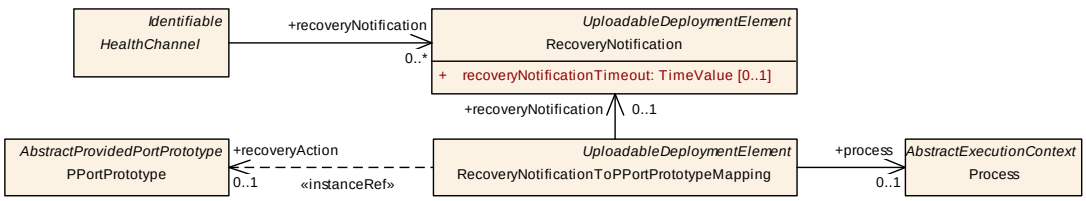


Figure 10.17: Modeling of `RecoveryNotification`

[constr_3613]{DRAFT} **Reference to a PhmSupervisionRecoveryNotificationInterface in the context of a HealthChannelSupervision** ⌈If the RecoveryNotification is aggregated by a HealthChannelSupervision then the RecoveryNotificationToPPortPrototypeMapping shall refer to a PPortPrototype in the role recoveryAction typed by PhmSupervisionRecoveryNotificationInterface **at the time when the creation of the manifest is finished**.⌋()

[constr_3614]{DRAFT} **Reference to a PhmHealthChannelRecoveryNotificationInterface in the context of a HealthChannelExternalStatus** ⌈If the RecoveryNotification is aggregated by a HealthChannelExternalStatus then the RecoveryNotificationToPPortPrototypeMapping shall refer to a PPortPrototype in the role recoveryAction typed by PhmHealthChannelRecoveryNotificationInterface **at the time when the creation of the manifest is finished**.⌋()

Class	**RecoveryNotification**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This meta-class represents a PHM action that can trigger a recovery operation inside a piece of State Management software. **Tags:** atp.recommendedPackage=RecoveryNotifications			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
recovery Notification Timeout	TimeValue	0..1	attr	The maximum acceptable amount of time (in seconds), Platform Health Management waits for an acknowledgement by State Management after sending the notification.

Table 10.34: RecoveryNotification

Class	**RecoveryNotificationToPPortPrototypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	This meta-class represents the ability to associate a RecoveryNotification to a PPortPrototype while also being able to identify the respective Process in which the actual recovery executes. **Tags:** atp.recommendedPackage=RecoveryNotificationMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
process	Process	0..1	ref	Reference to the process which represents the State Management instance that the recovery notification shall be applied to.
recoveryAction	PPortPrototype	0..1	iref	This reference identifies the PortPrototype to be addressed as part of a PHM recovery. **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
recovery Notification	RecoveryNotification	0..1	ref	This reference identifies the applicable Recovery Notification to be mapped.

Table 10.35: RecoveryNotificationToPPortPrototypeMapping

10.4 Time Synchronization Deployment

10.4.1 Overview

This chapter explains the configuration of the Time Synchronization functional cluster.

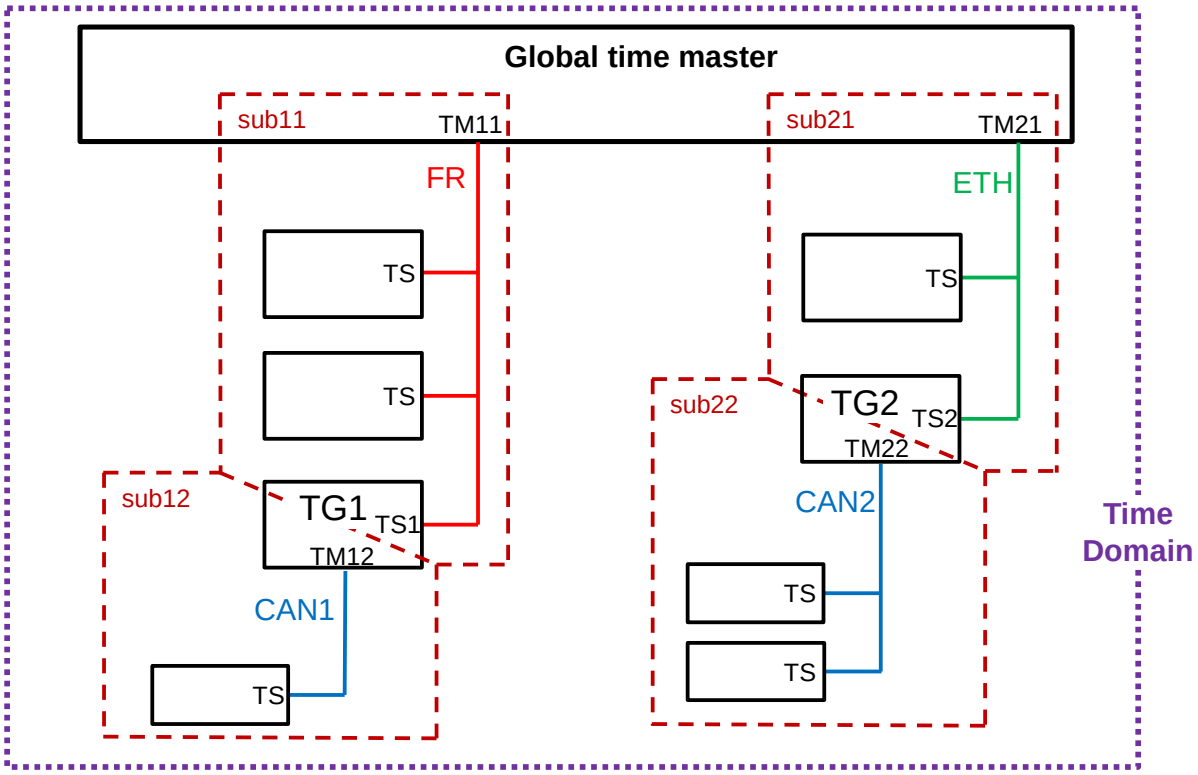
An adaptive AUTOSAR application can utilize several (synchronized) Time-Base Resources which are provided by the Time Synchronization functional cluster [15].

The intended interaction of an adaptive AUTOSAR application with Time Synchronization is described in chapter 3.3.6.

Since an adaptive `Machine` is usually collaborating with other `Machine`s (adaptive) and ECUs (classic), special focus has been put on the vehicle wide definition of synchronized time.

For a detailed specification please refer to the *Global Time Synchronization* chapter in the *System Template* [18].

Figure 10.18 provides an example system view on time domains and their transportation over diverse networks. In the scope of the *AUTOSAR adaptive platform* the focus is put on the Ethernet interaction with the rest of the system.



TS: GlobalTime slave
TM: GlobalTime master
TG: GlobalTime gateway

Figure 10.18: Example setup of Synchronized Global Time in AUTOSAR

10.4.2 Time Synchronization functional cluster configuration

The representation of the Time Synchronization functional cluster [15] within one specific `Machine` is defined by the `TimeSyncModuleInstantiation`. The `Machine` has the ability to define a set of `moduleInstantiations`, where a specialization can be the `TimeSyncModuleInstantiation`.

Class	**TimeSyncModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class defines the attributes for the Time Synchronization configuration on a specific machine.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
timeBase	TimeBaseResource	*	aggr	This aggregation defines the configured Time Bases for Time Synchronization.

Table 10.36: TimeSyncModuleInstantiation

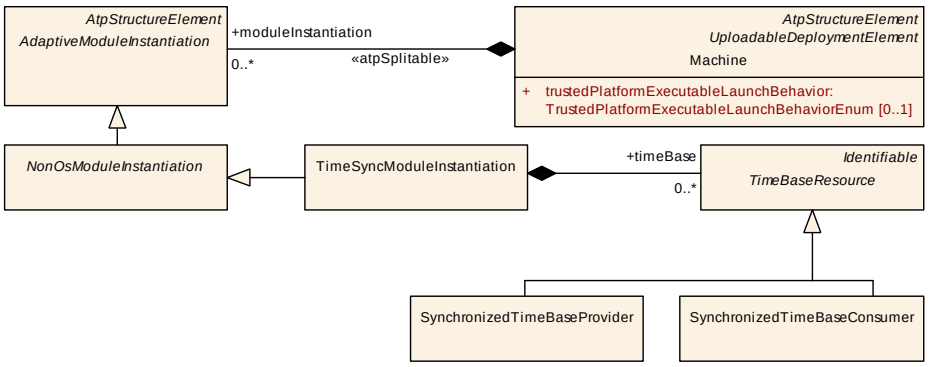


Figure 10.19: Modeling of `TimeSyncModuleInstantiation`

10.4.3 Time Base

The `TimeSyncModuleInstantiation` represents the actual instance of the Time Synchronization functional cluster executed on a specific `Machine`. In the scope of the `TimeSyncModuleInstantiation` the Time-Base Resources are defined.

[TPS_MANI_03539]{DRAFT} **Definition of Time-Base Resources** ⌈The meta-class `TimeSyncModuleInstantiation` has the ability to define a set of Time-Base Resources of kind `TimeBaseResource` in the role `timeBase`.⌋ *(RS_MANI_00040)*

There are several sub types of `TimeBaseResource` which will be explained in the following sections.

Class	***TimeBaseResource*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents the attributes of one Time Base Resource for Time Synchronization.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	SynchronizedTimeBaseConsumer, SynchronizedTimeBaseProvider			
Aggregated by	TimeSyncModuleInstantiation.timeBase			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 10.37: TimeBaseResource

10.4.3.1 Synchronized time base

When configuring a synchronized time base many configuration aspects are already provided by the definition of the `GlobalTimeDomain` and are specified in the *System Template* [18] and associated to `MachineDesign`.

As for the configuration of the `TimeSyncModuleInstantiation` the usage of the `SynchronizedTimeBaseProvider` respectively `SynchronizedTimeBaseConsumer` defines the interaction with the `GlobalTimeDomain`.

[TPS_MANI_03541]{DRAFT} **Definition of `SynchronizedTimeBaseConsumer`** ⌈The meta-class `SynchronizedTimeBaseConsumer` defines a Time Base which is synchronized with a time coming from the network. With the reference `SynchronizedTimeBaseConsumer.networkTimeConsumer` to a `GlobalTimeSlave` the relation to the system model is established.⌋*(RS_MANI_00040)*

[TPS_MANI_03542]{DRAFT} **Definition of `SynchronizedTimeBaseProvider`** ⌈The meta-class `SynchronizedTimeBaseProvider` defines a Time Base which is propagated to a time on the network. With the reference `SynchronizedTimeBaseProvider.networkTimeProvider` to a `GlobalTimeMaster` the relation to the system model is established.⌋*(RS_MANI_00040)*

Some aspects of the Synchronized Time Base for the *provider* role are not available in the system model, those are provided with the `TimeSyncCorrection`.

[TPS_MANI_03543]{DRAFT} **Definition of time sync correction attributes** ⌈The meta-class `TimeSyncCorrection` defines the attributes required to specify the time sync correction behavior of a `SynchronizedTimeBaseProvider`. The `SynchronizedTimeBaseProvider` aggregates the `TimeSyncCorrection` in the role `timeSyncCorrection`.⌋*(RS_MANI_00040)*

The synchronized global time feature also supports the definition of *offset* time domains.

[TPS_MANI_03547]{DRAFT} **Definition of *offset* time domains** ⌈A `GlobalTimeDomain` which has a `offsetTimeDomain` reference defined is considered an *offset*

time domain. The reference source is the *offset* time domain. The reference *target* is the synchronized time domain.⌋ *(RS_MANI_00040)*

The *offset* time domain is applicable to `GlobalTimeMaster` (therefore also `SynchronizedTimeBaseProvider`) and `GlobalTimeSlave` (therefore also `SynchronizedTimeBaseConsumer`).

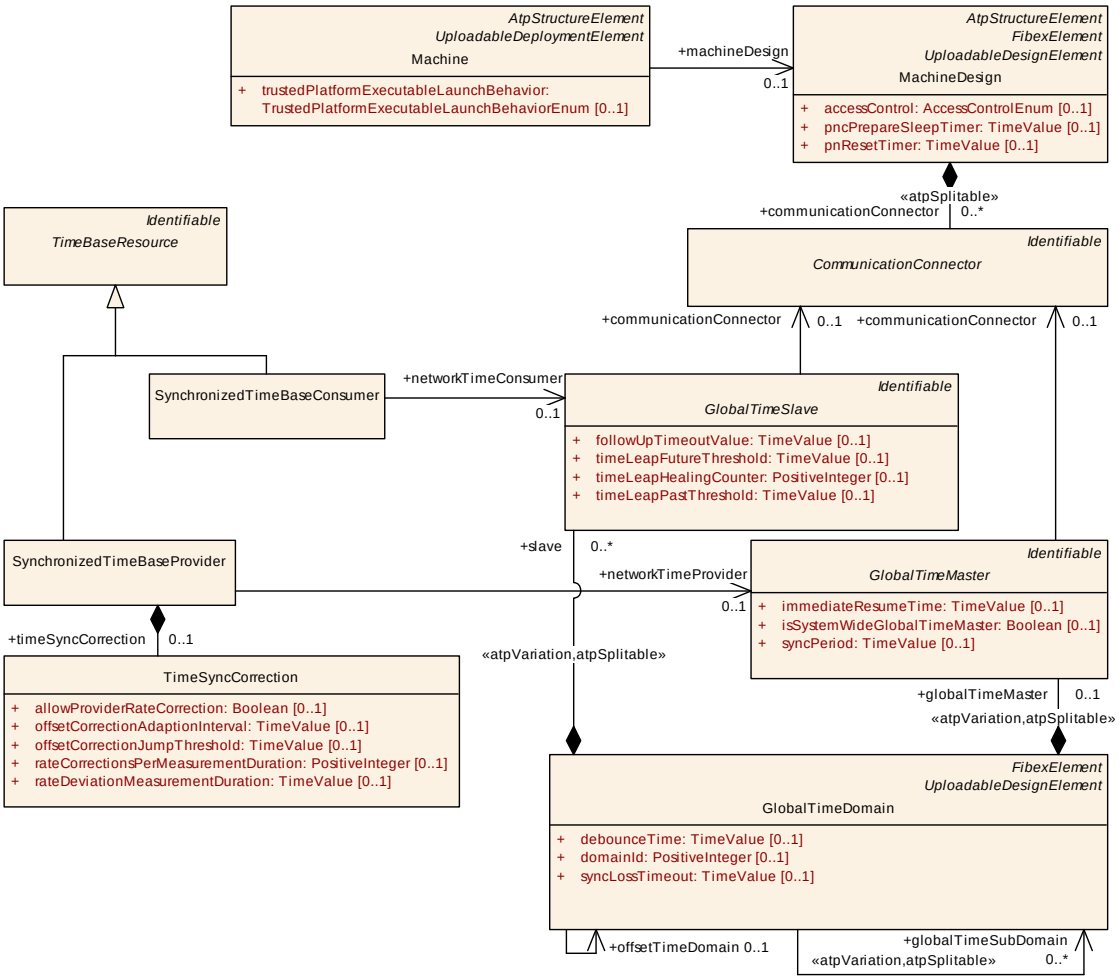


Figure 10.20: Modeling of synchronized time bases

Class	**SynchronizedTimeBaseConsumer**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents a Synchronized Time Base Consumer.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimeBaseResource*			
Aggregated by	TimeSyncModuleInstantiation.timeBase			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
networkTime Consumer	GlobalTimeSlave	0..1	ref	This reference defines the GlobalTime Consumer which is synchronized with this Time Base.

Table 10.38: SynchronizedTimeBaseConsumer

[constr_10184]{DRAFT} **Multiplicity of the reference in the role SynchronizedTimeBaseConsumer.networkTimeConsumer** ⌈For each SynchronizedTimeBaseConsumer, the reference in the role networkTimeConsumer shall exist **at the time when the creation of the manifest is finished.**⌋()

Class	**SynchronizedTimeBaseProvider**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents a Synchronized Time Base Provider.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *TimeBaseResource*			
Aggregated by	TimeSyncModuleInstantiation.timeBase			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
networkTime Provider	GlobalTimeMaster	0..1	ref	This reference defines the GlobalTime Provider which is synchronized with this Time Base.
timeSync Correction	TimeSyncCorrection	0..1	aggr	This aggregation defines the attributes used for the correction of time synchronization.

Table 10.39: SynchronizedTimeBaseProvider

[constr_10185]{DRAFT} **Multiplicity of the reference in the role SynchronizedTimeBaseProvider.networkTimeProvider** ⌈For each SynchronizedTimeBaseProvider, the reference in the role networkTimeProvider shall exist **at the time when the creation of the manifest is finished.**⌋()

Class	**TimeSyncCorrection**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents the attributes used for the correction of time synchronization.			
Base	*ARObject*			
Aggregated by	SynchronizedTimeBaseProvider.timeSyncCorrection			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allowProvider RateCorrection	Boolean	0..1	attr	Defines whether the rate correction value of a Time Base can be set by means of the method setRateCorrection(). false: rate correction cannot be set by method setRate Correction(). true: rate correction can be set by method setRate Correction().
offsetCorrection AdaptionInterval	TimeValue	0..1	attr	Defines the interval during which the adaptive rate correction cancels out the rate and time deviation. Unit: seconds.
offsetCorrection JumpThreshold	TimeValue	0..1	attr	Threshold for the correction method. Deviations below this value will be corrected by a linear reduction over a defined timespan. Values equal and greater than this value will be corrected by immediately setting the correct time and rate in form of a jump. Unit: seconds.
rateCorrections Per Measurement Duration	PositiveInteger	0..1	attr	Number of simultaneous rate measurements to determine the current rate deviation.
rateDeviation Measurement Duration	TimeValue	0..1	attr	Time span used to calculate the rate deviation. Unit: seconds.

Table 10.40: TimeSyncCorrection

10.4.3.2 Persistent Time Base value storage

The Time Synchronization functional cluster allows to define a storage for a SynchronizedTimeBaseProvider time base. During shutdown the value of that specific time base is stored in a PersistencyDeploymentElement (either PersistencyKeyValuePair or PersistencyFile) and during startup the value is read from persistency and restored at the time base.

[TPS_MANI_03632]{DRAFT} **Semantics of TimeBaseProviderToPersistencyMapping** ⌈The TimeBaseProviderToPersistencyMapping defines that the referenced SynchronizedTimeBaseProvider (TimeBaseProviderToPersistencyMapping.timeBaseProvider) time value shall be stored to persistency during shutdown and restored from persistency during startup (TimeBaseProviderToPersistencyMapping.persistencyDeploymentElement).⌋*(RS_MANI_00040)*

It is also possible to define a cycle in which the time base value shall be stored periodically using the attribute TimeBaseProviderToPersistencyMapping.cyclicBackupInterval.

[TPS_MANI_03671]{DRAFT} **Periodic storage of time base value** ⌈The attribute TimeBaseProviderToPersistencyMapping.cyclicBackupInterval defines an interval in which the time base value shall be stored periodically to the TimeBaseProviderToPersistencyMapping.persistencyDeploymentElement.⌋*(RS_MANI_00040)*

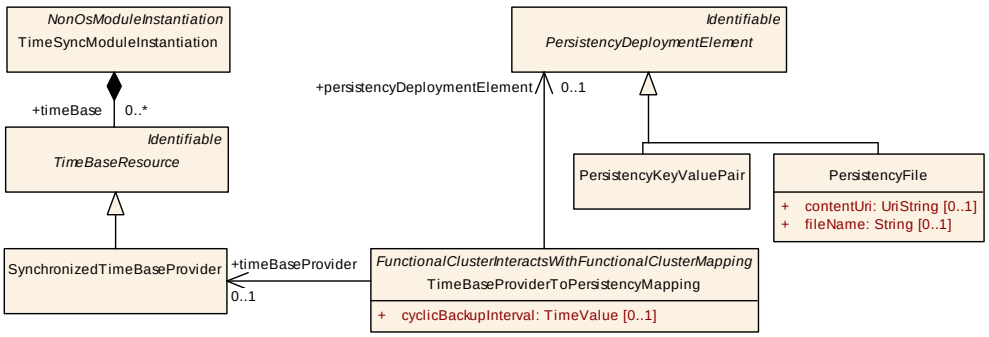


Figure 10.21: Modeling of TimeBaseProviderToPersistencyMapping

Class	**TimeBaseProviderToPersistencyMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class represents the ability to define a mapping between a TimeBaseProvider and a PersistencyDeploymentElement for the purpose of storing and retrieving the time value. **Tags:** atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**TimeBaseProviderToPersistencyMapping**			
cyclicBackup Interval	TimeValue	0..1	attr	Time interval in seconds to store the time base value periodically to persistence.
persistency Deployment Element	PersistencyDeployment Element	0..1	ref	This reference represents the PersistencyDeployment Element where the time value shall be stored in and retrieved from.
timeBase Provider	SynchronizedTimeBase Provider	0..1	ref	This reference represents the mapped TimeBase Provider.

Table 10.41: TimeBaseProviderToPersistencyMapping

[constr_3619]{DRAFT} **Mandatory references of `TimeBaseProviderToPersistencyMapping`** ⌈The references `TimeBaseProviderToPersistencyMapping.persistencyDeploymentElement` and `TimeBaseProviderToPersistencyMapping.timeBaseProvider` shall exist **at the time when the creation of the manifest is finished**.⌋ *()*

10.4.3.3 Ethernet synchronized time

As the *AUTOSAR adaptive platform* supports Ethernet as communication network also the time synchronization using Ethernet is supported.

In order to configure the behavior of the Ethernet time synchronization the specific sub-classes are used as shown in figure 10.22.

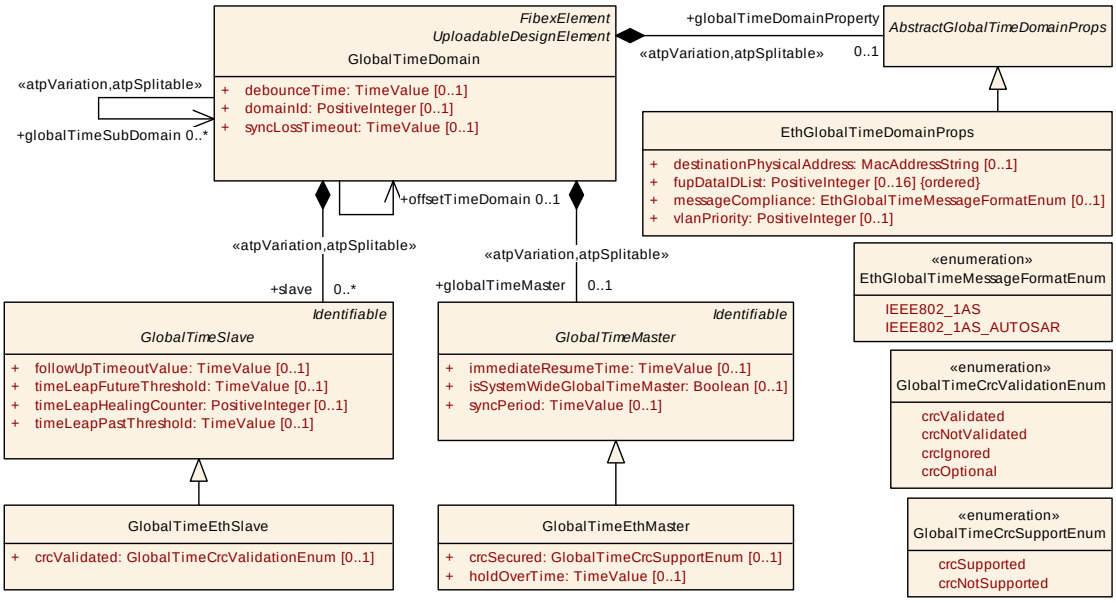


Figure 10.22: Modeling of Ethernet synchronized time

Class	**EthGlobalTimeDomainProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime::ETH			
Note	Enables the definition of Ethernet Global Time specific properties.			
Base	*ARObject*, *AbstractGlobalTimeDomainProps*			
Aggregated by	GlobalTimeDomain.globalTimeDomainProperty			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
crcFlags	EthTSynCrcFlags	0..1	aggr	Defines the fields of the message which shall be taken into account for CRC calculation and verification.
destination Physical Address	MacAddressString	0..1	attr	Defines the MAC multicast address the Ethernet time sync messages are communicated on.
fupDataIDList (ordered)	PositiveInteger	0..16	attr	The DataIDList for FUP messages to calculate CRC.
managed CouplingPort	EthGlobalTime ManagedCouplingPort	*	aggr	Collection of CouplingPorts which are managed in the scope of this Ethernet GlobalTimeDomain.
message Compliance	EthGlobalTimeMessage FormatEnum	0..1	attr	Defines the compliance of the Ethernet time sync messages to specific standards.
vlanPriority	PositiveInteger	0..1	attr	Defines which VLAN priority shall be assigned to a time sync message in case the message is sent using a VLAN tag.

Table 10.42: EthGlobalTimeDomainProps

Class	**EthTSynCrcFlags**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime::ETH			
Note	Defines the fields of the message which shall be taken into account for CRC calculation and verification.			
Base	*ARObject*			
Aggregated by	EthGlobalTimeDomainProps.crcFlags			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
crcCorrection Field	Boolean	0..1	attr	CorrectionField from the Follow_Up Message Header shall be included in CRC calculation.
crcDomain Number	Boolean	0..1	attr	DomainNumber from the Follow_Up Message Header shall be included in CRC calculation.
crcMessage Length	Boolean	0..1	attr	MessageLength from the Follow_Up Message Header shall be included in CRC calculation.
crcPrecise Origin Timestamp	Boolean	0..1	attr	PreciseOriginTimestamp from the Follow_Up Message Field shall be included in CRC calculation.
crcSequenceId	Boolean	0..1	attr	SequenceId from the Follow_Up Message Header shall be included in CRC calculation.
crcSourcePort Identity	Boolean	0..1	attr	SourcePortIdentity from the Follow_Up Message Header shall be included in CRC calculation.

Table 10.43: EthTSynCrcFlags

Primitive	**MacAddressString**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This primitive specifies a Mac Address. Notation: FF:FF:FF:FF:FF:FF Alternative notations, e.g. using dash instead of colon, or another grouping of numbers, is not allowed. **Tags:** xml.xsd.customType=MAC-ADDRESS-STRING xml.xsd.pattern=([0-9a-fA-F]{2}:){5}[0-9a-fA-F]{2} xml.xsd.type=string

Table 10.44: MacAddressString

Class	**EthGlobalTimeManagedCouplingPort**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime::ETH			
Note	Specifies a CouplingPort which is managed by an Ethernet Global Time Domain.			
Base	*ARObject*			
Aggregated by	EthGlobalTimeDomainProps.managedCouplingPort			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
couplingPort	CouplingPort	0..1	ref	Defines which CouplingPort is managed by this EthGlobal TimeManagedCouplingPort.
globalTimePort Role	GlobalTimePortRole Enum	0..1	attr	This attribute defines the port behavior.
globalTimeTx Period	TimeValue	0..1	attr	This attribute defines the TX period in seconds
pdelayLatency Threshold	TimeValue	0..1	attr	Threshold for calculated Pdelay. If a measured Pdelay exceeds pdelayLatencyThreshold, the measured Pdelay value is discarded.
pdelayRequest Period	TimeValue	0..1	attr	Defines the period for the pdelay request messages.
pdelayRespAnd RespFollowUp Timeout	TimeValue	0..1	attr	Timeout value for Pdelay_Resp and Pdelay_Resp_ Follow_Up after a Pdelay_Req has been transmitted resp. a Pdelay_Resp has been received. A value of 0 or not defining this attribute deactivates this timeout observation.
pdelay Response Enabled	Boolean	0..1	attr	Defines whether PDELAY RESPONSE and PDELAY RESPONSE FOLLOW UP shall be sent on this Coupling Port.

Table 10.45: EthGlobalTimeManagedCouplingPort

Enumeration	**EthGlobalTimeMessageFormatEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime::ETH
Note	Specifies which message formats are available to for the Ethernet time sync protocol.
Aggregated by	EthGlobalTimeDomainProps.messageCompliance
Literal	***Description***
IEEE802_1AS	Message format according to IEEE 802.1AS standard. **Tags:** atp.EnumerationLiteralIndex=0 xml.name=IEEE802-1AS
IEEE802_1AS_ AUTOSAR	Message format according to IEEE 802.1AS standard with AUTOSAR extensions. **Tags:** atp.EnumerationLiteralIndex=1 xml.name=IEEE802-1AS-AUTOSAR

Table 10.46: EthGlobalTimeMessageFormatEnum

Class	**GlobalTimeEthSlave**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime::ETH			
Note	This represents the specialization of the GlobalTimeSlave for Ethernet communication.			
Base	*ARObject*, *GlobalTimeSlave*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	GlobalTimeDomain.slave			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
crcValidated	GlobalTimeCrc ValidationEnum	0..1	attr	Definition of whether or not validation of the CRC is supported.

Table 10.47: GlobalTimeEthSlave

Enumeration	**GlobalTimeCrcValidationEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime
Note	This enumeration provides values for the evaluation of the CRC
Aggregated by	GlobalTimeCanSlave.crcValidated, GlobalTimeEthSlave.crcValidated, GlobalTimeFrSlave.crc Validated
Literal	***Description***
crcIgnored	The CRC is supposed to be ignored **Tags:** atp.EnumerationLiteralIndex=0
crcNotValidated	The CRC is not supposed to be present. If CRC is present the message is ignored. **Tags:** atp.EnumerationLiteralIndex=1
crcOptional	Either the CRC is present and then shall be validated or the CRC is not present and no CRC check is done. **Tags:** atp.EnumerationLiteralIndex=3
crcValidated	This CRC is supposed to be validated. **Tags:** atp.EnumerationLiteralIndex=2

Table 10.48: GlobalTimeCrcValidationEnum

Class	**GlobalTimeEthMaster**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime::ETH			
Note	This represents the specialization of the GlobalTimeMaster for Ethernet communication.			
Base	*ARObject*, *GlobalTimeMaster*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	GlobalTimeDomain.globalTimeMaster			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
crcSecured	GlobalTimeCrcSupport Enum	0..1	attr	Definition of whether or not CRC is supported. This is only relevant for selected bus systems.
holdOverTime	TimeValue	0..1	attr	This attribute defines the timeout for transmission of Sync and Follow_Up messages on Master ports in absence of reception of Sync and Follow_Up messages on Slave port.
subTlvConfig	EthTSynSubTlvConfig	0..1	aggr	Defines the subTLV fields which shall be included in the time sync message.

Table 10.49: GlobalTimeEthMaster

Enumeration	**GlobalTimeCrcSupportEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime
Note	This enumeration is used to define whether and how CRC on the TX side shall be utilized.
Aggregated by	GlobalTimeCanMaster.crcSecured, GlobalTimeEthMaster.crcSecured, GlobalTimeFrMaster.crc Secured
Literal	***Description***
crcNotSupported	This indicates that CRC is not supported **Tags:** atp.EnumerationLiteralIndex=0
crcSupported	This indicates that CRC is supported **Tags:** atp.EnumerationLiteralIndex=1

Table 10.50: GlobalTimeCrcSupportEnum

Class	**EthTSynSubTlvConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime::ETH			
Note	Defines the subTLV fields which shall be included in the time sync message.			
Base	*ARObject*			
Aggregated by	GlobalTimeEthMaster.subTlvConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ofsSubTlv	Boolean	0..1	attr	Defines whether an AUTOSAR Follow_Up TLV OFS Sub-TLV is used.
statusSubTlv	Boolean	0..1	attr	Defines whether an AUTOSAR Follow_Up TLV Status Sub-TLV is used.
timeSubTlv	Boolean	0..1	attr	Defines whether an AUTOSAR Follow_Up TLV Time Sub-TLV is used.
userDataSubTlv	Boolean	0..1	attr	Defines whether an AUTOSAR Follow_Up TLV UserData Sub-TLV is used.

Table 10.51: EthTSynSubTlvConfig

10.4.4 Time Base to Port Prototype mapping

The TimeBaseResource definition of chapter 10.4.3 and the RPortPrototype typed by a sub-class of AbstractSynchronizedTimeBaseInterface of chapter 3.3.6 have to be mapped to each other in order to define the binding of application software to the platform foundation software implementing the time synchronization.

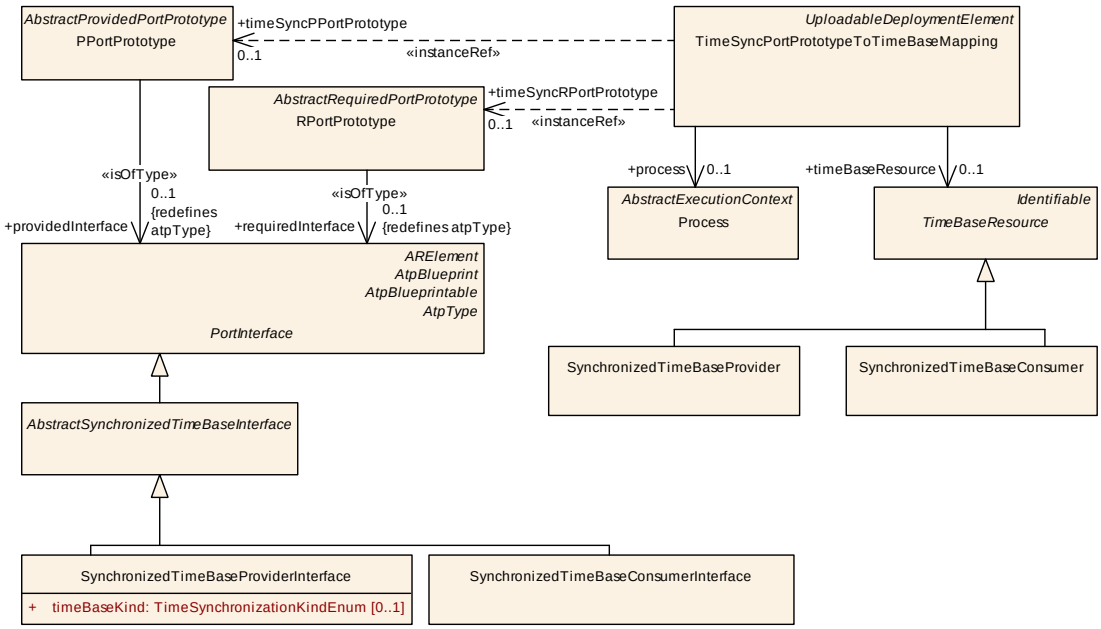


Figure 10.23: Modeling of TimeSyncPortPrototypeToTimeBaseMapping

[TPS_MANI_03548]{DRAFT} **Definition of TimeSyncPortPrototypeToTimeBaseMapping** ⌈A TimeSyncPortPrototypeToTimeBaseMapping is used to define a mapping between a TimeBaseResource and a RPortPrototype typed by a sub-class of AbstractSynchronizedTimeBaseInterface in the context of a Process.⌋ *(RS_MANI_00040)*

The TimeSyncPortPrototypeToTimeBaseMapping takes the Process into account so that every instantiation of an Executable (and the resulting instantiation of all the RPortPrototypes typed by a sub-class of AbstractSynchronizedTimeBaseInterface) can be mapped individually to TimeBaseResources.

Class	**TimeSyncPortPrototypeToTimeBaseMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::TimeSync			
Note	This meta-class provides the ability to map a PortPrototype typed by a AbstractSynchronizedTimeBase Interface to a TimeBaseResource in the context of a Process. **Tags:** atp.recommendedPackage=TimeSyncPortPrototypeToTimeBaseMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
process	Process	0..1	ref	Reference to the context Process this mapping applies to.
timeBase Resource	TimeBaseResource	0..1	ref	Reference to the mapped TimeBaseResource.
timeSyncPPort Prototype	PPortPrototype	0..1	iref	Instance reference to the mapped PPortPrototype typed by a AbstractSynchronizedTimeBaseInterface. **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
timeSyncRPort Prototype	RPortPrototype	0..1	iref	Instance reference to the mapped RPortPrototype typed by a AbstractSynchronizedTimeBaseInterface. **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 10.52: TimeSyncPortPrototypeToTimeBaseMapping

The example shown in figure 10.24 illustrates the mapping of RPortPrototypes typed by one of the sub-classes of AbstractSynchronizedTimeBaseInterface to actually configured TimeBaseResources at the Time Sync Management.

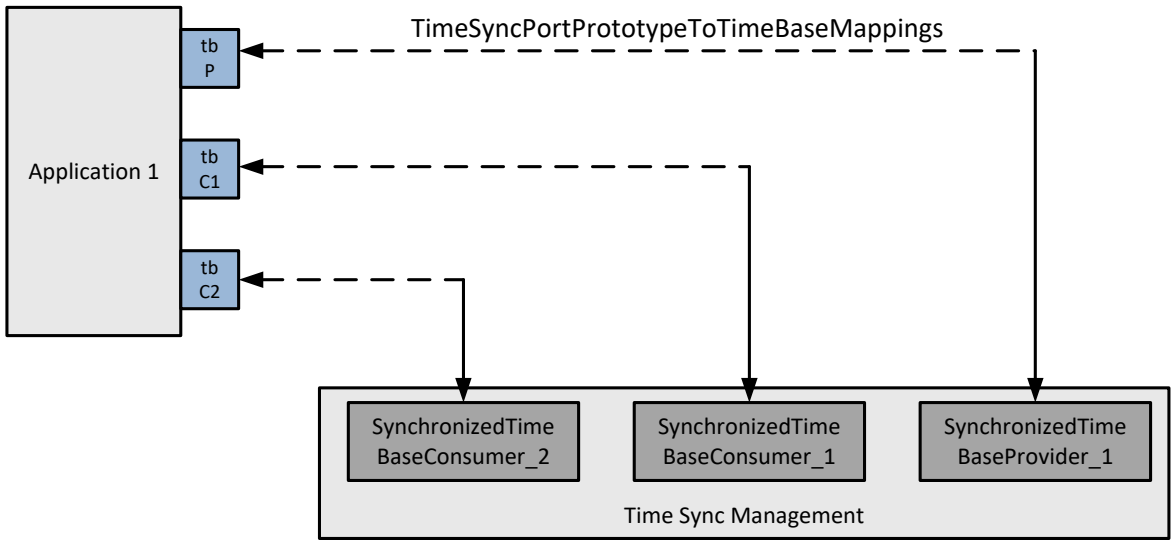


Figure 10.24: Example PortPrototype to TimeBase mapping

10.5 DoIP configuration

[TPS_MANI_03164]{DRAFT} **`Machine`-specific configuration settings for DoIP** ⌈The `Machine`-specific configuration settings for DoIP are collected in `DoIpInstantiation`.⌋*(RS_MANI_00023)*

Class	**DoIpInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
Note	This meta-class defines the attributes for the DoIP configuration on a specific machine.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eid	PositiveUnlimitedInteger	0..1	attr	Configured EID (Entity ID) used for VehicleIdentification Request.
entityStatusMaxByteFieldUse	Boolean	0..1	attr	This attribute is used to distinguish the optional support of the Max data size element of a diagnostic entity status response.
gid	PositiveUnlimitedInteger	0..1	attr	Configured GID (Group ID) used for VehicleIdentification Request. If configured, take this value (and set "Further action required" byte to 0x00="No further action required"), if not configured use ServiceInterface Do IPGroupIdentification to retrieve GID and 'further action required' values.
gidInvalidity Pattern	PositiveInteger	0..1	attr	Specifies the Byte pattern that is used for response messages if no valid GID could be retrieved. Only the value '0' or '255' is allowed.
logicalAddress	PositiveInteger	0..1	attr	Describes the logical address of the DoIP entity, which is used for VehicleAnnouncement and RoutingActivation responses.
maxRequest Bytes	PositiveInteger	0..1	attr	Specifies the maximum allowed bytes of a DoIP message request without the DoIP header.
network Interface	DoIpNetwork Configuration	*	aggr	Network interface specific DoIP properties.
request Configuration	DoIpRequest Configuration	*	aggr	Request configuration that is used to determine whether an incoming DiagnosticMessage request needs to be interpreted as PHYSICAL or FUNCTIONAL. Any request with target address not within the configured target address range will be rejected.
vinInvalidity Pattern	PositiveInteger	0..1	attr	Specifies the Byte pattern that is used for response messages if no valid VIN could be retrieved. Only the value '0' or '255' is allowed.

Table 10.53: DoIpInstantiation

[constr_10186]{DRAFT} **Multiplicity of attribute `DoIpInstantiation.entityStatusMaxByteFieldUse`** ⌈For each `DoIpInstantiation`, the attribute `entityStatusMaxByteFieldUse` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10187]{DRAFT} **Multiplicity of attribute `DoIpInstantiation.gidInvalidityPattern`** ⌈For each `DoIpInstantiation`, the attribute `gidInvalidityPattern` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10188]{DRAFT} **Multiplicity of attribute DoIpInstantiation.logicalAddress** ⌈For each DoIpInstantiation, the attribute logicalAddress shall exist **at the time when the creation of the manifest is finished.**⌋()

[constr_10189]{DRAFT} **Multiplicity of attribute DoIpInstantiation.maxRequestBytes** ⌈For each DoIpInstantiation, the attribute maxRequestBytes shall exist **at the time when the creation of the manifest is finished.**⌋()

[constr_10190]{DRAFT} **Multiplicity of attribute DoIpInstantiation.vinInvalidityPattern** ⌈For each DoIpInstantiation, the attribute vinInvalidityPattern shall exist **at the time when the creation of the manifest is finished.**⌋()

[constr_3425]{DRAFT} **Restriction of DoIpInstantiations on a Machine** ⌈Each Machine shall aggregate at most one DoIpInstantiation in the role moduleInstantiation **at the time when the creation of the manifest is finished.**⌋()

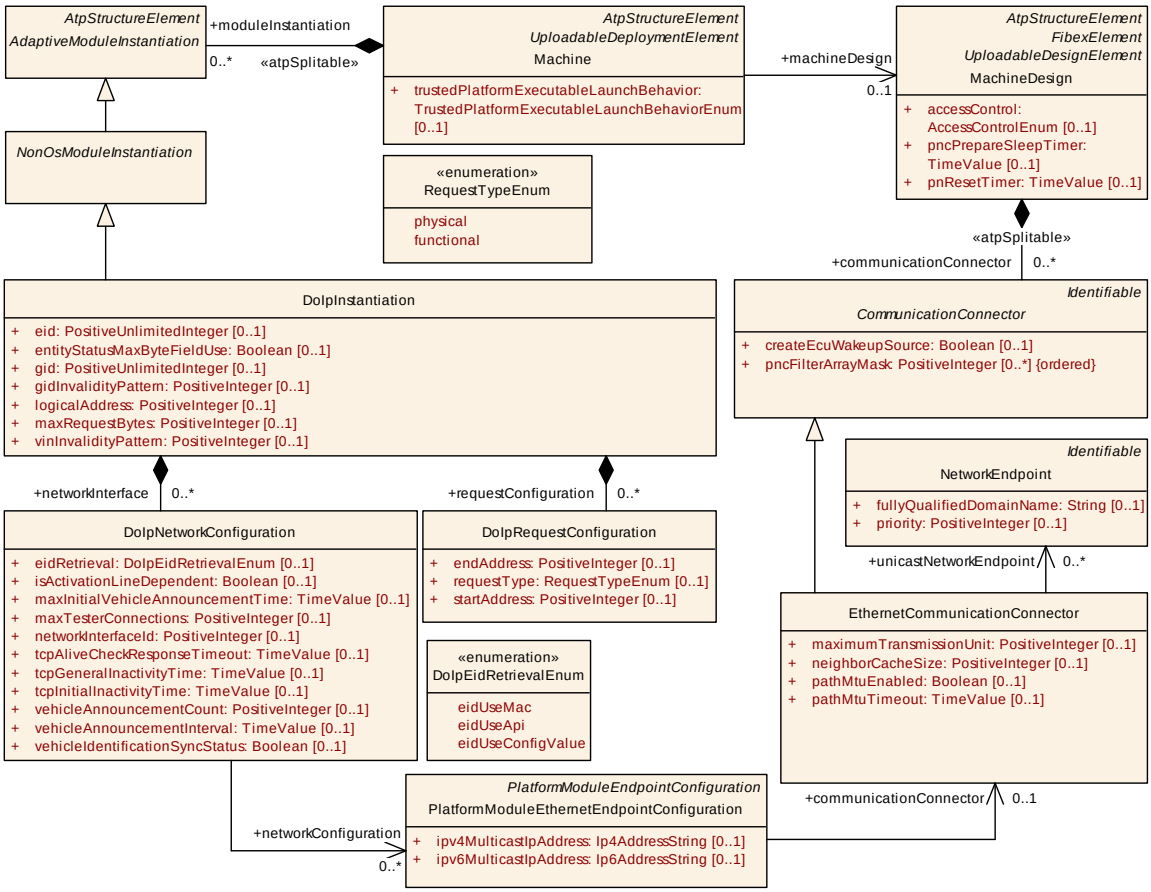


Figure 10.25: DoIP configuration

[constr_3495]{DRAFT} **Supported value range for attribute DoIpInstantiation.eid** ⌈The supported value range of attribute DoIpInstantiation.eid is limited

to the interval [0..281474976710655] **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3496]{DRAFT} **Supported value range for attribute DoIpInstantiation.gid** ⌈The supported value range of attribute DoIpInstantiation.gid is limited to the interval [0..281474976710655] **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3497]{DRAFT} **Supported value range for attribute DoIpInstantiation.maxRequestBytes** ⌈The supported value range of attribute DoIpInstantiation.maxRequestBytes is limited to the interval [0..4294967295] **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3498]{DRAFT} **Supported value range for attribute DoIpInstantiation.logicalAddress** ⌈The supported value range of attribute DoIpInstantiation.logicalAddress is limited to the interval [0..65535] **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_03165]{DRAFT} **Network Interface configuration for DoIP** ⌈The DoIpNetworkConfiguration contains all configuration settings that are specific for a configured network connection. The network connection is configured with the PlatformModuleEthernetEndpointConfiguration that is referenced by the DoIpNetworkConfiguration in the role networkConfiguration.

The tcpPort and udpPort references to the ApApplicationEndpoint are used to configure the Transport Protocol (Udp or Tcp) and the used Port number. The IP Address is configured in the NetworkEndpoint that is referenced by the PlatformModuleEthernetEndpointConfiguration via the EthernetCommunicationConnector.⌋*(RS_MANI_00023)*

Class	**DoIpNetworkConfiguration**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This element collects DoIP properties that are network interface specific.			
Base	*ARObject*			
Aggregated by	DoIpInstantiation.networkInterface			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eidRetrieval	DoIpEidRetrievalEnum	0..1	attr	This attribute defines how DoIp Entitiy Identification is retrieved.
isActivationLine Dependent	Boolean	0..1	attr	This attribute defines whether the network interface • is started "on-demand" when an activation line is sensed or • is always available.
maxInitial Vehicle Announcement Time	TimeValue	0..1	attr	Upper bound for the time to wait in [s] for sending first vehicle announcement message after IP address assignment. Represents parameter A_DoIP_Announce_Wait of ISO 13400-2:2019. The value of this timing shall be determined randomly in the closed interval [0..max InitialVehicleAnnouncementTime].

▽

△

Class	DoIpNetworkConfiguration			
maxTester Connections	PositiveInteger	0..1	attr	Maximum amount of tester connections that shall be maintained at one time before alive check is performed.
network Configuration	PlatformModule EthernetEndpoint Configuration	*	ref	Network configuration (Protocol, Port, IP Address) for transmission of DoIP messages on a specific VLAN.
network InterfaceId	PositiveInteger	0..1	attr	This attribute defines the identifier for the DoIPInterface.
tcpAliveCheck Response Timeout	TimeValue	0..1	attr	Timeout in [s] for waiting for a response to an Alive Check request before the connection is considered to be disconnected. Represents parameter T_TCP_AliveCheck of ISO 13400-2:2019.
tcpGeneral InactivityTime	TimeValue	0..1	attr	Timeout in [s] for maximum inactivity of a TCP socket connection before the DoIP module will close the according socket connection. Represents parameter T_TCP_General_Inactivity of ISO 13400-2:2019.
tcpInitial InactivityTime	TimeValue	0..1	attr	Timeout in [s] used for initial inactivity of a connected TCP socket connection directly after socket connection. Represents parameter T_TCP_Initial_Inactivity of ISO 13400-2:2019.
vehicle Announcement Count	PositiveInteger	0..1	attr	Number of vehicle announcement messages on IP address assignment. Represents parameter A_DoIP_Announce_Num of ISO 13400-2:2019.
vehicle Announcement Interval	TimeValue	0..1	attr	Time to wait in [s] for sending subsequent vehicle announcement messages. Represents parameter A_DoIP_Announce_Interval of ISO 13400-2:2019.
vehicle Identification SyncStatus	Boolean	0..1	attr	Defines if the optional VIN/GID synchronization status is used additionally in the vehicle identification/ announcement.

Table 10.54: DoIpNetworkConfiguration

[TPS_MANI_03323]{DRAFT} **Support for TLS in DoIP communication** ⌈DoIP communication that is secured by TLS is modeled by a `DoIpNetworkConfiguration` that references the `PlatformModuleEthernetEndpointConfiguration` that in turn references `SecureComProps` in the roles:

- `secureComPropsForTcp` - contains the credentials to secure the DoIP communication over TCP
- `secureComPropsForUdp` - contains the credentials to secure the DoIP communication over UDP

⌋*(RS_MANI_00023)*

[constr_10191]{DRAFT} **Multiplicity of attribute `DoIpNetworkConfiguration.isActivationLineDependent`** ⌈For each `DoIpNetworkConfiguration`, the attribute `isActivationLineDependent` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10192]{DRAFT} **Multiplicity of attribute `DoIpNetworkConfiguration.maxInitialVehicleAnnouncementTime`** ⌈For each `DoIpNetworkConfiguration`, the attribute `maxInitialVehicleAnnouncementTime` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10193]{DRAFT} **Multiplicity of attribute DoIpNetworkConfiguration.maxTesterConnections** ⌈For each DoIpNetworkConfiguration, the attribute maxTesterConnections shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3734]{DRAFT} **Upper multiplicity of reference in the role DoIpNetworkConfiguration.networkConfiguration** ⌈In the context of DoIpNetworkConfiguration, the reference in the role networkConfiguration shall exist at most once **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10194]{DRAFT} **Multiplicity of attribute DoIpNetworkConfiguration.networkInterfaceId** ⌈For each DoIpNetworkConfiguration, the attribute networkInterfaceId shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10195]{DRAFT} **Multiplicity of attribute DoIpNetworkConfiguration.vehicleIdentificationSyncStatus** ⌈For each DoIpNetworkConfiguration, the attribute vehicleIdentificationSyncStatus shall exist **at the time when the creation of the manifest is finished**.⌋*()*

Please note that it is possible to define several networkInterfaces in a DoIpInstantiation. For each network connection individual configuration settings can be set with the attributes that are defined in the DoIpNetworkConfiguration element, e.g. it is possible to configure the vehicle announcement for different network connections differently.

[TPS_MANI_03294]{DRAFT} **Semantics of DoIpNetworkConfiguration.eidRetrieval** ⌈If DoIpNetworkConfiguration.eidRetrieval is set to:

- eidUseConfigValue, then take the value from DoIpInstantiation.eid
- eidUseApi, then DoIpInstantiation.eid shall be ignored and the API DiagnosticDoIPEntityIdentification shall be used to retrieve the eid
- eidUseMac, then DoIpInstantiation.eid shall be ignored and the MAC shall be taken as as the eid

⌋*(RS_MANI_00023)*

[constr_5343]{DRAFT} **Usage of DoIpNetworkConfiguration.eidRetrieval** ⌈If DoIpNetworkConfiguration.eidRetrieval is set to eidUseConfigValue then DoIpInstantiation.eid shall exist and a value shall be assigned to it **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_03218]{DRAFT} **Default value for the attribute tcpInitialInactivityTime of meta-class DoIpNetworkConfiguration** ⌈If no value for the attribute DoIpNetworkConfiguration.tcpInitialInactivityTime is defined then the default value of 2 seconds shall be assumed.⌋*(RS_MANI_00023)*

[TPS_MANI_03219]{DRAFT} **Default value for the attribute tcpGeneralInactivityTime of meta-class DoIpNetworkConfiguration** ⌈If no value for the attribute

DoIpNetworkConfiguration.tcpGeneralInactivityTime is defined then the default value of 300 seconds shall be assumed.⌋*(RS_MANI_00023)*

[TPS_MANI_03220]{DRAFT} **Default value for the attribute vehicleAnnouncementCount of meta-class DoIpNetworkConfiguration** ⌈If no value for the attribute DoIpNetworkConfiguration.vehicleAnnouncementCount is defined then the default value of 3 shall be assumed.⌋*(RS_MANI_00023)*

[TPS_MANI_03221]{DRAFT} **Default value for the attribute vehicleAnnouncementInterval of meta-class DoIpNetworkConfiguration** ⌈If no value for the attribute DoIpNetworkConfiguration.vehicleAnnouncementInterval is defined then the default value of 0,5 seconds shall be assumed.⌋*(RS_MANI_00023)*

[TPS_MANI_03222]{DRAFT} **Default value for the attribute tcpAliveCheckResponseTimeout of meta-class DoIpNetworkConfiguration** ⌈If no value for the attribute DoIpNetworkConfiguration.tcpAliveCheckResponseTimeout is defined then the default value of 0,5 seconds shall be assumed.⌋*(RS_MANI_00023)*

During vehicle discovery the DoIP module responds by informing the tester about its own address, configured as the logicalAddress. The tester will approach the ECU under this UDS target address, thus the ECU should have a SoftwareCluster that is configured to respond to this UDS target address.

The list of available target addresses may or may not be obtainable from the SoftwareCluster with the logicalAddress.

In some cases, this SoftwareCluster may have the ability to inform the tester which other existing physical and/or logical addresses are available.

Class	**DoIpRequestConfiguration**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
Note	This meta-class specifies a range of target addresses and its interpretation as either physical or functional request.			
Base	*ARObject*			
Aggregated by	DoIpInstantiation.requestConfiguration			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
endAddress	PositiveInteger	0..1	attr	End address for range of target-addresses (including this address).
requestType	RequestTypeEnum	0..1	attr	Determines the type of request.
startAddress	PositiveInteger	0..1	attr	Start address for range of target-addresses (including this address).

Table 10.55: DoIpRequestConfiguration

[constr_10197]{DRAFT} **Multiplicity of attribute DoIpRequestConfiguration.endAddress** ⌈For each DoIpRequestConfiguration, the attribute endAddress shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10198]{DRAFT} **Multiplicity of attribute DoIpRequestConfiguration.requestType** ⌈For each DoIpRequestConfiguration, the attribute requestType shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10199]{DRAFT} **Multiplicity of attribute DoIpRequestConfiguration.startAddress** ⌈For each DoIpRequestConfiguration, the attribute startAddress shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3492]{DRAFT} **DoIpInstantiation.logicalAddress shall be defined as member in the DoIpRequestConfiguration** ⌈The DoIpInstantiation.logicalAddress shall be a member of the intervals of available physical addresses configured for the DoIpInstantiation in the requestConfiguration **at the time when the creation of the manifest is finished**.⌋*()*

On top of that, there is the expectation that the configured DiagnosticCommonProps.diagnosticAddresses deployed to the Machine fit to the intervals defined in the context of the DoIpInstantiation in the requestConfiguration.

Please note that the DoIpRequestConfiguration corresponds to Table 39 that is defined in ISO-13400-2:2019 [22].

[constr_3499]{DRAFT} **Supported value range for attribute DoIpRequestConfiguration.startAddress** ⌈The supported value range of attribute DoIpRequestConfiguration.startAddress is limited to the interval [0..65535] **at the time when the creation of the manifest is finished**.⌋*()*

[constr_5000]{DRAFT} **Supported value range for attribute DoIpRequestConfiguration.endAddress** ⌈The supported value range of attribute DoIpRequestConfiguration.endAddress is limited to the interval [0..65535] **at the time when the creation of the manifest is finished**.⌋*()*

10.6 Log and Trace module configuration

The Log and Trace functionality in AUTOSAR supports the monitoring of applications and provides means to forward logging information onto the communication bus, the console, to the file system, or to the ARTI trace API. Forwarding to ARTI trace API is only valid for modeled messages.

The logging information is put into a standardized delivery and presentation format that is described in more detail in the Log and Trace Protocol specification [23].The format contains meta-data that identifies for example the application that produces the logging information.

This chapter describes settings that are available in the Machine Manifest to configure the logging framework and defines the relation between the application design and the deployment model.

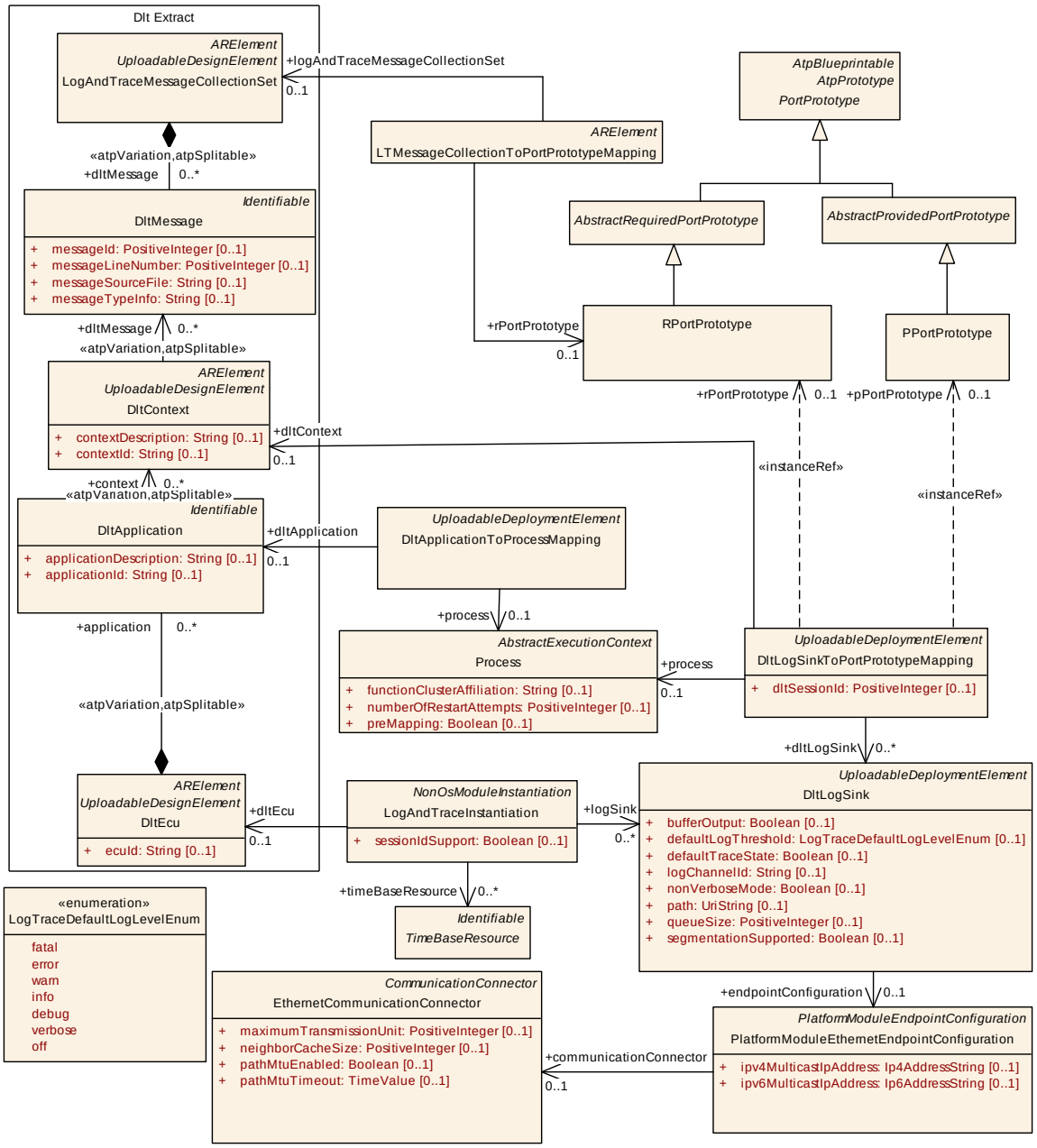


Figure 10.26: Log and Trace module configuration

Please note that the modeling elements described in this chapter are using elements from the Log And Trace Extract Template that defines a AUTOSAR platform independent format for description of log and trace messages and the source that produces these log and trace messages.

10.6.1 Log and trace deployment

[TPS_MANI_03162]{DRAFT} **`Machine`-specific configuration settings for the Log and Trace functional cluster** ⌈The `Machine`-specific configuration settings for the

Log and Trace functional cluster are collected in LogAndTraceInstantiation.⌋ *(RS_MANI_00023)*

[TPS_MANI_03282]{DRAFT} **Assignment of a Dlt Ecu Identifier to a LogAndTraceInstantiation** ⌈The Machine is represented in the Log And Trace Extract by the DltEcu that is referenced from the LogAndTraceInstantiation in the role dltEcu. The referenced DltEcu defines the ecuId that is transported in the standard header of the log and trace message.⌋ *(RS_MANI_00023)*

Class	**LogAndTraceInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::LogAndTrace			
Note	This meta-class defines the attributes for the Log&Trace configuration on a specific machine.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dltEcu	DltEcu	0..1	ref	Reference to the Ecu representation in the Log And Trace Extract.
logSink	DltLogSink	*	ref	Reference to output sinks for log or trace messages that are produced on the Machine.
sessionId Support	Boolean	0..1	attr	This attribute defines whether the sessionId is used or not.
timeBase Resource	TimeBaseResource	*	ref	This reference is used to describe to which time base the Log and Trace module has access. From the Time Base Resource the Log and Trace module gets the needed information to generate the time stamp.

Table 10.56: LogAndTraceInstantiation

[constr_5275]{DRAFT} **Existence of LogAndTraceInstantiation.dltEcu** ⌈For each LogAndTraceInstantiation the reference to DltEcu in the role dltEcu shall exist **at the time when the creation of the manifest is finished**.⌋ *()*

[constr_5276]{DRAFT} **Existence of LogAndTraceInstantiation.logSink** ⌈Each LogAndTraceInstantiation shall reference at least one DltLogSink in the role logSink **at the time when the creation of the manifest is finished**.⌋ *()*

[constr_3729]{DRAFT} **Upper multiplicity of reference in the role LogAndTraceInstantiation.timeBaseResource** ⌈In the context of LogAndTraceInstantiation, the reference in the role timeBaseResource shall exist at most once **at the time when the creation of the manifest is finished**.⌋ *()*

[TPS_MANI_03274]{DRAFT} **Configuration of log and trace sinks** ⌈The output sinks for log or trace messages that are produced on the Machine are defined by DltLogSinks that are referenced in the role logSink from the LogAndTraceInstantiation.⌋ *(RS_MANI_00023)*

Class	**DltLogSink**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::LogAndTrace			
Note	The meta-class defines the output sink for DltLogMessages **Tags:** atp.recommendedPackage=DltLogSinks			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bufferOutput	Boolean	0..1	attr	This attribute defines whether a buffer is used in case that the output sink is the console.
defaultLog Threshold	LogTraceDefaultLog LevelEnum	0..1	attr	This attribute allows to set a log level Threshold for Log Level filtering.
defaultTrace State	Boolean	0..1	attr	This attributes defines the default trace status.
endpoint Configuration	PlatformModule EthernetEndpoint Configuration	0..1	ref	Network configuration (Protocol, Port, IP Address) for transmission of dlt messages on a specific VLAN.
logChannelId	String	0..1	attr	This attribute identifies the LogChannel for usage within the Log And Trace protocol.
nonVerbose Mode	Boolean	0..1	attr	This attribute defines whether this DltLogSink supports non-Verbose Dlt messages. If disabled only verbose mode messages shall be used.
path	UriString	0..1	attr	This attribute defines the path to the file that is used as output sink.
queueSize	PositiveInteger	0..1	attr	Length of the queue (in which messages can be stored before processing) in the unit "Log message".
segmentation Supported	Boolean	0..1	attr	If enabled, segmentation will be used for DLT messages that are larger than EthernetCommunication Connector.maximumTransmissionUnit referenced via Dlt LogSink.endpointConfiguration.

Table 10.57: DltLogSink

Enumeration	**LogTraceDefaultLogLevelEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Dlt
Note	This enum defines available log&trace log levels that may be used to define the severity level of a log message.
Aggregated by	DltLogChannel.logTraceDefaultLogThreshold, DltLogSink.defaultLogThreshold
Literal	***Description***
debug	Detailed information for programmers **Tags:** atp.EnumerationLiteralIndex=4
error	Error with impact to correct functionality **Tags:** atp.EnumerationLiteralIndex=1
fatal	Fatal error **Tags:** atp.EnumerationLiteralIndex=0
info	High level information **Tags:** atp.EnumerationLiteralIndex=3
off	logging is turned off **Tags:** atp.EnumerationLiteralIndex=6

△

Enumeration	LogTraceDefaultLogLevelEnum
verbose	Verbose debug message **Tags:** atp.EnumerationLiteralIndex=5
warn	Warning if correct behavior cannot be ensured **Tags:** atp.EnumerationLiteralIndex=2

Table 10.58: LogTraceDefaultLogLevelEnum

[TPS_MANI_03283]{DRAFT} **Standardized values for attribute `DltLogSink.category`** ⌈AUTOSAR reserves the following values for attribute `DltLogSink.category`:

- `DLT_LOGSINK_REMOTE`
- `DLT_LOGSINK_DLT`
- `DLT_LOGSINK_FILE`
- `DLT_LOGSINK_CONSOLE`
- `DLT_LOGSINK_ARTI`

⌋*(RS_MANI_00023)*

[TPS_MANI_02384]{DRAFT} **`DltLogSink` with `category` `DLT_LOGSINK_REMOTE`** ⌈`DltLogSink` with `category` `DLT_LOGSINK_REMOTE` provides means to forward logging information to a Dlt log channel defined by `logChannelId`.⌋*(RS_MANI_00023)*

[TPS_MANI_02385]{DRAFT} **`DltLogSink` with `category` `DLT_LOGSINK_DLT`** ⌈`DltLogSink` with `category` `DLT_LOGSINK_DLT` provides means to forward logging information onto on a specific VLAN configured by `endpointConfiguration` via a Dlt log channel defined by `logChannelId`.⌋*(RS_MANI_00023)*

[TPS_MANI_02386]{DRAFT} **`DltLogSink` with `category` `DLT_LOGSINK_FILE`** ⌈`DltLogSink` with `category` `DLT_LOGSINK_FILE` provides means to forward logging information to a file in `path` of a file system.⌋*(RS_MANI_00023)*

[TPS_MANI_02387]{DRAFT} **`DltLogSink` with `category` `DLT_LOGSINK_CONSOLE`** ⌈`DltLogSink` with `category` `DLT_LOGSINK_CONSOLE` provides means to forward logging information to the console.⌋*(RS_MANI_00023)*

[TPS_MANI_03325]{DRAFT} **`DltLogSink` with `category` `DLT_LOGSINK_ARTI`** ⌈`DltLogSink` with `category` `DLT_LOGSINK_ARTI` provides means to forward all modeled messages without filtering to ARTI trace API.⌋*(RS_MANI_00023)*

[TPS_MANI_03163]{DRAFT} **Network configuration for Log and Trace messages** ⌈The output channel on Ethernet for Log and Trace messages is configured with the `PlatformModuleEthernetEndpointConfiguration` that is referenced by the `DltLogSink` in the role `endpointConfiguration`. The attributes `tcpPort` and `udpPort` are used to configure the Transport Protocol (Udp or Tcp) and the used Port

number. The IP Address is configured in the NetworkEndpoint that is referenced by the PlatformModuleEthernetEndpointConfiguration via the EthernetCommunicationConnector.⌋(*RS_MANI_00023*)

[constr_5277]{DRAFT} **applicable DltLogSink categorys vs. DltLogSink attributes** ⌈

Category	Applicable to ...					
	DltLogSink.logChannelId	DltLogSink.endpointConfiguration	DltLogSink.path	DltLogSink.bufferOutput	DltLogSink.nonVerboseMode	DltLogSink.segmentationSupported
DLT_LOGSINK_REMOTE	x					
DLT_LOGSINK_DLT	x	x			x	x
DLT_LOGSINK_FILE			x			
DLT_LOGSINK_CONSOLE				x		
DLT_LOGSINK_ARTI						

This rule shall be imposed **at the time when the creation of the manifest is finished**

⌋()

[constr_5278]{DRAFT} **DltLogSink with category DLT_LOGSINK_REMOTE is only allowed to be referenced by DltLogSinkToPortPrototypeMapping** ⌈DltLogSink with category DLT_LOGSINK_REMOTE shall not be referenced by LogAndTraceInstantiation in the role logSink **at the time when the creation of the manifest is finished**.⌋()

[constr_5279]{DRAFT} **DltLogSink with category DLT_LOGSINK_DLT is only allowed to be referenced by LogAndTraceInstantiation** ⌈DltLogSink with category DLT_LOGSINK_DLT shall not be referenced by DltLogSinkToPortPrototypeMapping in the role dltLogSink **at the time when the creation of the manifest is finished**.⌋()

[constr_5281]{DRAFT} **Existence of DltLogSink.defaultTraceState** ⌈For each DltLogSink, attribute defaultTraceState shall exist **at the time when the creation of the manifest is finished**.⌋()

10.6.2 Relation between design and deployment

The relation between the DltLogSink and applications that are monitored and are producing the logging information is created with the DltLogSinkToPortPrototypeMapping.

[TPS_MANI_02388]{DRAFT} **Semantics of DltLogSinkToPortPrototypeMapping** ⌈Meta-class DltLogSinkToPortPrototypeMapping has the ability to map a specific PortPrototype referenced in the role rPortPrototype or pPortPrototype to a DltLogSink referenced in the role dltLogSink. The dltContext reference to the DltContext that is defined in the Log And Trace Extract defines the relationship to DltMessages that the application produces.

The mapping also comprises a reference to meta-class Process in order to accommodate for the fact that identical combinations of DltLogSink and PortPrototype may or may not apply for a given Process that represents the enclosing Executable at runtime.⌋*(RS_MANI_00023)*

[constr_5282]{DRAFT} **Existence of DltLogSinkToPortPrototypeMapping.process** ⌈Each DltLogSinkToPortPrototypeMapping shall reference a Process in the role process **at the time when the creation of the manifest is finished**.⌋*()*

[constr_5283]{DRAFT} **Existence of DltLogSinkToPortPrototypeMapping.dltLogSink** ⌈Each DltLogSinkToPortPrototypeMapping shall reference at least one DltLogSink in the role dltLogSink **at the time when the creation of the manifest is finished**.⌋*()*

[constr_5284]{DRAFT} **Existence of DltLogSinkToPortPrototypeMapping.dltContext** ⌈Each DltLogSinkToPortPrototypeMapping shall reference a DltContext in the role dltContext **at the time when the creation of the manifest is finished**.⌋*()*

[constr_5285]{DRAFT} **Existence of PortPrototype references in DltLogSinkToPortPrototypeMapping** ⌈Each DltLogSinkToPortPrototypeMapping shall reference exactly one PortPrototype in the role rPortPrototype or pPortPrototype **at the time when the creation of the manifest is finished**.⌋*()*

Class	**DltLogSinkToPortPrototypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::LogAndTrace			
Note	This meta-class maps a PortPrototype to an output sink of a log and trace message. **Tags:** atp.recommendedPackage=DltLogSinkToPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

▽

△

Class	DltLogSinkToPortPrototypeMapping			
dltContext	DltContext	0..1	ref	Assignement of the DltContext that groups log and trace messages that will be transmitted to the DltLogSink.
dltLogSink	DltLogSink	*	ref	Reference to the output sink to which the log or trace message will be transmitted,
dltSessionId	PositiveInteger	0..1	attr	This attribute allows distinguishing log/trace messages from different instances of the same SW-C.
pPortPrototype	PPortPrototype	0..1	iref	Reference to PPortPrototype that is mapped to the DltLog Sink. **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference represents the process required as context for the mapping.
rPortPrototype	RPortPrototype	0..1	iref	Reference to RPortPrototype that is mapped to a DltLog Sink **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 10.59: DltLogSinkToPortPrototypeMapping

[constr_5286]{DRAFT} **Restriction applicable for `DltLogSinkToPortPrototypeMapping.rPortPrototype`** ⌈The reference `DltLogSinkToPortPrototypeMapping.rPortPrototype` shall only be used for a `RPortPrototype` typed by a `LogAndTraceInterface` or by a `ServiceInterface` **at the time when the creation of the manifest is finished**.⌋*()*

[constr_5287]{DRAFT} **Restriction applicable for `DltLogSinkToPortPrototypeMapping.pPortPrototype`** ⌈The reference `DltLogSinkToPortPrototypeMapping.pPortPrototype` shall only be used for a `PPortPrototype` typed by a `ServiceInterface` **at the time when the creation of the manifest is finished**.⌋*()*

With the possibility to map a `DltLogSink` to a `PortPrototype` typed by a `ServiceInterface` it is possible to describe the ara::com communication that is monitored by Log And Trace.

[TPS_MANI_03286]{DRAFT} **Assignment of `DltApplication` to a `Process`** ⌈Meta-class `DltApplicationToProcessMapping` has the ability to map a specific `DltApplication` that defines the `applicationId` used in the Log And Trace Protocol to a `Process`.⌋*(RS_MANI_00023)*

[constr_5292]{DRAFT} **Assigned `dltSessionId` shall be consistent for the same `PortPrototype`** ⌈If several `DltLogSinkToPortPrototypeMapping`s are referencing the same `PortPrototype` in the role `rPortPrototype` or `pPortPrototype` then the value for the `dltSessionId` in all these `DltLogSinkToPortPrototypeMapping`s shall be the same **at the time when the creation of the manifest is finished**.⌋*()*

Class	**DltApplicationToProcessMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::LogAndTrace			
Note	This element assigns a DltApplicationId to a Process. **Tags:** atp.recommendedPackage=DltApplicationToProcessMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dltApplication	DltApplication	0..1	ref	Reference to a DltApplication that defines the application Id
process	Process	0..1	ref	Reference to the process that is assigned to a Log And Trace applicationId.

Table 10.60: DltApplicationToProcessMapping

[constr_5288]{DRAFT} **Existence of `process` reference in `DltApplicationToProcessMapping`** ⌈Each `DltApplicationToProcessMapping` shall reference a `Process` in the role `process` **at the time when the creation of the manifest is finished**.⌋*()*

[constr_5289]{DRAFT} **Existence of `dltApplication` reference in `DltApplicationToProcessMapping`** ⌈Each `DltApplicationToProcessMapping` shall reference a `DltApplication` in the role `dltApplication` **at the time when the creation of the manifest is finished**.⌋*()*

10.7 Network Management configuration

10.7.1 UDP Network Management

[TPS_MANI_03166]{DRAFT} **`Machine`-specific configuration settings for NM module** ⌈The `Machine`-specific configuration settings for Nm are collected in `NmInstantiation`.⌋*(RS_MANI_00023)*

[TPS_MANI_03226]{DRAFT} **Collection of partialNetworks and VLANs in `NmNetworkHandle`** ⌈The `NmNetworkHandle` element is used to describe a collection of `partialNetwork`s and `vlan`s that can be controlled collectively by the State Management.⌋*(RS_MANI_00023)*

The `UdpNmCluster` with all included `UdpNmNode`s is described in the `System` design model. With the reference `NmNode.machine` the relation between the `System` design model and the `NmInstantiation` on a `Machine` is established.

Typically, the `System` design model is provided by an OEM that defines the network configuration and provides all configuration settings that are relevant for a network management cluster to an integrator. The NM configuration options that will typically be set by an Integrator are collected in the `NmInstantiation` element. The Machine Manifest delivery to configure UdpNm consists of both, the `NmInstantiation` settings together with the `UdpNmCluster` and `UdpNmNode` settings.

The NmConfig element is a wrapper that contains all network management specific configuration settings in the System model.

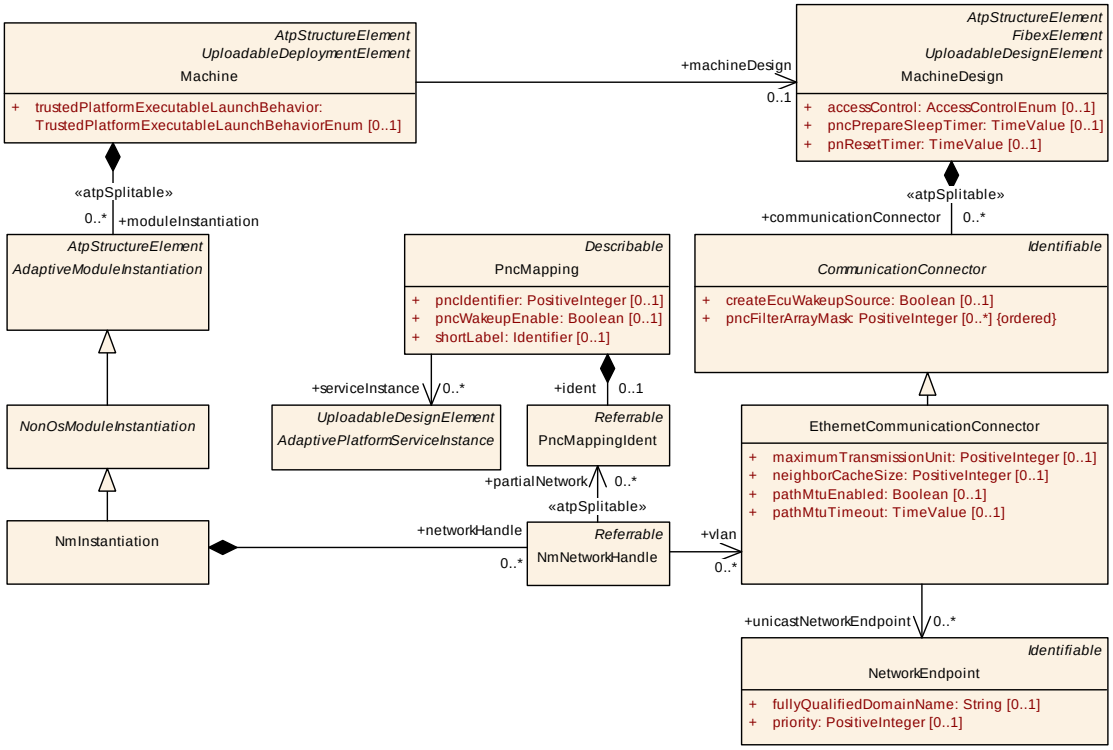


Figure 10.27: Network configuration for Nm

Class	**NmInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
Note	This meta-class defines the attributes for the Nm configuration on a specific machine.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
networkHandle	NmNetworkHandle	*	aggr	Supported NmNetworkHandles used to control Partial Network Clusters/VLANs.

Table 10.61: NmInstantiation

Class	**NmNetworkHandle**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
Note	Group of partialNetworks and/or VLANs that can be controlled collectively.			
Base	*ARObject*, *Referrable*			
Aggregated by	NmInstantiation.networkHandle			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	NmNetworkHandle			
partialNetwork	PncMappingIdent	*	ref	Reference to a Partial Network that is included in the Nm NetworkHandle. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=partialNetwork
vlan	EthernetCommunication Connector	*	ref	Reference to a VLAN that is included in the NmNetwork Handle.

Table 10.62: NmNetworkHandle

Class	PncMappingIdent			
Package	M2::AUTOSARTemplates::SystemTemplate::PncMapping			
Note	This meta-class is created to add the ability to become the target of a reference to the non-Referrable PncMapping.			
Base	*ARObject*, *Referrable*			
Aggregated by	PncMapping.ident			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 10.63: PncMappingIdent

AUTOSAR Adaptive Network Management is based on periodic NM messages, which are received by all `UdpNmNode`s in the `UdpNmCluster` via multicast. Reception of NM packets indicates that sending `UdpNmNode`s want to keep the `UdpNmCluster` awake.

If any node is ready to go to sleep mode, it stops sending NM messages, but as long as NM packets from other `UdpNmNode`s are received, it postpones transition to sleep mode.

[TPS_MANI_03167]{DRAFT} **Network configuration for Nm** ⌈The UDP multicast connection over which Network Management messages are transported is configured with the `UdpNmNetworkConfiguration` that is aggregated by the `UdpNmCluster` in the role `networkConfiguration`. The attribute `udpPort` is used to configure the port number over which the Nm message is transmitted and received. The IP Address is configured either by `ipv4MulticastIpAddress` or `ipv6MulticastIpAddress`.⌋*(RS_MANI_00023)*

Class	UdpNmNetworkConfiguration			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SystemDesign			
Note	This meta-class defines the attributes for the configuration of a UDP port and UDP multicast IP address of the Nm communication on a VLAN.			
Base	*ARObject*			
Aggregated by	UdpNmCluster.networkConfiguration			
Attribute	Type	Mult.	Kind	Note
ipv4MulticastIp Address	Ip4AddressString	0..1	attr	Multicast IPv4 Address to which the message will be transmitted.
ipv6MulticastIp Address	Ip6AddressString	0..1	attr	Multicast IPv6 Address to which the message will be transmitted

▽

△

Class	**UdpNmNetworkConfiguration**			
priority	PositiveInteger	0..1	attr	This attribute defines the VLAN frame priority for messages on the Socket defined by the udpPort and the multicast IP address. Values from 0 (best effort) to 7 (highest) are allowed.
udpPort	PositiveInteger	0..1	attr	This attribute allows to configure a udp port number that is used for reception and transmission of UdpNm messages.

Table 10.64: UdpNmNetworkConfiguration

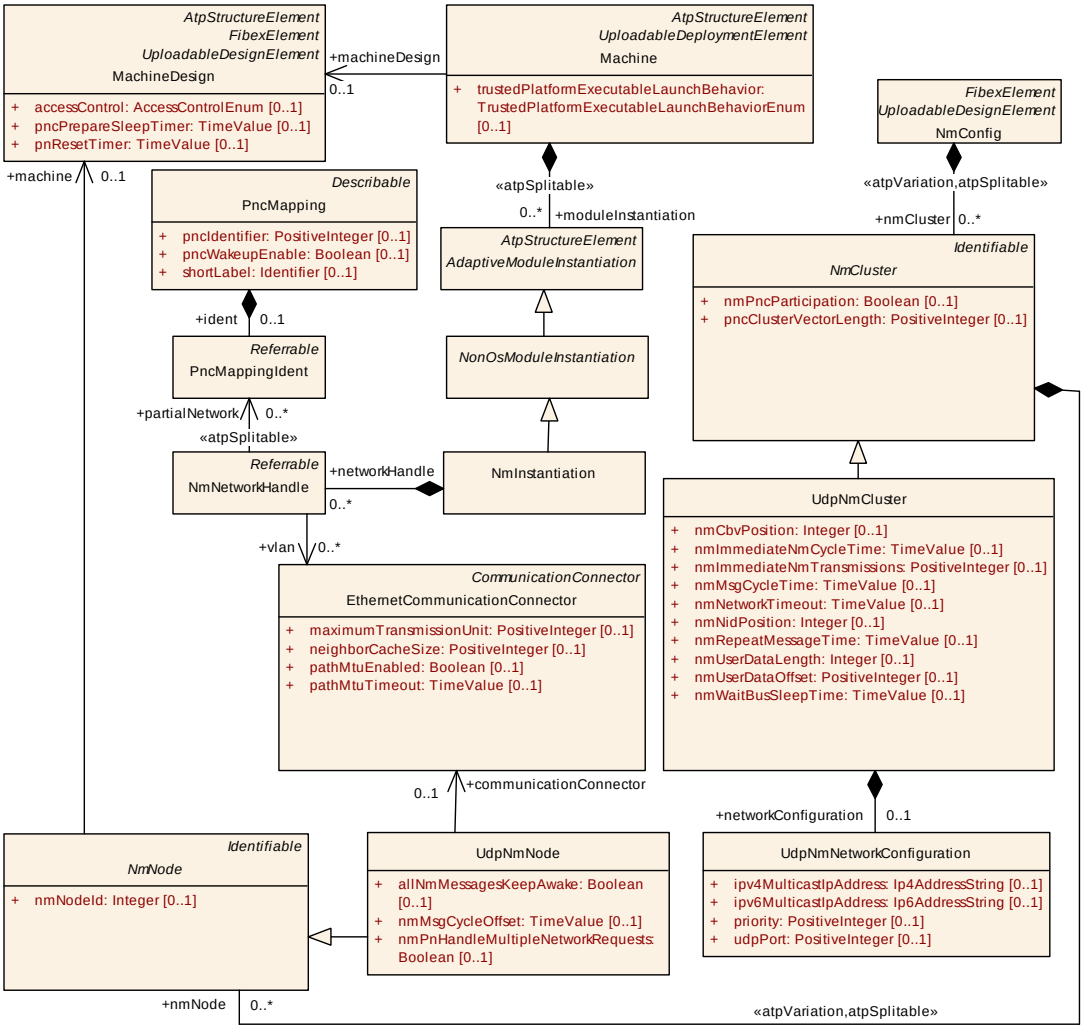


Figure 10.28: NM Cluster configuration

[TPS_MANI_03279]{DRAFT} **Priority of Nm messages** ⌈The attribute `priority` in the `UdpNmNetworkConfiguration` can be used to assign a VLAN priority to Network Management messages that are transmitted over the Socket defined by the `UdpNmNetworkConfiguration.udpPort` and `ipv4MulticastIpAddress` or `ipv6MulticastIpAddress`.⌋*(RS_MANI_00023)*

[constr_3419]{DRAFT} **Allowed usage of `UdpNmNetworkConfiguration` attributes** ⌈The `UdpNmNetworkConfiguration` that is aggregated by `UdpNmCluster` in the role `networkConfiguration` shall have either

- `ipv4MulticastIpAddress` or
- `ipv6MulticastIpAddress`.

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

Class	**NmConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	Contains the all configuration elements for AUTOSAR Nm. **Tags:** atp.recommendedPackage=NmConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
nmCluster	NmCluster	*	aggr	Collection of NM Clusters atpVariation: Derived, because cluster can be variable. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=nmCluster.shortName, nmCluster.variation Point.shortLabel vh.latestBindingTime=postBuild

Table 10.65: NmConfig

Class	***NmCluster*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	Set of NM nodes coordinated with use of the NM algorithm.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	CanNmCluster, FlexrayNmCluster, UdpNmCluster			
Aggregated by	NmConfig.nmCluster			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Cluster	CommunicationCluster	0..1	ref	Association to a CommunicationCluster in the topology description.
nmNode	NmNode	*	aggr	Collection of NmNodes of the NmCluster. atpVariation: Derived, because NmNode can be variable. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=nmNode.shortName, nmNode.variation Point.shortLabel vh.latestBindingTime=postBuild
nmPnc Participation	Boolean	0..1	attr	Defines whether this NmCluster contributes to the partial network mechanism.

▽

△

Class	***NmCluster*** (abstract)			
pncCluster VectorLength	PositiveInteger	0..1	attr	Optionally defines the length of the PNC Vector per CommunicationCluster (and VLAN in case of UdpNm). If not defined then System.pncVectorLength applies. Should only make the PNC Vector shorter (or same length as defined in System.pncVectorLength).

Table 10.66: NmCluster

Class	**UdpNmCluster**			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	Udp specific NmCluster attributes			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *NmCluster*, *Referrable*			
Aggregated by	NmConfig.nmCluster			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
network Configuration	UdpNmNetwork Configuration	0..1	aggr	Configuration of a UDP port and UDP multicast IP address of the Nm communication on a VLAN.
nmCbvPosition	Integer	0..1	attr	Defines the position of the control bit vector within the Nm Pdu (Byte position). If this attribute is not configured, the Control Bit Vector is not used.
nmImmediate NmCycleTime	TimeValue	0..1	attr	Defines the immediate NmPdu cycle time in seconds which is used for nmImmediateNmTransmissions NmPdu transmissions. This attribute is only valid if nmImmediate NmTransmissions is greater one.
nmImmediate Nm Transmissions	PositiveInteger	0..1	attr	Defines the number of immediate NmPdus which shall be transmitted. If the value is zero no immediate NmPdus are transmitted. The cycle time of immediate NmPdus is defined by nmImmediateNmCycleTime.
nmMsgCycle Time	TimeValue	0..1	attr	Period of a NmPdu in seconds. It determines the periodic rate in the periodic transmission mode with bus load reduction and is the basis for transmit scheduling in the periodic transmission mode without bus load reduction.
nmNetwork Timeout	TimeValue	0..1	attr	Network Timeout for NmPdus in seconds. It denotes the time how long the UdpNm shall stay in the Network Mode before transition into Prepare Bus-Sleep Mode shall take place.
nmNidPosition	Integer	0..1	attr	Defines the byte position of the source node identifier within the NmPdu. If this attribute is not configured, the Node Identification is not used.
nmRepeat MessageTime	TimeValue	0..1	attr	Timeout for Repeat Message State in seconds. Defines the time how long the NM shall stay in the Repeat Message State.
nmUserData Length	Integer	0..1	attr	Defines the length in bytes of the user data contained in the Nm message. User data excludes the PNC bit vector.
nmUserData Offset	PositiveInteger	0..1	attr	Specifies the offset (in bytes) of the user data information in the NM message. User data excludes the PNC bit vector.
nmWaitBus SleepTime	TimeValue	0..1	attr	Timeout for bus calm down phase in seconds. It denotes the time how long the CanNm shall stay in the Prepare Bus-Sleep Mode before transition into Bus-Sleep Mode shall take place.
vlan	EthernetPhysical Channel	0..1	ref	Reference to the vlan (represented by the Ethernet PhysicalChannel) this UdpNmCluster shall apply to.

Table 10.67: UdpNmCluster

Class	***NmNode*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	The linking of NmEcus to NmClusters is realized via the NmNodes.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	CanNmNode, FlexrayNmNode, UdpNmNode			
Aggregated by	*NmCluster*.nmNode			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
machine	MachineDesign	0..1	ref	Reference to the machine that contains the NmNode.
nmNodeId	Integer	0..1	attr	Node identifier of local NmNode. Shall be unique in the NmCluster.

Table 10.68: NmNode

Class	**UdpNmNode**			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	Udp specific NM Node attributes.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *NmNode*, *Referrable*			
Aggregated by	*NmCluster*.nmNode			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
allNmMessages KeepAwake	Boolean	0..1	attr	Specifies if Nm drops irrelevant NM PDUs. false: Only NM PDUs with a Partial Network Information Bit (PNI) = true and containing a Partial Network request for this ECU trigger the standard RX indication handling and thus keep the ECU awake true: Every NM PDU triggers the standard RX indication handling and keeps the ECU awake
communication Connector	EthernetCommunication Connector	0..1	ref	Reference to the CommunicationConnector that represents the UdpNmNode in the topology description.
nmMsgCycle Offset	TimeValue	0..1	attr	Node specific time offset in the periodic transmission node. It determines the start delay of the transmission. Specified in seconds.
nmPnHandle MultipleNetwork Requests	Boolean	0..1	attr	Specifies if NM performs an additional transition from Network Mode to Repeat Message State (true) or not (false).

Table 10.69: UdpNmNode

10.7.2 Relation between Application Software and Network Handles

A `PPortPrototype` typed by a `NetworkManagementPortInterface` can make requests towards a status change of a network handle. But it is necessary to indicate which network handle is associated with which `PPortPrototype`, This relation can be indicated by means of the `NetworkHandlePortMapping`.

[constr_10441]{DRAFT} **Restriction for `NetworkHandlePortMapping.pPortPrototypeInExecutable`** ⌈A `PPortPrototype` referenced in the role `NetworkHandlePortMapping.pPortPrototypeInExecutable` shall only be typed by a `NetworkManagementPortInterface` **at the time before the generation of the ara API starts**.⌋()

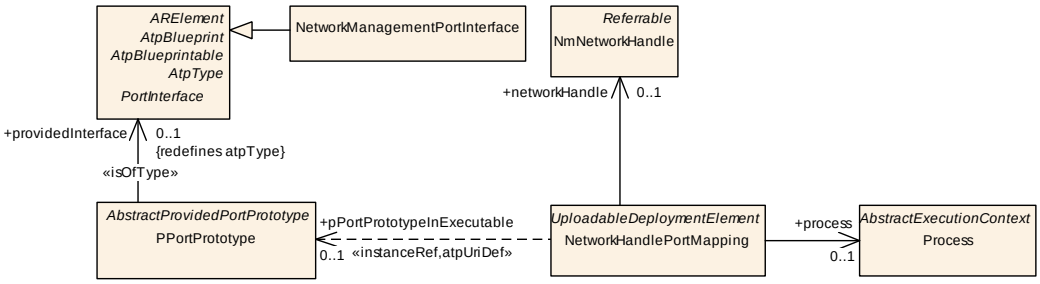


Figure 10.29: Modeling of the mapping between application software and network handle

Class	**NetworkHandlePortMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModuleImplementation			
Note	This class is used to associate a PortPrototype with a network handle in order to control the network handle from the PortPrototype **Tags:** atp.recommendedPackage=NetworkHandleMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
networkHandle	NmNetworkHandle	0..1	ref	This reference identifies the applicable network handle.
pPortPrototypeInExecutable	PPortPrototype	0..1	iref	This reference identifies the applicable PortPrototype in the context of an enclosing Executable. **Stereotypes:** atpUriDef **InstanceRef implemented by:** PPortPrototypeInExecutableInstanceRef
process	Process	0..1	ref	This reference identifies the process applicable for the state management

Table 10.70: NetworkHandlePortMapping

10.7.3 Relation between Function Group states and Network Handles

[TPS_MANI_01343]{DRAFT} **Relation between `Function Group State`s and `NmNetworkHandle`** ⌈An essential part of the network management on the *AUTOSAR adaptive platform* is the understanding which `Function Group State` requires network access. This relation can be expressed by means of meta-class `NmHandleToFunctionGroupStateMapping`.⌋ *(RS_MANI_00023)*

The relation between `Function Group State`s and network handles is in principle bi-directional, i.e. the following use cases exist for the creation of an `NmHandleToFunctionGroupStateMapping`:

- The application software wants to indicate which `Function Group State`s require network access. This means that attribute `mappingDirection` of the first `NmHandleToFunctionGroupStateMapping` shall be set to `NmHandleMappingDirectionEnum.functionGroupStateToNmHandle`.

- The application software wants to indicate which Function Group State shall be set if the network goes down. This means that attribute mappingDirection shall be set to NmHandleMappingDirectionEnum.nmHandleInactiveToFunctionGroupState.
- The application software wants to indicate which Function Group State shall be set if the network goes up. This means that attribute mappingDirection shall be set to NmHandleMappingDirectionEnum.nmHandleActiveToFunctionGroupState.

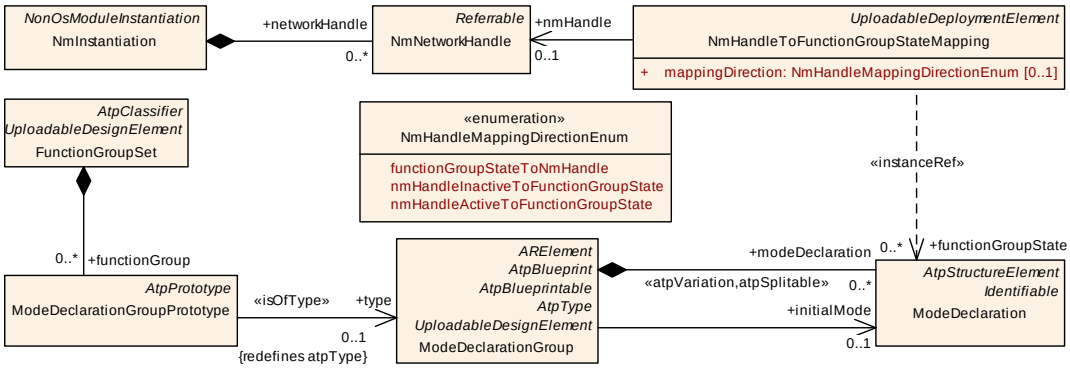


Figure 10.30: Mapping between NmNetworkHandle and Function Group States

Class	**NmHandleToFunctionGroupStateMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class represents the ability to create a mapping between an NmNetworkHandle and a collection of function group states. This way, the impact of function groups on the network management can be specified. **Tags:** atp.recommendedPackage=NmHandleToFunctionGroupStateMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
functionGroup State	ModeDeclaration	*	iref	This reference identifies the collection of function group states in the context of the mapping. **InstanceRef implemented by:** FunctionGroupStateIn FunctionGroupSetInstanceRef
mapping Direction	NmHandleMapping DirectionEnum	0..1	attr	This attribute describes the direction of the mapping.
nmHandle	NmNetworkHandle	0..1	ref	This reference identifies the applicable NmNetwork Handle in the context of the mapping.

Table 10.71: NmHandleToFunctionGroupStateMapping

Enumeration	**NmHandleMappingDirectionEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation
Note	This enumeration provides direction values for the mapping of NM handles to function group states.

▽

△

Enumeration	**NmHandleMappingDirectionEnum**
Aggregated by	NmHandleToFunctionGroupStateMapping.mappingDirection
Literal	***Description***
functionGroupState ToNmHandle	The purpose of the mapping is to indicate which function group state requires network access. **Tags:** atp.EnumerationLiteralIndex=0
nmHandleActiveTo FunctionGroupState	The purpose of the mapping is to indicate that the function group shall be switched to a given state if the network handle becomes active. **Tags:** atp.EnumerationLiteralIndex=2
nmHandleInactive ToFunctionGroup State	The purpose of the mapping is to indicate that the function group shall be switched to a given state if the network handle becomes inactive. **Tags:** atp.EnumerationLiteralIndex=1

Table 10.72: NmHandleMappingDirectionEnum

[TPS_MANI_01365]{DRAFT} **Semantics of attribute `NmHandleToFunctionGroupStateMapping.mappingDirection` set to `functionGroupStateToNmHandle`** ⌈If the attribute `NmHandleToFunctionGroupStateMapping.mappingDirection` is set to the value `NmHandleMappingDirectionEnum.functionGroupStateToNmHandle`, then the semantics of the `NmHandleToFunctionGroupStateMapping` is that the referenced `nmHandle` is requested to become active if the `Function Group` to which the referenced `functionGroupState` belongs to is switched to the referenced `Function Group State`, formalized by meta-class `ModeDeclaration`.⌋*(RS_MANI_00023)*

Please note that the multiplicity of the reference `NmHandleToFunctionGroupStateMapping.functionGroupState` allows for the creation of a "shortcut" such that several `Function Group State`s can simultaneously be indicated as requiring communication in the context of a single `NmHandleToFunctionGroupStateMapping`, as depicted in Figure 10.31.

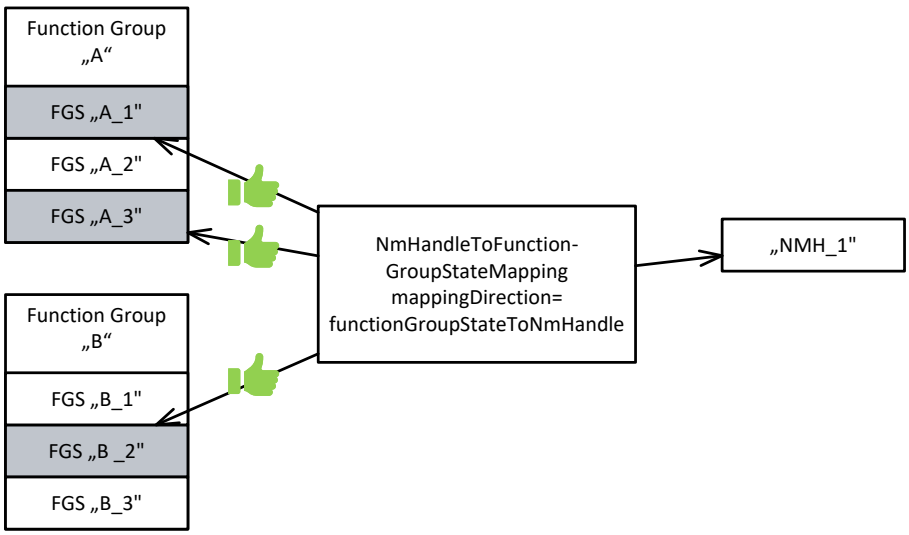


Figure 10.31: Mapping from `Function Group` to NM handle

[TPS_MANI_01366]{DRAFT} **Semantics of attribute NmHandleToFunctionGroupStateMapping.mappingDirection set to nmHandleInactiveToFunctionGroupState** ⌈If the attribute NmHandleToFunctionGroupStateMapping.mappingDirection is set to the value NmHandleMappingDirectionEnum.nmHandleInactiveToFunctionGroupState, then the semantics of the NmHandleToFunctionGroupStateMapping is that if the referenced nmHandle becomes **inactive**, the Function Group to which the referenced functionGroupState belongs to is switched to the referenced Function Group State, formalized by meta-class ModeDeclaration.⌋*(RS_MANI_00023)*

[TPS_MANI_01367]{DRAFT} **Semantics of attribute NmHandleToFunctionGroupStateMapping.mappingDirection set to nmHandleActiveToFunctionGroupState** ⌈If the attribute NmHandleToFunctionGroupStateMapping.mappingDirection is set to the value NmHandleMappingDirectionEnum.nmHandleActiveToFunctionGroupState, then the semantics of the NmHandleToFunctionGroupStateMapping is that if the referenced nmHandle becomes **active**, the function group to which the referenced functionGroupState belongs to is switched to the referenced Function Group State, formalized by meta-class ModeDeclaration.⌋*(RS_MANI_00023)*

It is important to note that the mapping from NM Handle to Function Group State can only identify on Function Group State per Function Group because it is obviously impossible to activate two Function Group States of the same Function Group at the same time.

Nevertheless, if an NM handle becomes active/inactive, it is very likely that this event has an impact on several SoftwareClusters. And because Function Groups cannot be shared among SoftwareClusters, it is necessary to be able to establish a 1:n communication pattern for this case under a precondition described by [constr_10101] and sketched by Figure 10.32.

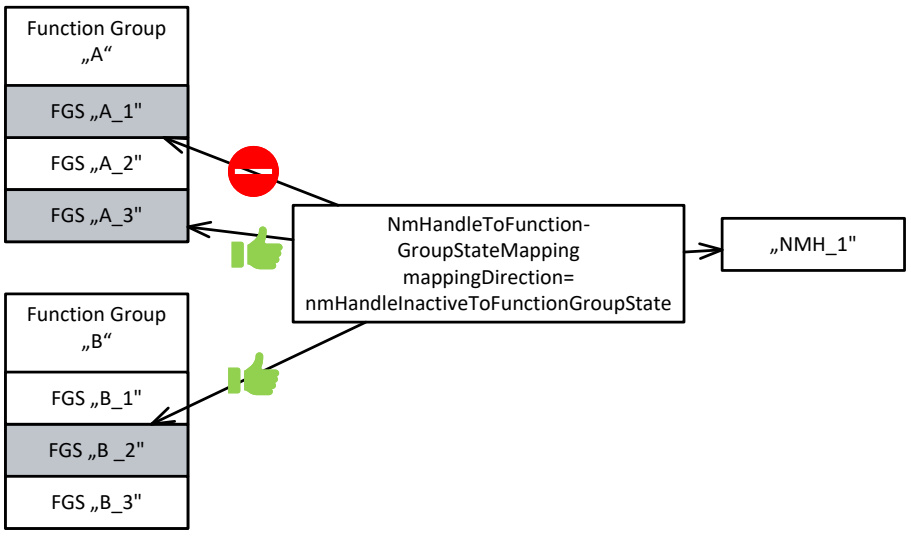


Figure 10.32: Mapping from NM handle to Function Group

[constr_10101]{DRAFT} **Attribute NmHandleToFunctionGroupStateMapping.mappingDirection is set to nmHandleActiveToFunctionGroupState or nmHandleInactiveToFunctionGroupState** ⌈If the value of attribute NmHandleToFunctionGroupStateMapping.mappingDirection is set to the value NmHandleMappingDirectionEnum.nmHandleActiveToFunctionGroupState or NmHandleMappingDirectionEnum.nmHandleInactiveToFunctionGroupState, then the reference NmHandleToFunctionGroupStateMapping.functionGroupState shall not refer to two (or more) ModeDeclarations **of the same** ModeDeclarationGroup **at the time when the creation of the manifest is finished**.⌋*()*

10.7.3.1 UdpNm configuration constraints

Please note that the Classic Platform and the Adaptive Platform are using the same model for configuration of UdpNm. Some Classic Platform features like the NmCoordinator and UdpNmClusterCoupling are not supported in Adaptive Autosar.

But the TPS_SystemTemplate [18] contains a more detailed description of UdpNmClusters and UdpNmNodes and defines modeling constraints that are also valid for the Adaptive Platform.

The following constraints of the TPS_SystemTemplate [18] shall be considered if a UdpNmCluster with UdpNmNodes is described:

- [constr_3078]
- [constr_3079]
- [constr_3080]
- [constr_5223]
- [constr_5224]
- [constr_5225]
- [constr_5226]

Note that in the context of the *AUTOSAR adaptive platform*, the imposition time for these constraints shall be **at the time when the system design is complete**.

In addition, the following Adaptive Platform specific constraints are valid:

[constr_5227]{DRAFT} **Mandatory elements of UdpNmCluster** ⌈The following attributes shall always be defined for the UdpNmCluster:

- nmMsgCycleTime
- nmNetworkTimeout
- nmRepeatMessageTime

- nmWaitBusSleepTime
- communicationCluster

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

[constr_5228]{DRAFT} **Partial Networking timing constraint** ⌈For Partial Networking the following timing constraints shall be ensured: (MachineDesign.pnResetTimer + MachineDesign.pncPrepareSleepTimer) < UdpNmCluster.nmNetworkTimeout.

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

[constr_10098]{DRAFT} **Relation of MachineDesign.pnResetTimer and UdpNmCluster.nmMsgCycleTime** ⌈For the configuration of the partial networking timing, the following condition shall be ensured: MachineDesign.pnResetTimer > UdpNmCluster.nmMsgCycleTime.

This rule shall be imposed **at the time when the creation of the manifest is finished**,⌋()

10.7.4 CAN XL Network Management

The network management for CAN XL is modeled in a very similar way as the UDP NM. Please find details in Figure 10.33.

Although the CAN XL is entirely used to tunnel Ethernet communication, it still needs to be able to participate in the regular network management of the CAN cluster.

For this purpose, a relation between a EthernetCommunicationConnector and a CanNmNode that represents the local Machine in the CAN NM networking needs to be created.

The CAN ID used to transmit NM frames is configured in CanNmNode.canXlNmProps.canNmTxCanId.

The CAN IDs of received NM frames are configured by the respective attributes of the CanNmNodes connected to the same CanNmCluster.

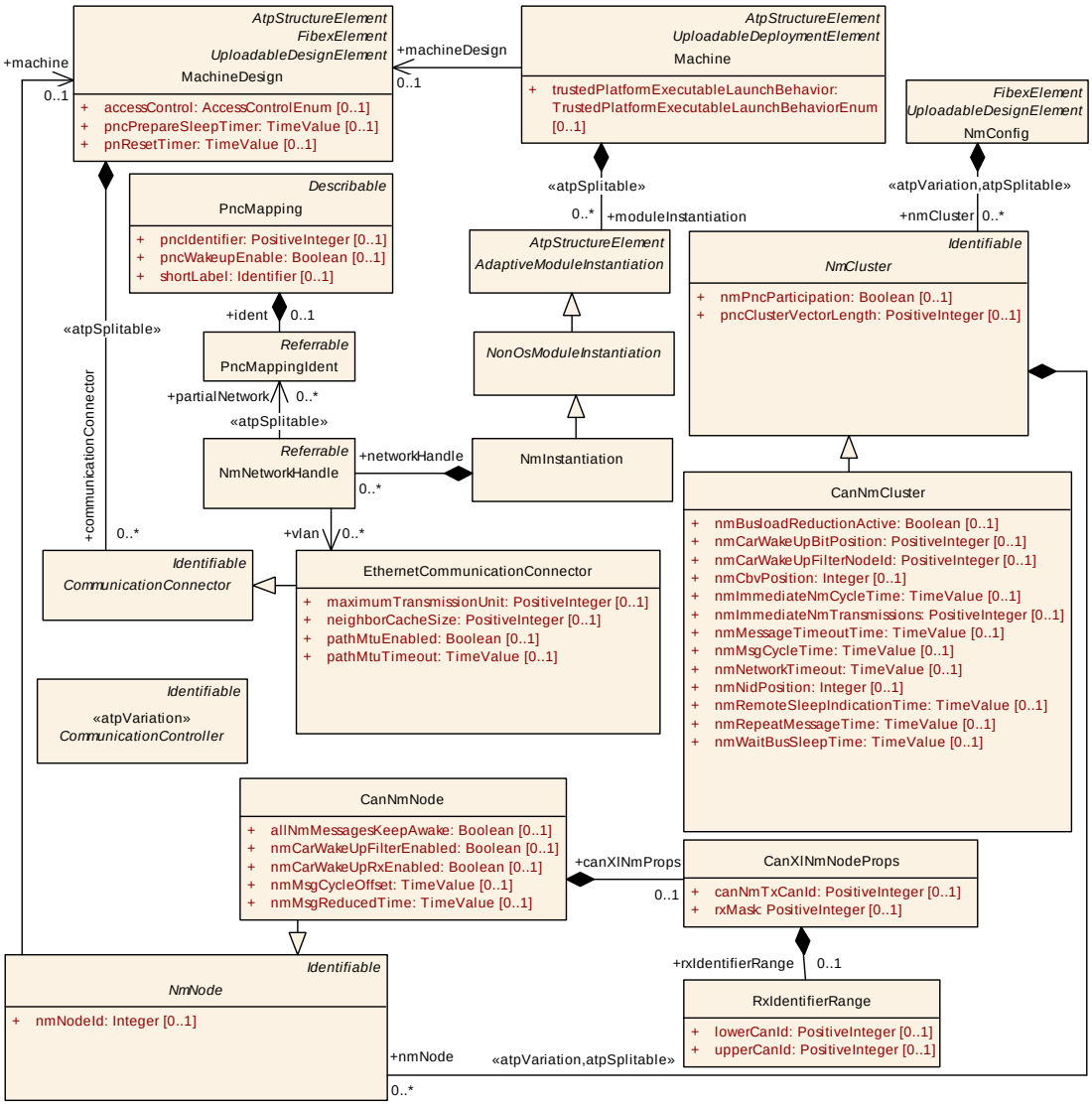


Figure 10.33: NM configuration for CAN XL

Class	**CanNmNode**			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	CAN specific NM Node attributes.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *NmNode*, *Referrable*			
Aggregated by	*NmCluster*.nmNode			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allNmMessages KeepAwake	Boolean	0..1	attr	Specifies if Nm drops irrelevant NM PDUs. false: Only NM PDUs with a Partial Network Information Bit (PNI) = true and containing a Partial Network request for this ECU trigger the standard RX indication handling and thus keep the ECU awake true: Every NM PDU triggers the standard RX indication handling and keeps the ECU awake

△

Class	**CanNmNode**			
canXlNmProps	CanXlNmNodeProps	0..1	aggr	This aggregation contributes the CAN IDs for the execution of CAN NM on the adaptive platform.
nmCarWakeUp FilterEnabled	Boolean	0..1	attr	If this attribute is set to true the CareWakeUp filtering is supported.
nmCarWakeUp RxEnabled	Boolean	0..1	attr	If set to true this attribute enables the support of CarWake Up bit evaluation in received NmPdus.
nmMsgCycle Offset	TimeValue	0..1	attr	Node specific time offset in the periodic transmission node. It determines the start delay of the transmission. Specified in seconds.
nmMsg ReducedTime	TimeValue	0..1	attr	Node specific bus cycle time in the periodic transmission mode with bus load reduction. Specified in seconds.

Table 10.73: CanNmNode

Class	**CanXlNmNodeProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	Definition of CAN attributes in the context of CanXL usage on the adaptive platform.			
Base	*ARObject*			
Aggregated by	CanNmNode.canXlNmProps			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
canNmTxCanId	PositiveInteger	0..1	attr	This attribute is used to define the identifier the CAN NM frame shall use on the CAN network.
rxIdentifier Range	RxIdentifierRange	0..1	aggr	Optional definition of a CanId range.
rxMask	PositiveInteger	0..1	attr	Identifier mask which denotes the relevant bits in the CAN Identifier. Together with the identifier, this parameter defines a CAN identifier range.

Table 10.74: CanXlNmNodeProps

Class	**RxIdentifierRange**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Can::CanCommunication			
Note	Optional definition of a CanId range to reduce the effort of specifying every possible FrameTriggering within the defined Id range during reception. All frames received within a range are mapped to the same Pdu that is passed to a upper layer module (e.g. Nm, CDD, PduR).			
Base	*ARObject*			
Aggregated by	CanFrameTriggering.rxIdentifierRange, CanXlNmNodeProps.rxIdentifierRange, IEEE1722TpAcfCanPart. canIdentifierRange			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
lowerCanId	PositiveInteger	0..1	attr	This attribute can be used together with the upperCanId attribute to define a range of CanIds.
upperCanId	PositiveInteger	0..1	attr	This attribute can be used together with the lowerCanId attribute to define a range of CanIds.

Table 10.75: RxIdentifierRange

Class	**CanNmCluster**			
Package	M2::AUTOSARTemplates::SystemTemplate::NetworkManagement			
Note	Can specific NmCluster attributes			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *NmCluster*, *Referrable*			
Aggregated by	NmConfig.nmCluster			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
nmBusload ReductionActive	Boolean	0..1	attr	It determines if bus load reduction for the respective Can Nm channel is active or not.
nmCarWakeUp BitPosition	PositiveInteger	0..1	attr	Specifies the bit position of the CarWakeUp within the Nm Pdu.
nmCarWakeUp FilterNodeId	PositiveInteger	0..1	attr	Source node identifier for CarWakeUp filtering.
nmCbvPosition	Integer	0..1	attr	Defines the position of the control bit vector within the Nm Pdu (Byte position). If this attribute is not configured, the Control Bit Vector is not used.
nmImmediate NmCycleTime	TimeValue	0..1	attr	Defines the immediate NmPdu cycle time in seconds which is used for nmImmediateNmTransmissions NmPdu transmissions. This parameter is only valid if CanNm ImmediateNmTransmissions is greater one.
nmImmediate Nm Transmissions	PositiveInteger	0..1	attr	Defines the number of immediate NmPdus which shall be transmitted. If the value is zero no immediate NmPdus are transmitted. The cycle time of immediate NmPdus is defined by nmImmediateNmCycleTime.
nmMessage TimeoutTime	TimeValue	0..1	attr	Timeout of an NmPdu in seconds. It determines how long the NM shall wait with notification of transmission failure while communication errors occur on the bus.
nmMsgCycle Time	TimeValue	0..1	attr	Period of a NmPdu in seconds. It determines the periodic rate in the periodic transmission mode with bus load reduction and is the basis for transmit scheduling in the periodic transmission mode without bus load reduction.
nmNetwork Timeout	TimeValue	0..1	attr	Network Timeout for NmPdus in seconds It denotes the time how long the CanNm shall stay in the Network Mode before transition into Prepare Bus-Sleep Mode shall take place.
nmNidPosition	Integer	0..1	attr	Defines the byte position of the source node identifier within the NmPdu. If this attribute is not configured, the Node Identification is not used.
nmRemote SleepIndication Time	TimeValue	0..1	attr	Timeout for Remote Sleep Indication in seconds. It defines the time how long it shall take to recognize that all other nodes are ready to sleep.
nmRepeat MessageTime	TimeValue	0..1	attr	Timeout for Repeat Message State in seconds. Defines the time how long the NM shall stay in the Repeat Message State.
nmWaitBus SleepTime	TimeValue	0..1	attr	Timeout for bus calm down phase in seconds. It denotes the time how long the CanNm shall stay in the Prepare Bus-Sleep Mode before transition into Bus-Sleep Mode shall take place.

Table 10.76: CanNmCluster

10.8 Update and Configuration Management

10.8.1 Overview

[TPS_MANI_01226]{DRAFT} **Machine-specific configuration settings for the UCM module** ⌈The Machine-specific configuration settings for UCM are collected in abstract meta-class `UcmModuleInstantiation` and its two concrete specializations:

- `UcmMasterModuleInstantiation` represents the configuration of a *UCM Master*.
- `UcmSubordinateModuleInstantiation` represents the configuration of a *UCM Subordinate*.

⌋*(RS_MANI_00023)*

Please find more information about meta-class `UcmMasterModuleInstantiation` in section 10.8.3 and about `UcmSubordinateModuleInstantiation` in section 10.8.4.

Class	***UcmModuleInstantiation*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Ucm			
Note	This meta-class represents the ability to define the deployment of a UCM instantiation.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Subclasses	UcmMasterModuleInstantiation, UcmSubordinateModuleInstantiation			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
identifier	String	0..1	attr	This represents the identification of a UCM.
maxBlockSize	PositiveInteger	0..1	attr	This attribute denotes the maximum block size (unit: bytes) used in the UCM implementation.
version	StrongRevisionLabel String	0..1	attr	This attribute defines the software version of the UCM on this platform. Note that the definition of the version is required if the ability of the SoftwarePackage to require a minimum version of the UCM is utilized.

Table 10.77: UcmModuleInstantiation

[constr_10428]{DRAFT} **Existence of attribute `UcmModuleInstantiation.identifier` in subclasses** ⌈The Attribute `identifier` shall **not** exist in a `UcmMasterModuleInstantiation` **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10200]{DRAFT} **Multiplicity of attribute `UcmModuleInstantiation.identifier`** ⌈For each `UcmModuleInstantiation`, the attribute `identifier` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

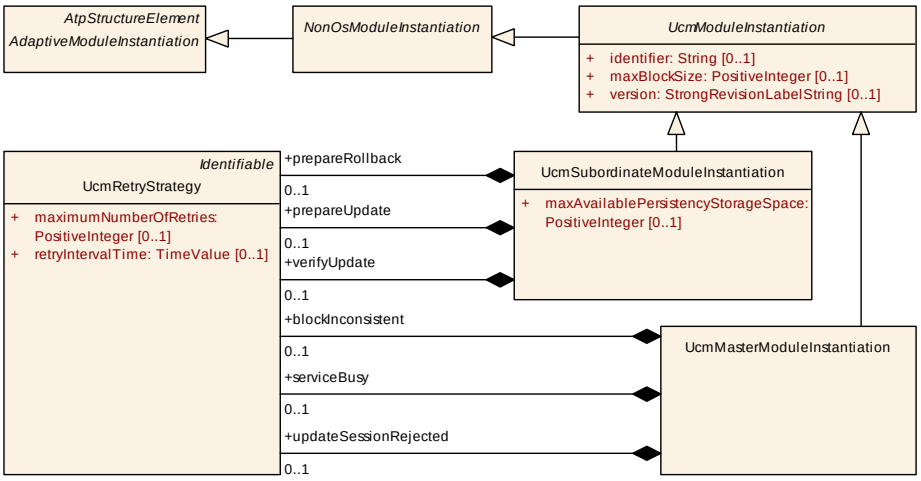


Figure 10.34: Modeling of `UcmModuleInstantiation`

[TPS_MANI_01227]{DRAFT} **Semantics of attribute `UcmModuleInstantiation.identifier`** ⌈Attribute `UcmModuleInstantiation.identifier` shall be used to identify a specific UCM on a specific `Machine` during a service discovery run by a master UCM or VUM.⌋ *(RS_MANI_00023)*

The usage of attribute `UcmModuleInstantiation.identifier` is documented in Figure 10.35. The master UCM or VUM acts as a client in a service discovery that is configured to search for *any* server.

The individual UCMs offer their service and then the master UCM as the client calls a specific `method` in the server's `ServiceInterface` to reveal the `identifier` of each server.

In the case of this example there are three slave UCMs with `identifier` set to 1, 2, and 3. The master UCM instantiates a proxy for each of the slave UCMs such that the value of the respective `identifier` can be retrieved from the proxy in order to be able to communicate with a specific slave UCM.

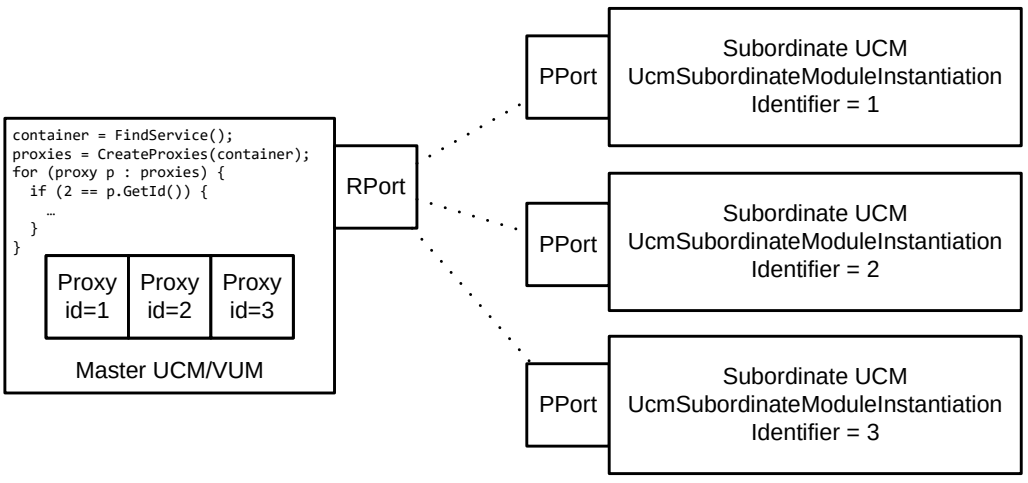


Figure 10.35: Identification of subordinate UCM modules by the master UCM

The master UCM or VUM can then instantiate proxies for each service offer and programatically access the respective server going forward.

[constr_1691]{DRAFT} **UcmModuleInstantiation.identifier shall be unique** ⌈The value of attribute UcmModuleInstantiation.identifier shall be unique for each Machine in a given vehicle **at the time when the creation of the manifest is finished**.⌋*()*

10.8.2 Retry Strategy

[TPS_MANI_01375]{DRAFT} **Semantics of meta-class UcmRetryStrategy** ⌈Both the UcmMasterModuleInstantiation and the UcmSubordinateModuleInstantiation have the ability to retry certain activities if these activities don't succeed. The configuration of such retry cycles is done by means of meta-class UcmRetryStrategy.⌋*(RS_MANI_00023)*

Class	**UcmRetryStrategy**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Ucm			
Note	This meta-class describes the configuration of the retry strategy for a sub-class of UcmModule Implementation.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	UcmMasterModuleInstantiation.blockInconsistent, UcmMasterModuleInstantiation.serviceBusy, UcmMasterModuleInstantiation.updateSessionRejected, UcmSubordinateModuleInstantiation.prepareRollback, UcmSubordinateModuleInstantiation.prepareUpdate, UcmSubordinateModuleInstantiation.verifyUpdate			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
maximumNumberOfRetries	PositiveInteger	0..1	attr	This attribute defines the maximum number of time the UCM module instantiation shall attempt a retry.
retryIntervalTime	TimeValue	0..1	attr	This attribute defines the time (in seconds) between two retry attempts.

Table 10.78: UcmRetryStrategy

[constr_10105]{DRAFT} **Existence of UcmRetryStrategy.maximumNumberOfRetries** ⌈For each UcmRetryStrategy, attribute maximumNumberOfRetries shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10106]{DRAFT} **Existence of UcmRetryStrategy.retryIntervalTime** ⌈For each UcmRetryStrategy, attribute retryIntervalTime shall exist **at the time when the creation of the manifest is finished**.⌋*()*

10.8.3 UCM Master Module Instantiation

[TPS_MANI_01376]{DRAFT} **Semantics of meta-class UcmMasterModuleInstantiation** ⌈Meta-class UcmMasterModuleInstantiation is used to configure the deployment of a *UCM Master* and the various retry strategies associated with the *UCM Master*.⌋*(RS_MANI_00023)*

Class	UcmMasterModuleInstantiation			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Ucm			
Note	This meta-class represents the ability to define the deployment of a UCM Master instantiation.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*, *UcmModuleInstantiation*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
block Inconsistent	UcmRetryStrategy	0..1	aggr	This attribute defines the retry strategy of the UCM Master for the case that the block is inconsistent.
serviceBusy	UcmRetryStrategy	0..1	aggr	This attribute defines the retry strategy of the UCM Master for the case that the service is busy.
updateSession Rejected	UcmRetryStrategy	0..1	aggr	This attribute defines the retry strategy of the UcmMaster for the case that the update session is rejected.

Table 10.79: UcmMasterModuleInstantiation

[constr_10107]{DRAFT} **Existence of the attribute `UcmMasterModuleInstantiation.blockInconsistent`** ⌈The attribute `UcmMasterModuleInstantiation.blockInconsistent` shall **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10108]{DRAFT} **Existence of the attribute `UcmMasterModuleInstantiation.serviceBusy`** ⌈The attribute `UcmMasterModuleInstantiation.serviceBusy` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10109]{DRAFT} **Existence of the attribute `UcmMasterModuleInstantiation.updateSessionRejected`** ⌈The attribute `UcmMasterModuleInstantiation.updateSessionRejected` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

10.8.4 UCM Subordinate Module Instantiation

[TPS_MANI_01377]{DRAFT} **Semantics of meta-class `UcmSubordinateModuleInstantiation`** ⌈Meta-class `UcmSubordinateModuleInstantiation` is used to configure the deployment of a *UCM Subordinate* and the various retry strategies associated with the *UCM Subordinate*.⌋*(RS_MANI_00023)*

Class	UcmSubordinateModuleInstantiation			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Ucm			
Note	This meta-class represents the ability to define the deployment of a UCM Subordinate instantiation.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*, *UcmModuleInstantiation*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**UcmSubordinateModuleInstantiation**			
maxAvailable Persistency StorageSpace	PositiveInteger	0..1	attr	This attribute names the maximum amount of space available for persistent data handled by the Persistency of installed packages. The UCM needs to figure out from traversing the minimum storage requirement from existing PersistencyDeployments whether specific packages can be installed from the perspective of available storage space. Note that the minimum storage requirement of PersistencyDeployment needs to include space for the handling of the storage, which shall be calculated by the tooling that creates the manifest information inside the package.
prepareRollback	UcmRetryStrategy	0..1	aggr	This attribute identifies the configuration of prepare rollback retries initiated by the Ucm Subordinate.
prepareUpdate	UcmRetryStrategy	0..1	aggr	This attribute identifies the configuration of prepare update retries initiated by the Ucm Subordinate.
verifyUpdate	UcmRetryStrategy	0..1	aggr	This attribute identifies the configuration of verify update retries initiated by the Ucm Subordinate.

Table 10.80: UcmSubordinateModuleInstantiation

[constr_10374]{DRAFT} **Existence of the attribute `UcmSubordinateModuleInstantiation.verifyUpdate`** ⌈The attribute `UcmSubordinateModuleInstantiation.verifyUpdate` shall **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10375]{DRAFT} **Existence of the attribute `UcmSubordinateModuleInstantiation.prepareUpdate`** ⌈The attribute `UcmSubordinateModuleInstantiation.prepareUpdate` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10376]{DRAFT} **Existence of the attribute `UcmSubordinateModuleInstantiation.prepareRollback`** ⌈The attribute `UcmSubordinateModuleInstantiation.prepareRollback` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10508]{DRAFT} **Existence of the attribute `UcmSubordinateModuleInstantiation.maxAvailablePersistencyStorageSpace`** ⌈The attribute `UcmSubordinateModuleInstantiation.maxAvailablePersistencyStorageSpace` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10110]{DRAFT} **Existence of `UcmSubordinateModuleInstantiation` on a `Machine`** ⌈For each `Machine`, a `ProcessToMachineMapping` shall exist that refers in the role `machine` to the `Machine` and in the role `nonOsModuleInstantiation` to a `UcmSubordinateModuleInstantiation`.

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

10.9 IAM configuration

The definition of the deployment for the *Identity and Access Manager* represents the creation of actual grants, as opposed to the definition of grants on design level.

One important aspect of the modeling on deployment level is that it is not intended to include a large portion of the design model. The goal is to keep the deployment part as self-contained as possible.

While this approach represents a significant benefit for the size of deployment models it also creates some sort of disconnect between design and deployment. In other words, the connection of the modeling of a specific `Grant` to the respective intent in the design model is not immediately obvious.

To mitigate this issue, AUTOSAR introduced the `GrantDesign` that in turn allows for the identification of the corresponding intent modeling. When loading the design model and deployment model together into a suitable tool it would still be possible to run an analysis in terms of completeness of the overall IAM configuration.

The enforcement of access restrictions is not mandatory for a `Machine`. Therefore, the existence of a `Grant` by itself is not sufficient to activate the IAM mechanisms.

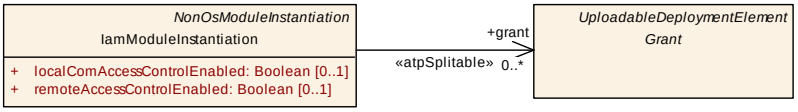


Figure 10.36: Modeling of the `IamModuleInstantiation`

[constr_1695]{DRAFT} **Semantics of a `Grant` depends on the existence of `IamModuleInstantiation`** ⌈The existence of `Grant`s shall only be enforced if in the context of the enclosing `Machine` an `IamModuleInstantiation` has been defined and is referencing the `Grant` **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3677]{DRAFT} **`ComGrant`s referencing DDS Service Instances** ⌈`ComGrant`s associated with `DdsProvidedServiceInstance`s or `DdsRequiredServiceInstance`s via the `serviceInstance` attribute shall not be referenced by `IamModuleInstantiation.grant`, since access control in the DDS Network Binding is delegated to DDS Security **at the time when the creation of the manifest is finished**.⌋*()*

Class	***Grant*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement
Note	This meta-class serves as the abstract base class for defining specific Grants **Tags:** atp.Status=candidate
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*
Subclasses	*ComGrant*, ComOfferServiceGrant, *RawDataStreamGrant*

▽

△

Class	***Grant*** (abstract)			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.81: Grant

Class	***ComGrant*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class serves as the abstract base class for defining specific ComGrants **Tags:** atp.Status=candidate			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	ComEventGrant, ComFieldGrant, ComMethodGrant, ComTriggerGrant			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
remoteSubject	AbstractIamRemote Subject	*	ref	This optional reference defines the remoteSubject that is allowed to access the defined Object via the Grant. **Tags:** atp.Status=candidate
serviceInstance	AdaptivePlatform ServiceInstance	0..1	ref	This reference identifies the applicable AdaptivePlatform ServiceInstance for which the grant applies. **Tags:** atp.Status=candidate

Table 10.82: ComGrant

[constr_10201]{DRAFT} **Multiplicity of of the reference in the role `ComGrant.serviceInstance`** ⌈For each `ComGrant`, the reference in the role `serviceInstance` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

Further information about DDS Security can be found in section 11.5.3.

10.9.1 Com Grant Deployment

10.9.1.1 Com Field Grant Deployment

[TPS_MANI_01237]{DRAFT} **Semantics of meta-class `ComFieldGrant`** ⌈Meta-class `ComFieldGrant` shall be used to award access to a given `field` (identified by means of the reference to meta-class `ServiceFieldDeployment` in the role `serviceDeployment`) in the context of a given `AdaptivePlatformServiceInstance` referenced in the role `serviceInstance`.⌋*(RS_MANI_00060)*

In other words, if a given `AdaptivePlatformServiceInstance` and the respective `ServiceFieldDeployment` are not referenced from a `ComFieldGrant` and an `IamModuleInstantiation` exists **then this specific communication shall be suppressed**.

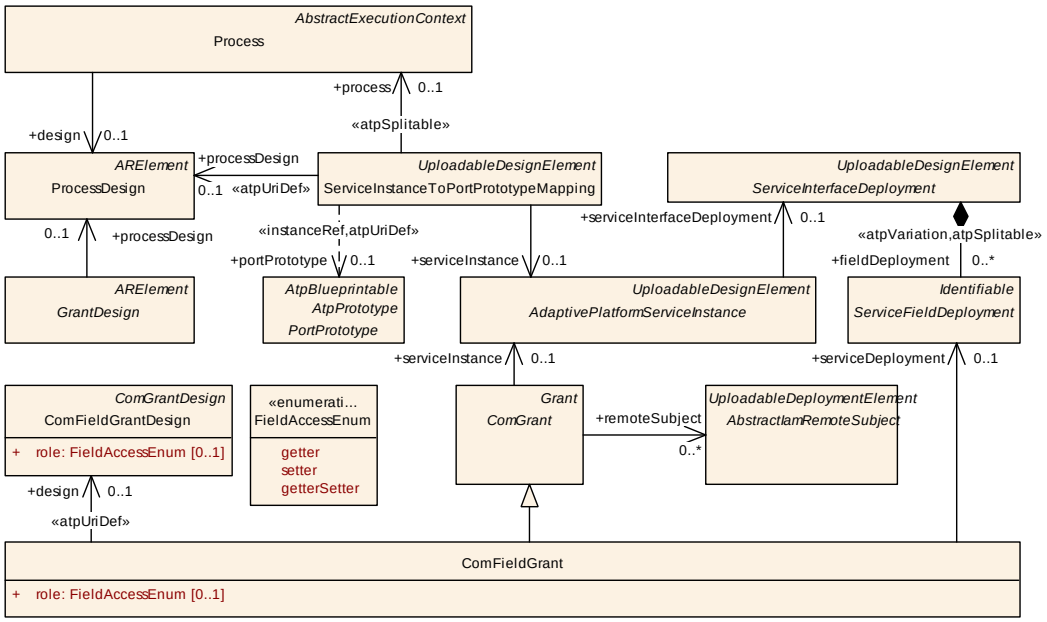


Figure 10.37: Modeling of the `ComFieldGrant`

Class	**ComFieldGrant**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to grant access to a ServiceInterface.field. **Tags:** atp.Status=candidate atp.recommendedPackage=Grants			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrant*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ComFieldGrantDesign	0..1	ref	This reference identifies the ComFieldGrantDesign that the enclosing ComFieldGrant was created from. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate
role	FieldAccessEnum	0..1	attr	This attribute provides the ability to further specify the access to the ServiceInterface.field. **Tags:** atp.Status=candidate
service Deployment	ServiceField Deployment	0..1	ref	This reference identifies the applicable deployment within the context of an AdaptivePlatformServiceInstance for which the grant applies. **Tags:** atp.Status=candidate

Table 10.83: ComFieldGrant

[constr_10202]{DRAFT} **Multiplicity of attribute `ComFieldGrant.role`** ⌈For each `ComFieldGrant`, the attribute `role` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10203]{DRAFT} **Multiplicity of the reference in the role `ComFieldGrant.serviceDeployment`** ⌈For each `ComFieldGrant`, the reference in the role `serviceDeployment` shall exist **at the time when the creation of the manifest is finished.**⌋*()*

10.9.1.2 Com Method Grant Deployment

[TPS_MANI_01238]{DRAFT} **Semantics of meta-class `ComMethodGrant`** ⌈Meta-class `ComMethodGrant` shall be used to clear the call of a given `method` (identified by means of the reference to meta-class `ServiceMethodDeployment` in the role `serviceDeployment`) in the context of a given `AdaptivePlatformServiceInstance` referenced in the role `serviceInstance`.⌋*(RS_MANI_00060)*

In other words, if a given `AdaptivePlatformServiceInstance` and the respective `ServiceMethodDeployment` are not referenced from a `ComMethodGrant` and an `IamModuleInstantiation` exists **then this specific communication shall be suppressed**.

Class	**ComMethodGrant**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to grant access to a ServiceInterface.method. **Tags:** atp.Status=candidate atp.recommendedPackage=Grants			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrant*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ComMethodGrant Design	0..1	ref	This reference identifies the ComMethodGrantDesign that the enclosing ComMethodGrant was created from. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate
service Deployment	ServiceMethod Deployment	0..1	ref	This reference identifies the applicable deployment within the context of an AdaptivePlatformServiceInstance for which the grant applies. **Tags:** atp.Status=candidate

Table 10.84: ComMethodGrant

[constr_10204]{DRAFT} **Multiplicity of the reference in the role `ComMethodGrant.serviceDeployment`** ⌈For each `ComMethodGrant`, the reference in the role `serviceDeployment` shall exist **at the time when the creation of the manifest is finished.**⌋*()*

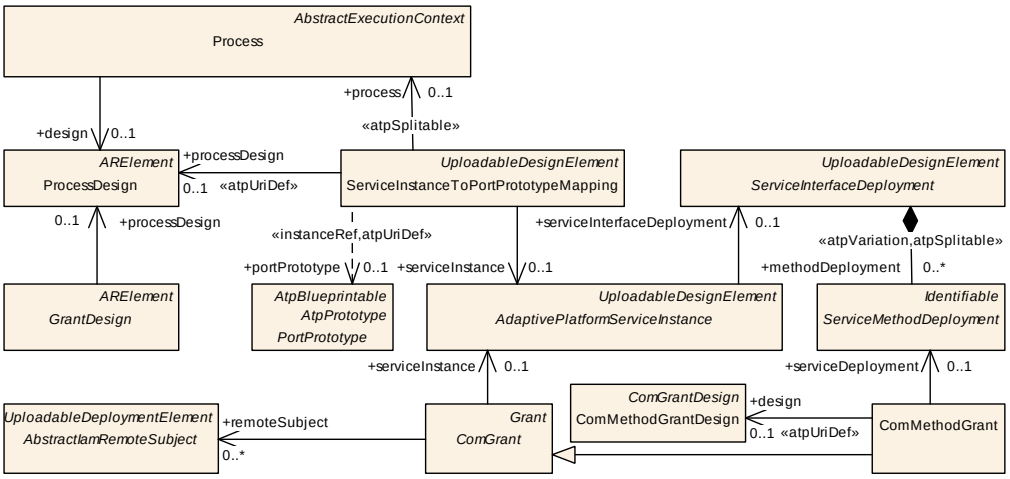


Figure 10.38: Modeling of the `ComMethodGrant`

10.9.1.3 Com Event Grant Deployment

[TPS_MANI_01239]{DRAFT} **Semantics of meta-class `ComEventGrant`** ⌈Meta-class `ComEventGrant` shall be used to award access to a given `event` (identified by means of the reference to meta-class `ServiceEventDeployment` in the role `serviceDeployment`) in the context of a given `AdaptivePlatformServiceInstance` referenced in the role `serviceInstance`.⌋*(RS_MANI_00060)*

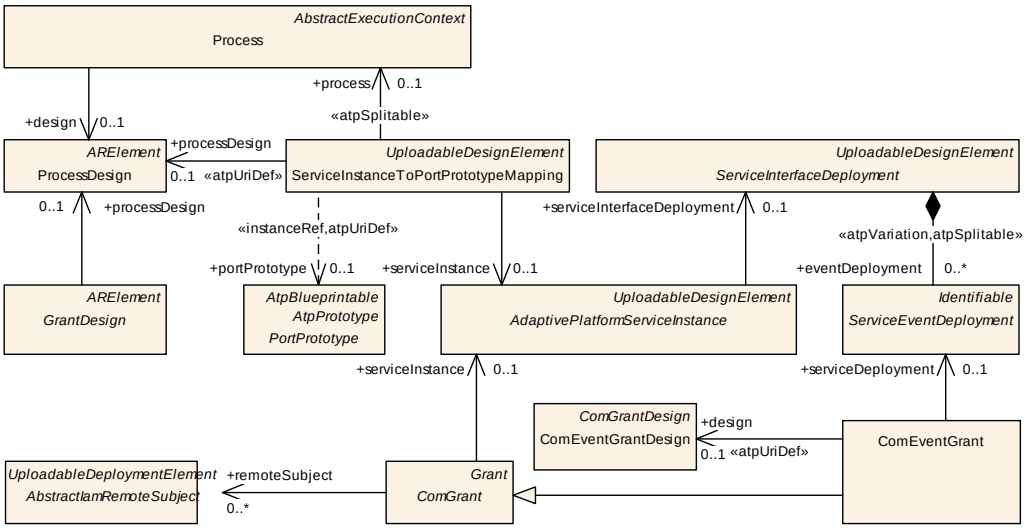


Figure 10.39: Modeling of the `ComEventGrant`

In other words, if a given `AdaptivePlatformServiceInstance` and the respective `ServiceEventDeployment` are not referenced from a `ComEventGrant` and an `IamModuleInstantiation` exists **then this specific communication shall be suppressed**.

[constr_5366]{DRAFT} **Allowed target of `ComEventGrant.serviceDeployment` reference** ⌈`ComEventGrant` is allowed to reference only a `ServiceEventDeployment` in the role `serviceDeployment` that in turn references a `VariableDataPrototype` in the role `event` **`at the time when the creation of the manifest is finished`**.⌋*()*

Class	ComEventGrant			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to grant access to a ServiceInterface.event. **Tags:** atp.Status=candidate atp.recommendedPackage=Grants			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrant*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ComEventGrantDesign	0..1	ref	This reference identifies the ComEventGrantDesign that the enclosing ComEventGrant was created from. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate
service Deployment	ServiceEvent Deployment	0..1	ref	This reference identifies the applicable deployment within the context of an AdaptivePlatformServiceInstance for which the grant applies. **Tags:** atp.Status=candidate

Table 10.85: ComEventGrant

[constr_10205]{DRAFT} **Multiplicity of the reference in the role `ComEventGrant.serviceDeployment`** ⌈For each `ComEventGrant`, the reference in the role `serviceDeployment` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

10.9.1.4 Com Trigger Grant Deployment

[TPS_MANI_03322]{DRAFT} **Semantics of meta-class `ComTriggerGrant`** ⌈Meta-class `ComTriggerGrant` shall be used to award access to a given `trigger` (identified by means of the reference to meta-class `ServiceEventDeployment` in the role `serviceDeployment`) in the context of a given `AdaptivePlatformServiceInstance` referenced in the role `serviceInstance`.⌋*(RS_MANI_00060)*

In other words, if a given `AdaptivePlatformServiceInstance` and the respective `ServiceEventDeployment` for a `Trigger` are not referenced from a `ComTriggerGrant` and an `IamModuleInstantiation` exists then this specific communication shall be suppressed.

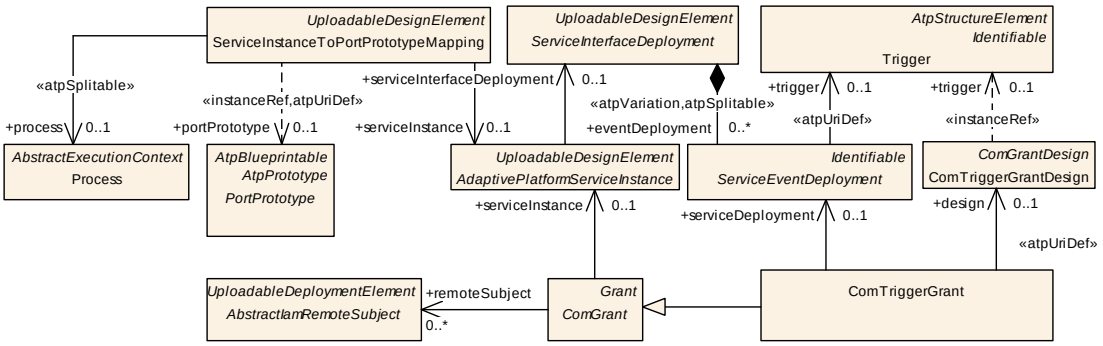


Figure 10.40: Modeling of the `ComTriggerGrant`

Class	**ComTriggerGrant**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to grant access to a ServiceInterface.trigger **Tags:** atp.Status=candidate atp.recommendedPackage=Grants			
Base	*ARElement*, *ARObject*, *CollectableElement*, *ComGrant*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ComTriggerGrant Design	0..1	ref	This reference identifies the ComTriggerGrantDesign that the enclosing ComTriggerGrant was created from **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate
service Deployment	ServiceEvent Deployment	0..1	ref	This reference identifies the applicable deployment within the context of an AdaptivePlatformServiceInstance for which the grant applies. **Tags:** atp.Status=candidate

Table 10.86: ComTriggerGrant

[constr_5367]{DRAFT} **Allowed target of `ComTriggerGrant.serviceDeployment` reference** ⌈`ComTriggerGrant` is allowed to reference only a `ServiceEventDeployment` in the role `serviceDeployment` that in turn references a `Trigger` in the role `trigger` **at the time when the creation of the manifest is finished**.⌋*()*

[constr_5368]{DRAFT} **Multiplicity of the reference in the role `ComTriggerGrant.serviceDeployment`** ⌈For each `ComTriggerGrant`, the reference in the role `serviceDeployment` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

10.9.1.5 Com Offer Service Grant Deployment

The enforcement of service discovery rights is modeled by means of meta-classes `ComOfferServiceGrant`.

There is no need for a dedicated grant for service discovery on the client side because the existence of one of the dedicated grants to method, field, event, or trigger is taken as the implicit intent to also do service discovery.

[TPS_MANI_01240]{DRAFT} **Semantics of meta-class ComOfferServiceGrant** ⌈Meta-class ComOfferServiceGrant shall be used to award the right to offer the referenced AdaptivePlatformServiceInstance.⌋*(RS_MANI_00060)*

Class	**ComOfferServiceGrant**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to grant the offering of a service. **Tags:** atp.Status=candidate atp.recommendedPackage=Grants			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ComOfferServiceGrant Design	0..1	ref	This reference identifies the ComOfferServiceGrant Design that the enclosing ComOfferServiceGrant was created from. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate
serviceInstance	AdaptivePlatform ServiceInstance	0..1	ref	This reference identifies the AdaptivePlatformService Instances for which the grant applies. **Tags:** atp.Status=candidate

Table 10.87: ComOfferServiceGrant

[constr_10206]{DRAFT} **Multiplicity of the reference in the role ComOfferServiceGrant.serviceInstance** ⌈For each ComOfferServiceGrant, the reference in the role serviceInstance shall exist **at the time when the creation of the manifest is finished**.⌋*()*

10.9.2 Grant Deployment for Raw Streaming Data

The definition of abstract meta-class RawDataStreamGrant on the level of deployment complements the existence of RawDataStreamGrantDesign on design level.

[TPS_MANI_01307]{DRAFT} **Semantics of meta-class EthernetRawDataStreamGrant** ⌈Meta-class EthernetRawDataStreamGrant provides the deployment-level IAM semantics for raw data streams that run on TCP/IP sockets. For this purpose, the reference in the role ethernetRawDataStreamMapping to meta-class EthernetRawDataStreamMapping exists.⌋*(RS_MANI_00060, RS_MANI_00067)*

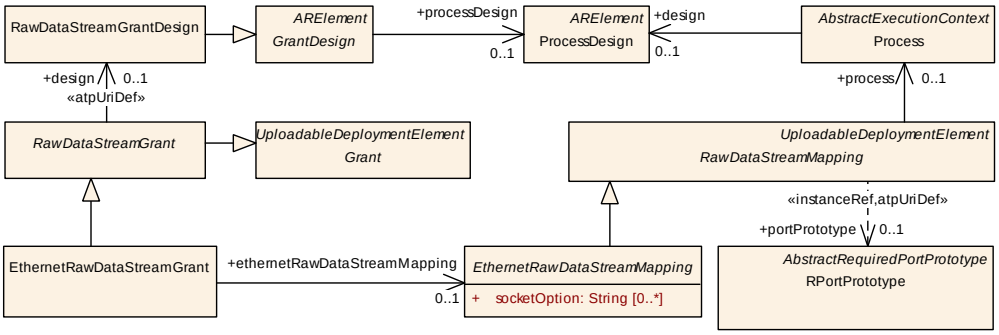


Figure 10.41: Modeling of the `RawDataStreamGrant`

Class	***RawDataStreamGrant*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This abstract meta-class represents the ability to define the IAM configuration for a RawDataStream on deployment level. **Tags:** atp.Status=candidate			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	EthernetRawDataStreamGrant			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	RawDataStreamGrant Design	0..1	ref	This reference identifies the RawDataStreamGrantDesign that the enclosing RawDataStreamEventGrant was created from. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate

Table 10.88: RawDataStreamGrant

Class	**EthernetRawDataStreamGrant**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to define the IAM configuration for a EthernetRawDataStream on deployment level. **Tags:** atp.Status=candidate atp.recommendedPackage=Grants			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Grant*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *RawDataStreamGrant*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackage Element*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ethernetRaw DataStream Mapping	EthernetRawData StreamMapping	0..1	ref	This reference identifies the applicable EthernetRawData Stream to which the enclosing EthernetRawDataStream Grant shall apply. **Tags:** atp.Status=candidate

Table 10.89: EthernetRawDataStreamGrant

10.9.3 Remote access control

The overview of the remote access control was already introduced in chapter 3.6.3. Please note that for the modeling of AbstractIamRemoteSubjects the same approach is used in the Design and in the Deployment. So if AbstractIamRemoteSubjects were defined during the design phase they can be taken over into the deployment.

This chapter defines how ComGrants with a defined remoteSubjects shall be interpreted.

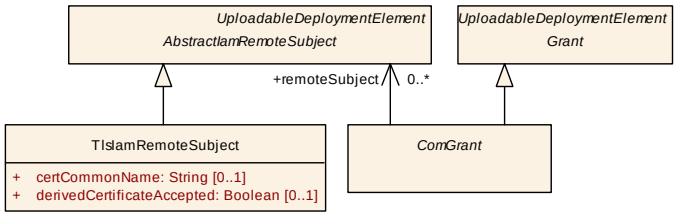


Figure 10.42: Modeling of the relation of ComGrants to the corresponding remoteSubjects

[TPS_MANI_03245]{DRAFT} **Definition of ComMethodGrant.remoteSubjects on server side** ⌈If the ComMethodGrant references a ProvidedApServiceInstance in the role serviceInstance then the ComMethodGrant.remoteSubject reference defines the remoteSubjects that are allowed to call the defined method that is referenced by the ComMethodGrant in the role serviceDeployment.⌋*(RS_MANI_00060)*

[TPS_MANI_03246]{DRAFT} **Definition of ComMethodGrant.remoteSubjects on client side** ⌈If the ComMethodGrant references a RequiredApServiceInstance in the role serviceInstance then the ComMethodGrant.remoteSubject reference defines the remoteSubjects to which a ServiceMethodDeployment.method call is allowed to be sent.⌋*(RS_MANI_00060)*

[TPS_MANI_03247]{DRAFT} **Definition of ComEventGrant.remoteSubjects on provider side** ⌈If the ComEventGrant references a ProvidedApServiceInstance in the role serviceInstance then the ComEventGrant.remoteSubject reference defines the remoteSubjects to which an ServiceEventDeployment.event is allowed to be sent.⌋*(RS_MANI_00060)*

[TPS_MANI_03248]{DRAFT} **Definition of ComEventGrant.remoteSubjects on receiver side** ⌈If the ComEventGrant references a RequiredApServiceInstance in the role serviceInstance then the ComEventGrant.remoteSubject reference defines the remoteSubjects from which an ServiceEventDeployment.event is allowed to be received.⌋*(RS_MANI_00060)*

[TPS_MANI_03249]{DRAFT} **Definition of ComFieldGrant.remoteSubjects on provider side** ⌈If the ComFieldGrant references a ProvidedApServiceInstance in the role serviceInstance then the ComFieldGrant.remoteSubject reference defines the remoteSubjects that are allowed to access the Field that is referenced

by the ServiceFieldDeployment that in turn is referenced by the ComFieldGrant in the role serviceDeployment. This allows the following communication between the local Machine and the AbstractIamRemoteSubject:

- the Field notifier is allowed to be sent to remoteSubjects,
- if ComFieldGrant.role equals setter or getterSetter, then the setter call is allowed to be received from the remoteSubjects,
- if ComFieldGrant.role equals setter or getterSetter, then the setter return is allowed to be sent to remoteSubjects,
- if ComFieldGrant.role equals getter or getterSetter, then the getter call is allowed to be received from the remoteSubjects,
- if ComFieldGrant.role equals getter or getterSetter, then the getter return is allowed to be sent to remoteSubjects.

⌋(*RS_MANI_00060*)

[TPS_MANI_03250]{DRAFT} **Definition of ComFieldGrant.remoteSubjects on client side** ⌈If the ComFieldGrant references a RequiredApServiceInstance in the role serviceInstance then the ComFieldGrant.remoteSubject reference defines the remoteSubjects that are allowed to provide the Field for access that is referenced by the ServiceFieldDeployment that in turn is referenced by the ComFieldGrant in the role serviceDeployment. This will allow the following communication between the local Machine and the AbstractIamRemoteSubject:

- the Field notifier is allowed to be received from remoteSubjects,
- if ComFieldGrant.role equals setter or getterSetter, then the setter call is allowed to be sent to the remoteSubjects,
- if ComFieldGrant.role equals setter or getterSetter, then the setter return is allowed to be received from remoteSubjects,
- if ComFieldGrant.role equals getter or getterSetter, the getter call is allowed to be sent to the remoteSubjects,
- if ComFieldGrant.role equals getter or getterSetter, the getter return is allowed to be received from remoteSubjects.

⌋(*RS_MANI_00060*)

Please note that a ComGrant with the remoteSubject reference defines that a remote access control for this ComGrant will be performed. Such a ComGrant with the remoteSubject reference does not enforce any Machine local access restrictions.

To enforce local access restrictions for the same ServiceInterface element of the same AdaptivePlatformServiceInstance an additional ComGrant needs to be defined that points to the same serviceInstance and to the same ComFieldGrant.serviceDeployment or ComEventGrant.serviceDeployment or

ComMethodGrant.serviceDeployment but does not contain the remoteSubject reference.

10.10 Crypto Deployment

This chapter explains the configuration of the Crypto functional cluster and the interaction of application Software with the Crypto stack [16].

[TPS_MANI_03260]{DRAFT} **Semantics of meta-class CryptoModuleInstantiation** ⌈The representation of the Crypto functional cluster [16] within one specific Machine is defined by the CryptoModuleInstantiation.⌋ *(RS_MANI_00023)*

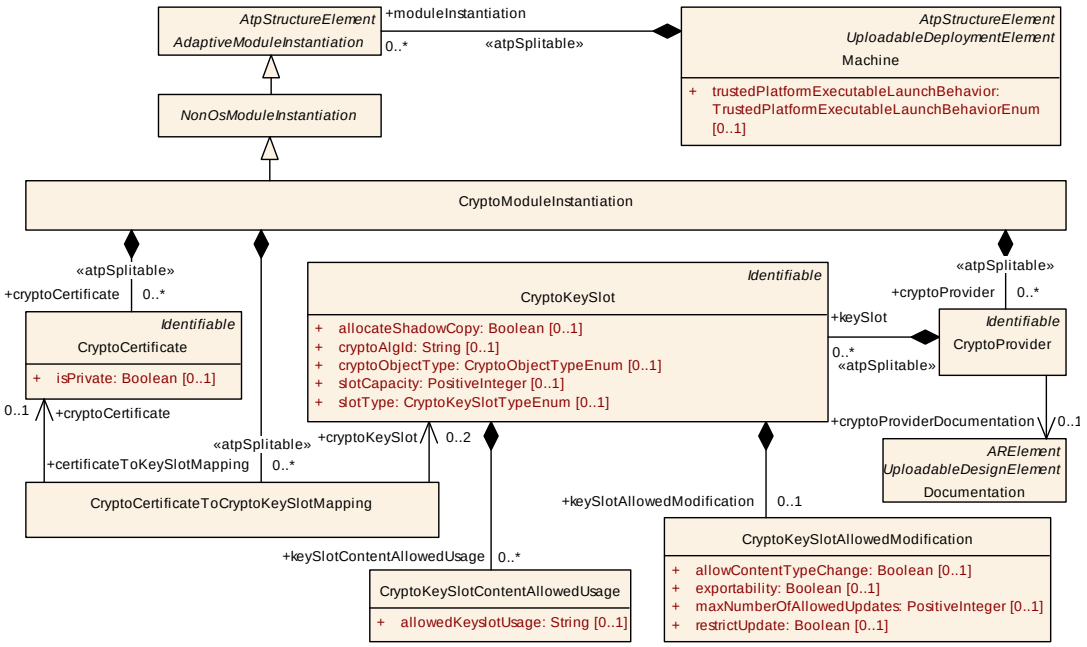


Figure 10.43: CryptoModuleInstantiation Overview

Class	**CryptoModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class defines the configuration for the Crypto stack on a specific machine.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
certificateToKeySlotMapping	CryptoCertificateToCryptoKeySlotMapping	*	aggr	List of CryptoCertificateToCryptoKeySlotMappings available in the CryptoStack. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=certificateToKeySlotMapping

△

Class	CryptoModuleInstantiation			
crypto Certificate	CryptoCertificate	*	aggr	List of CryptoCertificates managed in the CryptoStack **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=cryptoCertificate.shortName
cryptoProvider	CryptoProvider	*	aggr	List of CryptoProviders provided by the CryptoStack **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=cryptoProvider.shortName

Table 10.90: CryptoModuleInstantiation

10.10.1 Crypto Provider

[TPS_MANI_03261]{DRAFT} **Support of `CryptoProvider`s** ⌈The Crypto functional cluster is able to support multiple `CryptoProvider`s. Each `CryptoProvider` implements a software- or a hardware-based cryptographic library.⌋*(RS_MANI_00023)*

Class	CryptoProvider			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	CryptoProvider implements cryptographic primitives (algorithms) supported by the stack. Implementation of this component may be software or hardware based (HSM/TPM).			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CryptoModuleInstantiation.cryptoProvider			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cryptoProvider Documentation	Documentation	0..1	ref	Documentation of the CryptoProvider that describes the implemented cryptographic primitives.
keySlot	CryptoKeySlot	*	aggr	This aggregation represents the key slots that are allocated by the CryptoProvider. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=keySlot.shortName

Table 10.91: CryptoProvider

Please note that each `CryptoProvider` can be described in more detail with the `cryptoProviderDocumentation` that uses the `Documentation` element to provide means for a more detailed description.

[TPS_MANI_03262]{DRAFT} **Semantics of `CryptoProviderToPortPrototypeMapping`** ⌈Meta-class `CryptoProviderToPortPrototypeMapping` has the ability to map a specific `PortPrototype` referenced in the role `portPrototype` to a `CryptoProvider` referenced in the role `cryptoProvider`.

The mapping also comprises a reference to meta-class `process` in order to accommodate for the fact that identical combinations of `cryptoProvider` and `portPrototype` may or may not apply for a given `Process` that represents the enclosing `Executable` at runtime.⌋*(RS_MANI_00023)*

Class	**CryptoProviderToPortPrototypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to define a mapping between a CryptoProvider on deployment level to a given PortPrototype that is typed by a CryptoProviderInterface. **Tags:** atp.recommendedPackage=CryptoProviderToPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cryptoProvider	CryptoProvider	0..1	ref	This reference represents the mapped cryptoProvider.
portPrototype	RPortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference represents the process required as context for the mapping.

Table 10.92: CryptoProviderToPortPrototypeMapping

[constr_10207]{DRAFT} **Multiplicity of the reference in the role `CryptoProviderToPortPrototypeMapping.cryptoProvider`** ⌈For each `CryptoProviderToPortPrototypeMapping`, the reference in the role `cryptoProvider` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10208]{DRAFT} **Multiplicity of the reference in the role `CryptoProviderToPortPrototypeMapping.process`** ⌈For each `CryptoProviderToPortPrototypeMapping`, the reference in the role `process` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

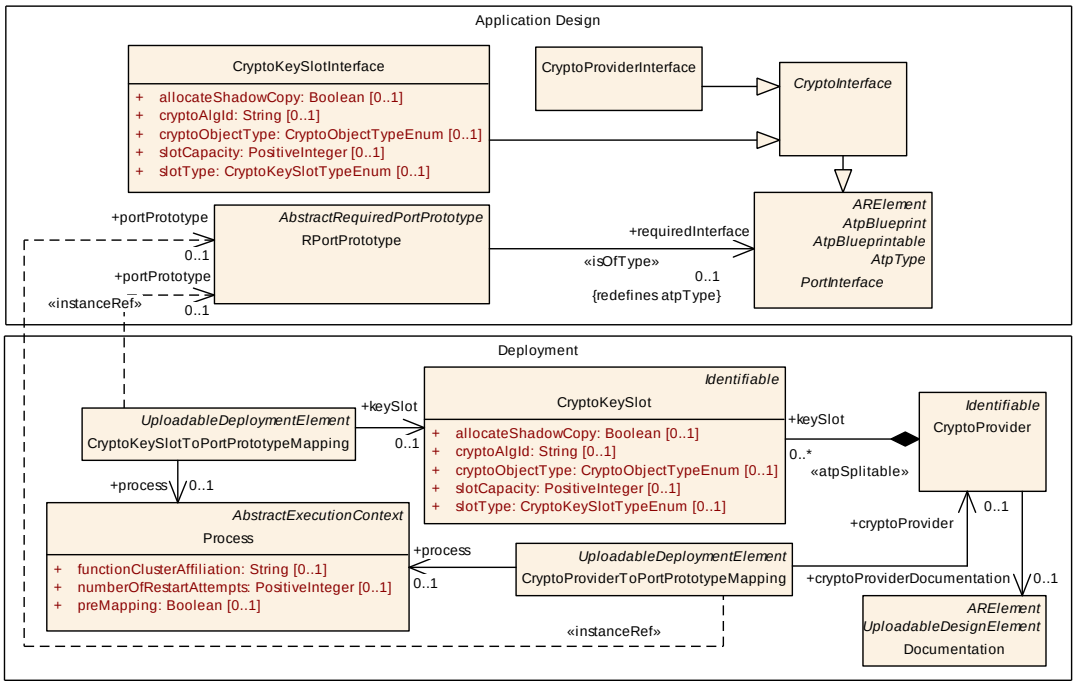


Figure 10.44: Mapping of CryptoProviders and CryptoKeySlots to RPortPrototypes

[constr_5240]{DRAFT} **Restriction applicable for `CryptoProviderToPortPrototypeMapping.portPrototype`** ⌈The reference `CryptoProviderToPortPrototypeMapping.portPrototype` shall only be used for an `RPortPrototype` typed by a `CryptoProviderInterface` **at the time when the creation of the manifest is finished**.⌋*()*

The application developer uses the `ara::core::InstanceSpecifier` as local identifier in the API call that represents the path to the modeled `RPortPrototype`. The Integrator maps the `RPortPrototype` in the deployment model with the `CryptoProviderToPortPrototypeMapping` to the concrete `CryptoProvider`.

10.10.2 Crypto Key Slot

The Key Storage Provider is responsible for the storage of different type of key material. The crypto objects that are stored by the Key Storage Provider are represented as `CryptoKeySlot`s in the model.

[TPS_MANI_03263]{DRAFT} **Assignment of `CryptoKeySlot`s to `CryptoProvider`s** ⌈Crypto objects that are used by the `CryptoProvider` are described by the `CryptoKeySlot`s that are aggregated by the `CryptoProvider` in the role `keySlot`.⌋*(RS_MANI_00023)*

Class	**CryptoKeySlot**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to define a concrete key to be used for a crypto operation. **Tags:** atp.ManifestKind=MachineManifest			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CryptoProvider.keySlot			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allocateShadowCopy	Boolean	0..1	attr	This attribute defines whether a shadow copy of this Key Slot shall be allocated to enable rollback of a failed Key Slot update campaign (see interface BeginTransaction).
cryptoAlgId	String	0..1	attr	This attribute defines a crypto algorithm restriction (kAlgId Any means without restriction). The algorithm can be specified partially: family & length, mode, padding. Future Crypto Providers can support some crypto algorithms that are not well known/ standardized today, therefore AUTOSAR doesn't provide a concrete list of crypto algorithms' identifiers and doesn't suppose usage of numerical identifiers. Instead of this a provider supplier should provide string names of supported algorithms in accompanying documentation. The name of a crypto algorithm shall follow the rules defined in the specification of cryptography for Adaptive Platform.
cryptoObjectType	CryptoObjectTypeEnum	0..1	attr	Object type that can be stored in the slot. If this field contains "Undefined" then mSlotCapacity must be provided and larger then 0. **Tags:** atp.Status=candidate

▽

△

Class	CryptoKeySlot			
keySlotAllowed Modification	CryptoKeySlotAllowed Modification	0..1	aggr	Restricts how this keySlot may be used **Tags:** atp.Status=candidate
keySlotContent AllowedUsage	CryptoKeySlotContent AllowedUsage	*	aggr	Restriction of allowed usage of a key stored to the slot. **Tags:** atp.Status=candidate
slotCapacity	PositiveInteger	0..1	attr	Capacity of the slot in bytes to be reserved by the stack vendor. One use case is to define this value in case that the cryptoObjectType is undefined and the slot size can not be deduced from cryptoObjectType and cryptoAlgId. "0" means slot size can be deduced from cryptoObject Type and cryptoAlgId.
slotType	CryptoKeySlotType Enum	0..1	attr	This attribute defines whether the keySlot is exclusively used by the Application; or whether it is used by Stack Services and managed by a Key Manager Application. **Tags:** atp.Status=candidate

Table 10.93: CryptoKeySlot

[TPS_MANI_03264]{DRAFT} **Semantics of `CryptoKeySlotToPortPrototypeMapping`** ⌈Meta-class `CryptoKeySlotToPortPrototypeMapping` has the ability to map a specific `PortPrototype` referenced in the role `portPrototype` to a `CryptoKeySlot` referenced in the role `keySlot`.

The mapping also comprises a reference to meta-class `process` in order to accommodate for the fact that identical combinations of `keySlot` and `portPrototype` may or may not apply for a given `Process` that represents the enclosing `Executable` at runtime.⌋*(RS_MANI_00023)*

Class	CryptoKeySlotToPortPrototypeMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to define a mapping between a CryptoKeySlot on deployment level to a given PortPrototype that is typed by a CryptoKeySlotInterface. **Tags:** atp.recommendedPackage=CryptoKeySlotToPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
keySlot	CryptoKeySlot	0..1	ref	This reference represents the mapped CryptoKeySlot.
portPrototype	RPortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference represents the process required as context for the mapping.

Table 10.94: CryptoKeySlotToPortPrototypeMapping

[constr_10209]{DRAFT} **Multiplicity of the reference in the role `CryptoKeySlotToPortPrototypeMapping.keySlot`** ⌈For each `CryptoKeySlotToPortPrototypeMapping`, the reference in the role `keySlot` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10210]{DRAFT} **Multiplicity of the reference in the role `CryptoKeySlotToPortPrototypeMapping.process`** ⌈For each `CryptoKeySlotToPortPrototypeMapping`, the reference in the role `process` shall exist **at the time when the creation of the manifest is finished.**⌋*()*

[constr_5241]{DRAFT} **Restriction applicable for `CryptoKeySlotToPortPrototypeMapping.portPrototype`** ⌈The reference `CryptoKeySlotToPortPrototypeMapping.portPrototype` shall only be used for an `RPortPrototype` typed by a `CryptoKeySlotInterface` **at the time when the creation of the manifest is finished.**⌋*()*

The application developer uses the `ara::core::InstanceSpecifier` as local identifier in the API call that represents the path to the modeled `RPortPrototype`. The Integrator maps the `RPortPrototype` in the deployment model with the `CryptoKeySlotToPortPrototypeMapping` to the concrete `CryptoKeySlot` that is stored by the Key Storage Provider. The information from the deployment model is therefore used to replace the local identifier from the Application Design by the concrete `CryptoKeySlot`.

10.10.3 Crypto Certificate

[TPS_MANI_03265]{DRAFT} **Support of `CryptoCertificate`s** ⌈Certificates stored by the Certificate Management Provider that is available in the Crypto functional cluster are modeled as `CryptoCertificate`s.⌋*(RS_MANI_00023)*

Class	**CryptoCertificate**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to model a cryptographic certificate.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CryptoModuleInstantiation.cryptoCertificate			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
isPrivate	Boolean	0..1	attr	This attribute controls the possibility to access the content of the CryptoCertificateSlot by Find() interfaces of the X509 Provider.

Table 10.95: CryptoCertificate

[TPS_MANI_03266]{DRAFT} **Semantics of `CryptoCertificateToCryptoKeySlotMapping`** ⌈The `CryptoCertificateToCryptoKeySlotMapping` is used to assign a private key and optionally a public key to the `CryptoCertificate`.⌋ *(RS_MANI_00023)*

Class	**CryptoCertificateToCryptoKeySlotMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment
Note	This meta-class represents the ability to define a mapping between a CryptoKeySlot and a Crypto Certificate.

▽

△

Class	**CryptoCertificateToCryptoKeySlotMapping**			
Base	*ARObject*			
Aggregated by	CryptoModuleInstantiation.certificateToKeySlotMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
crypto Certificate	CryptoCertificate	0..1	ref	This reference represents the mapped cryptoCertificate.
cryptoKeySlot	CryptoKeySlot	0..2	ref	This reference represents the mapped cryptoKeySlot.

Table 10.96: CryptoCertificateToCryptoKeySlotMapping

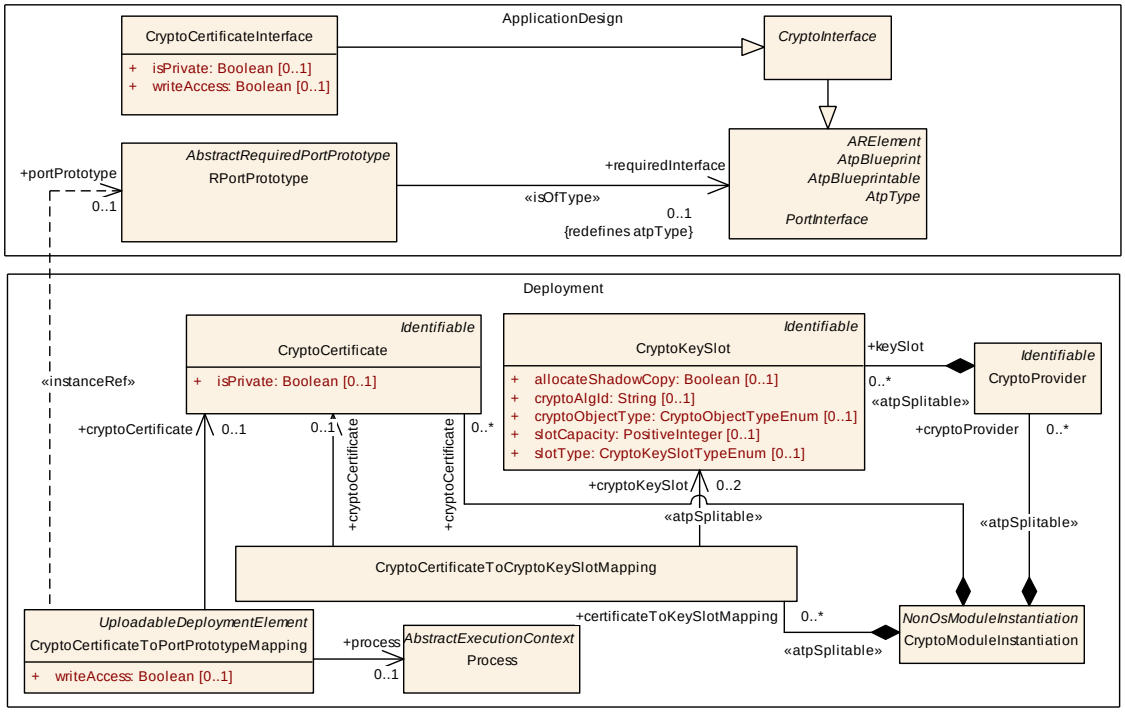


Figure 10.45: Mapping of CryptoCertificates to RPortPrototypes

[constr_10211]{DRAFT} **Multiplicity of the reference in the role `CryptoCertificateToCryptoKeySlotMapping.cryptoCertificate`** ⌈For each `CryptoCertificateToCryptoKeySlotMapping`, the reference in the role `cryptoCertificate` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[TPS_MANI_03267]{DRAFT} **Semantics of `CryptoCertificateToPortPrototypeMapping`** ⌈Meta-class `CryptoCertificateToPortPrototypeMapping` has the ability to map a specific `PortPrototype` referenced in the role `portPrototype` to a `CryptoCertificate` referenced in the role `cryptoCertificate`.

The mapping also comprises a reference to meta-class `process` in order to accommodate for the fact that identical combinations of `cryptoCertificate` and `portPrototype` may or may not apply for a given `Process` that represents the enclosing `Executable` at runtime.⌋*(RS_MANI_00023)*

Class	**CryptoCertificateToPortPrototypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to define a mapping between a CryptoCertificate on deployment level to a given PortPrototype that is typed by a CryptoCertificateInterface. **Tags:** atp.recommendedPackage=CryptoCertificateToPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
crypto Certificate	CryptoCertificate	0..1	ref	This reference represents the mapped cryptoCertificate.
portPrototype	RPortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference represents the process required as context for the mapping.
writeAccess	Boolean	0..1	attr	This attribute defines whether the application has write-access to the CryptoCertificate (true) or only read-access (false).

Table 10.97: CryptoCertificateToPortPrototypeMapping

[constr_5242]{DRAFT} **Restriction applicable for `CryptoCertificateToPortPrototypeMapping.portPrototype`** ⌈The reference `CryptoCertificateToPortPrototypeMapping.portPrototype` shall only be used for an `RPortPrototype` typed by a `CryptoCertificateInterface` **at the time when the creation of the manifest is finished**.⌋*()*

The application developer uses the `ara::core::InstanceSpecifier` as local identifier in the API call that represents the path to the modeled `RPortPrototype`. The Integrator maps the `RPortPrototype` in the deployment model with the `CryptoCertificateToPortPrototypeMapping` to the concrete `CryptoCertificate` that is stored by the Certificate Management Provider. The information from the deployment model is therefore used to replace the local identifier from the Application Design by the concrete `CryptoCertificate`.

10.11 IdsM Deployment

10.11.1 IdsM Instantiation

The definition of the deployment for the `Intrusion Detection System Manager` (IdsM) is modeled by means of the meta-class `IdsmModuleInstantiation`

Class	IdsmModuleInstantiation			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IntrusionDetectionSystem			
Note	This meta-class defines the attributes for the IdsM configuration on a specific machine. **Tags:** atp.Status=candidate			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *IdsPlatformInstantiation*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
reportable SecurityEvent	SecurityEventMapping	*	ref	Collection of reportable instances of security events. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=reportableSecurityEvent atp.Status=candidate

Table 10.98: IdsmModuleInstantiation

[constr_10021]{DRAFT} **Existence of `IdsmModuleInstantiation`** ⌈On each `Machine`, only one instance of the `Intrusion Detection System Manager` (modeled by `IdsmModuleInstantiation`) shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

This instance manages all the reported `SEv`s created by `SWCL`s or `Function Cluster`s on this `Adaptive Machine`.

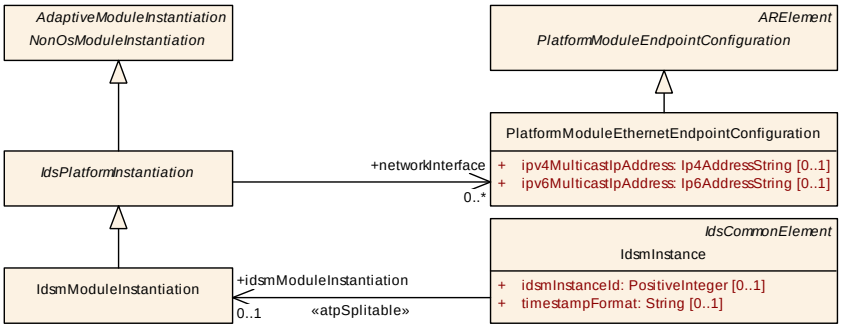


Figure 10.46: Specification of the `IdsmModuleInstantiation`

Meta-class `IdsmModuleInstantiation` is derived from the meta-class `IdsPlatformInstantiation` which acts as an abstract base class for all platform modules that contribute to the implementation of the Intrusion Detection System.

Class	***IdsPlatformInstantiation*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IntrusionDetectionSystem
Note	This meta-class acts as an abstract base class for platform modules that implement the intrusion detection system. **Tags:** atp.Status=candidate
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*
Subclasses	IdsmModuleInstantiation
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation

△

Class	*IdsPlatformInstantiation* (abstract)			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
network Interface	PlatformModule EthernetEndpoint Configuration	*	ref	This association contains the network configuration that shall be applied to an instance of an IDS entity. **Tags:** atp.Status=candidate
timeBase	TimeBaseResource	0..1	ref	This reference identifies the applicable time base resource. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=timeBase.timeBaseResource, time Base.variationPoint.shortLabel atp.Status=candidate vh.latestBindingTime=systemDesignTime

Table 10.99: IdsPlatformInstantiation

[constr_3728]{DRAFT} **Upper multiplicity of reference in the role `IdsPlatformInstantiation.networkInterface`** ⌈In the context of `IdsPlatformInstantiation`, the reference in the role `networkInterface` shall exist at most once **`at the time when the creation of the manifest is finished`**.⌋*()*

One of the tasks of an IdsM is to obtain a custom time stamp that corresponds to reported security events under certain circumstances (which are not relevant for the sake of discussing the modeling).

AUTOSAR supports two ways to obtain a time stamp for the purpose of *Intrusion Detection System Management*, but only one of these ways shall be used at any given time. This restriction is formalized in [constr_10501].

[constr_10501]{DRAFT} **Access to time stamps for *Intrusion Detection System Management*** ⌈Access to time stamps shall be configured in at most **one of two** possible ways:

- by reference in the role `timeBase` from `IdsPlatformInstantiation` to `TimeBaseResource` or
- by creating a `IdsmTimestampProviderMapping` that refers in the role `pPortPrototypeInExecutable` to a `PPortPrototype` defined by a piece of application software that represents a time-stamp provider.

⌋*()*

Please note that the first approach mentioned in [constr_10501] is documented in section 10.11.2, while second approach to obtain time-stamps mentioned in [constr_10501] is described in section 10.11.3, resp. [TPS_MANI_01440].

To be clear, the difference between the two approaches for obtaining a time stamp is that the first approach is implemented entirely on the level of the interaction of functional clusters, whereas the second approach involves the execution of application-level software.

10.11.2 Time Stamps for Security Events via Time-Base Resource

In general, there are two possible ways to obtain time stamps, as described in section 10.11.1. The approach described in this section is based on the creation of a reference to a `TimeBaseResource`.

Time stamps can be obtained from concrete sub-classes of the abstract base-class `TimeBaseResource`.

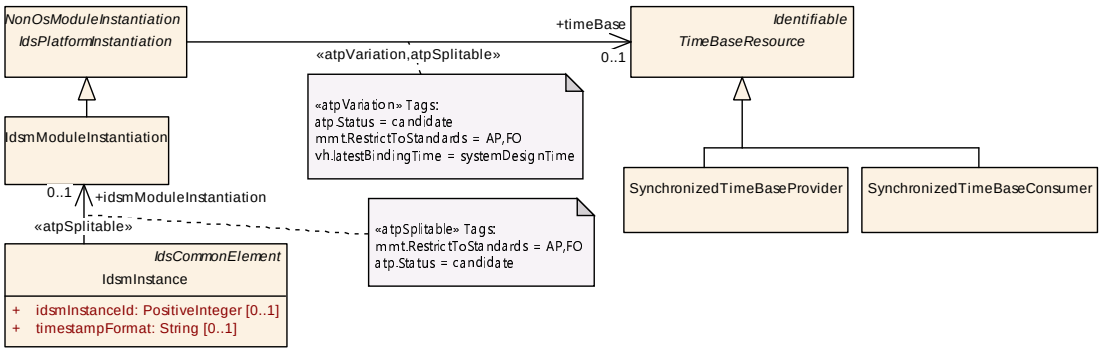


Figure 10.47: Creating an association between a `IdsmModuleInstantiation` and a `TimeBaseResource`

10.11.3 Mapping to Timestamp Provider

In general, there are two possible ways to obtain time stamps, as described in section 10.11.1. the approach described in this section is based on the creation of a `IdsmTimestampProviderMapping`.

[TPS_MANI_01440]{DRAFT} **Using the `IdsmTimestampProviderMapping` for the provision of timestamps in the context of the *Intrusion Detection System Management*** ⌈Meta-class `IdsmTimestampProviderMapping` creates an association between

- a `PPortPrototype` defined in the context of an `Executable` that implements application software to provide timestamps for the *Intrusion Detection System Management* and
- a `Process` in which the `Executable` is configured to run and
- the `IdsPlatformInstantiation` in which the *Intrusion Detection System Management* is implemented.

⌋*(RS_MANI_00069)*

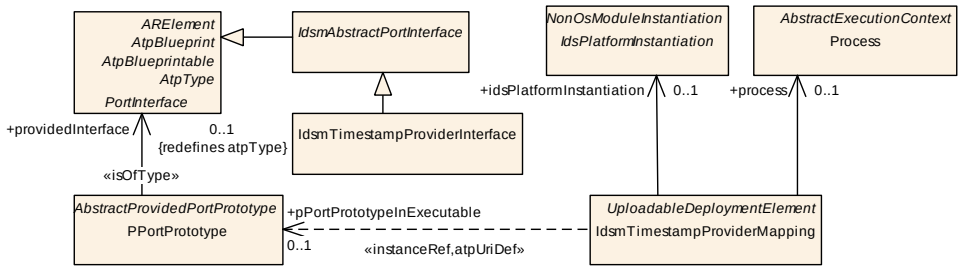


Figure 10.48: Mapping to timestamp provider

[constr_10491]{DRAFT} **Type of `PPortPrototype` referenced in the role `IdsmTimestampProviderMapping.pPortPrototypeInExecutable`** ⌈A `PPortPrototype` referenced in the role `IdsmTimestampProviderMapping.pPortPrototypeInExecutable` shall only be typed by a `IdsmTimestampProviderInterface` **at the time when the creation of the manifest is finished.**⌋ *()*

[constr_10492]{DRAFT} **`PPortPrototype` typed by a `IdsmTimestampProviderInterface`** ⌈A `PPortPrototype` that is typed by a `IdsmTimestampProviderInterface` shall only be referenced in the role `pPortPrototypeInExecutable` by a `IdsmTimestampProviderMapping` **at the time when the creation of the manifest is finished.**⌋ *()*

Class	**IdsmTimestampProviderMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IntrusionDetectionSystem			
Note	This meta-class represents the ability to define a mapping between an IdsMInstance and a Process on deployment level to a given PortPrototype that is typed by a IdsmTimestampProviderInterface. **Tags:** atp.recommendedPackage=IdsmProviderMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
idsPlatform Instantiation	IdsPlatformInstantiation	0..1	ref	This represents the IdsM functional cluster. **Tags:** atp.Status=candidate
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This reference identifies the mapped PortPrototype in the application software. **Stereotypes:** atpUriDef **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference identifies the process in which the application runs.

Table 10.100: IdsmTimestampProviderMapping

[constr_10493]{DRAFT} **Existence of `IdsmTimestampProviderMapping.idsPlatformInstantiation`** ⌈For each `IdsmTimestampProviderMapping`, the reference to an `IdsPlatformInstantiation` in the role `idsPlatformInstantiation` shall exist **at the time when the creation of the manifest is finished.**⌋ *()*

[constr_10494]{DRAFT} **Existence of `IdsmTimestampProviderMapping.pPortPrototypeInExecutable`** ⌈For each `IdsmTimestampProviderMapping`, the instance reference to a `PPortPrototype` in the role `pPortPrototypeInExecutable` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10495]{DRAFT} **Existence of `IdsmTimestampProviderMapping.process`** ⌈For each `IdsmTimestampProviderMapping`, the reference to a `Process` in the role `process` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

10.11.4 Mapping to Context Provider

[TPS_MANI_01441]{DRAFT} **Using the `IdsmContextProviderMapping` for the provision of context in the context of the *Intrusion Detection System Management*** ⌈Meta-class `IdsmContextProviderMapping` creates an association between

- a `PPortPrototype` defined in the context of an `Executable` that implements application software to provide context for the *Intrusion Detection System Management* and
- a `Process` in which the `Executable` is configured to run and
- the `IdsPlatformInstantiation` in which the *Intrusion Detection System Management* is implemented.

⌋*(RS_MANI_00069)*

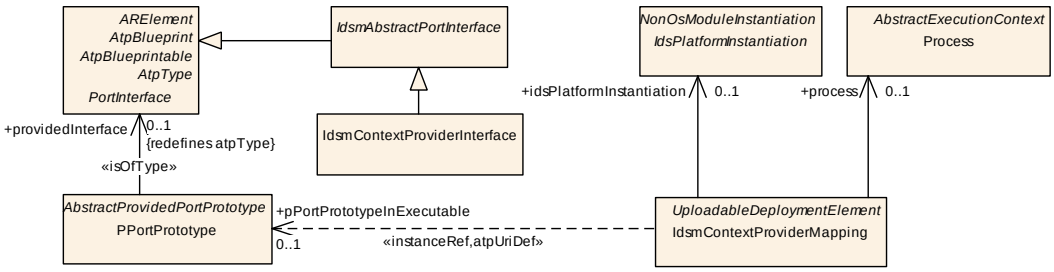


Figure 10.49: Mapping to context provider

[constr_10496]{DRAFT} **Type of `PPortPrototype` referenced in the role `IdsmContextProviderMapping.pPortPrototypeInExecutable`** ⌈A `PPortPrototype` referenced in the role `IdsmContextProviderMapping.pPortPrototypeInExecutable` shall only be typed by a `IdsmContextProviderInterface` **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10497]{DRAFT} **`PPortPrototype` typed by a `IdsmContextProviderInterface`** ⌈A `PPortPrototype` that is typed by a `IdsmContextProviderInterface` shall only be referenced in the role `pPortPrototypeInExecutable` by a `IdsmContextProviderMapping` **`at the time when the creation of the manifest is finished`**.⌋*()*

Class	**IdsmContextProviderMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IntrusionDetectionSystem			
Note	This meta-class represents the ability to define a mapping between an IdsMInstance and a Process on deployment level to a given PortPrototype that is typed by a IdsmContextProviderInterface. **Tags:** atp.recommendedPackage=IdsmProviderMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
idsPlatform Instantiation	IdsPlatformInstantiation	0..1	ref	This represents the IdsM functional cluster. **Tags:** atp.Status=candidate
pPortPrototype InExecutable	PPortPrototype	0..1	iref	This reference identifies the mapped PortPrototype in the application software. **Stereotypes:** atpUriDef **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference identifies the process in which the application runs.

Table 10.101: IdsmContextProviderMapping

[constr_10498]{DRAFT} **Existence of `IdsmContextProviderMapping.idsPlatformInstantiation`** ⌈For each `IdsmContextProviderMapping`, the reference to an `IdsPlatformInstantiation` in the role `idsPlatformInstantiation` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10499]{DRAFT} **Existence of `IdsmContextProviderMapping.pPortPrototypeInExecutable`** ⌈For each `IdsmContextProviderMapping`, the instance reference to a `PPortPrototype` in the role `pPortPrototypeInExecutable` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10500]{DRAFT} **Existence of `IdsmContextProviderMapping.process`** ⌈For each `IdsmContextProviderMapping`, the reference to a `Process` in the role `process` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

10.11.5 Deployment for Security Events

[TPS_MANI_01341]{DRAFT} **Security events that are actually reported by a local IdsM** ⌈The security events that are actually reported by a local IdsM are represented by meta-class `SecurityEventMapping`, referenced by `IdsmModuleInstantiation` in the role `reportableSecurityEvent`.⌋*(RS_MANI_00069)*

[TPS_MANI_01342]{DRAFT} **Semantics of `SecurityEventMapping`** ⌈The semantics of meta-class `SecurityEventMapping` is to identify the

- `PortPrototype` in the context of an `Executable` from which the security event is reported

- Process that runs the Executable, and from the Process the SecurityEventDefinition from which the SecurityEventMapping.id has been derived.

⌋(*RS_MANI_00069*)

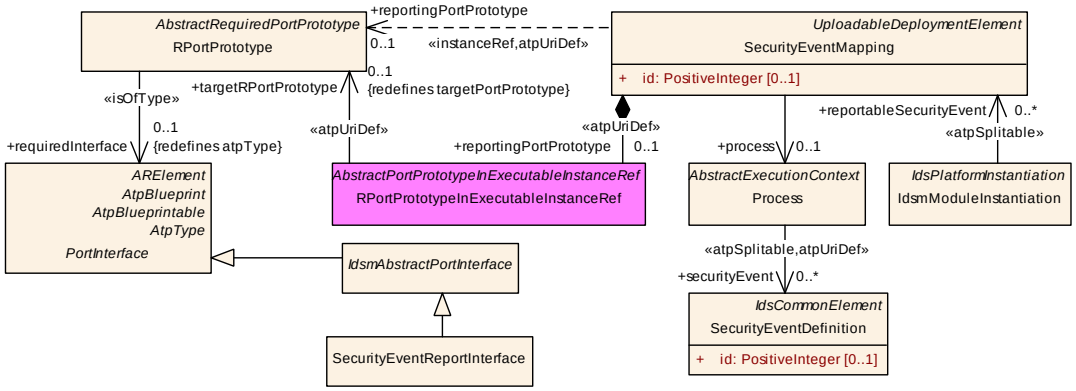


Figure 10.50: Modeling of between a SecurityEventMapping

Class	**SecurityEventMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IntrusionDetectionSystem			
Note	This meta-class represents a reportable instance of a security event. **Tags:** atp.Status=candidate atp.recommendedPackage=SecurityEventMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
id	PositiveInteger	0..1	attr	This attribute defines the numerical identification of the security event subject to deployment. **Tags:** atp.Status=candidate
process	Process	0..1	ref	This reference identifies the process in which context the seurity event is reported. **Tags:** atp.Status=candidate
reportingPort Prototype	RPortPrototype	0..1	iref	This instanceRef identifies the portPrototype over which the security event is reported. **Stereotypes:** atpUriDef **Tags:** atp.Status=candidate **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 10.102: SecurityEventMapping

[constr_10022]{DRAFT} **Restriction for SecurityEventMapping.process.securityEvent.id w.r.t SecurityEventMapping.id** ⌈The value of SecurityEventMapping.id shall also occur in one of the SecurityEventDefinition.id referenced in the role SecurityEventMapping.process.securityEvent **at the time when the creation of the manifest is finished.**⌋()

Rationale for [constr_10022]: during the creation of the IdsM deployment, the value of SecurityEventMapping.id shall be copied from one of the values in SecurityEventMapping.process.securityEvent.id.

10.12 Service-oriented Vehicle Diagnostics

10.12.1 Overview

The SOVD architecture [24] foresees the existence of an SOVD gateway that communicates with:

- an SOVD Client outside the vehicle's network
- the collection of SOVD Servers inside the vehicle's network.

Some of the configuration items of SOVD gateway and SOVD Server can be shared, but some are distinct for one of the two SOVD entities. This aspect is reflected in the formalization as SovdServerInstantiation and SovdGatewayInstantiation.

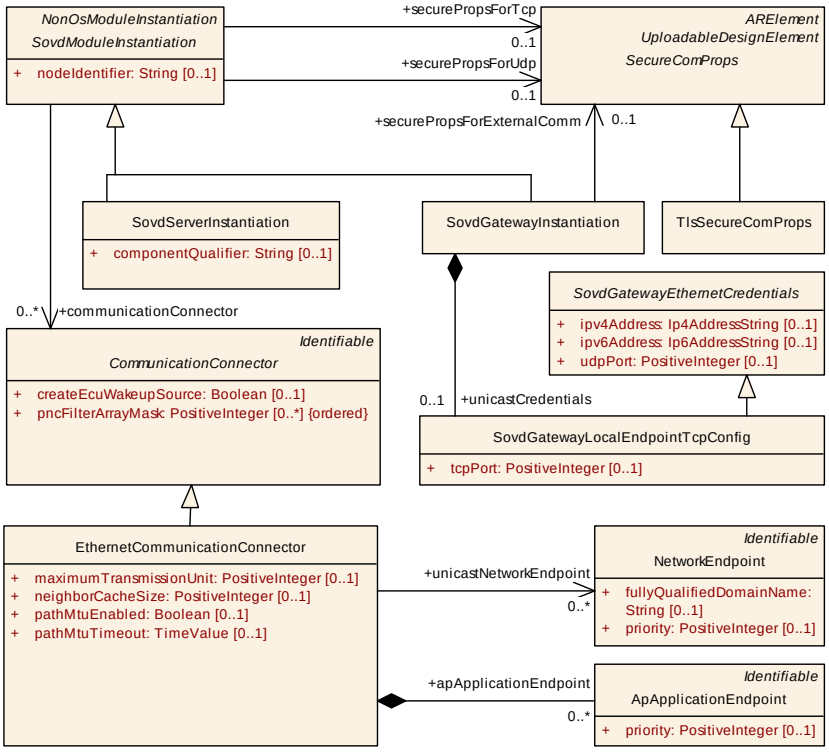


Figure 10.51: Modeling of subclasses of SovdModuleInstantiation

[TPS_MANI_01399]{DRAFT} **Modeling of the configuration of Service-oriented Vehicle Diagnostics** ⌈The deployment-level configuration of SOVD software is represented by abstract meta-class SovdModuleInstantiation and its two sub-classes:

- SovdServerInstantiation: representation of SOVD Server reachable via SOVD somewhere in the vehicle network.

- SovdGatewayInstantiation: representation of the “edge-node” that connects an SOVD Client with the SOVD Servers in the vehicle network. This is a logical role, i.e. the Machine on which an SovdGatewayInstantiation is running could also host one or more SovdServerInstantiations.

⌋(*RS_MANI_00023*, *RS_MANI_00070*)

[TPS_MANI_01400]{DRAFT} **Secure communication between SOVD Gateway and SOVD Server** ⌈The *unicast* communication between SOVD gateway and SOVD Server is modeled using the same concepts as for other vehicle-internal communication, i.e. by reference to a CommunicationConnector in the role SovdModuleInstantiation.communicationConnector.⌋(*RS_MANI_00023*, *RS_MANI_00070*)

[TPS_MANI_01401]{DRAFT} **Support for TLS in vehicle-internal SOVD communication** ⌈The internal SOVD communication can be secured by TLS. The configuration of the TLS credentials is modeled by means of the references

- SovdModuleInstantiation.securePropsForTcp
- SovdModuleInstantiation.securePropsForUdp

⌋(*RS_MANI_00023*, *RS_MANI_00070*)

Class	***SovdModuleInstantiation*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::SOVD			
Note	This abstract meta-class serves as the base class for meta-classes that describe the configuration of an SOVD module. **Tags:** atp.Status=candidate			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Subclasses	SovdGatewayInstantiation, SovdServerInstantiation			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Connector	Communication Connector	*	ref	This reference identifies the collection of communication connectors used by the SOVD module instantiation for vehicle-internal communication. **Tags:** atp.Status=candidate
nodeIdentifier	String	0..1	attr	This attribute represents the local hostname of the SOVD server to be used during the execution of the DNS-SD protocol. **Tags:** atp.Status=candidate
securePropsFor Tcp	SecureComProps	0..1	ref	This reference is used to identify the applicable TCP secure com properties for the external communication of the enclosing SOVD server. **Tags:** atp.Status=candidate
securePropsFor Udp	SecureComProps	0..1	ref	This reference is used to identify the applicable UDP secure com properties for the external communication of the enclosing SOVD server. **Tags:** atp.Status=candidate

Table 10.103: SovdModuleInstantiation

10.12.2 SOVD Server

[TPS_MANI_01402]{DRAFT} **Identification of the SOVD Server** ⌈The identification of the SOVD Server is supported by the configuration of the attribute SovdServerInstantiation.componentQualifier.

In other words, the SOVD Client communicates with the SOVD Server by putting the value of the componentQualifier into the *component-identifier part* of the query URL.⌋*(RS_MANI_00023, RS_MANI_00070)*

Class	SovdServerInstantiation			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::SOVD			
Note	This meta-class represents the configuration of an SOVD server. **Tags:** atp.Status=candidate			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*, *SovdModuleInstantiation*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
componentQualifier	String	0..1	attr	This attributes described the component qualifier used to compose an SOVD query. **Tags:** atp.Status=candidate

Table 10.104: SovdServerInstantiation

[constr_10400]{DRAFT} **Existence of SovdServerInstantiation.componentQualifier** ⌈For each SovdServerInstantiation, attribute componentQualifier shall exist **at the time when the creation of the manifest is finished**.⌋*()*

10.12.3 SOVD Gateway

The SOVD gateway represents the “edge-node” of the SOVD-related internal communication network towards the tester.

This means that the SOVD gateway’s configuration not only consists of the definition of internal communication paths, it also needs to expose credentials for the communication with a SOVD Client as well as the application of search functionality via *multicast DNS* (mDNS) [25] and *DNS Service Discovery* (DNS-SD) [26].

[TPS_MANI_01403]{DRAFT} **External communication of the SOVD gateway** ⌈The specification of external communication with the SOVD Client needs to cover the following cases:

- multicast DNS: usage of standardized mDNS credentials[2].
- DNS-SD: configuration of hostname in the form of SovdModuleInstantiation.nodeIdentifier.

[2]mDNS uses multicast UDP credentials: 224.0.0.251, port 5353 (IPv4) [27] [28] or FF02::FB (IPV6), port 5353 [29] [28], reserved by IANA.

- unicast communication with the diagnostic client: aggregation of SovdGatewayLocalEndpointTcpConfig in the role SovdGatewayInstantiation.unicastCredentials.

⌋(*RS_MANI_00023*, *RS_MANI_00070*)

[TPS_MANI_01404]{DRAFT} **Secure communication between SOVD gateway and SOVD Client** ⌈The communication between SOVD gateway and SOVD Client can be secured by TLS.

The TLS configuration for the SOVD gateway can be created by means of the reference in the role SovdGatewayInstantiation.securePropsForExternalComm.⌋ (*RS_MANI_00023*, *RS_MANI_00070*)

Class	**SovdGatewayInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::SOVD			
Note	This meta-class represents the configuration of an SOVD gateway. **Tags:** atp.Status=candidate			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*, *SovdModuleInstantiation*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
securePropsFor ExternalComm	SecureComProps	0..1	ref	This reference is used to identify the applicable secure com properties for the external communication of the enclosing SOVD gateway. **Tags:** atp.Status=candidate
unicast Credentials	SovdGatewayLocal EndpointTcpConfig	0..1	aggr	This aggragtion is used to to describe the unicast credentials for external (i.e. with a tester) communication of the enclosing SOVD gateway. **Tags:** atp.Status=candidate

Table 10.105: SovdGatewayInstantiation

Class	**SovdGatewayLocalEndpointTcpConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::SOVD			
Note	This meta-class provides the ability to define the TCP configuration of a local endpoint for external communication of an SOVD gateway. **Tags:** atp.Status=candidate			
Base	*ARObject*, *SovdGatewayEthernetCredentials*			
Aggregated by	SovdGatewayInstantiation.unicastCredentials			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
tcpPort	PositiveInteger	0..1	attr	This attribute describes the port number of the port used for TCP communication. **Tags:** atp.Status=candidate

Table 10.106: SovdGatewayLocalEndpointTcpConfig

[constr_10401]{DRAFT} **Existence of SovdGatewayLocalEndpointTcpConfig.tcpPort** ⌈For each SovdGatewayLocalEndpointTcpConfig, attribute tcpPort shall exist **at the time when the creation of the manifest is finished**.⌋ ()

Class	***SovdGatewayEthernetCredentials*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::SOVD			
Note	This meta-class represents the ability to define Ethernet credentials for the purpose of connecting a client to an SOVD gateway. **Tags:** atp.Status=candidate			
Base	*ARObject*			
Subclasses	SovdGatewayLocalEndpointTcpConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ipv4Address	Ip4AddressString	0..1	attr	This attribute represents the IPv4 address for the case that IPv4 is used for communication between the SOVD gateway and a client. **Tags:** atp.Status=candidate
ipv6Address	Ip6AddressString	0..1	attr	This attribute represents the IPv6 address for the case that IPv6 is used for communication between the SOVD gateway and a client. **Tags:** atp.Status=candidate
udpPort	PositiveInteger	0..1	attr	This attribute describes the port number of the port used for UDP communication. **Tags:** atp.Status=candidate

Table 10.107: SovdGatewayEthernetCredentials

[constr_10402]{DRAFT} **Existence of `SovdGatewayEthernetCredentials.ipv4Address` vs. `SovdGatewayEthernetCredentials.ipv6Address`** ⌈For each `SovdGatewayEthernetCredentials`, at least one of attributes

- `SovdGatewayEthernetCredentials.ipv4Address`
- `SovdGatewayEthernetCredentials.ipv6Address`

shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10403]{DRAFT} **Existence of `SovdGatewayEthernetCredentials.udpPort`** ⌈For each `SovdGatewayEthernetCredentials`, attribute `udpPort` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

10.13 Firewall Deployment

AUTOSAR supports an Ethernet packet filtering firewall as an additional security layer. Such a firewall checks, for example, the source and destination IP address, protocols (UDP and TCP) and port addresses. If the address information matches then the packet is considered secured.

Two kinds of firewalls are supported:

1. Stateful packet filtering firewalls
2. Stateless packet filtering firewalls

The stateful firewall monitors the state of network connections and analyzes the complete context of traffic and data packets. The context involves the metadata of the packets (e.g. the ports and IP address of the endpoint and destination) and the state refers to the policy based on the connection state. Approved traffic requests will be added to an approved list and will be allowed to proceed to the network.

The stateless firewall supports a rule-based filtering of traffic based on the evaluation of IP packet parameters without knowledge of the connection status (e.g. TCP session). The filtering can either be an allow list- or a block list-based.

[TPS_MANI_03296] Machine-specific configuration settings for Firewall ⌈The Machine-specific configuration settings for the Firewall module are collected in `AdaptiveFirewallModuleInstantiation`.⌋*()*

Class	**AdaptiveFirewallModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	This meta-class defines the attributes for the Firewall configuration on a specific machine. **Tags:** atp.Status=candidate			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
stateDep Firewall	StateDependentFirewall	*	ref	Firewall rules that are defined in a firewall state. **Tags:** atp.Status=candidate

Table 10.108: AdaptiveFirewallModuleInstantiation

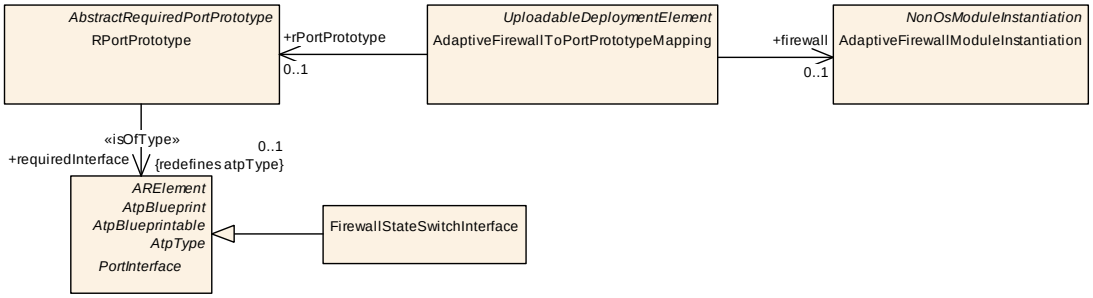


Figure 10.52: `AdaptiveFirewallModuleInstantiation` and the relationship to Application with a `RPortPrototype` typed by a `FirewallStateSwitchInterface`

Class	**AdaptiveFirewallToPortPrototypeMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall
Note	This meta-class maps the AdaptiveFirewall moduleInstantiation to the RPortPrototype that is typed by a FirewallModeSwitchInterface. **Tags:** atp.Status=candidate atp.recommendedPackage=AdaptiveFirewallToPortPrototypeMappings
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*

▽

△

Class	AdaptiveFirewallToPortPrototypeMapping			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
firewall	AdaptiveFirewallModule Instantiation	0..1	ref	Reference to the Firewall module **Tags:** atp.Status=candidate
rPortPrototype	RPortPrototype	0..1	ref	Reference to RPortPrototype typed by a FirewallMode SwitchInterface **Tags:** atp.Status=candidate

Table 10.109: AdaptiveFirewallToPortPrototypeMapping

[TPS_MANI_03297] Semantics of `AdaptiveFirewallToPortPrototypeMapping` ⌈Meta-class `AdaptiveFirewallToPortPrototypeMapping` has the ability to map a specific `RPortPrototype` typed by the `FirewallStateSwitchInterface` to the configured `AdaptiveFirewallModuleInstantiation`.⌋*()*

[constr_5358] `AdaptiveFirewallToPortPrototypeMapping.rPortPrototype` restriction ⌈The `AdaptiveFirewallToPortPrototypeMapping` is only allowed to reference a `RPortPrototype` that is typed by the `FirewallStateSwitchInterface`. This rule shall be imposed **at the time when the creation of the manifest is finished**.⌋*()*

Firewall functionality can be implemented on Host ECUs and on switches (as network firewalls). This is the reason why the `CouplingElement` (that defines a Switch) and the `Machine` (via the `AdaptiveFirewallModuleInstantiation`) both have an optional reference to the `StateDependentFirewall` that defines the entry point for the Firewall configuration.

The in-vehicle traffic may depend on the current state of the firewall statemachine. The firewall rules are adaptable to this and therefore the `StateDependentFirewall` is associated with a `firewallState`.

[TPS_MANI_03298] Semantics of `StateDependentFirewall` ⌈If the `firewallState` that is referenced by the `StateDependentFirewall` is active, then the firewall rule that is defined by the `StateDependentFirewall` shall be considered as active as well.⌋*()*

[constr_5348] Mandatory `initialMode` in `ModeDeclarationGroup` that is referenced by `StateDependentFirewall` ⌈The `ModeDeclarationGroup` that is referenced via a `ModeDeclaration` from `StateDependentFirewall` in the role `firewallState` shall define an `initialMode` **at the time when the creation of the manifest is finished**.⌋*()*

This `initialMode` is used in the case that the system boots up and no FirewallMode is yet set by an application.

[constr_5349] Mandatory `defaultAction` in `StateDependentFirewall` ⌈The `StateDependentFirewall` shall always define the attribute `defaultAction` **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_03299] Semantics of `FirewallRuleProps` ⌈Each `StateDependentFirewall` defines `FirewallRuleProps` that in turn define a rule expression (`matchingIngressRule` and `matchingEgressRule`) and an `action` that is performed in case that the rule expression matches.⌋*()*

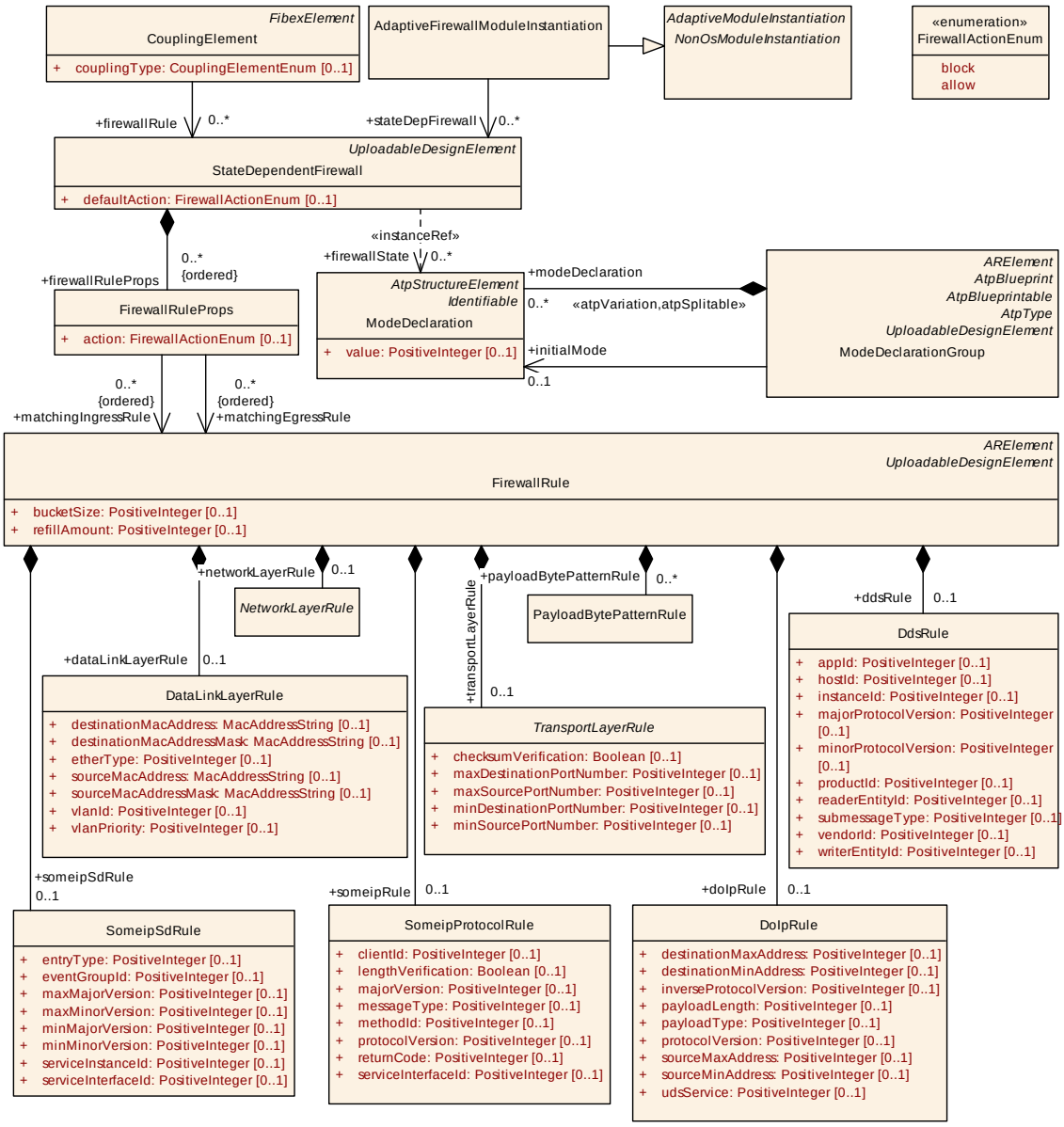


Figure 10.53: Modeling of the Firewall

[constr_5350] Mandatory `action` in `FirewallRuleProps` ⌈The `FirewallRuleProps` shall always define the attribute `action` **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_03300] Semantics of `FirewallRule` ⌈The `FirewallRule` defines an expression which matches network traffic against the conditions that are defined by the expression.⌋*()*

[TPS_MANI_03301] Semantics of `FirewallRule.refillAmount` and `FirewallRule.bucketSize` ⌈Every firewall matching rule can be associated with a rate limit for limiting network traffic. In AUTOSAR, the rate limit is implemented by a leaky bucket algorithm. The attributes `FirewallRule.refillAmount` and `FirewallRule.bucketSize` are used to configure the rate limit algorithm.⌋*()*

[constr_5351] `FirewallRule` is allowed to aggregate at most one protocol subelement ⌈A `FirewallRule` is allowed to aggregate either:

- `someipSdRule`
- `someipRule`
- `doIpRule`

This rule shall be imposed **`at the time when the creation of the manifest is finished`**.⌋*()*

If a firewall rule needs to cover different different Protocols (e.g. SOME/IP, DoIP) then several `FirewallRule`s need to be defined because of [constr_5351]. Please note that a `FirewallRule` is allowed to aggregate a `DataLinkLayerRule`, a `NetworkLayerRule` and a `TransportLayerRule` at the same time in addition to one of the subelements mentioned in [constr_5351].

Class	**StateDependentFirewall**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Firewall rules that are defined in a firewall state **Tags:** atp.Status=candidate atp.recommendedPackage=StateDependentFirewallRules			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultAction	FirewallActionEnum	0..1	attr	This attribute defines a defaultAction in case that the VehicleMode is not yet set. **Tags:** atp.Status=candidate
firewallRule Props (ordered)	FirewallRuleProps	*	aggr	Collection of firewall rules that apply in the vehicle mode **Tags:** atp.Status=candidate
firewallState	ModeDeclaration	*	iref	Reference to firewall states in which the Firewall is active. If one of the referenced ModeDeclarations is the current firewall state then the firewall rule shall be considered as active. **Tags:** atp.Status=candidate **InstanceRef implemented by:** FirewallStateInFirwall StateSwitchInterfaceInstanceRef

Table 10.110: StateDependentFirewall

Class	FirewallRule			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Firewall Rule that defines the control information in individual packets. **Tags:** atp.Status=candidate atp.recommendedPackage=FirewallRules			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
bucketSize	PositiveInteger	0..1	attr	This attribute defines the capacity of the queue for rate limitation (leaky-bucket Algorithm). **Tags:** atp.Status=candidate
dataLinkLayer Rule	DataLinkLayerRule	0..1	aggr	Configuration of rules on the Data Link Layer **Tags:** atp.Status=candidate
ddsRule	DdsRule	0..1	aggr	Configuration of firewall rules for DDS. **Tags:** atp.Status=candidate
doIpRule	DoIpRule	0..1	aggr	Configuration of firewall rules for DoIP messages **Tags:** atp.Status=candidate
networkLayer Rule	NetworkLayerRule	0..1	aggr	Configuration of rules on the Network Layer **Tags:** atp.Status=candidate
payloadByte PatternRule	PayloadBytePattern Rule	*	aggr	Configuration of generic firewall rules **Tags:** atp.Status=candidate
refillAmount	PositiveInteger	0..1	attr	This attribute defines the output rate that describes how many packets leave the queue per second (leaky-bucket Algorithm). **Tags:** atp.Status=candidate
someipRule	SomeipProtocolRule	0..1	aggr	Configuration of firewall rules for SOME/IP messages **Tags:** atp.Status=candidate
someipSdRule	SomeipSdRule	0..1	aggr	Configuration of firewall rules for SOME/IP Service Discovery messages **Tags:** atp.Status=candidate
transportLayer Rule	TransportLayerRule	0..1	aggr	Configuration of rules on the Transport Layer **Tags:** atp.Status=candidate

Table 10.111: FirewallRule

Class	FirewallRuleProps			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Firewall rule that is defined by an action that is performed if the referenced pattern matches. **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	StateDependentFirewall.firewallRuleProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
action	FirewallActionEnum	0..1	attr	Action that is performed by the firewall if the matching Rule is fulfilled. **Tags:** atp.Status=candidate

△

Class	**FirewallRuleProps**			
matchingEgress Rule (ordered)	FirewallRule	*	ref	This element defines an egress rule expression against which the network traffic is matched. **Tags:** atp.Status=candidate
matching IngressRule (ordered)	FirewallRule	*	ref	This element defines an ingress rule expression against which the network traffic is matched. **Tags:** atp.Status=candidate

Table 10.112: FirewallRuleProps

Enumeration	**FirewallActionEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall
Note	List of actions that the Firewall is able to perform. **Tags:** atp.Status=candidate
Aggregated by	FirewallRuleProps.action, StateDependentFirewall.defaultAction
Literal	*Description*
allow	Firewall allows the communication **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=candidate
block	Firewall blocks the communication **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=candidate

Table 10.113: FirewallActionEnum

10.13.1 Firewall rules on Datalink layer

[TPS_MANI_03302] Firewall rules on Data Link Layer ⌈The meta-class `DataLinkLayerRule` is used to define firewall rules on the Data Link Layer (second layer of the seven-layer OSI model).⌋ *()*

[TPS_MANI_03303] Filtering of packets with a single MAC address and MAC address range ⌈The `sourceMacAddressMask` is specifying which bits in the `sourceMacAddress` are static to define a source MAC Address or source MAC Address Range. For each bit position in the MAC mask a 1 indicates that the corresponding address bit is significant and a 0 indicates that the address bit is ignored. `destinationMacAddressMask` and `destinationMacAddress` shall be used in the same way to define a destination MAC Address or destination MAC Address Range.⌋ *()*

If, for example, only the first four octets of a MAC address are relevant, a `sourceMacAddressMask` of ff:ff:ff:ff:00:00 is used. A `sourceMacAddressMask` of ff:ff:ff:ff:ff:ff defines that all address bits are relevant and is used to define a single MAC address.

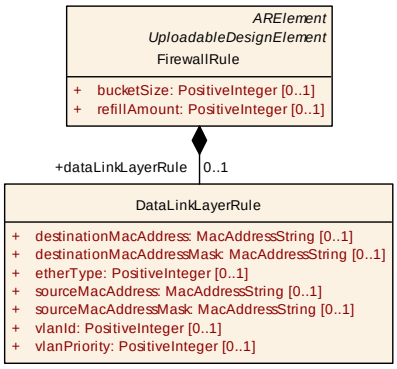


Figure 10.54: Modeling of Firewall rules on Data Link Layer

Class	**DataLinkLayerRule**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of filter rules on the DataLink layer **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	FirewallRule.dataLinkLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destinationMac Address	MacAddressString	0..1	attr	Filter to match packets with the destination MAC address. **Tags:** atp.Status=candidate
destinationMac AddressMask	MacAddressString	0..1	attr	Filter to match packets with the destination MAC address range. The destinationMacAddress with the destination MacAddressMask defines the MAC address range. **Tags:** atp.Status=candidate
etherType	PositiveInteger	0..1	attr	Filter to match packets based on the EtherType field in the Ethernet frame. The EtherType is used to indicate which protocol is encapsulated in the payload of the frame. **Tags:** atp.Status=candidate
sourceMac Address	MacAddressString	0..1	attr	Filter to match packets with the source MAC address. **Tags:** atp.Status=candidate
sourceMac AddressMask	MacAddressString	0..1	attr	Filter to match packets with the source MAC address range. The sourceMacAddress with the sourceMac AddressMask defines the MAC address range. **Tags:** atp.Status=candidate
vlanId	PositiveInteger	0..1	attr	Filter of packets with a specific VlanId. **Tags:** atp.Status=candidate
vlanPriority	PositiveInteger	0..1	attr	Filter of packets with a specific Vlan priority. **Tags:** atp.Status=candidate

Table 10.114: DataLinkLayerRule

10.13.2 Firewall rules on Network layer

[TPS_MANI_03304] Firewall rules on Network Layer ⌈The meta-class `NetworkLayerRule` is used to define firewall rules on the Network Layer (third layer of the seven-layer OSI model).⌋*()*

[TPS_MANI_03305] Filtering of packets with a single IP address and IP address range ⌈The `Ipv4Rule.sourceNetworkMask` (or `Ipv6Rule.sourceNetworkMask`) is specifying which bits in the `Ipv4Rule.sourceIpAddress` (or `Ipv6Rule.sourceIpAddress`) are static to define a source IP Address or source IP Address Range. `destinationIpAddress` and `destinationNetworkMask` shall be used in the same way to define a destination IP Address or destination IP Address Range.⌋*()*

If for example only the first three octets of a IP address are relevant, a `sourceNetworkMask` of 255.255.255.0 is used. The 255 octets in the mask define that the corresponding numbers in the IP address are static and never change. In this example the first three octets are fixed and the fourth octet is 0. This means that on the fourth octet all individual numbers from 1-254 are valid.

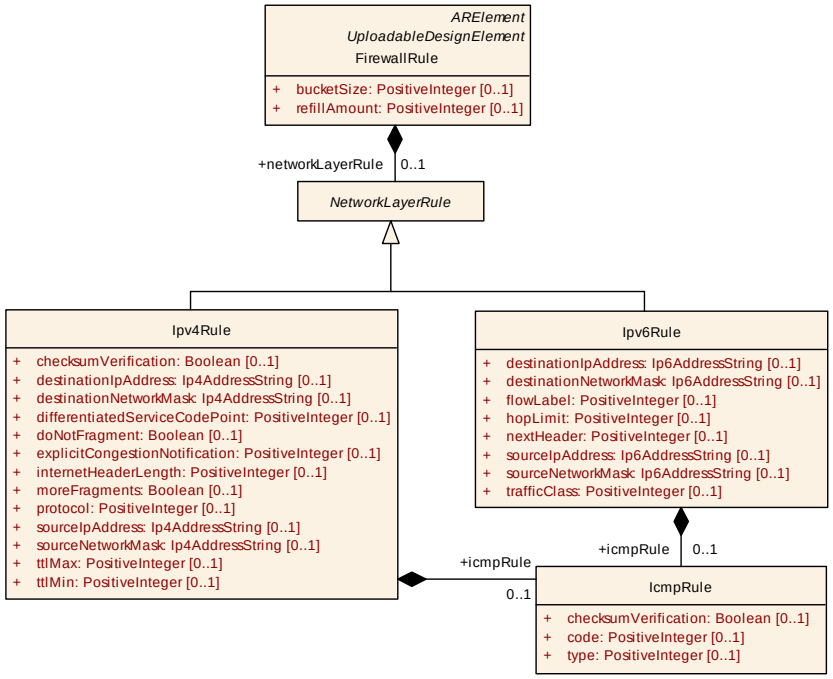


Figure 10.55: Modeling of Firewall rules on Network Layer

Class	***NetworkLayerRule*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of filter rules on the Network layer **Tags:** atp.Status=candidate			
Base	*ARObject*			
Subclasses	Ipv4Rule, Ipv6Rule			
Aggregated by	FirewallRule.networkLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.115: NetworkLayerRule

Class	**Ipv4Rule**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of filter rules on IPv4 level. **Tags:** atp.Status=candidate			
Base	*ARObject*, *NetworkLayerRule*			
Aggregated by	FirewallRule.networkLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checksum Verification	Boolean	0..1	attr	Defines whether a Ipv4 header checksum verification is performed or not. **Tags:** atp.Status=candidate
destinationIp Address	Ip4AddressString	0..1	attr	Filter to match packets with the destination IPv4 address. **Tags:** atp.Status=candidate
destination NetworkMask	Ip4AddressString	0..1	attr	Filter to match packets with the destination IPv4 address range. The destinationIpAddress with the destination NetworkMask defines the IP address range. **Tags:** atp.Status=candidate
differentiated ServiceCode Point	PositiveInteger	0..1	attr	Filter to match packets with a DSCP value. **Tags:** atp.Status=candidate
doNotFragment	Boolean	0..1	attr	Filter to match packets that have the doNotFragment bit in the Header set. **Tags:** atp.Status=candidate
explicit Congestion Notification	PositiveInteger	0..1	attr	Filter to match packets with a ECN code point. **Tags:** atp.Status=candidate
icmpRule	IcmpRule	0..1	aggr	Configuration of filter rules for ICMP (Internet Control Message Protocol). **Tags:** atp.Status=candidate
internetHeader Length	PositiveInteger	0..1	attr	Filter to match packets with a minimum ipv4 header length. **Tags:** atp.Status=candidate
moreFragments	Boolean	0..1	attr	Filter to match packets that have the moreFragments flag in the Header set. **Tags:** atp.Status=candidate
protocol	PositiveInteger	0..1	attr	Filter to match packets with a IP protocol number . **Tags:** atp.Status=candidate
sourceIp Address	Ip4AddressString	0..1	attr	Filter to match packets with the source IPv4 address. **Tags:** atp.Status=candidate
sourceNetwork Mask	Ip4AddressString	0..1	attr	Filter to match packets with the source IPv4 address range. The sourceIpAddress with the sourceNetwork Mask defines the IP address range. **Tags:** atp.Status=candidate
ttlMax	PositiveInteger	0..1	attr	Filter to match packets with a maximum ttl value (TimeTo Live defines the lifetime of data on the network). **Tags:** atp.Status=candidate
ttlMin	PositiveInteger	0..1	attr	Filter to match packets with a minimum ttl value (TimeTo Live defines the lifetime of data on the network). **Tags:** atp.Status=candidate

Table 10.116: Ipv4Rule

Class	**Ipv6Rule**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of filter rules on IPv6 level. **Tags:** atp.Status=candidate			
Base	*ARObject*, *NetworkLayerRule*			
Aggregated by	FirewallRule.networkLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destinationIp Address	Ip6AddressString	0..1	attr	Filter to match packets with the destination IPv6 address. **Tags:** atp.Status=candidate
destination NetworkMask	Ip6AddressString	0..1	attr	Filter to match packets with the destination IPv6 address range. The destinationIpAddress with the destination NetworkMask defines the MAC address range. **Tags:** atp.Status=candidate
flowLabel	PositiveInteger	0..1	attr	Filter to match packets with a defined flow label. **Tags:** atp.Status=candidate
hopLimit	PositiveInteger	0..1	attr	Filter to match packets with a minimum hop limit. **Tags:** atp.Status=candidate
icmpRule	IcmpRule	0..1	aggr	Configuration of filter rules for ICMP (Internet Control Message Protocol). **Tags:** atp.Status=candidate
nextHeader	PositiveInteger	0..1	attr	Filter to match packets with a defined type of an extension header. **Tags:** atp.Status=candidate
sourceIp Address	Ip6AddressString	0..1	attr	Filter to match packets with the source IPv6 address. **Tags:** atp.Status=candidate
sourceNetwork Mask	Ip6AddressString	0..1	attr	Filter to match packets with the source IPv6 address range. The sourceIpAddress with the sourceNetwork Mask defines the IP address range. **Tags:** atp.Status=candidate
trafficClass	PositiveInteger	0..1	attr	Filter to match packets with a defined traffic class or priority. **Tags:** atp.Status=candidate

Table 10.117: Ipv6Rule

Class	**IcmpRule**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of filter rules for ICMP (Internet Control Message Protocol). **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	Ipv4Rule.icmpRule, Ipv6Rule.icmpRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checksum Verification	Boolean	0..1	attr	Defines whether a Icmp header checksum verification is performed or not. **Tags:** atp.Status=candidate
code	PositiveInteger	0..1	attr	Filter to match packets with the Icmp code. **Tags:** atp.Status=candidate
type	PositiveInteger	0..1	attr	Filter to match packets with the Icmp type. **Tags:** atp.Status=candidate

Table 10.118: IcmpRule

10.13.3 Firewall rules on Transport layer

[TPS_MANI_03306] Firewall rules on Transport Layer ⌈The meta-class `TransportLayerRule` is used to define firewall rules on the Transport Layer (fourth layer of the seven-layer OSI model).⌋*()*

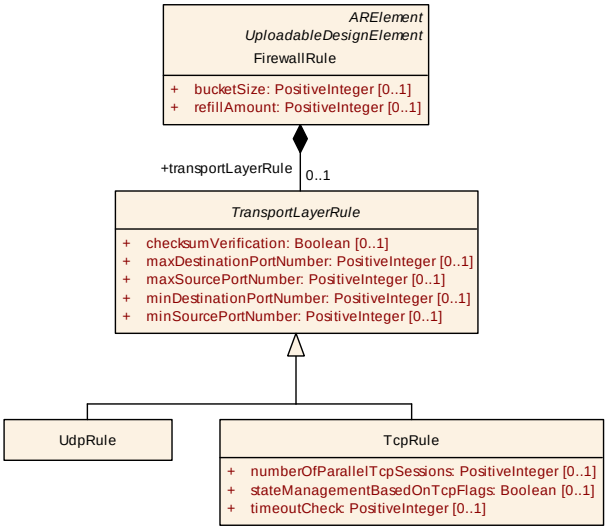


Figure 10.56: Modeling of Firewall rules on Transport Layer

Class	***TransportLayerRule*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of filter rules on Transport Layer level. **Tags:** atp.Status=candidate			
Base	*ARObject*			
Subclasses	TcpRule, UdpRule			
Aggregated by	FirewallRule.transportLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checksum Verification	Boolean	0..1	attr	Defines whether checksum verification is performed or not. **Tags:** atp.Status=candidate
maxDestination PortNumber	PositiveInteger	0..1	attr	Filter to match packets with the maximum destination UDP/TCP port number. **Tags:** atp.Status=candidate
maxSourcePort Number	PositiveInteger	0..1	attr	Filter to match packets with the maximum source UDP/ TCP port number. **Tags:** atp.Status=candidate
minDestination PortNumber	PositiveInteger	0..1	attr	Filter to match packets with the minimum destination UDP/TCP port number. **Tags:** atp.Status=candidate
minSourcePort Number	PositiveInteger	0..1	attr	Filter to match packets with the minimum source UDP/ TCP port number. **Tags:** atp.Status=candidate

Table 10.119: TransportLayerRule

Class	**UdpRule**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of UDP filter rules. **Tags:** atp.Status=candidate			
Base	*ARObject*, *TransportLayerRule*			
Aggregated by	FirewallRule.transportLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.120: UdpRule

Class	**TcpRule**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of TCP filter rules. **Tags:** atp.Status=candidate			
Base	*ARObject*, *TransportLayerRule*			
Aggregated by	FirewallRule.transportLayerRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
numberOf ParallelTcp Sessions	PositiveInteger	0..1	attr	This attribute defines the maximal number of TCP Sessions that are allowed to be established. **Tags:** atp.Status=candidate
state Management BasedOnTcp Flags	Boolean	0..1	attr	This attribute defines whether the StateManagement is based on TCP flags or not. **Tags:** atp.Status=candidate
timeoutCheck	PositiveInteger	0..1	attr	This attribute defines the TCP Session timeout in seconds **Tags:** atp.Status=candidate

Table 10.121: TcpRule

10.13.4 Firewall rules via byte pattern

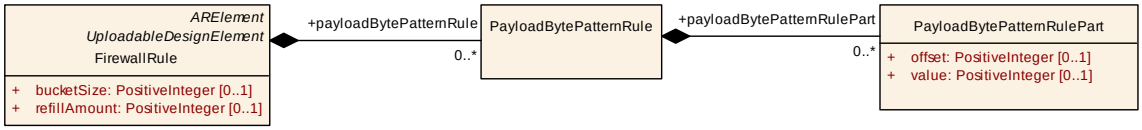


Figure 10.57: Modeling of Firewall rules via byte pattern

[TPS_MANI_03307] Byte-Pattern Firewall rule ⌈The meta-class `PayloadBytePatternRule` is used to define a generic firewall rule that specifies individually the value for each byte of the Ethernet frame. Each `PayloadBytePatternRulePart` that is aggregated by `PayloadBytePatternRule` defines one byte in the Ethernet frame. The `offset` defines the position in the Ethernet Frame and the `value` defines the byteValue.⌋*()*

Class	PayloadBytePatternRule			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of a generic firewall rule that defines the individual bytes of a message that shall match. **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	FirewallRule.payloadBytePatternRule			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
payloadByte PatternRulePart	PayloadBytePattern RulePart	*	aggr	Configuration of bytes in the message, **Tags:** atp.Status=candidate

Table 10.122: PayloadBytePatternRule

Class	PayloadBytePatternRulePart			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of one byte in the datagram, **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	PayloadBytePatternRule.payloadBytePatternRulePart			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
offset	PositiveInteger	0..1	attr	This attribute defines the byte offset in the datagram (start byte of the Ethernet frame, i.e. offset 0 corresponds to the first byte of the destination MAC address). **Tags:** atp.Status=candidate
value	PositiveInteger	0..1	attr	This attribute defines the byteValue (0..255) in the datagram. **Tags:** atp.Status=candidate

Table 10.123: PayloadBytePatternRulePart

10.13.5 Firewall deep packet inspection of SOME/IP SD and SOME/IP messages

[TPS_MANI_03308] Deep inspection of SOME/IP SD messages ⌈The meta-class `SomeipSdRule` is used to define a firewall rule for SOME/IP SD messages.⌋*()*

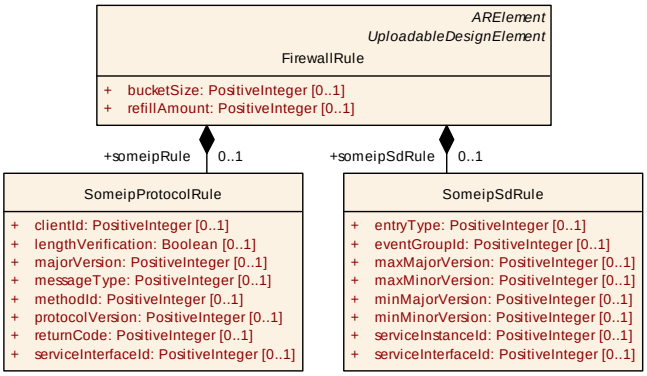


Figure 10.58: Modeling of SOME/IP Firewall rules

[TPS_MANI_03309] Deep inspection of SOME/IP messages ⌈The meta-class `SomeipProtocolRule` is used to define a firewall rule for SOME/IP messages.⌋*()*

Class	**SomeipProtocolRule**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of SOME/IP firewall rules **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	FirewallRule.someipRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientId	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the clientId in the SOME/IP header matches. **Tags:** atp.Status=candidate
length Verification	Boolean	0..1	attr	Defines whether length verification is performed or not. **Tags:** atp.Status=candidate
majorVersion	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the majorVersion in the SOME/IP header matches. **Tags:** atp.Status=candidate
messageType	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the messageType in the SOME/IP header matches. **Tags:** atp.Status=candidate
methodId	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the methodId in the SOME/IP header matches. **Tags:** atp.Status=candidate
protocolVersion	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the protocol Version in the SOME/IP header matches. **Tags:** atp.Status=candidate
returnCode	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the returnCode in the SOME/IP header matches. **Tags:** atp.Status=candidate
serviceInterface Id	PositiveInteger	0..1	attr	Filter for SOME/IP messages in which the service InterfaceId in the SOME/IP header matches. **Tags:** atp.Status=candidate

Table 10.124: SomeipProtocolRule

Class	**SomeipSdRule**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of SOME/IP Service Discovery firewall rules **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	FirewallRule.someipSdRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
entryType	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the entryType in the SOME/IP header matches. **Tags:** atp.Status=candidate

▽

△

Class	SomeipSdRule			
eventGroupId	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the eventGroup Id in the SOME/IP header matches. **Tags:** atp.Status=candidate
maxMajor Version	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the Major Version in the SOME/IP header is smaller or equal than maxMajorVersion. **Tags:** atp.Status=candidate
maxMinor Version	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the Minor Version in the SOME/IP header is smaller or equal than maxMinorVersion. **Tags:** atp.Status=candidate
minMajor Version	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the Major Version in the SOME/IP header is greater or equal than minMajorVersion. **Tags:** atp.Status=candidate
minMinor Version	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the Minor Version in the SOME/IP header is greater or equal than minMinorVersion. **Tags:** atp.Status=candidate
serviceInstance Id	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the service InstanceId in the SOME/IP header matches. **Tags:** atp.Status=candidate
serviceInterface Id	PositiveInteger	0..1	attr	Filter for SOME/IP SD messages in which the service InterfaceId in the SOME/IP header matches. **Tags:** atp.Status=candidate

Table 10.125: SomeipSdRule

10.13.6 Firewall deep packet inspection of DoIP messages

[TPS_MANI_03310] Deep inspection of DoIP messages ⌈The meta-class `DoIpRule` is used to define a firewall rule for DoIP messages.⌋*()*

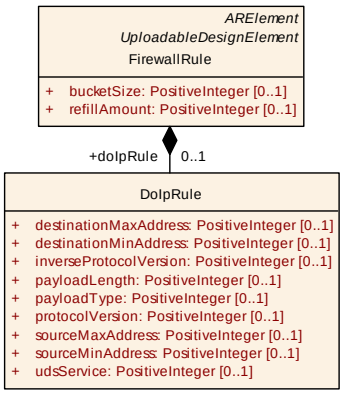


Figure 10.59: Modeling of DoIP Firewall rules

Class	**DoIpRule**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall			
Note	Configuration of a generic firewall rule **Tags:** atp.Status=candidate			
Base	*ARObject*			
Aggregated by	FirewallRule.doIpRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destinationMax Address	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the destination Address is smaller or equal than destinationMaxAddress. **Tags:** atp.Status=candidate
destinationMin Address	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the destination Address is greater or equal than destinationMinAddress. **Tags:** atp.Status=candidate
inverseProtocol Version	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the inverseprotocolVersion in the DoIP header matches. **Tags:** atp.Status=candidate
payloadLength	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the payload Length in the DoIP header matches. **Tags:** atp.Status=candidate
payloadType	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the payloadType in the DoIP header matches. **Tags:** atp.Status=candidate
protocolVersion	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the protocol Version in the DoIP header matches. **Tags:** atp.Status=candidate
sourceMax Address	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the source Address is smaller or equal than sourceMaxAddress. **Tags:** atp.Status=candidate
sourceMin Address	PositiveInteger	0..1	attr	Filter to match DoIP messages in which the source Address is greater or equal than sourceMinAddress.. **Tags:** atp.Status=candidate
udsService	PositiveInteger	0..1	attr	Filter to match DoIP messages that contain the uds Service. **Tags:** atp.Status=candidate

Table 10.126: DoIpRule

10.13.7 Firewall deep packet inspection of DDS messages

[TPS_MANI_03311] Deep inspection of DDS messages ⌈The meta-class `DdsRule` is used to define a firewall rule for DDS messages.⌋*()*

[constr_5352] **`DdsRule.submessageType` value restriction** ⌈The value of `DdsRule.submessageType` is restricted to the following values:

- 0x01 (PAD)
- 0x06 (ACKNACK)
- 0x07 (HEARTBEAT)

- 0x08 (GAP)
- 0x09 (INFO_TS)
- 0x0c (INFO_SRC)
- 0x0d (INFO_REPLY_IP4)
- 0x0e (INFO_DST)
- 0x0f (INFO_REPLY)
- 0x12 (NACK_FRAG)
- 0x13 (HEARTBEAT_FRAG)
- 0x15 (DATA)
- 0x16 (DATA_FRAG)

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

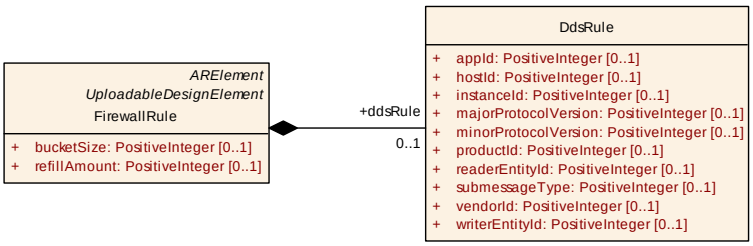


Figure 10.60: Modeling of DDS Firewall rules

[constr_5353] `DdsRule.readerEntityId` and `DdsRule.writerEntityId` value restriction ⌈The value of `DdsRule.readerEntityId` and `DdsRule.writerEntityId` is only allowed to be set if the value of `DdsRule.submessageType` is set to one of the following values:

- 0x06 (ACKNACK)
- 0x07 (HEARTBEAT)
- 0x08 (GAP)
- 0x15 (DATA)

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

Class	**DdsRule**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Firewall
Note	Configuration of a DDS firewall rule **Tags:** atp.Status=candidate

▽

△

Class	**DdsRule**			
Base	*ARObject*			
Aggregated by	FirewallRule.ddsRule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
appId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the appId in the DDSI-RTPS header and the INFO_DST (0x0E) submessage matches. **Tags:** atp.Status=candidate
hostId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the hostId in the DDSI-RTPS header and the INFO_DST (0x0E) submessage matches. **Tags:** atp.Status=candidate
instanceId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the instanceId in the DDSI-RTPS header and the INFO_DST (0x0E) submessage matches. **Tags:** atp.Status=candidate
majorProtocol Version	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the major ProtocolVersion in the DDSI-RTPS header matches. **Tags:** atp.Status=candidate
minorProtocol Version	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the minor ProtocolVersion in the DDSI-RTPS header matches. **Tags:** atp.Status=candidate
productId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the productId in the DDSI-RTPS header matches. **Tags:** atp.Status=candidate
readerEntityId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the readerEntity ID in a DDSI-RTPS submessage matches **Tags:** atp.Status=candidate
submessage Type	PositiveInteger	0..1	attr	Defines the allowed submessage type in the DDSI-RTPS message **Tags:** atp.Status=candidate
vendorId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the vendorId in the DDSI-RTPS header matches. **Tags:** atp.Status=candidate
writerEntityId	PositiveInteger	0..1	attr	Filter for DDSI-RTPS messages in which the writerEntity ID in a DDSI-RTPS submessage matches **Tags:** atp.Status=candidate

Table 10.127: DdsRule

10.14 State Management Deployment

10.14.1 Overview

On the *AUTOSAR adaptive platform*, state management can be implemented as a mixture of generated and manually written code. The generated code is created based on a model that is described in this chapter.

Another aspect of this hybrid nature of the state management is that it sometimes acts like an application and sometimes it acts like a functional cluster.

The entry point for the deployment-level configuration of the State Management on the AUTOSAR adaptive platform is represented by meta-class `StateManagementModuleInstantiation`, a sub-class of `NonOsModuleInstantiation`.

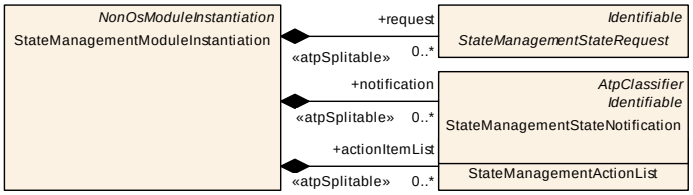


Figure 10.61: Modeling of the `StateManagementModuleInstantiation`

Class	**StateManagementModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This meta-class represents the deployment-level configuration of the state management on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
actionItemList	StateManagement ActionList	*	aggr	This represents the collection of action item lists defined in the context of the enclosing state management module. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=actionItemList.shortName atp.Status=draft
notification	StateManagementState Notification	*	aggr	This aggregation represents the state switch notifications handled by the state manager. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=notification.shortName atp.Status=draft
request	StateManagementState Request	*	aggr	This aggregation represents the state requests handled by the state manager. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=request.shortName atp.Status=draft

Table 10.128: StateManagementModuleInstantiation

The job of the State Management on the AUTOSAR adaptive platform is to control two kinds of states:

- the state of a `Function Group`, modeled as a `ModeDeclarationGroupPrototype` owned by a `FunctionGroupSet`, that is referenced by a `SoftwareCluster` in the role `claimedFunctionGroup`.

Note that these states are only indirectly controlled, because the actual state switch for Function Groups is in the responsibility of the Execution Management.

- the state of an “internal” state machine, owned by StateManagementStateNotifications.

10.14.2 Requests to State Management

[TPS_MANI_01391]{DRAFT} **Semantics of StateManagementStateRequest** ⌈Meta-class StateManagementStateRequest that is aggregated directly at the StateManagementModuleInstantiation represents the ability to configure requests to State Management.⌋ *(RS_MANI_00023)*

Please note that meta-class StateManagementStateRequest is modeled as an abstract class so that the concrete sub-classes that correspond to different flavors of requests (which are also respected for the definition of applicable PortInterfaces, see section 3.3.4) can be individually considered.

In particular, the State Management accepts requests for state changes in two different flavors:

- “triggers” (i.e. request for a state change from select functional clusters), modeled by mean of meta-class StateManagementRequestTrigger, see section 10.14.2.1.
- “errors” (i.e. error report from a functional cluster), modeled by means of meta-class StateManagementRequestError, see section 10.14.2.2.

As depicted in Figure 10.62, StateManagementStateRequest also references an RPortPrototype as the source of inputs from outside the State Management.

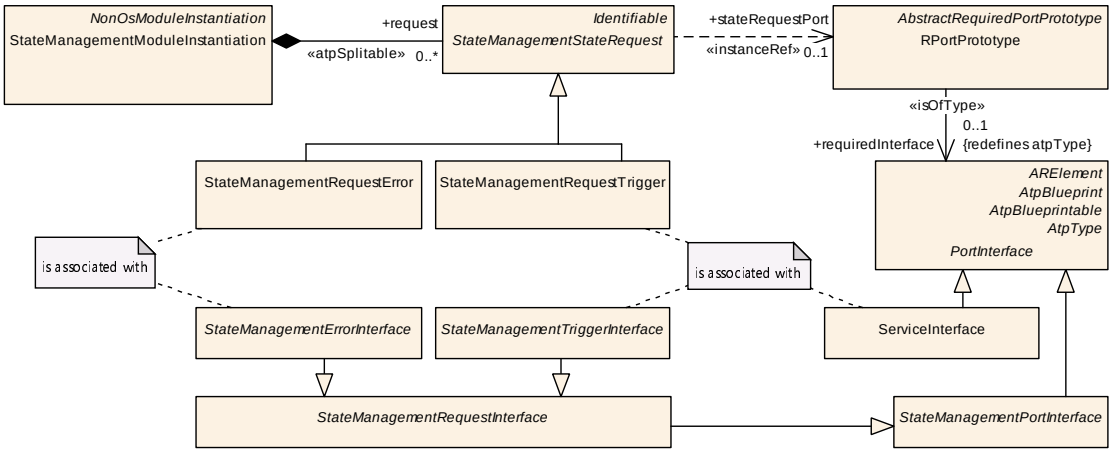


Figure 10.62: Modeling of StateManagementStateRequest

Figure 10.63 also shows that the configuration of State Management on the AUTOSAR adaptive platform is modeled as a hybrid of deployment-level `StateManagementModuleInstantiation` and the definition of an `Executable` that is executed as a `Process`.

The connection between the `StateManagementModuleInstantiation` and the `Executable`/`Process` is created by `ProcessToMachineMapping` that references both the `StateManagementModuleInstantiation` and the `Process`, as depicted in Figure 10.1.

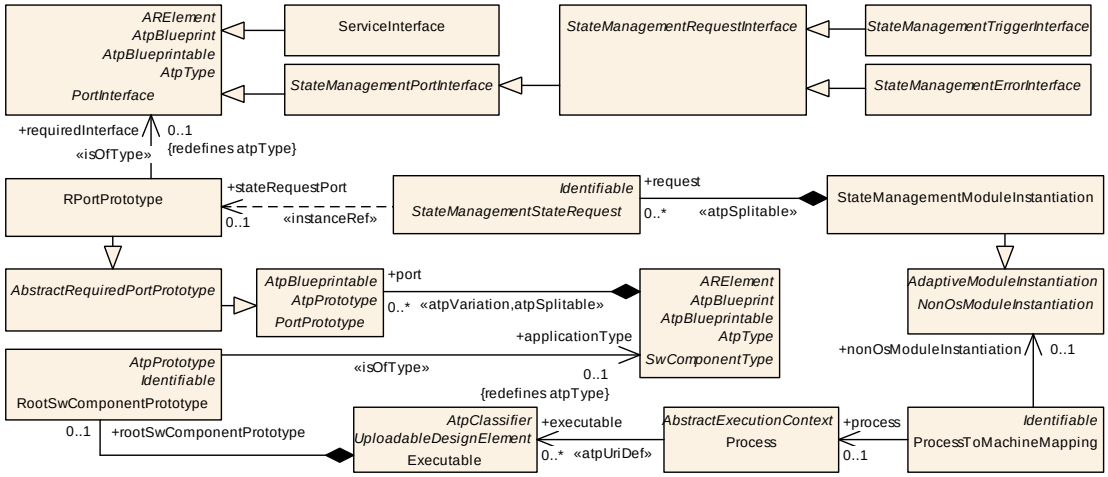


Figure 10.63: Modeling of `StateManagementStateRequest` including the association with the modeling of `Executable`

Class	***StateManagementStateRequest*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This abstract class serves as the base class for state requests on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	StateManagementRequestError, StateManagementRequestTrigger			
Aggregated by	StateManagementModuleInstantiation.request			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
stateRequest Port	RPortPrototype	0..1	iref	This represents the RPortPrototype in the application software that is issuing the request for state change. **Tags:** atp.Status=draft **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef

Table 10.129: StateManagementStateRequest

[constr_10390]{DRAFT} **Existence of reference `StateManagementStateRequest.stateRequestPort`** ⌈Unless the `StateManagementStateRequest` is referenced from a `NmInteractsWithSmMapping`, the reference in the role `stateRequestPort` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10511]{DRAFT} **Reference `StateManagementStateRequest.stateRequestPort` shall not exist if `StateManagementStateRequest` is referenced from `NmInteractsWithSmMapping`** ⌈The reference StateManagementStateRequest.stateRequestPort shall not exist if StateManagementStateRequest is referenced from NmInteractsWithSmMapping.⌋*()*

10.14.2.1 Trigger Requests to State Management

Trigger requests to State Management are configured by means of meta-class StateManagementRequestTrigger.

As depicted by Figure 10.62, the existence of a StateManagementRequestTrigger is associated with either a subclass of StateManagementTriggerInterface or a ServiceInterface. This relation is formalized in [constr_10384].

[constr_10384]{DRAFT} **`PortInterface` used for trigger state requests** ⌈Each RPortPrototype that is referenced by a StateManagementRequestTrigger shall be typed by ether

- a subclass of StateManagementTriggerInterface or
- a ServiceInterface.

This rule shall be imposed **`at the time when the creation of the manifest is finished`**.⌋*()*

Class	**StateManagementRequestTrigger**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This meta-class has the ability to configure a trigger request to the state management. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementStateRequest*			
Aggregated by	StateManagementModuleInstantiation.request			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
rule	StateManagement RequestRule	*	aggr	This aggregation represents the collection of rules applicable for the trigger request. **Tags:** atp.Status=draft

Table 10.130: StateManagementRequestTrigger

10.14.2.2 Error Requests to State Management

Error submissions to State Management are configured by means of meta-class StateManagementRequestError, see Figure 10.62.

As depicted by Figure 10.62, the existence of a StateManagementRequestTrigger is associated with a subclass of StateManagementTriggerInterface. This relation is formalized in [constr_10385].

[constr_10385]{DRAFT} **PortInterface used for error state requests** ⌈Each RPortPrototype that is referenced by a StateManagementRequestError shall be typed by subclass of StateManagementErrorInterface **at the time when the creation of the manifest is finished.**⌋*()*

Class	**StateManagementRequestError**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This meta-class has the ability to configure the submission of an error to the state management. **Tags:** atp.Status=draft atp.recommendedPackage=StateManagementRequests			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementStateRequest*			
Aggregated by	StateManagementModuleInstantiation.request			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
rule	StateManagement RequestRule	*	aggr	This aggregation represents the collection of rules applicable for the error request. **Tags:** atp.Status=draft

Table 10.131: StateManagementRequestError

10.14.3 Notifications from State Management

[TPS_MANI_01392]{DRAFT} **Semantics of StateManagementStateNotification** ⌈Notifications are used to make state users aware of state changes. Meta-class StateManagementStateNotification is aggregated at StateManagementModuleInstantiation and in turn aggregates ModeDeclarationGroupPrototype and therefore one StateManagementStateNotification becomes the **owner of one internal state machine.**⌋*(RS_MANI_00023)*

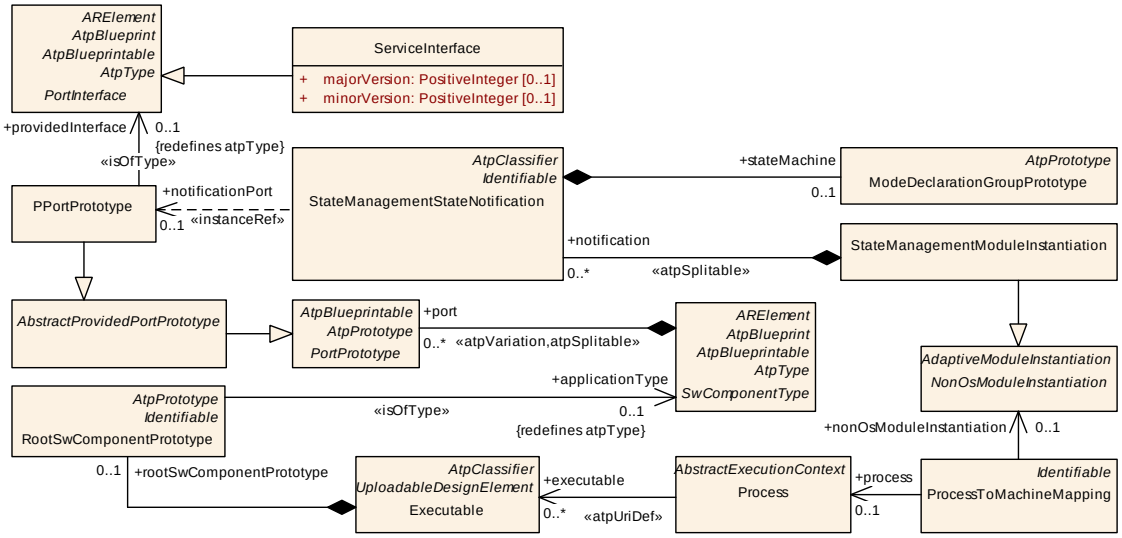


Figure 10.64: Modeling of StateManagementStateNotification

Please note that it is positively supported that several StateManagementStateNotification aggregate a ModeDeclarationGroupPrototype typed by the identical ModeDeclarationGroup as their internal state machine. This way, several instances of the same "type" of state machine are created.

A StateManagementModuleInstantiation references a PPortPrototype (typed by a ServiceInterface) that shall be taken to provide the actual state notification.

[constr_10388]{DRAFT} **Restriction for a PortInterface used for state switch notifications** ⌈Each PPortPrototype that is referenced by a StateManagementStateNotification shall be typed by a ServiceInterface **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10391]{DRAFT} **Existence of attribute StateManagementStateNotification.notificationPort** ⌈For each StateManagementStateNotification, the aggregation in the role notificationPort shall exist **at the time when the creation of the manifest is finished**.⌋*()*

Figure 10.64 also depicts the relation between the modeled Executable that implements the State Management functionality and the deployment aspects represented by the StateManagementModuleInstantiation.

Class	**StateManagementStateNotification**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This meta-class represents the ability to formalize state notifications on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft			
Base	*ARObject*, *AtpClassifier*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	StateManagementModuleInstantiation.notification			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
notificationPort	PPortPrototype	0..1	iref	This instanceRef identifies the PPortPrototype over which the notification is to be conveyed. **Tags:** atp.Status=draft **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
stateMachine	ModeDeclarationGroup Prototype	0..1	aggr	This aggregation represents the existence of an actual state machine. **Tags:** atp.Status=draft

Table 10.132: StateManagementStateNotification

10.14.4 Rules

10.14.4.1 Overview

[TPS_MANI_01384]{DRAFT} **Semantics of StateManagementRequestRule** ⌈The purpose of StateManagementRequestRule is to evaluate an incoming request against a given comparison value.

If the evaluation of the formula owned by the StateManagementRequestRule yields true, then a state machine (modeled as a ModeDeclarationGroupPrototype) is forwarded from the current state (represented by meta-class ModeDeclaration) to the nextState, as specified by the definition of the specific subclass of StateManagementRequestRule.⌋*(RS_MANI_00023)*

Figure 10.65: Modeling of StateManagementRequestRule

Class	**StateManagementRequestRule**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This meta-class represents a rule for deciding about a state change. **Tags:** atp.Status=draft			
Base	*ARObject*			
Aggregated by	StateManagementRequestError.rule, StateManagementRequestTrigger.rule			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
formula	StateManagement CompareFormula	0..1	aggr	This aggregation represents the definition of the formula for the StateManagementRequestRule **Tags:** atp.Status=draft
nextState	ModeDeclaration	0..1	iref	This reference identifies the state to be switched to if the condition is fulfilled. **Tags:** atp.Status=draft **InstanceRef implemented by:** ModeDeclarationInState ManagementStateNotificationInstanceRef

Table 10.133: StateManagementRequestRule

[constr_10392]{DRAFT} **Existence of attribute StateManagementRequestRule.formula** ⌈For each StateManagementRequestRule, the aggregation in the role formula shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10393]{DRAFT} **Existence of reference in the role StateManagementRequestRule.nextState** ⌈For each StateManagementRequestRule, the reference in the role nextState shall exist **at the time when the creation of the manifest is finished**.⌋*()*

The general approach for rule formulation is very generic, although the State Management for the time being has only little requirements for rule evaluation: one value is checked against a reference value. But since the complexity of rule formulation might grow over time, the model is already prepared for future extensions.

The meta-model supports the formulation of hierarchical comparisons such that state changes could be evaluated against not only a single input, but also – where applicable – collection of inputs.

The execution of state changes is subject to the evaluation of rules that are owned by the respective request classes. This relation is depicted in Figure 10.66.

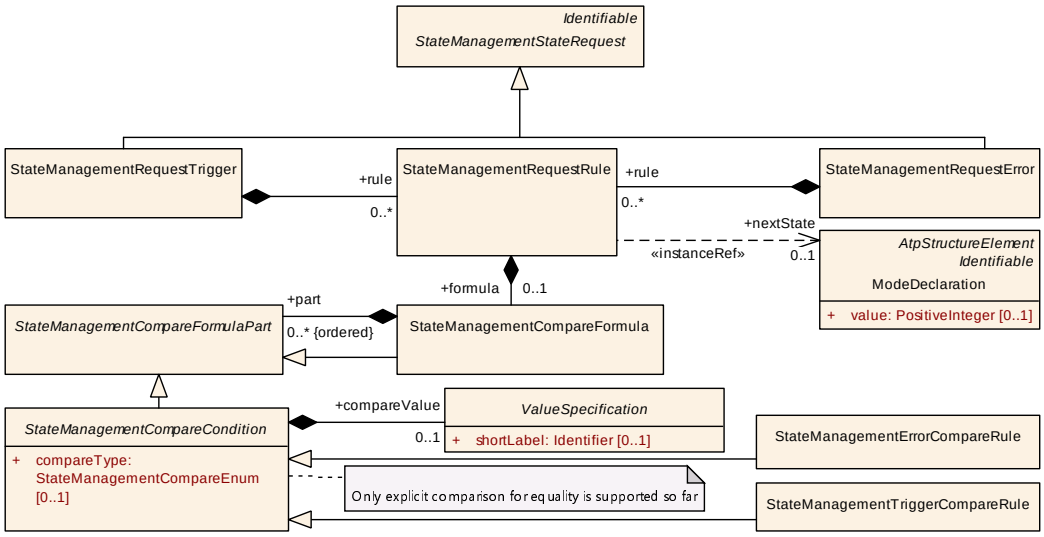


Figure 10.66: Relation between `StateManagementStateRequest` and `StateManagementRequestRule`

The states that are involved in the evaluation of State Management rules are entirely "internal" states, `Function Group State`s are not affected and shall[3] not be used for rule evaluation.

Class	**StateManagementCompareFormula**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	A StateManagementCompareFormula embodies the computation instruction that is to be evaluated at runtime to determine if the aggregating request rule yields true or false. The formula itself consists of parts which are combined by the logical operations specified by StateManagementCompareFormula .op. **Tags:** atp.Status=draft			
Base	*ARObject*, *StateManagementCompareFormulaPart*			
Aggregated by	StateManagementCompareFormula.part, StateManagementRequestRule.formula			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
part (ordered)	StateManagement CompareFormulaPart	*	aggr	This aggregation represents the collection of formula parts that can be combined by logical operators. **Tags:** atp.Status=draft

Table 10.134: StateManagementCompareFormula

Enumeration	**StateManagementLogicalOperatorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement
Note	Logical AND and OR operation for the evaluation of rules in state management. **Tags:** atp.Status=draft
Literal	***Description***

▽

[3]The nature of the underlying `instanceRef` modeling would unfortunately not be able to prohibit a brute-force referencing of function groups states.

△

Enumeration	**StateManagementLogicalOperatorEnum**
logicalAnd	This value represents the semantics of a logical "and" **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=draft
logicalOr	This value represents the semantics of a logical "or" **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=draft

Table 10.135: StateManagementLogicalOperatorEnum

Class	***StateManagementCompareFormulaPart*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	A StateManagementCompareFormulaPart can either be a atomic condition, e.g. a StateManagement TriggerCompareRule or StateManagementErrorCompareRule, or a StateManagementCompareFormula again, which allows arbitrary nesting. **Tags:** atp.Status=draft			
Base	*ARObject*			
Subclasses	*StateManagementCompareCondition*, StateManagementCompareFormula			
Aggregated by	StateManagementCompareFormula.part			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 10.136: StateManagementCompareFormulaPart

Class	***StateManagementCompareCondition*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	StateManagementCompareConditions are atomic conditions. They are based on the idea of a comparison at runtime of some variable data with something constant. The type of the comparison (==, !=, <, <=, ...) is specified in StateManagementCompareCondition.compareType. **Tags:** atp.Status=draft			
Base	*ARObject*, *StateManagementCompareFormulaPart*			
Subclasses	StateManagementErrorCompareRule, StateManagementTriggerCompareRule			
Aggregated by	StateManagementCompareFormula.part			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
compareType	StateManagement CompareEnum	0..1	attr	This attributes represents the concrete type of the comparison. **Tags:** atp.Status=draft
compareValue	ValueSpecification	0..1	aggr	This aggregation represents the reference value against which the value obtained from request shall be compared to. **Tags:** atp.Status=draft

Table 10.137: StateManagementCompareCondition

[constr_10394]{DRAFT} **Existence of attribute `StateManagementCompareCondition.compareType`** ⌈For each `StateManagementCompareCondition`, the aggregation in the role `compareType` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10395]{DRAFT} **Existence of attribute `StateManagementCompareCondition.compareValue`** ⌈For each `StateManagementCompareCondition`, the aggregation in the role `compareValue` shall exist **at the time when the creation of the manifest is finished.**⌋*()*

Enumeration	**StateManagementCompareEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement
Note	Enumeration for the type of a comparison of values usually expressed by the following operators: ==, !=, <, <=, >, >= **Tags:** atp.Status=draft
Aggregated by	*StateManagementCompareCondition*.compareType
Literal	***Description***
isEqual	Equal **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=draft

Table 10.138: StateManagementCompareEnum

10.14.4.2 Trigger Rules

The modeling of the rules for trigger requests is depicted in Figure 10.66 and Figure 10.67.

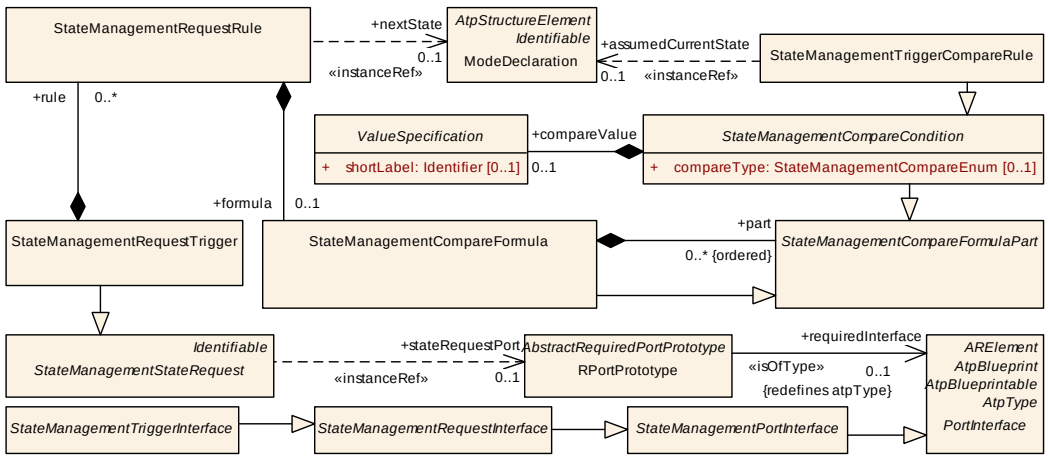


Figure 10.67: Modeling of `StateManagementRequestRule` for the "trigger" case

As depicted in Figure 10.67, the definition of the trigger request rule involves two `instanceRef`s to `ModeDeclarationGroup`, as further explained in

[TPS_MANI_01835]{DRAFT} **Semantics of `StateManagementRequestTrigger.rule`** ⌈The semantics of `StateManagementRequestTrigger.rule` is mainly defined by the existence of two `instanceRef`s:

- The reference in the role `StateManagementRequestTrigger.rule.nextState` denotes the state that shall be switched to if the rule evaluates to `true`.

- The additional reference in the role `StateManagementTriggerCompareRule.assumedCurrentState` represents an input to the rule evaluation. In other words, the `rule.formula` evaluates to `true` if
  - the input value obtained from the `PortPrototype` referenced in the role `StateManagementRequestTrigger.stateRequestPort` typed by either a sub-class of `StateManagementTriggerInterface` or a `ServiceInterface` is equal the compare value **and**
  - the reference `StateManagementTriggerCompareRule.assumedCurrentState` identifies the current state.

⌋()

Class	**StateManagementTriggerCompareRule**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This meta-class represents the configuration of a compare rule for the processing of a trigger request. **Tags:** atp.Status=draft			
Base	*ARObject*, *StateManagementCompareCondition*, *StateManagementCompareFormulaPart*			
Aggregated by	StateManagementCompareFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
assumed CurrentState	ModeDeclaration	0..1	iref	This reference denotes the assumed current state for the given compare rule for trigger values. **Tags:** atp.Status=draft **InstanceRef implemented by:** ModeDeclarationInStateManagementStateNotificationInstanceRef

Table 10.139: StateManagementTriggerCompareRule

[constr_10396]{DRAFT} **Existence of reference in the role `StateManagementTriggerCompareRule.assumedCurrentState`** ⌈For each `StateManagementTriggerCompareRule`, the reference in the role `assumedCurrentState` shall exist **at the time when the creation of the manifest is finished**.⌋()

10.14.4.3 Error Rules

The modeling of the rules for error submissions is depicted in Figure 10.66 and Figure 10.68.

[TPS_MANI_01836]{DRAFT} **Semantics of `StateManagementRequestError.rule`** ⌈The semantics of the `rule` is that the affected state machine shall switch to the `nextState` if the `formula` evaluates to `true`.⌋(*RS_MANI_00023*)

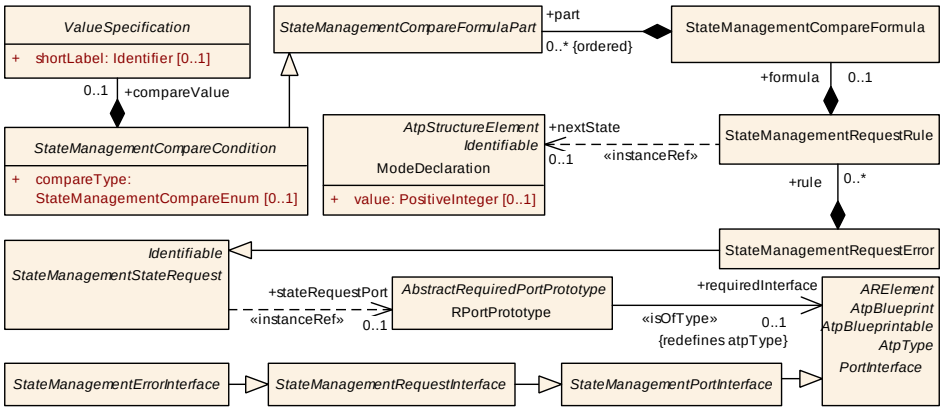


Figure 10.68: Modeling of `StateManagementRequestRule` for the "error" case

Class	**StateManagementErrorCompareRule**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This meta-class represents the configuration of a compare rule for the processing of an error submission. **Tags:** atp.Status=draft			
Base	*ARObject*, *StateManagementCompareCondition*, *StateManagementCompareFormulaPart*			
Aggregated by	StateManagementCompareFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.140: StateManagementErrorCompareRule

10.14.5 Action Items

[TPS_MANI_01387]{DRAFT} **Semantics of meta-class `StateManagementActionList`** ⌈Whenever a state machine switches to a different state, one or more actions shall be executed in response.

This functionality is modeled by means of the existence of a `StateManagementActionList` that aggregates abstract class `StateManagementActionItem`s as an **ordered** collection, i.e. the order of the execution of individual `StateManagementActionItem`s is significant.

The concrete sub-classes

- `StateManagementStateMachineActionItem`
- `StateManagementSyncActionItem`
- `StateManagementSetFunctionGroupStateActionItem`

of `StateManagementActionItem` define the specific action item that needs to be executed.

The StateManagementActionList refers to a ModeDeclaration (an “internal” state, as opposed to a Function Group State) as the representation of the state to which the state machine was switched to.⌋(*RS_MANI_00023*)

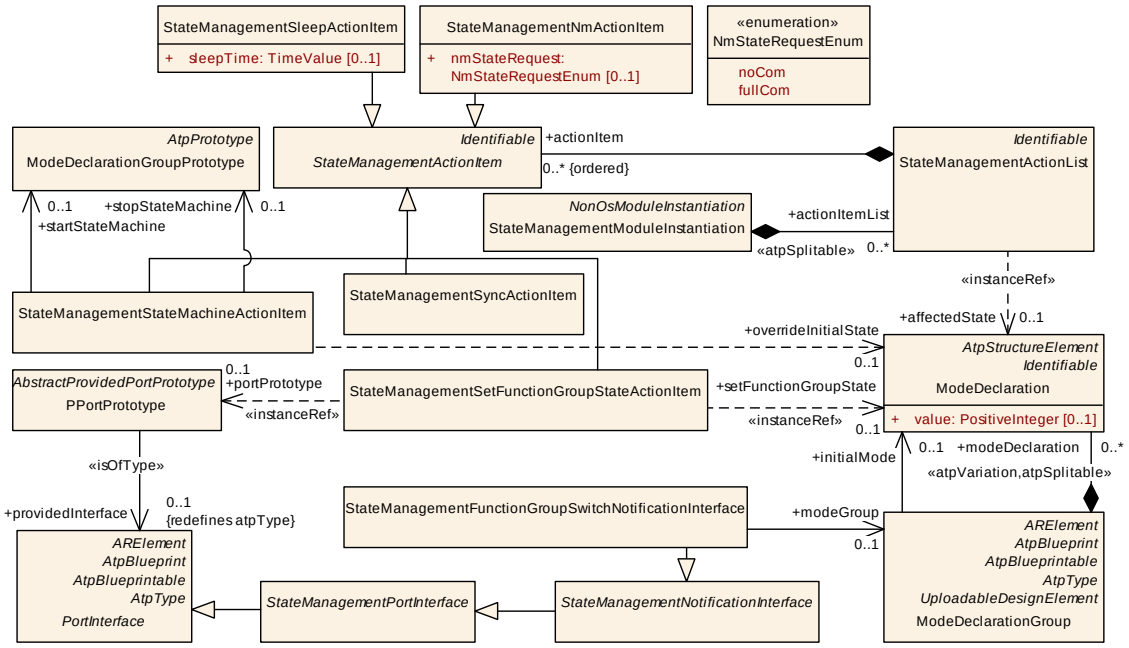


Figure 10.69: Modeling of StateManagementActionList

Class	**StateManagementActionList**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This meta-class represents the ability to define an action list that is associated with a state of a state machine. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	StateManagementModuleInstantiation.actionItemList			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
actionItem (ordered)	StateManagement ActionItem	*	aggr	This represents the collection of action items in the context of the action item list. **Tags:** atp.Status=draft
affectedState	ModeDeclaration	0..1	iref	This reference identifies the state for which the referencing action list applies. **Tags:** atp.Status=draft **InstanceRef implemented by:** ModeDeclarationInState ManagementStateNotificationInstanceRef

Table 10.141: StateManagementActionList

[constr_10405]{DRAFT} **Existence of reference in the role StateManagementActionList.affectedState** ⌈For each StateManagementActionList, the reference in the role affectedState shall exist **at the time when the creation of the manifest is finished**.⌋()

Class	***StateManagementActionItem*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This meta-class represents an action item that is executed in response to a state change. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	StateManagementNmActionItem, StateManagementSetFunctionGroupStateActionItem, StateManagementSleepActionItem, StateManagementStateMachineActionItem, StateManagementSyncActionItem			
Aggregated by	StateManagementActionList.actionItem			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 10.142: StateManagementActionItem

10.14.5.1 State Machine State Machine Action Item

[TPS_MANI_01388]{DRAFT} **Semantics of the meta-class `StateManagementStateMachineActionItem`** ⌈A `StateManagementStateMachineActionItem` can be used to start or stop a state machine in the context of State Management.

This activity is typically needed to interact with subordinate State Management software that comes with a specific `SoftwareCluster` and is responsible for the State Management of the enclosing `SoftwareCluster`.⌋*(RS_MANI_00023)*

[TPS_MANI_01436]{DRAFT} **Relation of `StateManagementStateMachineActionItem` to internal state machine instances** ⌈The modeling of the relation between a `StateManagementStateMachineActionItem` and the affected state machines (represented by `ModeDeclarationGroupPrototype` assumes that the internal state machines are located in the context of the same Executable/Process as the `StateManagementStateMachineActionItem`.

In other words, an inter-process communication is not foreseen for starting and stopping internal state machines.⌋*()*

As mentioned by [TPS_MANI_01388], the purpose of the `StateManagementStateMachineActionItem` is to either start or stop a given state machine. There is therefore no valid use case for both references in the role `startStateMachine` and `stopStateMachine` to exist at the same time.

[constr_10386]{DRAFT} **Existence of references `StateManagementStateMachineActionItem.startStateMachine` and `stopStateMachine`** ⌈For each `StateManagementStateMachineActionItem`, **at most one** of the two references

- `startStateMachine`
- `stopStateMachine`

shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[TPS_MANI_01437]{DRAFT} **Standardized values of attribute StateManagementStateNotification.stateMachine.category** ⌈The following values of attribute StateManagementStateNotification.stateMachine.category are standardized:

STATE_MANAGEMENT_CONTROLLER the aggregated StateManagementStateNotification.stateMachine represents the state machine that starts all other state machines.

STATE_MANAGEMENT_AGENT the aggregated StateManagementStateNotification.stateMachine represents a state machine that is started by a "controller".

⌋()

[constr_10489]{DRAFT} **Existence of StateManagementModuleInstantiation in the context of a Machine** ⌈Each Machine shall only aggregate at most one StateManagementModuleInstantiation in the role moduleInstantiation **at the time when the creation of the manifest is finished**.⌋()

[TPS_MANI_01438] Existence of StateManagementStateMachineActionItem.overrideInitialState ⌈If the instance reference in the role StateManagementStateMachineActionItem.overrideInitialState exists, then the references

- StateManagementStateMachineActionItem.startStateMachine.type.initialMode
- StateManagementStateMachineActionItem.stopStateMachine.type.initialMode

shall be ignored.⌋()

Class	**StateManagementStateMachineActionItem**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This meta-class represents a state management action item to start or stop a state machine. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementActionItem*			
Aggregated by	StateManagementActionList.actionItem			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
overrideInitial State	ModeDeclaration	0..1	iref	The referenced ModeDeclaration shall be considered the initial state of the context ModeDeclarationGroup Prototype and the corresponding reference Mode DeclarationGroup.initialMode shall be ignored. **Tags:** atp.Status=draft **InstanceRef implemented by:** ModeDeclarationInState ManagementStateNotificationInstanceRef
startState Machine	ModeDeclarationGroup Prototype	0..1	ref	This reference identifies the state machine that shall be started when the enclosing action list item is executed. **Tags:** atp.Status=draft

▽

△

Class	StateManagementStateMachineActionItem			
stopState Machine	ModeDeclarationGroup Prototype	0..1	ref	This reference identifies the state machine that shall be stopped when the enclosing action list item is executed. **Tags:** atp.Status=draft

Table 10.143: StateManagementStateMachineActionItem

10.14.5.2 Sync Action Item

[TPS_MANI_01389]{DRAFT} **Semantics of meta-class `StateManagementSyncActionItem`** ⌈A `StateManagementSyncActionItem` can be used to introduce a synchronization point at which the workflow waits for the completion of actions that could (in theory) be executed in parallel.

In other words, any `StateManagementActionItem` positioned after the `StateManagementSyncActionItem` can only be executed if the `StateManagementActionItem` placed before the `StateManagementSyncActionItem` are finished.⌋*(RS_MANI_00023)*

Class	StateManagementSyncActionItem			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This meta-class represents a state management action item to synchronize state machines. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementActionItem*			
Aggregated by	StateManagementActionList.actionItem			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table 10.144: StateManagementSyncActionItem

10.14.5.3 Function Group Action Item

[TPS_MANI_01390]{DRAFT} **Semantics of meta-class `StateManagementSetFunctionGroupStateActionItem`** ⌈The purpose of `StateManagementSetFunctionGroupStateActionItem` is to enable the switch of the state machine representing a specific `Function Group` to a new `Function Group State`, identified by means of the reference in the role `StateManagementSetFunctionGroupStateActionItem.setFunctionGroupState`.⌋*(RS_MANI_00023)*

Each `StateManagementSetFunctionGroupStateActionItem` references

- a `ModeDeclaration` that represents a `Function Group State` and
- a `PPortPrototype` over which the switch to the new `Function Group State` shall be communicated.

It is obviously very important that the ModeDeclarationGroup that is used in the context of the definition of the Function Group is identical[4] to the definition of the StateManagementFunctionGroupSwitchNotificationInterface.modeGroup. This consequence is the basis for the definition of [constr_10387].

[constr_10387]{DRAFT} **Consistency of StateManagementSetFunctionGroupStateActionItem.portPrototype and StateManagementSetFunctionGroupStateActionItem.setFunctionGroupState** ⌈For each StateManagementSetFunctionGroupStateActionItem, the ModeDeclarationGroup used to type the ModeDeclaration that is referenced in the role setFunctionGroupState shall be identical to the ModeDeclarationGroup referenced in the role modeGroup from the StateManagementFunctionGroupSwitchNotificationInterface that is used to type the PPortPrototype that is referenced in the role portPrototype from the affected StateManagementSetFunctionGroupStateActionItem.

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

Please note that the relation described in [constr_10387] is depicted in Figure 10.69.

Class	**StateManagementSetFunctionGroupStateActionItem**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This meta-class represents a state management action item to set a specific state in a specific function group. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementActionItem*			
Aggregated by	StateManagementActionList.actionItem			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
portPrototype	PPortPrototype	0..1	iref	This reference identifies the PortPrototype over which the function group state switch shall be communicated. **Tags:** atp.Status=draft **InstanceRef implemented by:** PPortPrototypeIn ExecutableInstanceRef
setFunction GroupState	ModeDeclaration	0..1	iref	This reference identifies the funtion group step that shall become active after the action step terminates. **Tags:** atp.Status=draft **InstanceRef implemented by:** FunctionGroupStateIn FunctionGroupSetInstanceRef

Table 10.145: StateManagementSetFunctionGroupStateActionItem

[constr_10397]{DRAFT} **Existence of reference in the role StateManagementSetFunctionGroupStateActionItem.portPrototype** ⌈For each StateManagementSetFunctionGroupStateActionItem, the reference in the role portPrototype shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[4]Technically, it would be sufficient if the two affected ModeDeclarationGroups were compatible. But the utilization of the concept of compatibility doesn't (yet) have the same foothold on the *AUTOSAR adaptive platform* as it has on the *AUTOSAR classic platform*.

[constr_10398]{DRAFT} **Existence of reference in the role StateManagementSetFunctionGroupStateActionItem.setFunctionGroupState** ⌈For each StateManagementSetFunctionGroupStateActionItem, the reference in the role setFunctionGroupState shall exist **at the time when the creation of the manifest is finished**.⌋()

10.14.5.4 Network Management Action Item

[TPS_MANI_01443]{DRAFT} **Semantics of meta-class StateManagementNmActionItem** ⌈A StateManagementNmActionItem can be used to request a communication state of the network management to become active. At the moment, the following communication states are supported:

- NmStateRequestEnum.noCom: network management switches communication off.
- NmStateRequestEnum.fullCom: network management switches communication on.

⌋()

Please note that the switch is applied on the granularity of the so-called *network management handle*, formalized as NmNetworkHandle. The relation of the StateManagementNmActionItem to the NmNetworkHandle is defined by meta-class SmInteractsWithNmMapping.

Class	**StateManagementNmActionItem**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This meta-class represents a state management action item to interact with the network management. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementActionItem*			
Aggregated by	StateManagementActionList.actionItem			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
nmState Request	NmStateRequestEnum	0..1	attr	This attribute defines the target network management state that is requested by state management. **Tags:** atp.Status=draft

Table 10.146: StateManagementNmActionItem

Enumeration	**NmStateRequestEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement
Note	This enumeration defines the description of states that can be requested from the network management. **Tags:** atp.Status=draft
Aggregated by	StateManagementNmActionItem.nmStateRequest
Literal	***Description***

▽

△

Enumeration	**NmStateRequestEnum**
fullCom	This literal represents that case that full communication should be possible. **Tags:** atp.EnumerationLiteralIndex=1 atp.Status=draft
noCom	This literal represents that case that no communication should be possible. **Tags:** atp.EnumerationLiteralIndex=0 atp.Status=draft

Table 10.147: NmStateRequestEnum

10.14.5.5 Sleep Action Item

[TPS_MANI_01445]{DRAFT} **Semantics of meta-class `StateManagementSleepActionItem`** ⌈A `StateManagementSleepActionItem` can be used to make the execution of a `StateManagementActionList` pause for some time.

The purpose is to give external code time for completion until the execution of the `StateManagementActionList` commences.⌋*()*

[TPS_MANI_01444]{DRAFT} **Semantics of attribute `StateManagementSleepActionItem.sleepTime`** ⌈Attribute `StateManagementSleepActionItem.sleepTime` can be used to define a time interval, after which the action represented by the `StateManagementSleepActionItem` is expected to be complete. The action is considered failed should the timeout expire without completion.⌋*()*

[constr_10512]{DRAFT} **Restriction for the value of `StateManagementSleepActionItem.sleepTime`** ⌈For each `StateManagementSleepActionItem`, the value of attribute `sleepTime` shall only have a value greater than 0 **at the time when the creation of the manifest is finished**.⌋*()*

Class	**StateManagementSleepActionItem**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This action item can be used to universally implement afterrun. One specific use case for afterrun comes up in the context of network management. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *StateManagementActionItem*			
Aggregated by	StateManagementActionList.actionItem			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
sleepTime	TimeValue	0..1	attr	This attribute represents the amount of time that the execution of the StateManagementActionItemList is supposed to go to sleep. **Tags:** atp.Status=draft

Table 10.148: StateManagementSleepActionItem

10.15 Ethernet Endpoint Configuration for Platform Modules

The `PlatformModuleEthernetEndpointConfiguration` allows the configuration of a socket address in form of a TCP/UDP port and an IP Address that is used by a Platform module.

Class	**PlatformModuleEthernetEndpointConfiguration**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation			
Note	This meta-class defines the attributes for the configuration of a port, protocol type and IP address of the communication on a VLAN. **Tags:** atp.recommendedPackage=PlatformModuleEndpointConfigurations			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PlatformModuleEndpointConfiguration*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Connector	EthernetCommunication Connector	0..1	ref	Reference to the CommunicationConnector (VLAN) for which the network configuration is defined.
ipv4MulticastIp Address	Ip4AddressString	0..1	attr	Multicast IPv4 Address to which the message will be transmitted.
ipv6MulticastIp Address	Ip6AddressString	0..1	attr	Multicast IPv6 Address to which the message will be transmitted.
secureCom PropsForTcp	SecureComProps	0..1	ref	Reference to communication security configuration settings that are valid for the tcp unicast endpoint (Tcp Port + unicast IP Address) defined by the PlatformModule EthernetEndpointConfiguration.
secureCom PropsForUdp	SecureComProps	0..1	ref	Reference to communication security configuration settings that are valid for the udp unicast endpoint (Udp Port + unicast IP Address) defined by the PlatformModule EthernetEndpointConfiguration.
tcpPort	ApApplicationEndpoint	0..1	ref	This reference allows to configure a tcp port number.
udpPort	ApApplicationEndpoint	0..1	ref	This reference allows to configure a udp port number.

Table 10.149: PlatformModuleEthernetEndpointConfiguration

[constr_3414]{DRAFT} **Allowed usage of `PlatformModuleEthernetEndpointConfiguration` attributes that are allowed to be used to configure the network communication in the different platform modules** ⌈

	Element	
`PlatformModuleEthernetEndpointConfiguration` attributes	**Usage in `DoIpInstantiation`**	**Usage in `DltLogSink`**
`tcpPort`	Optional	Optional
`udpPort`	Optional	Optional
`ipv4MulticastIpAddress`	N/A	N/A
`ipv6MulticastIpAddress`	N/A	N/A
`communicationConnector`	Mandatory	Mandatory

This rule shall be imposed **`at the time when the creation of the manifest is finished`**

⌋()

[constr_5372] SecureComProps for a PlatformModuleEthernetEndpointConfiguration that contains a UDP configuration ⌈Only a PlatformModuleEthernetEndpointConfiguration that is referencing an ApApplicationEndpoint in the role udpPort is allowed to reference SecureComProps in the role secureComPropsForUdp.⌋*()*

[constr_5373] SecureComProps for a PlatformModuleEthernetEndpointConfiguration that contains a TCP configuration ⌈Only a PlatformModuleEthernetEndpointConfiguration that is referencing an ApApplicationEndpoint in the role tcpPort is allowed to reference SecureComProps in the role secureComPropsForTcp.⌋*()*

11 Service Instance Manifest

11.1 ApplicationEndpoint

[TPS_MANI_03280]{DRAFT} **Semantics of `ApApplicationEndpoint`** ⌈The `ApApplicationEndpoint` defines a UDP or TCP Port on a `EthernetCommunicationConnector` and defines together with the `NetworkEndpoint` that is referenced by the same `EthernetCommunicationConnector` in the role `unicastNetworkEndpoint` a local endpoint of a network communication path, i.e. a Socket.⌋*(RS_MANI_00014)*

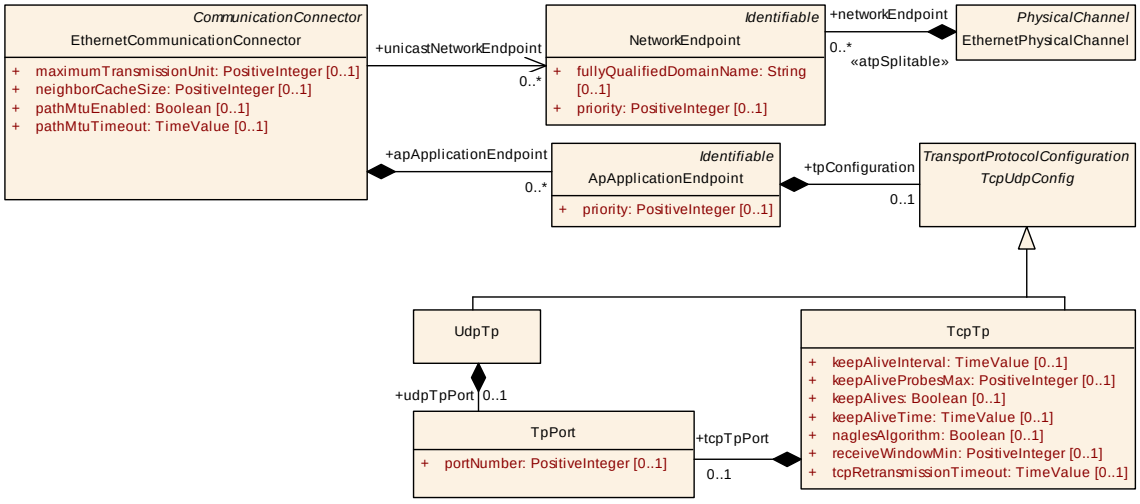


Figure 11.1: ApApplicationEndpoint definition

Class	**ApApplicationEndpoint**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::APApplicationEndpoint			
Note	An application endpoint is the endpoint on an Ecu in terms of application addressing (e.g. UDP or TCP Port).			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EthernetCommunicationConnector.apApplicationEndpoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
priority	PositiveInteger	0..1	attr	This attribute defines the VLAN frame priority where values from 0 (best effort) to 7 (highest) are allowed.
tpConfiguration	TcpUdpConfig	0..1	aggr	Configuration of the used transport protocol.

Table 11.1: ApApplicationEndpoint

Class	***TcpUdpConfig*** (abstract)
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology
Note	Tcp or Udp Transport Protocol Configuration.
Base	*ARObject*, *TransportProtocolConfiguration*
Subclasses	TcpTp, UdpTp

▽

△

Class	**TcpUdpConfig** (abstract)			
Aggregated by	ApApplicationEndpoint.tpConfiguration, ApplicationEndpoint.tpConfiguration, RtpTp.tcpUdpConfig			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 11.2: TcpUdpConfig

Class	**UdpTp**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Content Model for UDP configuration.			
Base	*ARObject*, *TcpUdpConfig*, *TransportProtocolConfiguration*			
Aggregated by	ApApplicationEndpoint.tpConfiguration, ApplicationEndpoint.tpConfiguration, RtpTp.tcpUdpConfig			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
udpTpPort	TpPort	0..1	aggr	Udp Port configuration.

Table 11.3: UdpTp

Class	**TcpTp**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Content Model for TCP configuration.			
Base	*ARObject*, *TcpUdpConfig*, *TransportProtocolConfiguration*			
Aggregated by	ApApplicationEndpoint.tpConfiguration, ApplicationEndpoint.tpConfiguration, HttpTp.tcpTpConfig, RtpTp.tcpUdpConfig			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
keepAlive Interval	TimeValue	0..1	attr	Specifies the interval in seconds between subsequent keepalive probes.
keepAlive ProbesMax	PositiveInteger	0..1	attr	Maximum number of times that TCP retransmits an individual data segment before aborting the connection.
keepAlives	Boolean	0..1	attr	Indicates if Keep-Alive messages are sent.
keepAliveTime	TimeValue	0..1	attr	Specifies the time in seconds between the last data packet sent and the first keepalive probe.
naglesAlgorithm	Boolean	0..1	attr	Indicates if Nagle's Algorithm is used.
receiveWindow Min	PositiveInteger	0..1	attr	Minimum size of the TCP receive window in bytes.
tcp Retransmission Timeout	TimeValue	0..1	attr	Defines the timeout in seconds before an unacknowledged TCP segment is sent again. If the tcp RetransmissionTimeout is not defined or set to "INF", no TCP segments shall be re-transmitted.
tcpTpPort	TpPort	0..1	aggr	TCP Port configuration.

Table 11.4: TcpTp

[TPS_MANI_03281]{DRAFT} **Port specific TCP configuration settings** ⌈The configuration settings in the `TcpTp` element of an `ApApplicationEndpoint` overwrite potential TCP settings that are available in `TcpProps` referenced by the `MachineDesign` that aggregates the `EthernetCommunicationConnector` that in turn contains the `ApApplicationEndpoint`.⌋*(RS_MANI_00014)*

Class	**TpPort**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Dynamic or direct assignment of a PortNumber.			
Base	*ARObject*			
Aggregated by	TcpTp.tcpTpPort, UdpTp.udpTpPort			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
portNumber	PositiveInteger	0..1	attr	Port Number.

Table 11.5: TpPort

11.1.1 VLAN Priority

The priority is a 3-bit field which refers to the IEEE 802.1Q priority. It indicates the frame priority level. Values are from 0 (best effort) to 7 (highest); These values can be used to prioritize different classes of traffic (voice, video, data, etc.). The priority is contained in the Ethernet Header together with the vlanIdentifier.

The priority can be defined on different levels and can overwrite the defaultPriority on the VLAN:

1. NetworkEndpoint (for network traffic over the IP Address)
2. ApApplicationEndpoint (for TCP Port and/or UDP Port network traffic)
3. ProvidedSomeipServiceInstance (for network traffic resulting from the Service Instance)

Please find an example of the definition of VLAN priority in section B.2.5.

Please note that the definition of the priority in the SomeipSdServerServiceInstanceConfig can be used to define priorities for SD messages like Service Offer and SubscribeEventGroupAck that are resulting from ProvidedSomeipServiceInstances that are referencing the SomeipSdServerServiceInstanceConfig in the role sdServerConfig.

To prioritize SD messages that are sent from the service consumer to the service provider like FindService or SubscribeEventGroup, the priority in the SomeipSdClientServiceInstanceConfig can be used.

The SD messages are communicated over a UDP Port that is defined in SomeipServiceDiscovery.someipServiceDiscoveryPort and over a IP Multicast Address that is defined by SomeipServiceDiscovery.multicastSdIpAddress. In other words, the priority in the SomeipSdServerServiceInstanceConfig and SomeipSdClientServiceInstanceConfig can be used to prioritize network traffic that results from specific ServiceInstances on SD level.

11.2 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.

[TPS_MANI_03036]{DRAFT} **ServiceInterface deployment to a middleware transport layer** ⌈The ServiceInterfaceDeployment meta-class provides the ability to map a ServiceInterface to a middleware transport layer that is represented by a concrete class that is derived from the abstract ServiceInterfaceDeployment meta-class.⌋ *(RS_MANI_00008)*

The association between the ServiceInterfaceDeployment and the ServiceInterface implicitly also defines the relation between the technology specific version number of the service on the ServiceInterfaceDeployment and the service version of the ServiceInterface (defined by ServiceInterface.majorVersion and ServiceInterface.minorVersion)

[TPS_MANI_03617]{DRAFT} **Version mapping between ServiceInterface and ServiceInterfaceDeployment** ⌈The contract version of a ServiceInterface (majorVersion, minorVersion) shall be mapped to a version of the ServiceInterfaceDeployment for each transport layer.

This version mapping may lead to different version numbers for different ServiceInterfaceDeployments that refer to the same ServiceInterface. This allows to define different version numbers, on the same network or on different networks (e.g. VLANs).⌋ *(RS_MANI_00065)*

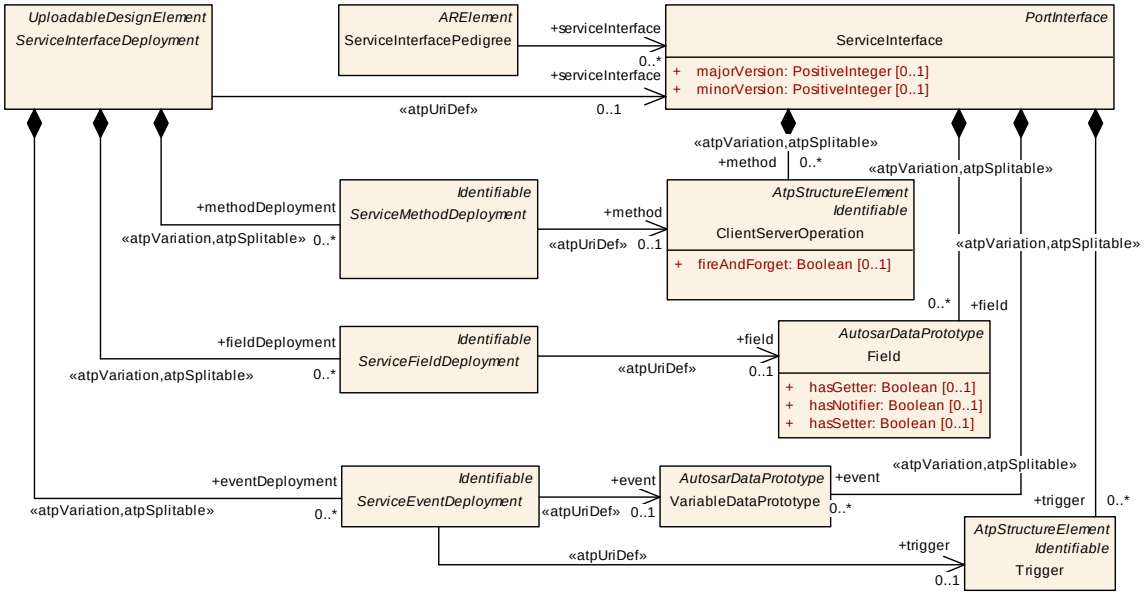


Figure 11.2: Deployment-related modeling of ServiceInterface

Note that transport layer specific constraints on the uniqueness of protocol credentials still have to be respected, e.g. [constr_1723].

In figure 11.3, the use-case of having one `PPortPrototype` typed by one `ServiceInterface` and having several `ProvidedSomeipServiceInstance`s on several VLANs is illustrated.

It is possible to have different `serviceInterfaceId`, `serviceInstanceId`, and `majorVersion` values on each individual VLAN. But also the use-case of providing both service instances on the same VLAN would be supported, as long as their SOME/IP credentials are unique (see [constr_1723]).

On different VLANs the service instance may also use the same SOME/IP credentials. But only in case that all service instance specific configuration settings (e.g. IP Multicast Address, SD Timing Properties) are exactly the same on both VLANs the same `AdaptivePlatformServiceInstance` can be used to model the communication on both VLANs.

If, for example, the `serviceInterfaceId`, `serviceInstanceId`, and `majorVersion` values of a service instance on each VLAN are the same, but different IP Multicast Addresses are used on the different VLANs for the EventGroup transmission, then an individual VLAN specific modeling of `AdaptivePlatformServiceInstance` is necessary as shown in figure 11.3.

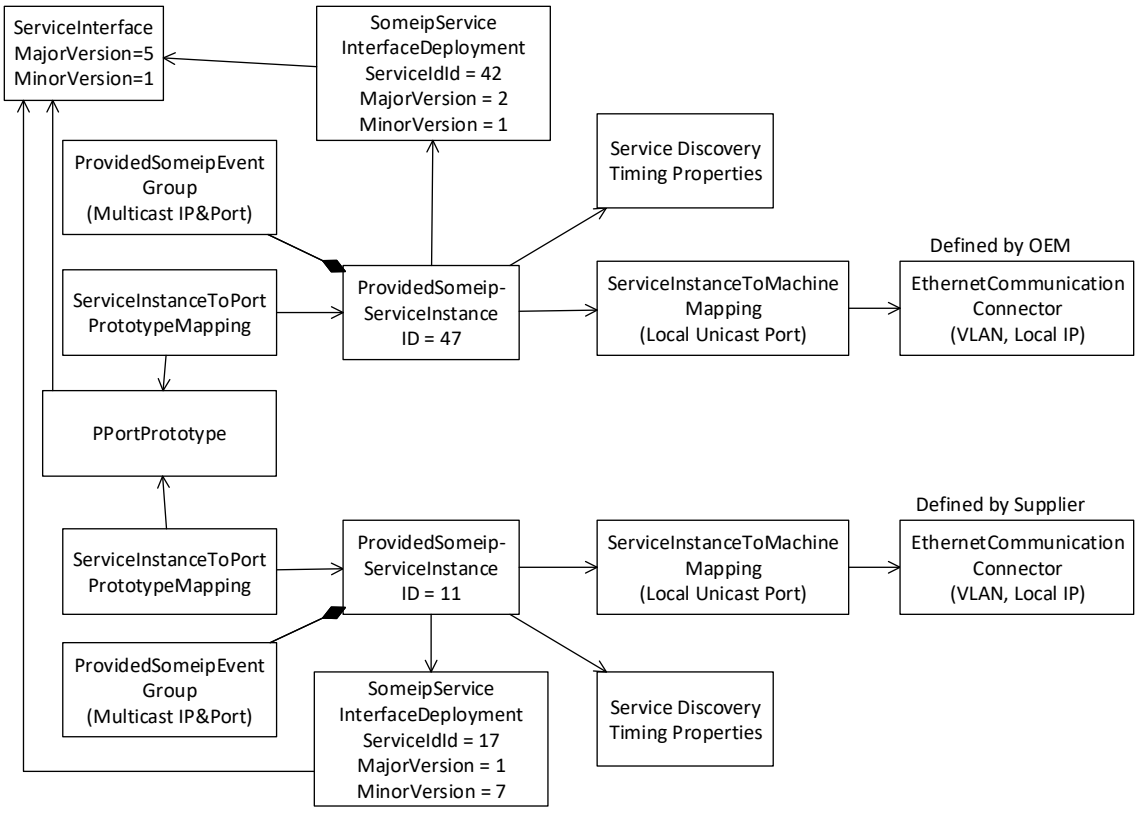


Figure 11.3: Example of 1 `PPortPrototype` mapped to 2 VLANs

Class	***ServiceInterfaceDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	Middleware transport layer specific configuration settings for the ServiceInterface and all contained ServiceInterface elements.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	DdsServiceInterfaceDeployment, SomeipServiceInterfaceDeployment, UserDefinedServiceInterface Deployment			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
event Deployment	ServiceEvent Deployment	*	aggr	Middleware transport layer specific configuration settings for an Event that is defined in the ServiceInterface. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=eventDeployment.shortName, event Deployment.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime
fieldDeployment	ServiceField Deployment	*	aggr	Middleware transport layer specific configuration settings for a Field that is defined in the ServiceInterface. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=fieldDeployment.shortName, field Deployment.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime
method Deployment	ServiceMethod Deployment	*	aggr	Middleware transport layer specific configuration settings for a method that is defined in the ServiceInterface. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=methodDeployment.shortName, method Deployment.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime
serviceInterface	ServiceInterface	0..1	ref	Reference to a ServiceInterface that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef

Table 11.6: ServiceInterfaceDeployment

[TPS_MANI_03037]{DRAFT} **Purpose of `ServiceMethodDeployment`** ⌈The `ServiceMethodDeployment` meta-class provides the ability to define middleware transport layer specific configuration settings relevant for a `method` that is defined in the context of a `ServiceInterface`.⌋*(RS_MANI_00008)*

[constr_3300]{DRAFT} **Allowed `ServiceMethodDeployment.method` references** ⌈The `ClientServerOperation` that is referenced by `ServiceMethodDeployment` in the role `method` shall be defined in the context of a `ServiceInterface` that is referenced by the `ServiceInterfaceDeployment` in the role `serviceInterface` that contains the `ServiceMethodDeployment` **`at the time when the creation of the manifest is finished`**.⌋*()*

[TPS_MANI_03038]{DRAFT} **Purpose of `ServiceEventDeployment`** ⌈The `ServiceEventDeployment` meta-class provides the ability to define middleware transport layer specific configuration settings relevant for an `event` or `trigger` that is defined in the context of a `ServiceInterface`.⌋*(RS_MANI_00008)*

[constr_3301]{DRAFT} **Allowed `ServiceEventDeployment.event` references** ⌈The `VariableDataPrototype` that is referenced by `ServiceEventDeployment` in the role `event` shall be defined in the context of a `ServiceInterface` that is referenced by the `ServiceInterfaceDeployment` in the role `serviceInterface` that contains the `ServiceEventDeployment` **at the time when the creation of the manifest is finished**.⌋*()*

[constr_5316]{DRAFT} **Allowed `ServiceEventDeployment.trigger` references** ⌈The `Trigger` that is referenced by `ServiceEventDeployment` in the role `trigger` shall be defined in the context of a `ServiceInterface` that is referenced by the `ServiceInterfaceDeployment` in the role `serviceInterface` that contains the `ServiceEventDeployment` **at the time when the creation of the manifest is finished**.⌋*()*

[constr_5317]{DRAFT} **`ServiceEventDeployment` not allowed to reference an `event` and a `trigger` at the same time** ⌈The `ServiceEventDeployment` element shall reference either:

- a `VariableDataPrototype` in the role `event` or
- a `Trigger` in the role `trigger`,

but not both at the same time.

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

[TPS_MANI_03039]{DRAFT} **Purpose of `ServiceFieldDeployment`** ⌈The `ServiceFieldDeployment` meta-class provides the ability to define middleware transport layer specific configuration settings relevant for a `field` that is defined in the context of a `ServiceInterface`.⌋*(RS_MANI_00008)*

[constr_3302]{DRAFT} **Allowed `ServiceFieldDeployment.field` references** ⌈The `Field` that is referenced by `ServiceFieldDeployment` in the role `field` shall be defined in the context of a `ServiceInterface` that is referenced by the `ServiceInterfaceDeployment` in the role `serviceInterface` that contains the `ServiceFieldDeployment` **at the time when the creation of the manifest is finished**.⌋*()*

Please note that a partial `ServiceInterfaceDeployment` that only covers certain parts of the corresponding `ServiceInterfaceDeployment` is not supported. This understanding is formalized in [constr_10029]

[constr_10029]{DRAFT} **`ServiceInterfaceDeployment` shall cover all elements of the corresponding `ServiceInterface`** ⌈If a `ServiceInterfaceDeployment` references a `ServiceInterface` in the role `serviceInterface`, then all `method`s, `field`s, `trigger`s, and `event`s defined in the context of the referenced `ServiceInterface` shall be referenced by respective `methodDeployment`s, `fieldDeployment`s, and `eventDeployment`s owned by the referencing `ServiceInterfaceDeployment` **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3715]{DRAFT} **Reference in the role `SomeipEventGroup.event`** ⌈In the context of a given `SomeipServiceInterfaceDeployment`, all aggregated `SomeipEventDeployment`s shall be referenced at least once in the role `event` by `SomeipEventGroup`s that in turn are aggregated at the same `SomeipServiceInterfaceDeployment` **at the time when the creation of the manifest is finished**.⌋*()*

Class	***ServiceMethodDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	This abstract meta-class represents the ability to specify a deployment of a Method to a middleware transport layer.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	SomeipMethodDeployment, UserDefinedMethodDeployment			
Aggregated by	*ServiceInterfaceDeployment*.methodDeployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
method	ClientServerOperation	0..1	ref	Reference to a method that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef

Table 11.7: ServiceMethodDeployment

Class	***ServiceEventDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	This abstract meta-class represents the ability to specify a deployment of an Event to a middleware transport layer.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	DdsEventDeployment, SomeipEventDeployment, UserDefinedEventDeployment			
Aggregated by	*ServiceInterfaceDeployment*.eventDeployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
event	VariableDataPrototype	0..1	ref	Reference to an Event that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef
trigger	Trigger	0..1	ref	Reference to a Trigger that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef

Table 11.8: ServiceEventDeployment

Class	***ServiceFieldDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	This abstract meta-class represents the ability to specify a deployment of a Field to a middleware transport layer.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	DdsFieldDeployment, SomeipFieldDeployment, UserDefinedFieldDeployment			
Aggregated by	*ServiceInterfaceDeployment*.fieldDeployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	***ServiceFieldDeployment*** (abstract)			
field	Field	0..1	ref	Reference to a Field that is deployed to a middleware transport layer. **Stereotypes:** atpUriDef

Table 11.9: ServiceFieldDeployment

[constr_10251]{DRAFT} **Multiplicity of the reference in the role `ServiceFieldDeployment.field`** ⌈For each `ServiceFieldDeployment`, the reference in the role `field` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

11.2.1 SOME/IP Service Interface Deployment

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

[TPS_MANI_03040]{DRAFT} **SOME/IP ServiceInterface binding** ⌈The `SomeipServiceInterfaceDeployment` meta-class provides the ability to bind a `ServiceInterface` to SOME/IP and to assign a SOME/IP Service identifier to the `ServiceInterface` with the `serviceInterfaceId` attribute.⌋*(RS_MANI_00024)*

The idea behind the `SomeipServiceInterfaceDeployment` is the definition of a common configuration set that is shared between the server that provides the `ServiceInterface` and all clients that are consuming the `ServiceInterface`. So it contains all relevant SOME/IP settings used for identification of the `ServiceInterface` and its content in messages on the network.

[constr_3410]{DRAFT} **Value range of `SomeipServiceInterfaceDeployment.serviceInterfaceId`** ⌈The value of `serviceInterfaceId` shall be in the range of 0..65535 **at the time when the creation of the manifest is finished**.⌋*()*

Please note that the SOME/IP MessageId that is 32 Bit long contains a 16 Bit `serviceInterfaceId`, a single bit that defines whether the message transports a method or an event and a 15 Bit `eventId` or `methodId`.

Please also consider [PRS_SOMEIPSD_00515] in [30] that defines special and reserved `serviceInterfaceId`s for SOME/IP and SOME/IP-SD.

[TPS_MANI_03041]{DRAFT} **Definition of SOME/IP EventGroups** ⌈The `SomeipServiceInterfaceDeployment.eventGroup` allows to define SOME/IP *EventGroups* that are included in the SOME/IP Service and provide a logical grouping of events and notification events used for publish/subscribe handling.⌋*(RS_MANI_00024)*

[constr_3304]{DRAFT} **Value of attribute `SomeipEventGroup.eventGroupId` shall be unique** ⌈The value of attribute `eventGroupId` shall be unique in the context of the enclosing `SomeipServiceInterfaceDeployment` **at the time when the creation of the manifest is finished.**⌋*()*

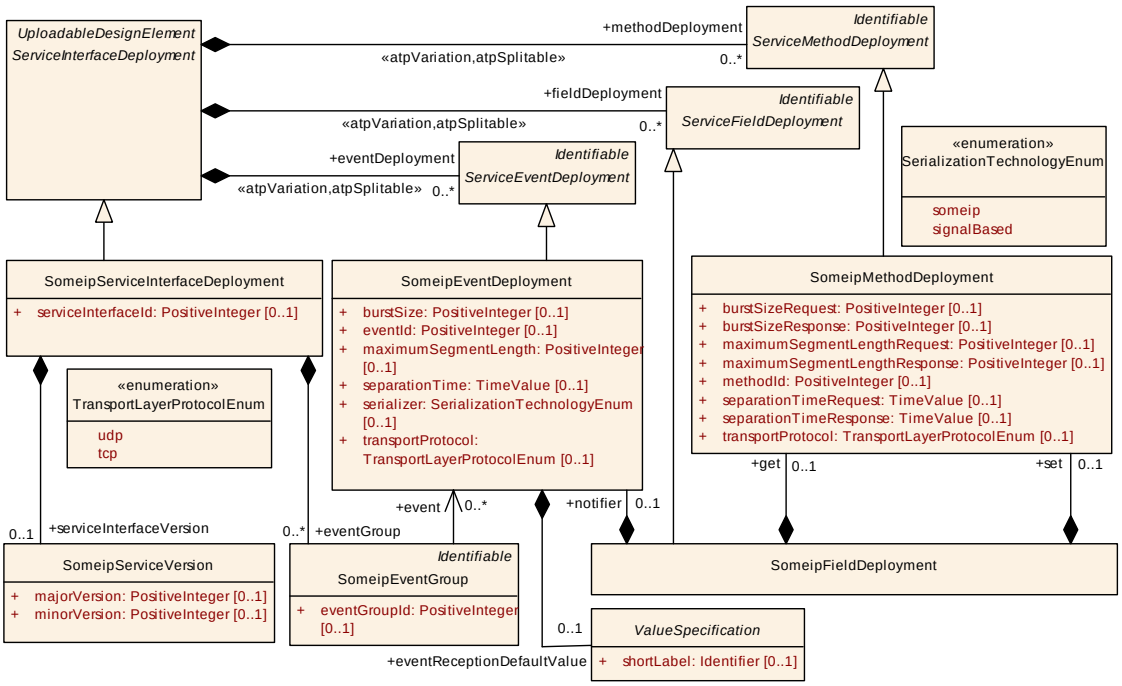


Figure 11.4: SOME/IP deployment of ServiceInterface

[TPS_MANI_03042]{DRAFT} **Definition of SOME/IP Service Version** ⌈The `SomeipServiceInterfaceDeployment.serviceInterfaceVersion` allows to define a major and a minor version for the SOME/IP Service.⌋*(RS_MANI_00024)*

[constr_3557]{DRAFT} **Mandatory `majorVersion` at `SomeipServiceInterfaceDeployment.serviceInterfaceVersion`** ⌈If the `SomeipServiceVersion` is aggregated at the `SomeipServiceInterfaceDeployment` in the role `serviceInterfaceVersion` then the attribute `SomeipServiceVersion.majorVersion` shall be defined **at the time when the creation of the manifest is finished.**⌋*()*

Class	**SomeipServiceInterfaceDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for a ServiceInterface. **Tags:** atp.recommendedPackage=ServiceInterfaceDeployments			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *ServiceInterfaceDeployment*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventGroup	SomeipEventGroup	*	aggr	SOME/IP EventGroups that are defined within the SOME/IP ServiceClass.

△

Class	**SomeipServiceInterfaceDeployment**			
serviceInterface Id	PositiveInteger	0..1	attr	Unique Identifier that identifies the ServiceInterface in SOME/IP. This Identifier is sent as Service ID in SOME/IP Service Discovery messages.
serviceInterface Version	SomeipServiceVersion	0..1	aggr	The SOME/IP major and minor Version of the Service.

Table 11.10: SomeipServiceInterfaceDeployment

[constr_10212]{DRAFT} **Multiplicity of attribute SomeipServiceInterfaceDeployment.serviceInterfaceId** ⌈For each SomeipServiceInterfaceDeployment, the attribute serviceInterfaceId shall exist **at the time when the creation of the manifest is finished.**⌋*()*

[constr_10213]{DRAFT} **Multiplicity of attribute SomeipServiceInterfaceDeployment.serviceInterfaceVersion** ⌈For each SomeipServiceInterfaceDeployment, the attribute serviceInterfaceVersion shall exist **at the time when the creation of the manifest is finished.**⌋*()*

Class	**SomeipEventGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	Grouping of events and notification events inside a ServiceInterface in order to allow subscriptions.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SomeipServiceInterfaceDeployment.eventGroup			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
event	SomeipEvent Deployment	*	ref	Reference to an event that is part of the EventGroup.
eventGroupId	PositiveInteger	0..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 11.11: SomeipEventGroup

[constr_10214]{DRAFT} **Multiplicity of attribute SomeipEventGroup.eventGroupId** ⌈For each SomeipEventGroup, the attribute eventGroupId shall exist **at the time when the creation of the manifest is finished.**⌋*()*

[TPS_MANI_03043]{DRAFT} **SOME/IP VariableDataPrototype binding** ⌈The SomeipEventDeployment meta-class provides the ability to bind a VariableDataPrototype to SOME/IP and to assign a SOME/IP Event identifier to the event with the eventId attribute.⌋*(RS_MANI_00024)*

[constr_3305]{DRAFT} **Value of attribute SomeipEventDeployment.eventId shall be unique** ⌈The value of eventId shall be unique in the context of the enclosing SomeipServiceInterfaceDeployment, unless SomeipEventDeployment.serializer is set to SerializationTechnologyEnum.signalBased **at the time when the creation of the manifest is finished.**⌋*()*

Please note that the value of each SomeipEventDeployment.eventId that is used in the context of an enclosing SomeipServiceInterfaceDeployment is allowed to

overlap with the value of each SomeipEventGroup.eventGroupId used in the same SomeipServiceInterfaceDeployment.

In other words, the range of valid numbers of SomeipEventDeployment.eventId is independent of the range of valid numbers used for SomeipEventGroup.eventGroupId and both attributes are allowed to use the same value in the context of the same SomeipServiceInterfaceDeployment.

For service-signal-translation according to [TPS_MANI_03667] special semantics applies as defined in [TPS_MANI_03666].

[constr_3408]{DRAFT} **Value range of SomeipEventDeployment.eventId** ⌈The value of eventId shall be in the range of 0..32767 **at the time when the creation of the manifest is finished**.⌋*()*

Please note that [PRS_SOMEIPSD_00517] in [30] defines special and reserved EVENT-IDs for SOME/IP and SOME/IP-SD that result in the eventId values of 0 and 32767.

[TPS_MANI_03050]{DRAFT} **Usage of SomeipEventDeployment.transportProtocol** ⌈The value of SomeipEventDeployment.transportProtocol defines over which Transport Layer Protocol the SomeipEventDeployment.event is provided.⌋*(RS_MANI_00024)*

[constr_5156]{DRAFT} **SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances** ⌈If SomeipEventDeployment.transportProtocol is set to udp then each ProvidedSomeipServiceInstance that refers the SomeipServiceInterfaceDeployment in the role serviceInterfaceDeployment shall only be mapped to a MachineDesign with a SomeipServiceInstanceToMachineMapping with a configured udpPort **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3308]{DRAFT} **SomeipEventDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances** ⌈If SomeipEventDeployment.transportProtocol is set to tcp then each ProvidedSomeipServiceInstance that refers the SomeipServiceInterfaceDeployment in the role serviceInterfaceDeployment shall only be mapped to a MachineDesign with a SomeipServiceInstanceToMachineMapping with a configured tcpPort **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_03067]{DRAFT} **SOME/IP segmentation of udp SomeipEventDeployments** ⌈If the maximumSegmentLength is set to a value and the data length is larger than maximumSegmentLength then SOME/IP shall segment the SomeipEventDeployment into several packets and transmit them over the network.

The sender shall wait the separationTime between the transmissions of segments. On the reception side, SOME/IP re-assembles the received SOME/IP segments to the original SomeipEventDeployment.⌋*(RS_MANI_00024)*

[constr_3351]{DRAFT} **SOME/IP segmentation allowed for udp SomeipEventDeployments** ⌈Attribute SomeipEventDeployment.maximumSegmentLength shall only be used if the value of attribute SomeipEventDeployment.transportProtocol is set to udp **at the time when the creation of the manifest is finished**.⌋*()*

As the SomeipServiceInterfaceDeployment is also used for the deployment of signal-based Pdus on Ethernet the attribute SomeipEventDeployment.serializer defines whether the someip or the signalBased serialization shall be used for a specific event.

[TPS_MANI_03615]{DRAFT} **SomeipEventDeployment.serializer equals someip** ⌈If the attribute SomeipEventDeployment.serializer is not defined or is set to the value someip then the event shall be serialized/de-serialized using the SOME/IP serializer.⌋*(RS_MANI_00063)*

[TPS_MANI_03591]{DRAFT} **SomeipEventDeployment.serializer equals signalBased** ⌈If the attribute SomeipEventDeployment.serializer is set to the value signalBased then the event shall be serialized/de-serialized using the signal-based approach.⌋*(RS_MANI_00063)*

This aspect is described in chapter 13.

Class	**SomeipEventDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for an Event.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceEventDeployment*			
Aggregated by	*ServiceInterfaceDeployment*.eventDeployment, SomeipFieldDeployment.notifier			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
burstSize	PositiveInteger	0..1	attr	Specifies the number of segments that shall be transmitted in a burst ignoring separationTime. SeparationTime will then only be applied between bursts. If not configured, SeparationTime will be applied between all frames.
eventId	PositiveInteger	0..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.
eventReception DefaultValue	ValueSpecification	0..1	aggr	Value used to fill the Event data on the receiver side, if less then expected data is received. The value is expected to cover the entire expected event network payload. The value specification is supposed to take the order of serialized representation of the data on the network, as opposed to the order of elements in a data type description.
maximum SegmentLength	PositiveInteger	0..1	attr	This attribute describes the length in bytes of the SOME/IP segment. This includes 8 bytes for the Request ID, Protocol Version, Interface Version, Message Type and Return Code and 4 additional SOME/IP TP bytes. If this attribute is set to a value and the data length is larger than maximumSegmentLength then the corresponding SOME/IP message will be segmented into smaller parts that are transmitted over the network.

△

Class	**SomeipEventDeployment**			
separationTime	TimeValue	0..1	attr	Sets the duration of the minimum time in seconds SOME/IP shall wait between the transmissions of segments.
serializer	SerializationTechnology Enum	0..1	attr	Defines which serialization technology shall be used.
transport Protocol	TransportLayerProtocol Enum	0..1	attr	This attribute defines over which Transport Layer Protocol this event is intended to be sent.

Table 11.12: SomeipEventDeployment

[constr_10215]{DRAFT} **Multiplicity of attribute `SomeipEventDeployment.eventId`** ⌈For each `SomeipEventDeployment`, the attribute `eventId` shall exist **at the time when the creation of the manifest is finished.**⌋*()*

[constr_10216]{DRAFT} **Multiplicity of attribute `SomeipEventDeployment.transportProtocol`** ⌈For each `SomeipEventDeployment`, the attribute `transportProtocol` shall exist **at the time when the creation of the manifest is finished.**⌋*()*

Enumeration	**SerializationTechnologyEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment
Note	This enumeration allows to choose a Serialization Technology.
Aggregated by	SomeipEventDeployment.serializer
Literal	***Description***
signalBased	Signal-Based serializer. **Tags:** atp.EnumerationLiteralIndex=1
someip	SOME/IP Serializer **Tags:** atp.EnumerationLiteralIndex=0

Table 11.13: SerializationTechnologyEnum

[constr_11000]{DRAFT} **`SomeipEventDeployment.eventReceptionDefaultValue` shall not be specified in the physical domain** ⌈Attribute `SomeipEventDeployment.eventReceptionDefaultValue` shall only be defined by

- a `NumericalValueSpecification` or
- a `TextValueSpecification`

or by

- a `RecordValueSpecification` or
- a `ArrayValueSpecification` or
- a `CompositeRuleBasedValueSpecification`

that (after further nesting levels consisting of `CompositeRuleBasedValueSpecification`, `RecordValueSpecification` and `ArrayValueSpecification` are resolved) only contain

- `NumericalValueSpecification`

- TextValueSpecification

at the time when the creation of the manifest is finished.⌋()

The point of [constr_11000] is that the **leaf** elements of the ValueSpecification aggregated in the role SomeipEventDeployment.eventReceptionDefaultValue shall only be NumericalValueSpecification or TextValueSpecification.

In other words, it is not allowed that "application-level" ValueSpecifications are used for this role.

[TPS_MANI_03044]{DRAFT} **SOME/IP ClientServerOperation binding** ⌈The SomeipMethodDeployment meta-class provides the ability to bind a ClientServerOperation to SOME/IP and to assign a SOME/IP Method identifier to the method with the methodId attribute.⌋(*RS_MANI_00024*)

[constr_3306]{DRAFT} **Value of attribute methodId shall be unique per SomeipServiceInterfaceDeployment** ⌈The value of methodId shall be unique in the context of the enclosing SomeipServiceInterfaceDeployment **at the time when the creation of the manifest is finished**.⌋()

[constr_3409]{DRAFT} **Value range of SomeipMethodDeployment.methodId** ⌈The value of methodId shall be in the range of 0..32767 **at the time when the creation of the manifest is finished**.⌋()

Please note that [PRS_SOMEIPSD_00517] in [30] defines special and reserved METHOD-IDs for SOME/IP and SOME/IP-SD that result in the methodId values of 0 and 32767.

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

[constr_3309]{DRAFT} **SomeipMethodDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances** ⌈If SomeipMethodDeployment.transportProtocol is set to udp then each ProvidedSomeipServiceInstance that refers the SomeipServiceInterfaceDeployment in the role serviceInterfaceDeployment shall only be mapped to a MachineDesign with a SomeipServiceInstanceToMachineMapping with a configured udpPort **at the time when the creation of the manifest is finished**.⌋()

[constr_3310]{DRAFT} **SomeipMethodDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances** ⌈If SomeipMethodDeployment.transportProtocol is set to tcp then each ProvidedSomeipServiceInstance that refers the SomeipServiceInterfaceDeployment in the role serviceInterfaceDeployment shall only be mapped to a MachineDesign with a SomeipServiceInstanceToMachineMapping with a configured tcpPort **at the time when the creation of the manifest is finished**.⌋()

[TPS_MANI_03068]{DRAFT} **SOME/IP segmentation of SomeipMethodDeployment Calls** ⌈If the maximumSegmentLengthRequest is set to a value and the data length is larger than maximumSegmentLengthRequest then SOME/IP shall segment the SomeipMethodDeployment Call-Message into several packets and transmit them over the network.

The sender shall wait the separationTimeRequest between the transmissions of segments. On the reception side, SOME/IP re-assembles the received SOME/IP segments to the original SomeipMethodDeployment Call-Message.⌋*(RS_MANI_00024)*

[TPS_MANI_03069]{DRAFT} **SOME/IP segmentation of SomeipMethodDeployment Responses** ⌈If the maximumSegmentLengthResponse is set to a value and the data length is larger than maximumSegmentLengthResponse then SOME/IP shall segment the SomeipMethodDeployment Response-Message into several packets and transmit them over the network.

The sender shall wait the separationTimeResponse between the transmissions of segments. On the reception side, SOME/IP re-assembles the received SOME/IP segments to the original SomeipMethodDeployment Response-Message.⌋*(RS_MANI_00024)*

[constr_3352]{DRAFT} **SOME/IP segmentation allowed for udp SomeipMethodDeployments** ⌈SomeipMethodDeployment.maximumSegmentLengthRequest and SomeipMethodDeployment.maximumSegmentLengthResponse shall only be used if SomeipMethodDeployment.transportProtocol is set to udp **at the time when the creation of the manifest is finished**.⌋*()*

Class	**SomeipMethodDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for a Method.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceMethodDeployment*			
Aggregated by	*ServiceInterfaceDeployment*.methodDeployment, SomeipFieldDeployment.get, SomeipFieldDeployment.set			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
burstSize Request	PositiveInteger	0..1	attr	Specifies the number of segments for the Method Call that shall be transmitted in a burst ignoring separation Time. SeparationTime will then only be applied between bursts. If not configured, SeparationTime will be applied between all frames.
burstSize Response	PositiveInteger	0..1	attr	Specifies the number of segments for the Method Response that shall be transmitted in a burst ignoring separationTime. SeparationTime will then only be applied between bursts. If not configured, SeparationTime will be applied between all frames.

▽

△

Class	**SomeipMethodDeployment**			
maximum SegmentLength Request	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.
maximum SegmentLength Response	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	0..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.
separationTime Request	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.
separationTime Response	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.
transport Protocol	TransportLayerProtocol Enum	0..1	attr	This attribute defines over which Transport Layer Protocol this method is intended to be sent.

Table 11.14: SomeipMethodDeployment

Class	**SomeipServiceInstanceToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class allows to map SomeipServiceInstances to a CommunicationConnector of a Machine. In this step the network configuration (IP Address, Transport Protocol, Port Number) for the ServiceInstance is defined. **Tags:** atp.recommendedPackage=ServiceInstanceToMachineMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInstanceToMachineMapping*, *UploadableDesignElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
remoteMulticast Config	SomeipRemote MulticastConfig	*	ref	This reference defines a remote multicast Address (IP Address, Port) that is used in a static configuration to setup the communication path between a service provider and service consumer. This reference shall ONLY be used if the remote address is determined from the configuration and not at runtime from the Service Discovery. **Tags:** atp.Status=candidate

▽

△

Class	**SomeipServiceInstanceToMachineMapping**			
remoteUnicast Config	SomeipRemoteUnicast Config	*	ref	In case that a static service connection is used and a single peer exists this element is used to statically configure the remote peer's address. **Tags:** atp.Status=candidate
tcpPort	ApApplicationEndpoint	0..1	ref	local TcpPort that will be used by the ServiceInstance for the communication.
udpCollection BufferSize Threshold	PositiveInteger	0..1	attr	Specifies the amount of data in bytes that shall be buffered for data transmission over the udp connection specified by this SomeipServiceInstanceToMachine Mapping. If this attribute is set to a value, then the data collection feature is enabled.
udpPort	ApApplicationEndpoint	0..1	ref	local UdpPort that will be used by the ServiceInstance for the communication.

Table 11.15: SomeipServiceInstanceToMachineMapping

[TPS_MANI_03057]{DRAFT} **SOME/IP `Field` binding** ⌈The `SomeipFieldDeployment` meta-class provides the ability to bind a `Field` to SOME/IP.

If the `Field` contains a notifier (`hasNotifier` = true), it is possible to assign a SOME/IP notifier identifier to the `field` by setting the value of attribute `SomeipFieldDeployment.notifier.eventId`.

If the `Field` contains a getter method (`hasGetter` = true), it is possible to assign a SOME/IP method 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 method identifier to the `field` by setting the value of attribute `SomeipFieldDeployment.set.methodId`⌋ *(RS_MANI_00024)*

Please note that each `methodId` and each `eventId` of a `SomeipFieldDeployment` shall be unique in the context of a `ServiceInterface` as defined in [constr_3306] and [constr_3305].

Class	**SomeipFieldDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	SOME/IP configuration settings for a Field.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceFieldDeployment*			
Aggregated by	*ServiceInterfaceDeployment*.fieldDeployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
get	SomeipMethod Deployment	0..1	aggr	This aggregation represents the setting of the get method.
notifier	SomeipEvent Deployment	0..1	aggr	This aggregation represents the settings of the notifier.
set	SomeipMethod Deployment	0..1	aggr	This aggregation represents the settings of the set method

Table 11.16: SomeipFieldDeployment

[constr_3362]{DRAFT} **SomeipEventDeployments aggregated by a SomeipFieldDeployment** ⌈A SomeipEventDeployment that is aggregated by a SomeipFieldDeployment in the role notifier shall not reference a VariableDataPrototype in the role event **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3363]{DRAFT} **SomeipMethodDeployments aggregated by a SomeipFieldDeployment** ⌈A SomeipMethodDeployment that is aggregated by a SomeipFieldDeployment in the role get or set shall not reference a ClientServerOperation in the role method **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_03227]{DRAFT} **Usage of ephemeral ports** ⌈Ephemeral ports are short-lived transport protocol ports that are allocated automatically by the communication middleware. In case the port number in the ApApplicationEndpoint (TpPort.portNumber) is configured to 0, an *ephemeral* port shall be used. If the port number is configured to a value different from 0, exactly that value shall be used.⌋*(RS_MANI_00024)*

11.2.2 DDS Service Interface Deployment

This chapter describes the DDS [31] deployment of a ServiceInterface.

[TPS_MANI_03525]{DRAFT} **DDS ServiceInterface binding** ⌈The DdsServiceInterfaceDeployment meta-class provides the ability to bind a ServiceInterface to DDS and to assign a DDS Service identifier to the ServiceInterface with the serviceInterfaceId attribute.⌋*(RS_MANI_00038)*

[constr_3690]{DRAFT} **DdsServiceInterfaceDeployment.serviceInterfaceId value shall not conflict with topic-based service discovery** ⌈The value "discovery" for DdsServiceInterfaceDeployment.serviceInterfaceId is reserved and shall not be used for modeled DdsServiceInterfaceDeployments Note that in the context of the *AUTOSAR adaptive platform*, the imposition time for these constraints shall be **at the time when the system design is complete**.⌋*()*

[constr_3690] is defined according to [SWS_CM_90508] of [10].

[TPS_MANI_03556]{DRAFT} **DDS-RPC Service Binding** ⌈The DdsServiceInterfaceDeployment meta-class provides the ability to configure the name of the DDS Request and Reply Topics associated with a DDS-RPC Service with the methodRequestTopicName and methodReplyTopicName attributes, respectively. DDS-RPC Services are the mechanisms specified in the OMG RPC over DDS specification (DDS-RPC [32]) to handle method calls with DDS.

The methodRequestTopicName and methodReplyTopicName attributes are optional, if unspecified they shall be configured as specified in the DDS-RPC specification.⌋*(RS_MANI_00038)*

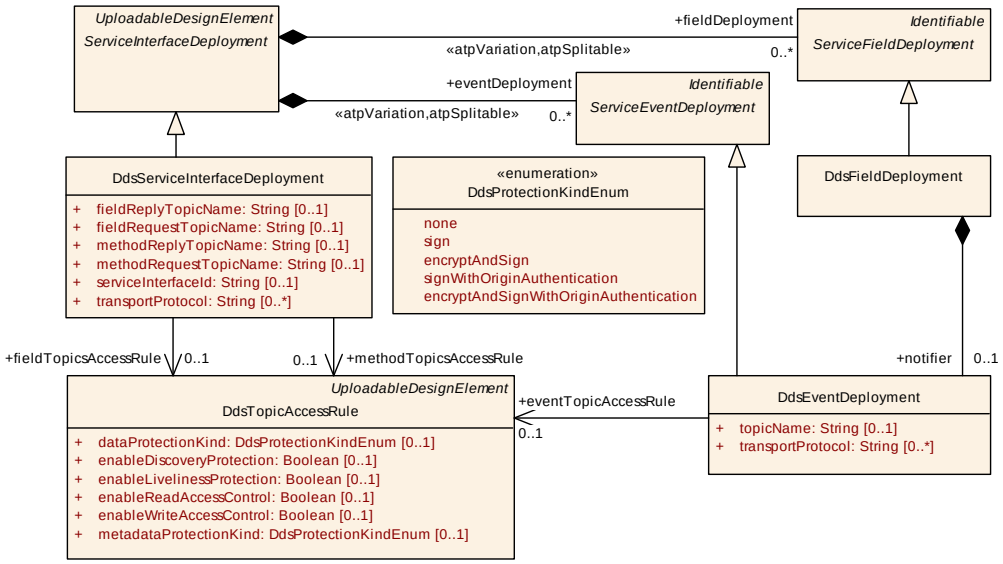


Figure 11.5: DDS deployment of ServiceInterface

[TPS_MANI_03526]{DRAFT} **DDS `VariableDataPrototype` binding** ⌈The `DdsEventDeployment` meta-class provides the ability to bind a `VariableDataPrototype` to DDS and to assign a DDS Topic to the `event` with the `topicName` attribute. Moreover, the meta-class provides the ability to configure the `transportProtocol`s over which the `VariableDataPrototype` may be accessed.⌋ *(RS_MANI_00038)*

Class	**DdsServiceInterfaceDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	DDS configuration settings for a ServiceInterface. **Tags:** atp.recommendedPackage=ServiceInterfaceDeployments			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *ServiceInterfaceDeployment*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
fieldReplyTopic Name	String	0..1	attr	Name of the DDS Reply Topic associated with the Field.
fieldRequest TopicName	String	0..1	attr	Name of the DDS Request Topic associated with the Field.
fieldTopics AccessRule	DdsTopicAccessRule	0..1	ref	DDS Security access rule applicable to the DDS Topics used for service interface field access methods (Get, Set).
methodReply TopicName	String	0..1	attr	Name of the DDS Reply Topic associated with the Method.
methodRequest TopicName	String	0..1	attr	Name of the DDS Request Topic associated with the Method.
methodTopics AccessRule	DdsTopicAccessRule	0..1	ref	DDS Security access rule applicable to the DDS Topics used for service interface methods.
serviceInterface Id	String	0..1	attr	Unique Identifier that identifies the ServiceInterface in DDS. This Identifier is encoded in the USER_DATA QoS of the DomainParticipant associated with the Service Instance and its value is propagated by DDS Discovery messages.
transport Protocol	String	*	attr	This attribute defines over which Transport Layer Protocol(s) this Method is intended to be sent.

Table 11.17: DdsServiceInterfaceDeployment

[constr_10217]{DRAFT} **Multiplicity of the attribute `DdsServiceInterfaceDeployment.serviceInterfaceId`** ⌈For each `DdsServiceInterfaceDeployment`, the attribute `serviceInterfaceId` shall exist **at the time when the creation of the manifest is finished.**⌋*()*

Class	**DdsEventDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	DDS configuration settings for an Event.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceEventDeployment*			
Aggregated by	DdsFieldDeployment.notifier, *ServiceInterfaceDeployment*.eventDeployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventTopic AccessRule	DdsTopicAccessRule	0..1	ref	DDS Security access rule applicable to the DDS Topics used for the service interface event.
topicName	String	0..1	attr	Name of the DDS Topic associated with the Event.
transport Protocol	String	*	attr	This attribute defines over which Transport Layer Protocol(s) this event is intended to be sent.

Table 11.18: DdsEventDeployment

[TPS_MANI_03557]{DRAFT} **DDS `ClientServerOperation` Binding** ⌈There exists no concrete subclass of `ServiceMethodDeployment` to bind a `ClientServerOperation` to DDS. This binding is done with the `DdsServiceInterfaceDeployment` (see [TPS_MANI_03556]).⌋*(RS_MANI_00038)*

[TPS_MANI_03558]{DRAFT} **DDS `Field` Binding** ⌈The `DdsFieldDeployment` meta-class provides the ability to bind a `Field` to DDS.

To bind the `Field`'s notification event the `notifier` is used to define a DDS Topic. To assign the get/set `Field` methods the `fieldRequestTopicName` and `fieldReplyTopicName` are used to define a DDS Topic.

The `fieldRequestTopicName` and `fieldReplyTopicName` attributes are optional, if unspecified they shall be configured as specified in the DDS-RPC specification.⌋*(RS_MANI_00038)*

[TPS_MANI_03622]{DRAFT} **DDS Transport Protocols are up to the stack implementer** ⌈Underlying transports below the RTPS protocol are not part of the DDS OMG standard (QoS APIs [31]) or (XML schema [33]). It is up to each DDS implementation vendor to decide which transports are supported and how those are expressed in APIs and XML.⌋*(RS_MANI_00038)*

Class	**DdsFieldDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	DDS configuration settings for a Field.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceFieldDeployment*			
Aggregated by	*ServiceInterfaceDeployment*.fieldDeployment			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
notifier	DdsEventDeployment	0..1	aggr	This aggregation represents the settings of the notifier.

Table 11.19: DdsFieldDeployment

[constr_3563]{DRAFT} **Mandatory topic name values** ⌈The attributes `methodRequestTopicName`, `methodReplyTopicName`, `fieldRequestTopicName`, `fieldReplyTopicName`, `topicName` shall specify string values, each of them unique within the service interface **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_03662]{DRAFT} **Configuration of Topic access rules** ⌈The `DdsTopicAccessRule` meta-class defines rules generally applicable to DDS Topics defined by Service Interface deployments, determining the protection mechanisms applicable for discovery, liveliness, access control, metadata and user data.

The `enableReadAccessControl` and `enableWriteAccessControl` attributes are not meant to interfere with or duplicate `ComGrant`-based access rules. These are just an additional mechanisms for integrators to fully disable access control for an specific topic at Service Interface Deployment level, in case that is desirable for an specific deployment scenario.⌋*(RS_MANI_00038)*

Class	**DdsTopicAccessRule**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	DDS Topic access rule definition. **Tags:** atp.recommendedPackage=DdsTopicAccessRules			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataProtection Kind	DdsProtectionKind Enum	0..1	attr	Defines the data protection policy applicable to metadata related to the DDS Topic(s).
enable Discovery Protection	Boolean	0..1	attr	Defines whether discovery protection mechanisms should apply to the DDS Topic(s).
enable Liveliness Protection	Boolean	0..1	attr	Defines whether liveliness protection mechanisms should apply to the DDS Topic(s).
enableRead AccessControl	Boolean	0..1	attr	Defines whether read access control mechanisms should apply to the DDS Topic(s).
enableWrite AccessControl	Boolean	0..1	attr	Defines whether write access control mechanisms should apply to the DDS Topic(s).
metadata ProtectionKind	DdsProtectionKind Enum	0..1	attr	Defines the data protection policy applicable to metadata related to the DDS Topic(s).

Table 11.20: DdsTopicAccessRule

[constr_3680]{DRAFT} **Existence of attributes for `DdsTopicAccessRule`** ⌈The following attributes of `DdsTopicAccessRule` shall exist **at the time when the creation of the manifest is finished**

- `enableDiscoveryProtection`
- `enableLivelinessProtection`
- `enableReadAccessControl`
- `enableWriteAccessControl`

- metadataProtectionKind
- dataProtectionKind

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

[constr_3681]{DRAFT} **Supported values of DdsTopicAccessRule.dataProtectionKind** ⌈Only values none, sign, or encryptAndSign from DdsProtectionKindEnum shall be used when setting DdsTopicAccessRule.dataProtectionKind **at the time when the creation of the manifest is finished**.⌋()

[constr_3683]{DRAFT} **Attributes referencing DdsTopicAccessRule** ⌈DdsServiceInterfaceDeployment.fieldTopicsAccessRule, DdsServiceInterfaceDeployment.methodTopicsAccessRule, and DdsEventDeployment.eventTopicAccessRule shall be set if the Service Interface Deployment is to be used by Service Instances relying in DDS Security (meaning DdsServiceInstanceToMachineMapping.secureComPropsForDds is defined) **at the time when the creation of the manifest is finished**.⌋()

11.2.3 User Defined Service Interface Deployment

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.

[TPS_MANI_03045]{DRAFT} **User-Defined ServiceInterface binding** ⌈The UserDefinedServiceInterfaceDeployment meta-class provides the ability to bind a ServiceInterface that is referenced in the role serviceInterface to a middleware technology that is not standardized by AUTOSAR.⌋(*RS_MANI_00014*)

Please note that UserDefinedServiceInterfaceDeployment is Identifiable and therefore it is able to describe special data (sdg) which is not represented by the standard model.

Class	**UserDefinedServiceInterfaceDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	UserDefined configuration settings for a ServiceInterface. **Tags:** atp.recommendedPackage=ServiceInterfaceDeployments			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *ServiceInterfaceDeployment*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 11.21: UserDefinedServiceInterfaceDeployment

[TPS_MANI_01165]{DRAFT} **Standardized value of `UserDefinedServiceInterfaceDeployment.category`** ⌈The AUTOSAR Standard reserves the following value for attribute `UserDefinedServiceInterfaceDeployment.category`:

- `SERVICE_INTERFACE_DEPLOYMENT_IPC`

It is possible to use a custom, non-standardized value for the attribute `UserDefinedServiceInterfaceDeployment.category` but this option comes with the obligation to use a value that is guaranteed to not clash with possible future extensions of the collection of standardized values.⌋*(RS_MANI_00014)*

IPC communication may or may not require configuration settings that nevertheless aren't standardized by AUTOSAR. The best support that the AUTOSAR standard can deliver is the provision of meta-classes that can be taken as the basis to define configuration settings by means of the definition of `Sdg`.

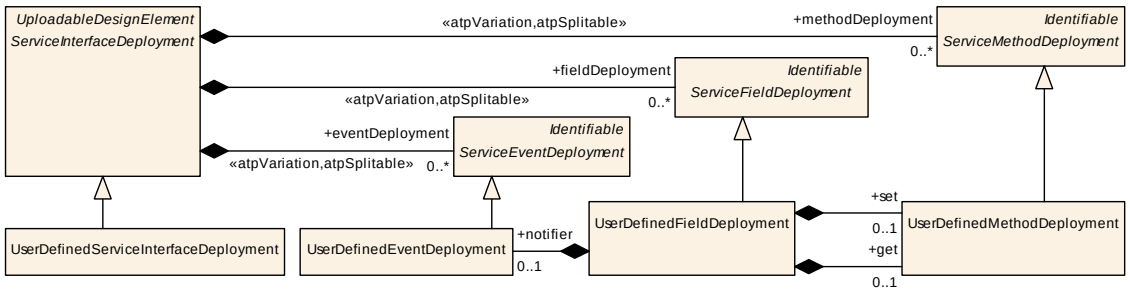


Figure 11.6: User defined deployment of ServiceInterface

[constr_1570]{DRAFT} **Restriction for `UserDefinedServiceInterfaceDeployment` of `category` `SERVICE_INTERFACE_DEPLOYMENT_IPC`** ⌈An `AdaptivePlatformServiceInstance` that references a `UserDefinedServiceInterfaceDeployment` of `category` `SERVICE_INTERFACE_DEPLOYMENT_IPC` shall **only** be referenced by a `UserDefinedServiceInstanceToMachineMapping` in the role `serviceInstance` that in turn references a `UserDefinedCommunicationConnector` **at the time when the creation of the manifest is finished**.⌋*()*

Class	**UserDefinedServiceInstanceToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class allows to map UserDefinedServiceInstances to a CommunicationConnector of a Machine. **Tags:** atp.recommendedPackage=ServiceInstanceToMachineMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInstanceToMachineMapping*, *UploadableDesignElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 11.22: UserDefinedServiceInstanceToMachineMapping

Rationale for [constr_1570]: for a local IPC binding it is sometimes necessary to define properties of the IPC system. And for this purpose the UserDefinedCommunicationConnector mapped to an AdaptivePlatformServiceInstance can be used to define global properties (e.g. for service discovery) of a given "IPC-Domain".

In other words, each defined UserDefinedCommunicationConnector may represent such an "IPC-Domain" that requires a dedicated configuration on the basis of the definition of Sdgs.

Class	**UserDefinedCommunicationConnector**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::CddSupport			
Note	This element allows the modeling of arbitrary Communication Connectors.			
Base	*ARObject*, *CommunicationConnector*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcuInstance.connector, MachineDesign.communicationConnector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 11.23: UserDefinedCommunicationConnector

[TPS_MANI_03046]{DRAFT} **User defined VariableDataPrototype binding** ⌈The UserDefinedEventDeployment meta-class provides the ability to bind a VariableDataPrototype that is referenced in the role event to a middleware technology that is not standardized by AUTOSAR.⌋*(RS_MANI_00014)*

Please note that UserDefinedEventDeployment is Identifiable and therefore it is able to describe special data (sdg) which is not represented by the standard model.

Class	**UserDefinedEventDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	UserDefined configuration settings for an Event.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceEventDeployment*			
Aggregated by	*ServiceInterfaceDeployment*.eventDeployment, UserDefinedFieldDeployment.notifier			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 11.24: UserDefinedEventDeployment

[TPS_MANI_03047]{DRAFT} **User defined ClientServerOperation binding** ⌈The UserDefinedMethodDeployment meta-class provides the ability to bind a ClientServerOperation that is referenced in the role method to a middleware technology that is not standardized by AUTOSAR.⌋*(RS_MANI_00014)*

Please note that UserDefinedMethodDeployment is Identifiable and therefore it is able to describe special data (sdg) which is not represented by the standard model.

Class	UserDefinedMethodDeployment			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	UserDefined configuration settings for a Method.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceMethodDeployment*			
Aggregated by	*ServiceInterfaceDeployment*.methodDeployment, UserDefinedFieldDeployment.get, UserDefinedFieldDeployment.set			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 11.25: UserDefinedMethodDeployment

[TPS_MANI_03048]{DRAFT} **User defined `Field` binding** ⌈The `UserDefinedFieldDeployment` meta-class provides the ability to bind a `Field` that is referenced in the role `field` to a middleware technology that is not standardized by AUTOSAR.⌋ *(RS_MANI_00014)*

Please note that `UserDefinedFieldDeployment` is `Identifiable` and therefore it is able to describe special data (sdg) which is not represented by the standard model.

Class	UserDefinedFieldDeployment			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInterfaceDeployment			
Note	UserDefined configuration settings for a Field.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceFieldDeployment*			
Aggregated by	*ServiceInterfaceDeployment*.fieldDeployment			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
get	UserDefinedMethod Deployment	0..1	aggr	This aggregation represents the settings of the get method
notifier	UserDefinedEvent Deployment	0..1	aggr	This aggregation represents the settings of the notifier.
set	UserDefinedMethod Deployment	0..1	aggr	This aggregation represents the settings of the set method

Table 11.26: UserDefinedFieldDeployment

[constr_3417]{DRAFT} **`UserDefinedEventDeployment`s aggregated by a `UserDefinedFieldDeployment`** ⌈A `UserDefinedEventDeployment` that is aggregated by a `UserDefinedFieldDeployment` in the role `notifier` shall not reference a `VariableDataPrototype` in the role `event` **at the time when the creation of the manifest is finished**.⌋ *()*

[constr_3418]{DRAFT} **`UserDefinedMethodDeployment`s aggregated by a `UserDefinedFieldDeployment`** ⌈A `UserDefinedMethodDeployment` that is aggregated by a `UserDefinedFieldDeployment` in the role `get` or `set` shall not reference a `ClientServerOperation` in the role `method` **at the time when the creation of the manifest is finished**.⌋ *()*

11.3 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.

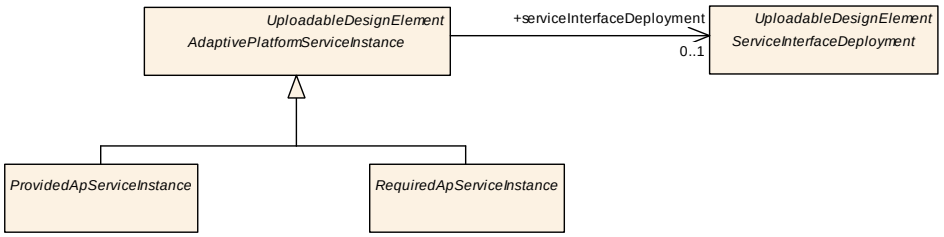


Figure 11.7: Modeling of the AdaptivePlatformServiceInstance

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 Manifest descriptions.

Class	***AdaptivePlatformServiceInstance*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a service instance in an abstract way.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	*ProvidedApServiceInstance*, *RequiredApServiceInstance*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
e2eEvent ProtectionProps	End2EndEvent ProtectionProps	*	aggr	This aggregation allows to protect an event or a field notifier that is defined inside of the ServiceInterface that is referenced by the ServiceInstance in the role service Interface.
e2eMethod ProtectionProps	End2EndMethod ProtectionProps	*	aggr	This aggregation allows to protect a method or a field getter or a field setter that is defined inside of the Service Interface that is referenced by the ServiceInstance in the role serviceInterface
secureCom Config	ServiceInterface ElementSecureCom Config	*	aggr	Configuration settings to secure the communication of ServiceInterface elements.
serviceInterface Deployment	ServiceInterface Deployment	0..1	ref	Reference to a ServiceInterfaceDeployment that identifies the ServiceInterface that is represented by the Service Instance.

Table 11.27: AdaptivePlatformServiceInstance

Class	***RequiredApServiceInstance*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in an abstract way.			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *Uploadable PackageElement*			
Subclasses	DdsRequiredServiceInstance, RequiredSomeipServiceInstance, RequiredUserDefinedServiceInstance			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 11.28: RequiredApServiceInstance

Class	***ProvidedApServiceInstance*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in an abstract way.			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *Uploadable PackageElement*			
Subclasses	DdsProvidedServiceInstance, ProvidedSomeipServiceInstance, ProvidedUserDefinedServiceInstance			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 11.29: ProvidedApServiceInstance

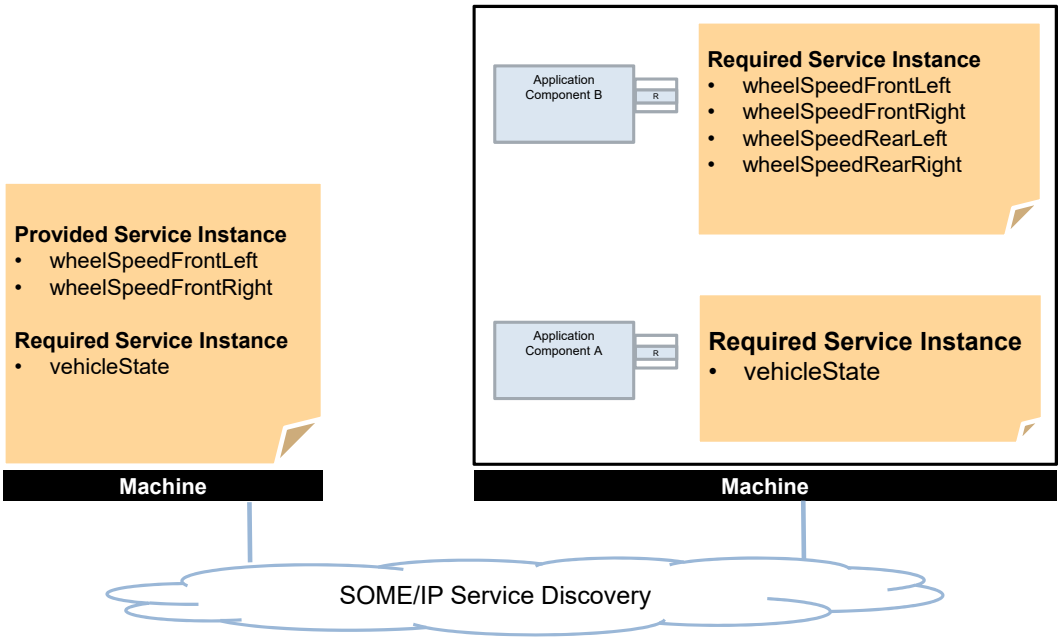


Figure 11.8: Different approaches for ServiceInstanceMapping

There are two alternative ways to relate an AdaptivePlatformServiceInstance with a MachineDesign as described in [TPS_MANI_03000] and [TPS_MANI_03001]. Figure Figure 11.8 shows both approaches in an example.

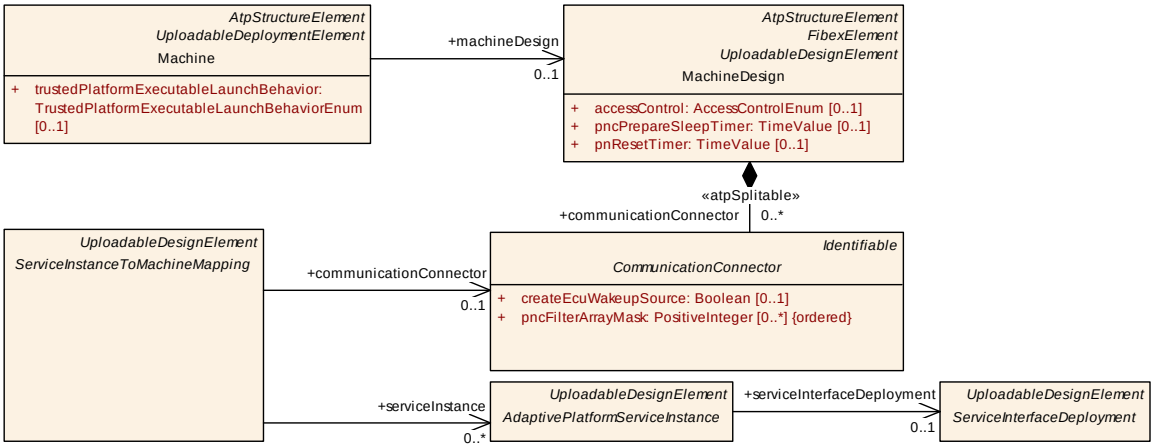


Figure 11.9: ServiceInstanceToMachineMapping

[TPS_MANI_03001]{DRAFT} **Mapping of AdaptivePlatformServiceInstance to a MachineDesign** ⌈ServiceInstanceToMachineMapping is used to assign one or several AdaptivePlatformServiceInstances to (via a Communication-Connector) a MachineDesign. This allows to define a “black box” machine view without any assumption on the application software but with all necessary information to configure the communication (e.g. SOME/IP).⌋ *(RS_MANI_00009)*

Class	***ServiceInstanceToMachineMapping*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class represents the ability to map one or several AdaptivePlatformServiceInstances to a CommunicationConnector of a Machine.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	DdsServiceInstanceToMachineMapping, SomeipServiceInstanceToMachineMapping, UserDefined ServiceInstanceToMachineMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Connector	Communication Connector	0..1	ref	Reference to the Machine to which the ServiceInstance is mapped.
secOcCom PropsFor Multicast	SecOcSecureCom Props	*	ref	Reference to communication security configuration settings that are valid for the udp multicast endpoint (Port + Multicast IP Address) defined by the ServiceInstanceTo MachineMapping.
secureCom PropsForTcp	SecureComProps	0..1	ref	Reference to communication security configuration settings that are valid for the tcp unicast endpoint (Tcp Port + Unicast IP Address) defined by the Service InstanceToMachineMapping.
secureCom PropsForUdp	SecureComProps	0..1	ref	Reference to communication security configuration settings that are valid for the udp unicast endpoint (Udp Port + Unicast IP Address) defined by the Service InstanceToMachineMapping.
serviceInstance	AdaptivePlatform ServiceInstance	*	ref	Reference to a ServiceInstance that is mapped to the Machine.

Table 11.30: ServiceInstanceToMachineMapping

[constr_5155]{DRAFT} **SomeipServiceInstanceToMachineMapping only supports a single Address Family** ⌈A SomeipServiceInstanceToMachineMapping shall only support a single Address Family, i.e. either IPv4 or IPv6. If IPv4 is defined for IP unicast communication according to [constr_3288] then the SomeipProvidedEventGroups in ProvidedSomeipServiceInstances that are referenced by the SomeipServiceInstanceToMachineMapping shall only define an ipv4MulticastIpAddress.

If IPv6 is defined for IP unicast communication according to [constr_3288] then the SomeipProvidedEventGroups in ProvidedSomeipServiceInstances that are referenced by the SomeipServiceInstanceToMachineMapping shall only define an ipv6MulticastIpAddress.

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

[TPS_MANI_03292]{DRAFT} **Semantic of SomeipServiceInstanceToMachineMapping.tcpPort** ⌈The SomeipServiceInstanceToMachineMapping.tcpPort defines a TCP-Port number that will be used in the following way:

- In case of a ProvidedSomeipServiceInstance, the value of attribute tcpPort will be transmitted in the SD-Offer message to the client and this port number will be used in the following way:
  - Method communication: over this port the ProvidedSomeipServiceInstance will receive the method call messages from the clients and over the same port the ProvidedSomeipServiceInstance will send the responses to the service consumers.
  - Event communication: over this port the ProvidedSomeipServiceInstance will transmit the event messages to the subscribed client in IP-Unicast case.
- In case of a RequiredSomeipServiceInstance, the value of attribute tcpPort will be transmitted in the SD-SubscribeEventGroup message to the server and this port number will be used in the following way:
  - Method communication: over this port the RequiredSomeipServiceInstance will send the method call messages to the ProvidedSomeipServiceInstance and over the same port the RequiredSomeipServiceInstance will receive the responses from the service provider.
  - Event communication: over this port the RequiredSomeipServiceInstance will receive the event messages in IP-Unicast case.

⌋*(RS_MANI_00009)*

[TPS_MANI_03293]{DRAFT} **Semantic of SomeipServiceInstanceToMachineMapping.udpPort** ⌈The SomeipServiceInstanceToMachineMapping.udpPort defines a UDP-Port number that will be used in the following way:

- In case of a ProvidedSomeipServiceInstance, the value of attribute udpPort will be transmitted in the SD-Offer message to the client and this port number will be used in the following way:
  - Method communication: over this port the ProvidedSomeipServiceInstance will receive the method call messages from the clients and over the same port the ProvidedSomeipServiceInstance will send the responses to the service consumers.
  - Event communication: over this port the ProvidedSomeipServiceInstance will transmit the event messages to the subscribed client in IP-Unicast case.
- In case of a RequiredSomeipServiceInstance, the value of attribute udpPort will be transmitted in the SD-SubscribeEventGroup message to the server and this port number will be used in the following way:
  - Method communication: over this port the RequiredSomeipServiceInstance will send the method call messages to the ProvidedSomeipServiceInstance and over the same port the RequiredSomeipServiceInstance will receive the responses from the service provider.
  - Event communication: over this port the RequiredSomeipServiceInstance will receive the event messages in IP-Unicast case.

⌋(*RS_MANI_00009*)

[constr_3487]{DRAFT} **TCP endpoint can only serve provided or required service instances exclusively** ⌈ServiceInstanceToMachineMapping is not allowed to refer to a ProvidedApServiceInstance and at the same time a RequiredApServiceInstance in the role serviceInstance if

- the ServiceInterfaceDeployment that is referenced by the ProvidedApServiceInstance in the role serviceInterfaceDeployment and
- the ServiceInterfaceDeployment that is referenced by the RequiredApServiceInstance in the role serviceInterfaceDeployment

both contain defined tcp content that is described by the transportProtocol attribute in the deployment elements of SOME/IP or DDS.

In other words a TCP endpoint can only serve provided or required service instances exclusively.

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

The reason for [constr_3487] is that the POSIX Socket API does not support the binding of several TCP sockets onto the same tuple <local IP address, local port>. But this would be necessary if a service is provided and consumed over the same TCP Endpoint.

[TPS_MANI_03000]{DRAFT} **Mapping of `AdaptivePlatformServiceInstance` to `PortPrototype`s** ⌈`ServiceInstanceToPortPrototypeMapping` is used to assign an `AdaptivePlatformServiceInstance` to a `PortPrototype` of a `SwComponentType`. This allows to define how specific `PortPrototype`s of a Software Component are represented in the middleware in terms of the service configuration.⌋*(RS_MANI_00011)*

In other words, the "outside" appearance of a `PortPrototype` from the middleware point of view is the `AdaptivePlatformServiceInstance`, or the concrete subclasses `RequiredApServiceInstance` and `ProvidedApServiceInstance`.

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.

[TPS_MANI_01316]{DRAFT} **Existence of `ServiceInstanceToPortPrototypeMapping.processDesign`** ⌈The reference `ServiceInstanceToPortPrototypeMapping.processDesign` shall only be used in a design-level modeling scenario where a pre-assignment of a given `ServiceInstanceToPortPrototypeMapping` to a specific `ProcessDesign` is intended.

By this means it is possible to express that one `Executable` is foreseen to be executed in multiple instances and it is also possible to assign service instances to each of the foreseen `ProcessDesign`s that represent instances of `Executable` at design time.⌋*(RS_MANI_00009)*

Class	**ServiceInstanceToPortPrototypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class represents the ability to assign a transport layer dependent ServiceInstance to a Port Prototype. With this mapping it is possible to define how specific PortPrototypes are represented in the middleware in terms of service configuration. **Tags:** atp.recommendedPackage=ServiceInstanceToPortPrototypeMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
portPrototype	PortPrototype	0..1	iref	Reference to a specific PortPrototype that represents the ServiceInstance. **Stereotypes:** atpUriDef **InstanceRef implemented by:** PortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	Reference to the Process in which the enclosing Service InstanceToPortPrototypeMapping is executed. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=process

▽

△

Class	**ServiceInstanceToPortPrototypeMapping**			
processDesign	ProcessDesign	0..1	ref	Reference to the ProcessDesign in which the Executable that contains the SoftwareComponent and the referenced PortPrototype is executed. **Stereotypes:** atpUriDef
serviceInstance	AdaptivePlatform ServiceInstance	0..1	ref	Reference to a ServiceInstance that is represented in the Software Component by the mapped group of Port Prototypes.

Table 11.31: ServiceInstanceToPortPrototypeMapping

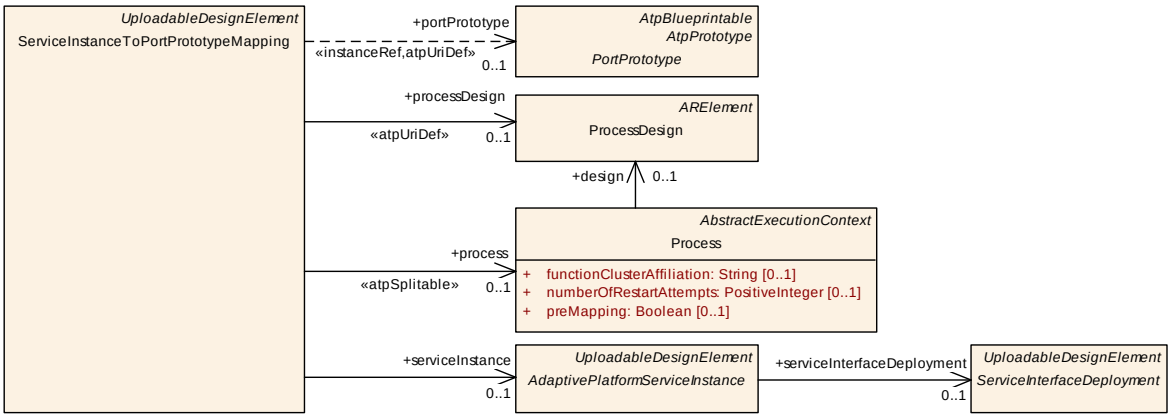


Figure 11.10: ServiceInstanceToPortPrototypeMapping

[TPS_MANI_01317]{DRAFT} **Existence of `ServiceInstanceToPortPrototypeMapping.process`** ⌈The reference `ServiceInstanceToPortPrototypeMapping.process` shall be only used in a deployment-level modeling where the integration of a `SoftwareCluster` is created. The reference has the role to identify the actual `Process` used in the execution manifest.

This reference overwrites a potentially existing reference to a `ProcessDesign` in the context of the enclosing `ServiceInstanceToPortPrototypeMapping`⌋*(RS_MANI_00009)*

Please note that if both `ServiceInstanceToPortPrototypeMapping.processDesign` and `process` exist, the latter gets the higher significance because it is created by an integrator who may overrule design decisions on the basis of superior knowledge about the context.

In such a case it is acceptable that the `Process` references a different `ProcessDesign` than the one referenced in the role `ServiceInstanceToPortPrototypeMapping.processDesign`. This is just the result of superior knowledge of the integrator over the designer.

11.3.1 SOME/IP Service Instance Deployment

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

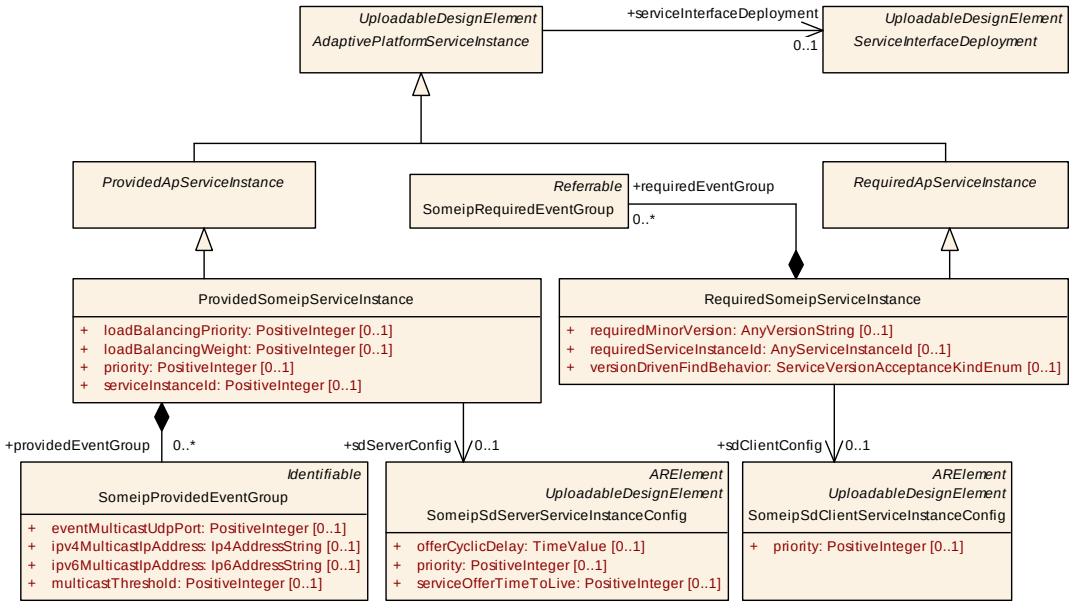


Figure 11.11: SOME/IP Service Instances

Primitive	**AnyServiceInstanceId**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This is a positive integer or the literal ALL (the value ANY is technically supported but deprecated) which can be denoted in decimal, octal and hexadecimal. The value is between 0 and 65535. **Tags:** xml.xsd.customType=ANY-SERVICE-INSTANCE-ID xml.xsd.pattern=[1-9][0-9]*|0[xX][0-9a-fA-F]+|0[0-7]*|0[bB][0-1]+|ANY|ALL xml.xsd.type=string

Table 11.32: AnyServiceInstanceId

11.3.1.1 Provided Service Instance

The `ProvidedSomeipServiceInstance` defines the `serviceInstanceId` for the Service Instance of the `SomeipServiceInterfaceDeployment` that is referenced with the `serviceInterfaceDeployment` reference.

It means that the Server on which the `ProvidedSomeipServiceInstance` is deployed offers the Service Instance over SOME/IP with the `serviceInstanceId` and `serviceInterfaceId`.

Class	**ProvidedSomeipServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in a concrete implementation on top of SOME/IP. **Tags:** atp.recommendedPackage=ServiceInstances			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *ProvidedApServiceInstance*, *Referrable*, *Uploadable DesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
capability Record (ordered)	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service.
eventProps	SomeipEventProps	*	aggr	Configuration settings for individual events that are provided by the ServiceInstance.
loadBalancing Priority	PositiveInteger	0..1	attr	This attribute is used to specify the priority in the load balancing option of SOME/IP that is added to the Offer Service. When a client searches for all service instances of a service, the client shall choose the service instance with highest priority if one is defined.
loadBalancing Weight	PositiveInteger	0..1	attr	This attribute is used to specify the weight in the load balancing option of SOME/IP that is added to the Offer Service. When a client searches for all service instances of a service, the client shall choose the service instance with highest priority if one is defined. If several service instances exist with the highest priority the service instance shall be chosen based on the weights of the service instances.
method ResponseProps	SomeipMethodProps	*	aggr	Configuration settings for individual methods that are provided by the ServiceInstance.
priority	PositiveInteger	0..1	attr	This attribute defines the VLAN frame priority for SOME/IP messages that are resulting from this ProvidedSomeip ServiceInstance (Method and Event communication). Values from 0 (best effort) to 7 (highest) are allowed.
providedEvent Group	SomeipProvidedEvent Group	*	aggr	List of EventGroups that are provided by the Service Instance.
sdServerConfig	SomeipSdServer ServiceInstanceConfig	0..1	ref	Server specific configuration settings relevant for the SOME/IP service discovery.
serviceInstance Id	PositiveInteger	0..1	attr	Identification number that is used by SOME/IP service discovery to identify the instance of the service. The value 65535 for service instance id is reserved and should not be used.

Table 11.33: ProvidedSomeipServiceInstance

[constr_10218]{DRAFT} **Multiplicity of reference in the role `ProvidedSomeipServiceInstance.sdServerConfig`** ⌈For each `ProvidedSomeipServiceInstance`, the reference in the role `sdServerConfig` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10219]{DRAFT} **Multiplicity of attribute `ProvidedSomeipServiceInstance.serviceInstanceId`** ⌈For each `ProvidedSomeipServiceInstance`, the attribute `serviceInstanceId` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_3287]{DRAFT} **Mandatory information of a ProvidedSomeipServiceInstance** ⌈The ProvidedSomeipServiceInstance shall always define the serviceInstanceId **at the time when the creation of the manifest is finished**.⌋*()*

[constr_1770]{DRAFT} **Value of ProvidedSomeipServiceInstance.serviceInstanceId** ⌈For each ProvidedSomeipServiceInstance.serviceInstanceId, the value shall be in the range 0..65534 **at the time when the creation of the manifest is finished**.⌋*()*

Rationale for [constr_1770]: on protocol level, the value 65535 represents the "ALL" semantics that can only be used in find messages.

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.

[constr_5338]{DRAFT} **ProvidedSomeipServiceInstance shall offer all SomeipEventGroups for subscription** ⌈In the scope of a ProvidedSomeipServiceInstance, SomeipProvidedEventGroups shall be defined such that

- every aggregated ProvidedSomeipServiceInstance.providedEventGroup references a SomeipEventGroup in the context of the SomeipServiceInterfaceDeployment referenced from the enclosing ProvidedSomeipServiceInstance in the role serviceInterfaceDeployment
- each SomeipEventGroup defined in the scope of the SomeipServiceInterfaceDeployment referenced from the enclosing ProvidedSomeipServiceInstance in the role serviceInterfaceDeployment shall be referenced from exactly one SomeipProvidedEventGroup aggregated in the role providedEventGroup in the scope of the enclosing ProvidedSomeipServiceInstance.

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

In other words each ProvidedSomeipServiceInstance shall offer all SomeipEventGroups for subscription that are defined in the SomeipServiceInterfaceDeployment and there shall not be two or more SomeipProvidedEventGroups in the ProvidedSomeipServiceInstance that are referencing to the same SomeipEventGroup in the SomeipServiceInterfaceDeployment.

[constr_5339]{DRAFT} **SomeipEventGroups of a SomeipServiceInterfaceDeployment shall be referenced at most once from a RequiredSomeipServiceInstance that instantiates the SomeipServiceInterfaceDeployment** ⌈Each SomeipEventGroup that is defined in a SomeipServiceInterfaceDeployment shall be referenced at most once from a SomeipRequiredEventGroup that is aggregated by the RequiredSomeipServiceInstance that is referencing the SomeipServiceInterfaceDeployment in the role serviceInterfaceDeployment **at the time when the creation of the manifest is finished**.⌋*()*

In other words there shall not be two or more SomeipRequiredEventGroups in the RequiredSomeipServiceInstance that are referencing to the same SomeipEventGroup in the SomeipServiceInterfaceDeployment.

[TPS_MANI_03168]{DRAFT} **Configuration of the SOME/IP load balancing option** ⌈The SOME/IP load balancing option is configurable per ProvidedSomeipServiceInstance with the two attributes loadBalancingPriority and loadBalancingWeight.⌋*(RS_MANI_00024)*

The SOME/IP load balancing option is used to prioritize different ProvidedSomeipServiceInstances that point to the same SomeipServiceInterfaceDeployment, so that a client chooses the service instance based on these settings. This option is attached to SOME/IP Offer Service entries.

[constr_3415]{DRAFT} **Value range of ProvidedSomeipServiceInstance.loadBalancingPriority** ⌈The value of attribute ProvidedSomeipServiceInstance.loadBalancingPriority shall be in the range of 0..65535 **at the time when the creation of the manifest is finished**.⌋*()*

Please note that according to SOME/IP a lower value means higher priority.

[constr_3416]{DRAFT} **Value range of ProvidedSomeipServiceInstance.loadBalancingWeight** ⌈The value of attribute ProvidedSomeipServiceInstance.loadBalancingWeight shall be in the range of 0..65535 **at the time when the creation of the manifest is finished**.⌋*()*

Please note that according to SOME/IP a higher value means higher probability to be chosen.

[constr_1723]{DRAFT} **ProvidedSomeipServiceInstance shall be unique in respect of serviceInstanceId, serviceInterfaceId and majorVersion on a VLAN** ⌈On a VLAN, each ProvidedSomeipServiceInstance shall have a different serviceInstanceId, serviceInterfaceId and majorVersion value combination.

In other words, no two ProvidedSomeipServiceInstances shall have the same serviceInstanceId, serviceInterfaceId and majorVersion value combination during runtime on the same VLAN.

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

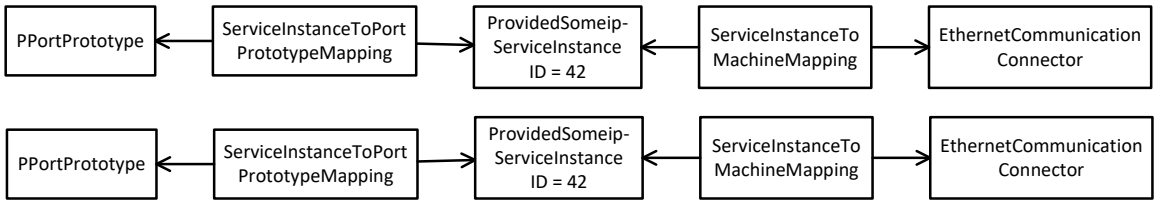


Figure 11.12: Scenario in which two ProvidedSomeipServiceInstances with the same credentials are provided on two VLANs

Figure 11.12 shows that different SOME/IP ServiceInstances with the same serviceInstanceId, serviceInterfaceId and majorVersion are provided on different VLANs. This is a valid setup according to [constr_1723].

In the following example where the same ProvidedSomeipServiceInstance is mapped to different PPortPrototypes and is provided on different VLANs the specification item [TPS_MANI_03236] applies. This means that only one of the PPortPrototypes is active at runtime at the same time and offers the ServiceInstance on two different VLANs.

Figure 11.13: Scenario in which a ProvidedSomeipServiceInstance is provided on two VLANs and is mapped to different PPortPrototypes from which latest at runtime only one is operational

[TPS_MANI_03236]{DRAFT} **Mapping of ProvidedSomeipServiceInstance to different PPortPrototypes** ⌈In case that the same ProvidedSomeipServiceInstance is mapped by several ServiceInstanceToPortPrototypeMappings to different PPortPrototypes it shall be ensured (latest at runtime) that only one of these mapped PPortPrototypes is actually operational at any given point in time.⌋ *(RS_MANI_00009)*

Please note that two ProvidedSomeipServiceInstance elements with the same credentials according to [constr_1723] may exist that both are mapped to different PPortPrototypes and the ProvidedSomeipServiceInstances are mapped by ServiceInstanceToMachineMappings to the same VLAN. At runtime only one of these PPortPrototypes shall be operational at any given point in time.

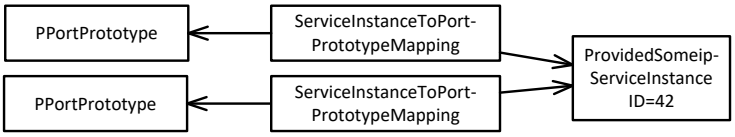


Figure 11.14: Static setup in which a ProvidedSomeipServiceInstance is mapped to two PPortPrototypes from which latest at runtime only one is operational

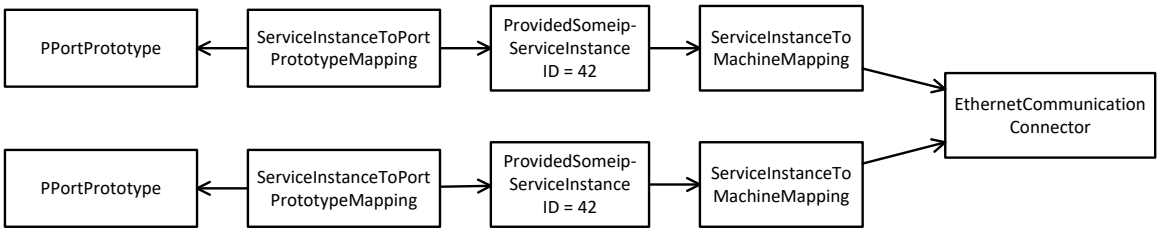


Figure 11.15: Static setup in which `ProvidedSomeipServiceInstance`s with the same credentials provided on the same VLAN are mapped to two different `PPortPrototype`s from which latest at runtime only one is operational

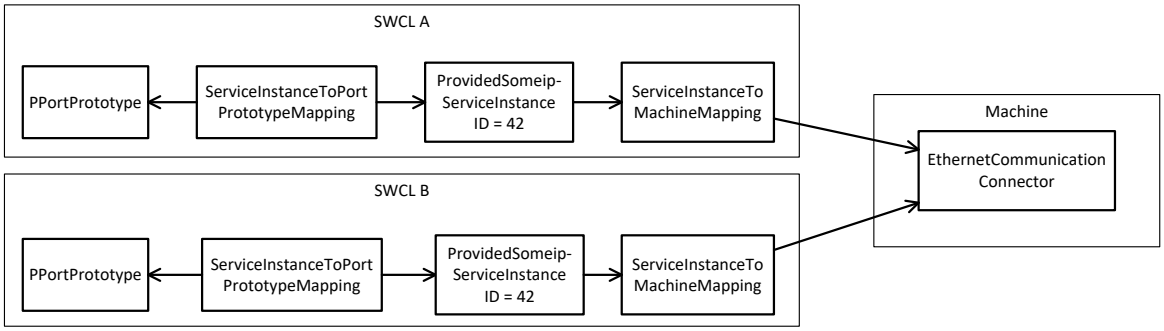


Figure 11.16: `SoftwareCluster` scenario in which `ProvidedSomeipServiceInstance`s with the same credentials provided on the same VLAN exist

Such a scenario may be created by the installation of two separate `SoftwareCluster`s as shown in the following figure. It is not possible to check such a setup since the `SoftwareCluster`s may be developed, integrated, and deployed independent from each other.

11.3.1.1.1 IP Configuration

In SOME/IP the Offer service entry references IPv4 or IPv6 Endpoint options to indicate to the client where the server accepts the method calls and where the server sends the event messages.

Such an Endpoint contains the IP address of the sender. The IP address configuration is described in this chapter.

[TPS_MANI_03002]{DRAFT} **IP configuration for a `ProvidedSomeipServiceInstance`** ⌈A `ProvidedSomeipServiceInstance` can be mapped to a `CommunicationConnector` of a `MachineDesign` with the `SomeipServiceInstanceToMachineMapping`.

With this mapping an assignment of the `ProvidedSomeipServiceInstance` to a unicast IP Address is established since the `EthernetCommunicationConnector`

refers to a NetworkEndpoint in the role unicastNetworkEndpoint.⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03003]{DRAFT} **ProvidedSomeipServiceInstance Fanout** ⌈It is allowed to map the same ProvidedSomeipServiceInstance to different CommunicationConnectors of a MachineDesign. In such a case, several SomeipServiceInstanceToMachineMappings shall be defined.

This allows for offering the same ProvidedSomeipServiceInstance on different VLANs or even on different CommunicationClusters.⌋*(RS_MANI_00009, RS_MANI_00024)*

[constr_3538]{DRAFT} **Only one ServiceInstanceToMachineMapping per technology and CommunicationConnector** ⌈Each AdaptivePlatformServiceInstance shall only be referenced up to once by a specific ServiceInstanceToMachineMapping subclass in the role serviceInstance where the ServiceInstanceToMachineMapping refer to the same CommunicationConnector **at the time when the creation of the manifest is finished**.⌋*()*

In other words, it is not allowed to define for the same service instance two ServiceInstanceToMachineMapping of the same kind (e.g. SomeipServiceInstanceToMachineMapping) which refer to the same CommunicationConnector.

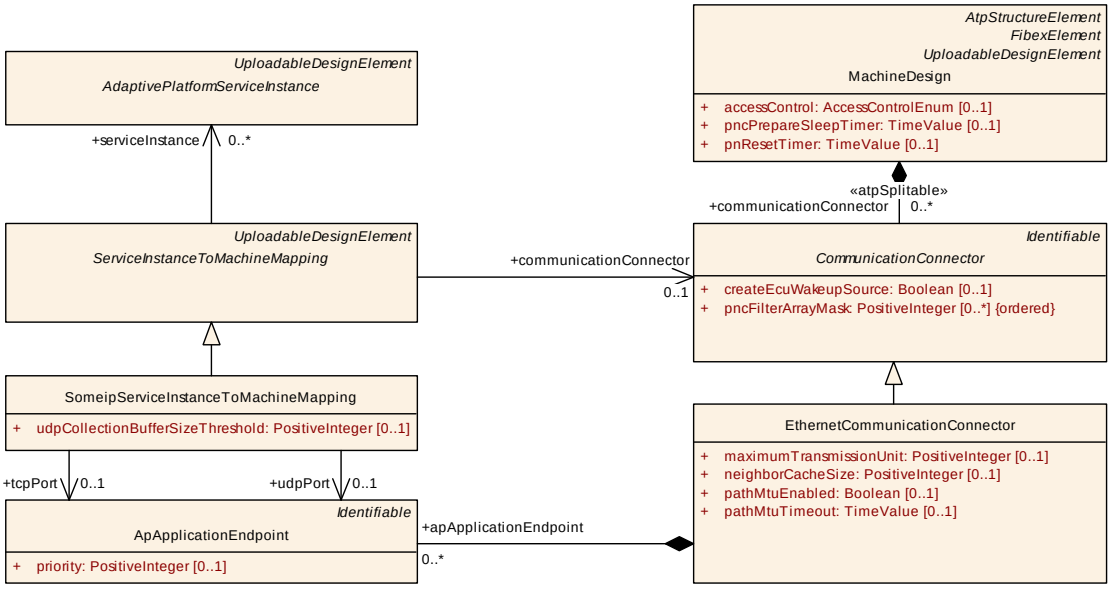


Figure 11.17: SomeipServiceInstanceToMachineMapping with TP and IP configuration

[TPS_MANI_03554]{DRAFT} **Several SomeipServiceInstanceToMachineMappings with equal settings** ⌈If

- one SomeipServiceInstanceToMachineMapping refers to several service instances in the role serviceInstance
- several SomeipServiceInstanceToMachineMappings with equal settings refer to several service instances in the role serviceInstance

- the combination of the two above applies

then for all the referenced service instances the same network connection (i.e. Ethernet socket) will be used.⌋*(RS_MANI_00009, RS_MANI_00024)*

[constr_5052]{DRAFT} **ProvidedSomeipServiceInstances of the same serviceInterface on one Machine** ⌈ProvidedSomeipServiceInstances that are referring to the same SomeipServiceInterfaceDeployment element or to several SomeipServiceInterfaceDeployments that all have the same SOME/IP credentials (SomeipServiceInterfaceDeployment.serviceInterfaceId and SomeipServiceInterfaceDeployment.serviceInterfaceVersion.majorVersion) shall not be mapped to the same combination of:

- IP address that is assigned by the SomeipServiceInstanceToMachineMapping with the reference to the EthernetCommunicationConnector that in turn references the NetworkEndpoint and
- UDP Port or TCP Port number that are defined by the SomeipServiceInstanceToMachineMapping.udpPort and SomeipServiceInstanceToMachineMapping.tcpPort references to the ApApplicationEndpoint.

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

The reason for this restriction is that the Instance IDs are only used for Service Discovery but are not contained in the SOME/IP header. So if for example two ProvidedSomeipServiceInstances of the same ServiceInterface are provided on the same machine and a client wants to call a method of one of these ProvidedSomeipServiceInstances the only possibility for the client to distinguish the ProvidedSomeipServiceInstances is the port number over which the individual ProvidedSomeipServiceInstances are provided.

[TPS_MANI_03555]{DRAFT} **Mix of SomeipServiceInstanceToMachineMapping and signal-based communication** ⌈SomeipServiceInstanceToMachineMapping defines service instance communication on a specific Ethernet socket and the same socket may also be used for signal-based communication at the same time.⌋*(RS_MANI_00009, RS_MANI_00024)*

Please note that the signal-based communication is described in section 13.

Via the definition of respective ISignalTriggering, PduTriggering, and SocketConnection for signal-based communication, the same values for Ethernet address and port may be defined as used at the SomeipServiceInstanceToMachineMapping.

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	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	*AbstractCanCluster*, EthernetCluster, FlexrayCluster, LinCluster, UserDefinedCluster			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
baudrate	PositiveUnlimitedInteger	0..1	attr	Channels speed in bits/s.
physical Channel	PhysicalChannel	*	aggr	This relationship defines which channel element belongs to which cluster. A channel shall be assigned to exactly one cluster, whereas a cluster may have one or more channels. Note: This atpSplitable property has no atp.Splitkey due to atpVariation (PropertySetPattern). **Stereotypes:** atpSplitable; atpVariation **Tags:** 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 11.34: 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*			
Subclasses	*AbstractCanCommunicationConnector*, EthernetCommunicationConnector, FlexrayCommunication Connector, UserDefinedCommunicationConnector			
Aggregated by	EcuInstance.connector, MachineDesign.communicationConnector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
createEcu WakeupSource	Boolean	0..1	attr	If this parameter is available and set to true then a channel wakeup source shall be created for the Physical Channel referencing this CommunicationConnector.
pncFilterArray Mask (ordered)	PositiveInteger	*	attr	Bit mask for NM-Pdu Payload used to configure the NM filter mask for the Network Management.

Table 11.35: CommunicationConnector

Class	**EthernetCommunicationConnector**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Ethernet specific attributes to the CommunicationConnector.			
Base	*ARObject*, *CommunicationConnector*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcuInstance.connector, MachineDesign.communicationConnector			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
apApplication Endpoint	ApApplicationEndpoint	*	aggr	Collection of Application Addresses that are used on the CommunicationConnector.
canXlProps	CanXlProps	*	ref	If the Ethernet frames handled by this Ethernet CommunicationConnector are tunneled through CAN XL, then this reference shall refer the CanXlProps which contains the specific configuration parameters of the CAN XL controller of the physical CAN XL connection to be used for tunneling.
maximum Transmission Unit	PositiveInteger	0..1	attr	This attribute specifies the maximum transmission unit in bytes.
neighborCache Size	PositiveInteger	0..1	attr	This attribute specifies the size of neighbor cache or ARP table in units of entries.
pathMtu Enabled	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.
unicastNetwork Endpoint	NetworkEndpoint	*	ref	Network Endpoint that defines the IPAddress of the machine.

Table 11.36: EthernetCommunicationConnector

[constr_3721]{DRAFT} **Upper multiplicity of reference in the role `EthernetCommunicationConnector.unicastNetworkEndpoint`** ⌈In the context of `EthernetCommunicationConnector`, the reference in the role `unicastNetworkEndpoint` shall exist at most once **`at the time when the system design is complete`**.⌋*()*

[constr_3722]{DRAFT} **Upper multiplicity of reference in the role `EthernetCommunicationConnector.canXlProps`** ⌈In the context of `EthernetCommunicationConnector`, the reference in the role `canXlProps` shall exist at most once **`at the time when the system design is complete`**.⌋*()*

[constr_3288]{DRAFT} **IP configuration restriction for `unicastNetworkEndpoints`** ⌈A `NetworkEndpoint` that is referenced by a `EthernetCommunicationConnector` in the role `unicastNetworkEndpoint` shall have either

- one `Ipv4Configuration` or
- one `Ipv6Configuration`

as `networkEndpointAddress` that is defined in the unicast IP range according to the rules defined in [TPS_MANI_03005] and [TPS_MANI_03006] **`at the time when the creation of the manifest is finished`**.⌋*()*

In SOME/IP, a server that offers a `ProvidedSomeipServiceInstance` is able to send events and notification events to an IP-Multicast address.

As the server needs to inform the client to which Multicast IP address the event messages will be sent, the *Subscribe Eventgroup Acknowledgement Entry* contains a reference to an IPv4 Multicast Option and/or an IPv6 Multicast Option.

[TPS_MANI_03004]{DRAFT} **IPv4 Multicast event destination address** ⌈Meta-class `SomeipProvidedEventGroup` defines the multicast IPv4 address to which the `event`s and notification events of the `SomeipProvidedEventGroup` are sent to with the attribute `ipv4MulticastIpAddress`.⌋*(RS_MANI_00009, RS_MANI_00024)*

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

[TPS_MANI_03005]{DRAFT} **IPv4 Multicast address range** ⌈The IPv4 addresses reserved for multicast communication are in the range 224.0.0.0 through 239.255.255.255. Addresses between 0.0.0.0 and 223.255.255.255 are reserved for unicast communication.⌋*(RS_MANI_00009, RS_MANI_00024)*

Please note that the endpoint 224.0.0.251:5353 is reserved by IANA [27] [28] for the utilization of mDNS (only relevant if SOVD is used).

[TPS_MANI_03006]{DRAFT} **IPv6 Multicast address range** ⌈IPv6 multicast addresses are distinguished from unicast addresses by the value of the high-order octet of the addresses: a value of 0xFF (binary 11111111) identifies an address as an address reserved for multicast communication; any other value identifies an address as a unicast address.⌋*(RS_MANI_00009, RS_MANI_00024)*

Please note that the endpoint [FF02::FB]:5353 is reserved by IANA [29] [28] for the utilization of mDNS (only relevant if SOVD is used).

Class	***NetworkEndpointAddress*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	To build a valid network endpoint address there has to be either one MAC multicast group reference or an ipv4 configuration or an ipv6 configuration.			
Base	*ARObject*			
Subclasses	Ipv4Configuration, Ipv6Configuration, MacMulticastConfiguration			
Aggregated by	NetworkEndpoint.networkEndpointAddress			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 11.37: NetworkEndpointAddress

11.3.1.1.2 TP Configuration

The IPv4 or IPv6 Endpoint option that is referenced in the SOME/IP Offer message contains besides the IP address the transport layer protocol (e.g. UDP or TCP), and the port number of the sender.

With the SomeipServiceInstanceToMachineMapping the Transport Layer configuration attributes are assigned to the ProvidedSomeipServiceInstance.

The same element contains the Transport Layer configuration attributes for the IPv4/IPv6 Multicast Option that may be used in the SOME/IP SubscribeEventGroupAck message.

[TPS_MANI_03007]{DRAFT} **Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance** ⌈The reference SomeipServiceInstanceToMachineMapping.udpPort to the ApApplicationEndpoint defines the Transport Protocol for a UDP communication.

This setting is used in an IPv4 or IPv6 Endpoint Option that is referenced by an OfferService entry.⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03008]{DRAFT} **Tcp Transport Protocol Configuration for ProvidedSomeipServiceInstance** ⌈The reference SomeipServiceInstanceToMachineMapping.tcpPort to the ApApplicationEndpoint defines the Transport Protocol for a TCP communication.

This setting is used in an IPv4 or IPv6 Endpoint Option that is referenced by an OfferService entry.⌋*(RS_MANI_00009, RS_MANI_00024)*

[TPS_MANI_03009]{DRAFT} **Tcp and Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance** ⌈It is allowed to set tcpPort and udpPort in the same SomeipServiceInstanceToMachineMapping.

Such a setting shall be used to indicate that one UDP endpoint and one TCP endpoint are referenced in the OfferService entry. It means that the Server provides the ProvidedSomeipServiceInstance over both Transport Protocols.⌋*(RS_MANI_00009, RS_MANI_00024)*

If a Tcp and Udp Transport Protocol Configuration is defined for a ProvidedSomeipServiceInstance as described in [TPS_MANI_03009] then the SOME/IP ServiceInterfaceDeployment settings decide which content of the ProvidedSomeipServiceInstance is transported over udp and which content is transported over tcp.

This is described in [TPS_MANI_03050] and [TPS_MANI_03051].

[TPS_MANI_03010]{DRAFT} **Udp Transport Protocol Configuration in case of IP-Multicast** ⌈The SomeipProvidedEventGroup.eventMulticastUdpPort defines the Transport Protocol Port Number for a UDP event communication in case IP-Multicast is used.

This setting is used in an IPv4 or IPv6 Multicast Option that is referenced by a SubscribeEventGroupAck Service entry.⌋*(RS_MANI_00009, RS_MANI_00024)*

[constr_3290]{DRAFT} **Transport Protocol attributes defined for a ProvidedSomeipServiceInstance** ⌈Each SomeipServiceInstanceToMachineMapping that is defined for a ProvidedSomeipServiceInstance shall define either

- a udpPort or
- a tcpPort or
- a udpPort and a tcpPort.

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

[TPS_MANI_03157]{DRAFT} **Enabling of data collection for UDP data transmission** ⌈The setting of the attribute SomeipServiceInstanceToMachineMapping.udpCollectionBufferSizeThreshold to a value enables the data collection for data transmission over the udpPort and unicastNetworkEndpoint defined on the EthernetCommunicationConnector that is referenced by the SomeipServiceInstanceToMachineMapping. In this case all event and method messages that are configured for data collection will be collected in the buffer until a transmission trigger arrives and the data transmission starts.⌋*(RS_MANI_00024)*

For configuration of transmission triggers please see [TPS_MANI_03158] and [TPS_MANI_03159].

11.3.1.1.3 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.

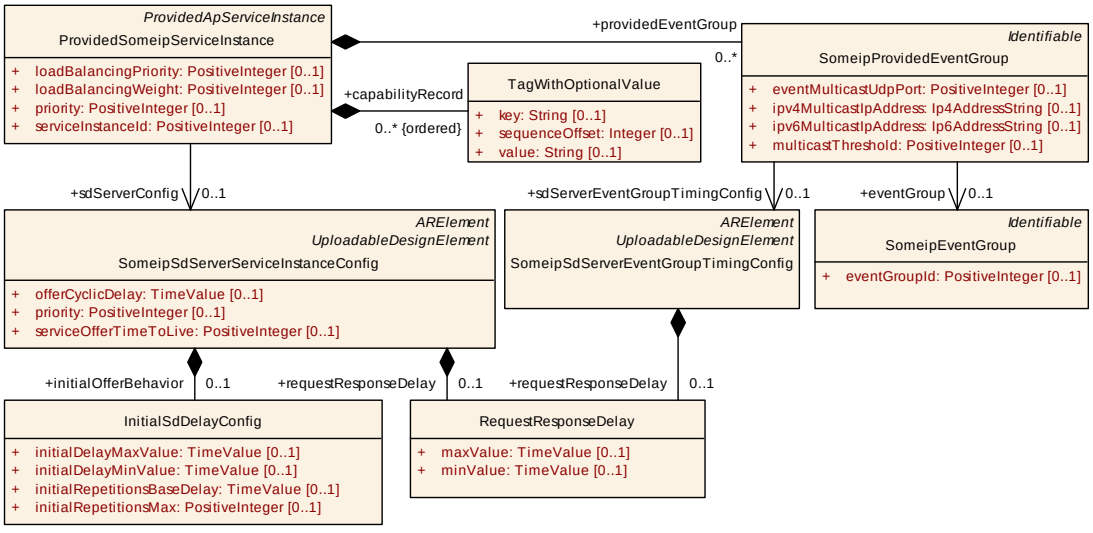


Figure 11.18: SOME/IP Service Discovery Server configuration settings

For every ProvidedSomeipServiceInstance on a Server different phases are existing:

- Down
- Available

- Initial Wait Phase
- Repetition Phase
- Main Phase

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

The number of `SomeipSdServerServiceInstanceConfig` elements determine how many timers shall actually be used by the middleware to keep the randomized times. Via the reference `ProvidedSomeipServiceInstance.sdServerConfig` each `ProvidedSomeipServiceInstance` defines to which timer it is assigned.

If several `ProvidedSomeipServiceInstance`s share the same timer then the expiration of that timer will lead a combined sending of service discovery messages.

Note that it is possible to define several `SomeipSdServerServiceInstanceConfig` elements with identical timing specification values in order to request several timer handling in the middleware.

[TPS_MANI_03230]{DRAFT} **Sharing timers for `ProvidedSomeipServiceInstance`** ⌈If several `ProvidedSomeipServiceInstance`s point to the same `SomeipSdServerServiceInstanceConfig` in the role `sdServerConfig` then all of these `ProvidedSomeipServiceInstance`s will share the same timers for their timing behavior. This will lead to combining several service discovery entries in one service discovery message.⌋*(RS_MANI_00024)*

Class	**SomeipSdServerServiceInstanceConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
Note	Server specific settings that are relevant for the configuration of SOME/IP Service-Discovery. **Tags:** atp.recommendedPackage=SomeipSdTimingConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initialOffer Behavior	InitialSdDelayConfig	0..1	aggr	Controls offer behavior of the server.
offerCyclicDelay	TimeValue	0..1	attr	Optional attribute to define cyclic offers. Cyclic offer is active, if the delay is set (in seconds) and greater then 0.
priority	PositiveInteger	0..1	attr	This attribute defines the VLAN frame priority for Service Discovery messages that result from ProvidedSomeip ServiceInstances that are referencing the SomeipSd ServerServiceInstanceConfig (OfferService, StopOffer Service, SubscribeEventGroupAck). Values from 0 (best effort) to 7 (highest) are allowed.

▽

△

Class	SomeipSdServerServiceInstanceConfig			
request ResponseDelay	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).
serviceOffer TimeToLive	PositiveInteger	0..1	attr	Defines the time in seconds the service offer is valid.

Table 11.38: SomeipSdServerServiceInstanceConfig

[TPS_MANI_03012]{DRAFT} **Initial Wait Phase configuration for a `ProvidedSomeipServiceInstance`** ⌈The Initial Wait Phase for a `ProvidedSomeipServiceInstance` is configured with the `initialOfferBehavior` and the two attributes `initialDelayMinValue` and `initialDelayMaxValue`.

When a calculated random timer based on these min and max values expires the first `OfferService` entry will be sent out.⌋*(RS_MANI_00024)*

When the calculated random timer expires, the Repetition Phase will be entered.

[TPS_MANI_03013]{DRAFT} **Repetition Phase configuration for a `ProvidedSomeipServiceInstance`** ⌈The Repetition Phase for a `ProvidedSomeipServiceInstance` is configured with the `initialOfferBehavior` and the two attributes `initialRepetitionsMax` and `initialRepetitionsBaseDelay`.⌋*(RS_MANI_00024)*

If the Repetition Phase is entered the Service Discovery waits for the `initialRepetitionsBaseDelay` and then sends an `OfferService` entry. If the amount of sent `OfferService` entries reaches `initialRepetitionsMax`, the Main Phase will be entered.

[TPS_MANI_03014]{DRAFT} **Main Phase configuration for a `ProvidedSomeipServiceInstance`** ⌈The Main Phase for a `ProvidedSomeipServiceInstance` is configured with the `offerCyclicDelay` attribute of `SomeipSdServerServiceInstanceConfig`.

The `OfferService` entry will be sent cyclically with an interval that is defined by the value of attribute `offerCyclicDelay`, if the value is greater than 0.⌋*(RS_MANI_00024)*

Class	InitialSdDelayConfig			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
Note	This element is used to configure the offer behavior of the server and the find behavior on the client.			
Base	*ARObject*			
Aggregated by	SdClientConfig.initialFindBehavior, SdServerConfig.initialOfferBehavior, SomeipSdClientServiceInstanceConfig.initialFindBehavior, SomeipSdServerServiceInstanceConfig.initialOfferBehavior			
Attribute	Type	Mult.	Kind	Note

▽

△

Class	**InitialSdDelayConfig**			
initialDelayMax Value	TimeValue	0..1	attr	Max Value in seconds to delay randomly the first offer (if aggregated in role initialOfferBehavior by SomeipSd ServerServiceInstanceConfig) or the transmission of a find message (if aggregated in role initialFindBehavior by SomeipSdClientServiceInstanceConfig).
initialDelayMin Value	TimeValue	0..1	attr	Min Value in seconds to delay randomly the first offer (if aggregated in role initialOfferBehavior by SomeipSd ServerServiceInstanceConfig) or the transmission of a find message (if aggregated in role initialFindBehavior by SomeipSdClientServiceInstanceConfig).
initial Repetitions BaseDelay	TimeValue	0..1	attr	The base delay for offer repetitions (if aggregated in role initialOfferBehavior by SomeipSdServerServiceInstance Config) or find repetitions (if aggregated in role initialFind Behavior by SomeipSdClientServiceInstanceConfig). Successive find messages have an exponential back off delay.
initial RepetitionsMax	PositiveInteger	0..1	attr	Describes the maximum amount of offer repetitions (if aggregated in role initialOfferBehavior by SomeipSd ServerServiceInstanceConfig) or the maximum amount of find repetitions (if aggregated in role initialFindBehavior by SomeipSdClientServiceInstanceConfig).

Table 11.39: InitialSdDelayConfig

[TPS_MANI_03015]{DRAFT} **TTL for Offer Service Entries** ⌈The lifetime of a `ProvidedSomeipServiceInstance` is configurable with the `serviceOfferTimeToLive` attribute of `SomeipSdServerServiceInstanceConfig`.

If the time that is configured by `serviceOfferTimeToLive` expires, the `ProvidedSomeipServiceInstance` is no longer offered.

If the `offerCyclicDelay` attribute of `SomeipSdServerServiceInstanceConfig` is not configured, or the configured value of `offerCyclicDelay` is 0, the configured value of `serviceOfferTimeToLive` shall be ignored, and the infinite value (0xFFFF) shall be used instead.⌋ *(RS_MANI_00024)*

Please note that the *System Template* [18] defines [constr_5382] and [constr_5383] that are also valid for `SomeipSdServerServiceInstanceConfig`s that are referenced by a `ProvidedSomeipServiceInstance`.

[TPS_MANI_03016]{DRAFT} **Servers `RequestResponseDelay` for received `FindService` entries** ⌈The Server will delay the `OfferService` answer to a received multicast `FindService` entry by the configured `SomeipSdServerServiceInstanceConfig.requestResponseDelay`.

The actual delay will be randomly chosen between the `maxValue` and `minValue`.⌋ *(RS_MANI_00024)*

Class	**RequestResponseDelay**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
Note	Time to wait before answering the query.			
Base	*ARObject*			
Aggregated by	SdClientConfig.requestResponseDelay, SdServerConfig.requestResponseDelay, SomeipSdClientEventGroupTimingConfig.requestResponseDelay, SomeipSdServerEventGroupTimingConfig.requestResponseDelay, SomeipSdServerServiceInstanceConfig.requestResponseDelay			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
maxValue	TimeValue	0..1	attr	Maximum allowable response delay to entries received by multicast in seconds.
minValue	TimeValue	0..1	attr	Minimum allowable response delay to entries received by multicast in seconds.

Table 11.40: RequestResponseDelay

Figure 11.19 shows an example of the different SOME/IP phases on the Server side.

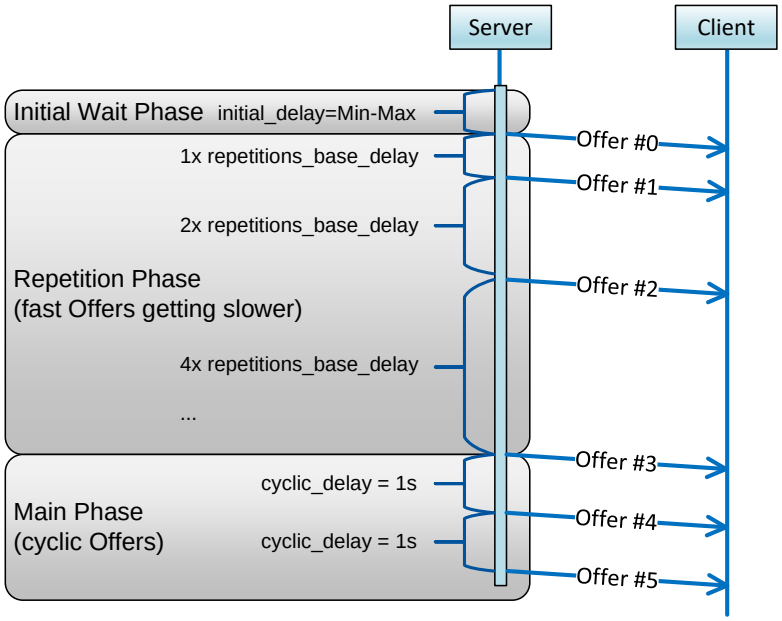


Figure 11.19: SOME/IP Server Timing example

SOME/IP allows for the specification of additional information about the `ProvidedSomeipServiceInstance` with the Capability Record that allows to transport arbitrary configuration strings (key/value pairs). This allows to encode additional information like the name of a service or its configuration.

[TPS_MANI_03017]{DRAFT} **Server Capability Records** ⌈A Capability Record (key/-value pair) on the Server side is configurable with the `capabilityRecord` and the two attributes `key` and `value`.⌋ *(RS_MANI_00024)*

Class	TagWithOptionalValue			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::TagWithOptionalValue			
Note	A tagged value is a combination of a tag (key) and a value that gives supplementary information that is attached to a model element. Please note that keys without a value are allowed.			
Base	*ARObject*			
Aggregated by	*AbstractServiceInstance*.capabilityRecord, Machine.environmentVariable, ProvidedSomeipService Instance.capabilityRecord, RequiredSomeipServiceInstance.capabilityRecord, SdClientConfig.capability Record, SdServerConfig.capabilityRecord, StartupConfig.environmentVariable			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
key	String	0..1	attr	Defines a key.
sequenceOffset	Integer	0..1	attr	The sequenceOffset attribute supports the use case where TagWithOptionalValue is aggregated as splitable. If multiple aggregations define the same value of attribute key then the order in which the value collection is merged might be significant. As an example consider the modeling of the $PATH environment variable by means of a meta class TagWithOptionalValue. The sequenceOffset describes the relative position of each contribution in the concatenated value. The contributions are sorted in increasing integer order.
value	String	0..1	attr	Defines the corresponding value.

Table 11.41: TagWithOptionalValue

[constr_10037]{DRAFT} **Existence of attribute `TagWithOptionalValue.sequenceOffset` in the context of attribute `capabilityRecord` owned by `ProvidedSomeipServiceInstance`, `RequiredSomeipServiceInstance`, `SdServerConfig`, `SdClientConfig`, or `AbstractServiceInstance`** ⌈For all `capabilityRecord` modeled in the context of `ProvidedSomeipServiceInstance`, `RequiredSomeipServiceInstance`, `SdServerConfig`, `SdClientConfig`, or `AbstractServiceInstance`, attribute `TagWithOptionalValue.sequenceOffset` shall not exist **`at the time when the creation of the manifest is finished`**.⌋*()*

11.3.1.1.4 Provided Event Group

The `ProvidedSomeipServiceInstance` aggregates a `SomeipProvidedEventGroup` in the role `providedEventGroup` that allows to define service instance specific configuration settings for a `SomeipEventGroup`.

Class	SomeipProvidedEventGroup			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	The meta-class represents the ability to configure ServiceInstance related communication settings on the provided side for each EventGroup separately.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ProvidedSomeipServiceInstance.providedEventGroup			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**SomeipProvidedEventGroup**			
eventGroup	SomeipEventGroup	0..1	ref	Reference to the SomeipEventGroup in the System Manifest for which the ServiceInstance related Event Group settings are valid.
eventMulticast UdpPort	PositiveInteger	0..1	attr	UdpPort configuration that is used for Event communication in the IP-Multicast case. During SOME/IP Service Discovery: Send in the SD-SubscribeEventGroupAck Message to client (answer to SD-SubscribeEventGroup). Event: This is the destination-port where the server sends the multicast event messages if the multicastThreshold is exceeded.
ipv4MulticastIp Address	Ip4AddressString	0..1	attr	Multicast IPv4 Address that is transmitted in the Event GroupSubscribeAck message.
ipv6MulticastIp Address	Ip6AddressString	0..1	attr	Multicast IPv6 Address that is transmitted in the Event GroupSubscribeAck message.
multicast Threshold	PositiveInteger	0..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 served with unicast and as soon as the 2nd client arrives both will be served by multicast. This does not influence the handling of initial events, which are served using unicast only.
sdServerEvent GroupTiming Config	SomeipSdServerEvent GroupTimingConfig	0..1	ref	Server Timing configuration settings that are EventGroup specific.

Table 11.42: SomeipProvidedEventGroup

[constr_10220]{DRAFT} **Multiplicity of attribute `SomeipProvidedEventGroup.multicastThreshold`** ⌈For each `SomeipProvidedEventGroup`, the attribute `multicastThreshold` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_03018]{DRAFT} **Usage of `SomeipProvidedEventGroup.multicastThreshold`** ⌈The switching between IP-Unicast and IP-Multicast is guided by the server with the `SomeipProvidedEventGroup.multicastThreshold` attribute and by the number of subscribed clients to the `SomeipProvidedEventGroup`.

The Server will change the transmission of events to Multicast if the `multicastThreshold` of the corresponding `SomeipProvidedEventGroup` is reached by the number of subscribed clients. If the number of subscribed clients is smaller than the configured `multicastThreshold`, the transmission of events takes place via unicast communication.⌋*(RS_MANI_00024)*

The following example shows the effect of the `multicastThreshold` in relation to the number of subscribed clients to the transmission of the SOME/IP event to the unicast or multicast destination address:

- If `multicastThreshold` is configured to 0, only the unicast IP address and the port will be used as destination address.

- If multicastThreshold is configured to 1, the first client will be served by multicast.
- If multicastThreshold is configured to 2, the first client will be served with unicast and as soon as the second client arrives both will be served by multicast, etc.

[constr_5332]{DRAFT} **Mandatory multicast endpoint in case of multicastThreshold different from 0** ⌈If SomeipProvidedEventGroup.multicastThreshold is configured to a value different from 0, then

- SomeipProvidedEventGroup.eventMulticastUdpPort

and either

- SomeipProvidedEventGroup.ipv4MulticastIpAddress or
- SomeipProvidedEventGroup.ipv6MulticastIpAddress

shall exist **at the time when the creation of the manifest is finished**.⌋()

[constr_5333]{DRAFT} **No multicast in case of TCP** ⌈If a SomeipProvidedEventGroup references only SomeipEventDeployments that have the attribute transportProtocol set to tcp (via SomeipProvidedEventGroup.eventGroup.event) then this SomeipProvidedEventGroup shall not have a SomeipProvidedEventGroup.multicastThreshold attribute or shall have the SomeipProvidedEventGroup.multicastThreshold set to 0 **at the time when the creation of the manifest is finished**.⌋()

[constr_5392] Assignment of the same event to several SomeipEventGroups is forbidden in case one of the SomeipEventGroups has the multicastThreshold set to to a value greater than 1 ⌈The same SomeipEventDeployment (Event) shall not be referenced by several SomeipEventGroups if these SomeipEventGroups

- are referenced by different SomeipProvidedEventGroups in the context of one Machine and
- one or more of these SomeipProvidedEventGroups has the multicastThreshold set to a value>1

⌋()

[constr_5392] is introduced to forbid scenarios in which the Service Provider is forced to transmit the same event over IP Unicast and IP Multicast in cases where the same event is assigned to several SomeipProvidedEventGroups and the multicastThreshold is reached for one SomeipProvidedEventGroup but not for the other ones.

[TPS_MANI_03020]{DRAFT} **Servers RequestResponseDelay for received SubscribeEventGroup entries** ⌈The Server will delay the SubscribeEventGroupAck

answer to a received SubscribeEventGroup message that was triggered by a multicast ServiceOffer by the configured SomeipSdClientEventGroupTimingConfig.requestResponseDelay.

The actual delay will be randomly chosen between the maxValue and minValue.⌋ *(RS_MANI_00024)*

Class	**SomeipSdServerEventGroupTimingConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
Note	EventGroup specific timing configuration settings. **Tags:** atp.recommendedPackage=SomeipSdTimingConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
request ResponseDelay	RequestResponseDelay	0..1	aggr	The Service Discovery shall delay answers to unicast messages triggered by multicast messages (e.g. Subscribe Eventgroup after Offer Service).

Table 11.43: SomeipSdServerEventGroupTimingConfig

11.3.1.1.5 ProvidedSomeipServiceInstance related event and method properties

[TPS_MANI_03154]{DRAFT} **ProvidedSomeipServiceInstance related configuration settings for events** ⌈The class SomeipEventProps that is aggregated by the ProvidedSomeipServiceInstance in the role eventProps allows for specifying ProvidedSomeipServiceInstance related configuration settings for events that are defined in the SomeipServiceInterfaceDeployment referenced by the ProvidedSomeipServiceInstance in the role serviceInterfaceDeployment.⌋ *(RS_MANI_00024)*

[TPS_MANI_03155]{DRAFT} **ProvidedSomeipServiceInstance related configuration settings for methods** ⌈The class SomeipMethodProps that is aggregated by the ProvidedSomeipServiceInstance in the role methodResponseProps allows for specifying ProvidedSomeipServiceInstance related configuration settings for a method response message.

The method is defined in the SomeipServiceInterfaceDeployment referenced by the ProvidedSomeipServiceInstance in the role serviceInterfaceDeployment.⌋ *(RS_MANI_00024)*

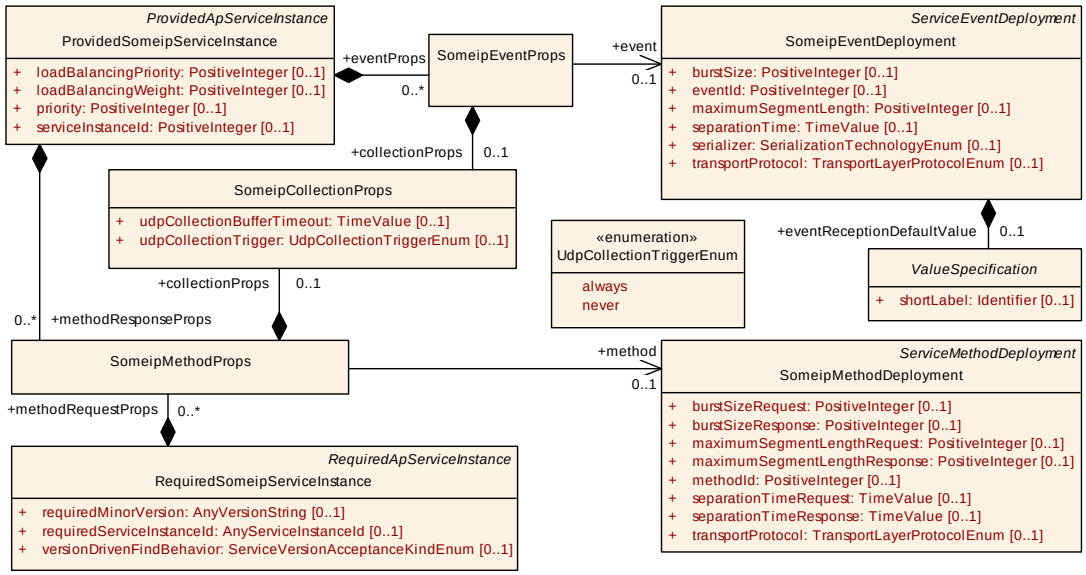


Figure 11.20: `ProvidedSomeipServiceInstance` related event and method properties

Class	**SomeipEventProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class allows to set configuration options for an event in the provided service instance.			
Base	*ARObject*			
Aggregated by	ProvidedSomeipServiceInstance.eventProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
collectionProps	SomeipCollectionProps	0..1	aggr	Collection of timing attributes configurable for an event that is provided by a Service Instance.
event	SomeipEvent Deployment	0..1	ref	Reference to the event for which the SomeipEventProps are applicable.

Table 11.44: SomeipEventProps

Class	**SomeipMethodProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class allows to set configuration options for a method in the service instance.			
Base	*ARObject*			
Aggregated by	ProvidedSomeipServiceInstance.methodResponseProps, RequiredSomeipServiceInstance.method RequestProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
collectionProps	SomeipCollectionProps	0..1	aggr	Collection of timing attributes configurable for a method that is provided or requested by a Service Instance.
method	SomeipMethod Deployment	0..1	ref	Reference to the method for which the SomeipMethod Props are applicable.

Table 11.45: SomeipMethodProps

Class	**SomeipCollectionProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	Collection of attributes that are configurable for an event that is provided by a ServiceInstance or for a method that is provided or requested by a ServiceInstance.			
Base	*ARObject*			
Aggregated by	SomeipEventProps.collectionProps, SomeipMethodProps.collectionProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
udpCollection BufferTimeout	TimeValue	0..1	attr	Maximum time, an outgoing message (event, method call or method response) may be delayed, due to data collection.
udpCollection Trigger	UdpCollectionTrigger Enum	0..1	attr	Defines whether the ServiceInterface element (event or method) contributes to the triggering of the udp data transmission if data collection is enabled.

Table 11.46: SomeipCollectionProps

Enumeration	**UdpCollectionTriggerEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment
Note	Defines whether the ServiceInterface element (event or method) contributes to the triggering of the udp data transmission if data collection is enabled.
Aggregated by	SomeipCollectionProps.udpCollectionTrigger
Literal	***Description***
always	ServiceInterface element will trigger the transmission of the data. **Tags:** atp.EnumerationLiteralIndex=0
never	ServiceInterface element will be buffered and will not trigger the transmission of the data. **Tags:** atp.EnumerationLiteralIndex=1

Table 11.47: UdpCollectionTriggerEnum

[TPS_MANI_03158]{DRAFT} **Configuration of a data collection on a `ProvidedServiceInstance` for transmission over udp** ⌈The attributes `udpCollectionBufferTimeout` and `udpCollectionTrigger` support the configuration of a data collection of several messages for transmission over Udp.

In the `ProvidedServiceInstance` all `method` responses and `event`s for which the `udpCollectionTrigger` is set to `never` will be collected in a buffer until a trigger arrives that starts the data transmission.

The following trigger options are supported:

- a message needs to be transmitted for which the `udpCollectionTrigger` is set to `always`.
- the `udpCollectionBufferTimeout` is reached for a message.
- the buffer size defined by the attribute `udpCollectionBufferSizeThreshold` is reached.

⌋*(RS_MANI_00024)*

11.3.1.2 Required Service Instance

[TPS_MANI_03059]{DRAFT} **RequiredSomeipServiceInstance.requiredServiceInstanceId** ⌈The RequiredSomeipServiceInstance defines the requiredServiceInstanceId of a SomeipServiceInterfaceDeployment that the client searches.

The client may search for a specific requiredServiceInstanceId or for ALL requiredServiceInstanceId of the serviceInterfaceDeployment.⌋*(RS_MANI_00024)*

[constr_10070]{DRAFT} **Value of RequiredSomeipServiceInstance.requiredServiceInstanceId** ⌈For each RequiredSomeipServiceInstance.requiredServiceInstanceId, the value shall be in the range 0..65534 or ALL **at the time when the creation of the manifest is finished**.⌋*()*

Class	**RequiredSomeipServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in a concrete implementation on top of SOME/IP. **Tags:** atp.recommendedPackage=ServiceInstances			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *RequiredApServiceInstance*, *Uploadable DesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blocklisted Version	SomeipServiceVersion	*	aggr	Collection of blocklisted versions.
capability Record (ordered)	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service.
methodRequest Props	SomeipMethodProps	*	aggr	Configuration settings for individual methods that are requested by the ServiceInstance.
requiredEvent Group	SomeipRequiredEvent Group	*	aggr	List of EventGroups that are used by the RequiredService Instance.
requiredMinor Version	AnyVersionString	0..1	attr	This attribute is used to configure for which minor version of the SomeIp ServiceInterface the Service Discovery will search. Value can be set to a number that represents the Minor Version of the searched service or to ANY.
requiredService InstanceId	AnyServiceInstanceId	0..1	attr	This attribute represents the ability to describe the required service instance ID.
sdClientConfig	SomeipSdClientService InstanceConfig	0..1	ref	Client specific configuration settings relevant for the SOME/IP service discovery.
versionDriven FindBehavior	ServiceVersion AcceptanceKindEnum	0..1	attr	Defines the service discovery find behavior.

Table 11.48: RequiredSomeipServiceInstance

[constr_10221]{DRAFT} **Multiplicity of reference in the role RequiredSomeipServiceInstance.sdClientConfig** ⌈For each RequiredSomeipServiceInstance, the reference in the role sdClientConfig shall exist **at the time when the creation of the manifest is finished**.⌋*()*

Class	**SomeipServiceVersion**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
Note	This meta-class represents the ability to describe a version of a SOME/IP Service.			
Base	*ARObject*			
Aggregated by	ConsumedServiceInstance.blocklistedVersion, RequiredSomeipServiceInstance.blocklistedVersion, SomeipServiceInterfaceDeployment.serviceInterfaceVersion			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
majorVersion	PositiveInteger	0..1	attr	Major Version of the ServiceInterface. **Tags:** xml.sequenceOffset=10
minorVersion	PositiveInteger	0..1	attr	Minor Version of the ServiceInterface. **Tags:** xml.sequenceOffset=20

Table 11.49: SomeipServiceVersion

Enumeration	**ServiceVersionAcceptanceKindEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances
Note	Defined the possible acceptance kinds for required service instances.
Aggregated by	ConsumedServiceInstance.versionDrivenFindBehavior, RequiredSomeipServiceInstance.versionDrivenFindBehavior
Literal	***Description***
exactOrAnyMinorVersion	Search for ANY or specific minor version service instance and select either ALL returned service instances (in case of ANY) or exactly the specific minor version service instances defined in requiredMinorVersion. **Tags:** atp.EnumerationLiteralIndex=0
minimumMinorVersion	Search for ANY minor version service instance and select only those service instances which have an equal or greater minor version than given in requiredMinorVersion. **Tags:** atp.EnumerationLiteralIndex=1

Table 11.50: ServiceVersionAcceptanceKindEnum

[TPS_MANI_03021]{DRAFT} **Requirements on the searched minor version from the client's point of view** ⌈The meta-class `RequiredSomeipServiceInstance` is able to make further specifications regarding the version of the service from the client's point of view.

For this purpose, the attribute `RequiredSomeipServiceInstance.requiredMinorVersion` exists and provides the ability to define the required minor version (`SomeipServiceVersion.minorVersion`).⌋*(RS_MANI_00024)*

Please note that the major version that the client searches for is already defined by the `SomeipServiceVersion.majorVersion` in the `SomeipServiceInterfaceDeployment`. It is therefore not possible to search for ANY major version, so the client looks always for a specific major version.

The minor version that may be defined by `SomeipServiceVersion.minorVersion` in the `SomeipServiceInterfaceDeployment` is irrelevant for the client and the service search and shall be ignored.

[TPS_MANI_03619]{DRAFT} **SOME/IP Service search for `requiredMinorVersion`** ⌈A `RequiredSomeipServiceInstance` is searching for a SOME/IP Service Instance `requiredMinorVersion`:

- in case versionDrivenFindBehavior = exactOrAnyMinorVersion: Service minor version that matches the value set in requiredMinorVersion or *ANY* minor version of the Service Instance in case the requiredMinorVersion is set to *ANY*
- in case versionDrivenFindBehavior = minimumMinorVersion: Service minor version that matches at least the value set in requiredMinorVersion or is higher

⌋(*RS_MANI_00009*)

[constr_3561]{DRAFT} **minimumMinorVersion and RequiredSomeipServiceInstance.requiredMinorVersion value** ⌈The RequiredSomeipServiceInstance.requiredMinorVersion shall not have the value *ANY* if versionDrivenFindBehavior = minimumMinorVersion **at the time when the creation of the manifest is finished**.⌋()

[TPS_MANI_03618]{DRAFT} **Usage of RequiredSomeipServiceInstance.blocklistedVersion** ⌈A service connection of a RequiredSomeipServiceInstance to a ProvidedSomeipServiceInstance is not considered for service discovery if the SomeipServiceVersion.minorVersion of the enclosing SomeipServiceInterfaceDeployment that is referenced by the ProvidedSomeipServiceInstance exists in the collection of SomeipServiceVersions aggregated at the RequiredSomeipServiceInstance in the role blocklistedVersion.⌋(*RS_MANI_00066*)

A typical scenario for using a blocklist may be: For a certain RequiredSomeipServiceInstance a certain compatible provider service version inside a system may not work which may have been identified after the design phase.

In order to keep the system running this certain provider version won't be considered in the service search if it has been blocklisted. Therefore, the RequiredSomeipServiceInstance may connect only to ProvidedSomeipServiceInstances that fulfill the search criteria and are not blocklisted.

[constr_3558]{DRAFT} **RequiredSomeipServiceInstance.blocklistedVersion is restricted to the usage of minorVersion** ⌈The majorVersion attribute shall not be used in the SomeipServiceVersion that is aggregated by the RequiredSomeipServiceInstance in the role blocklistedVersion **at the time when the creation of the manifest is finished**.⌋()

[constr_5115]{DRAFT} **Search for a specific SOME/IP ServiceInstance and for all SOME/IP ServiceInstances over the same RPortPrototype** ⌈A RequiredSomeipServiceInstance that configures the search for a specific ServiceInstance on SOME/IP (with concrete requiredServiceInstanceId) and a RequiredSomeipServiceInstance that configures the search for ALL ServiceInstances on SOME/IP (with requiredServiceInstanceId = ALL) that are mapped using ServiceInstanceToMachineMapping to the same EthernetCommunicationConnector (and therefore are searching for SOME/IP ServiceInstances on

the same VLAN) are not allowed to be mapped by ServiceInstanceToPortPrototypeMappings to the same RPortPrototype **at the time when the creation of the manifest is finished**.⌋()

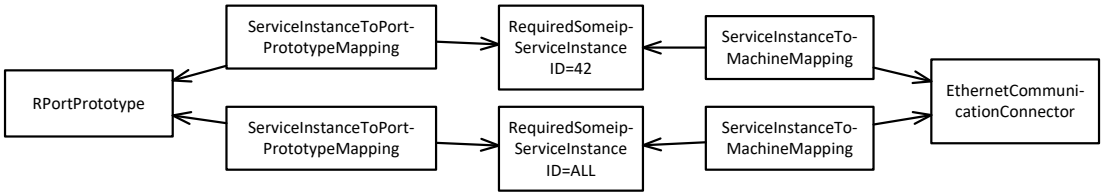


Figure 11.21: Depiction of not-allowed scenario

Please note that the depicted scenario in Figure 11.21 is covered by [constr_5115].

In other words the middleware is allowed to search for a concrete SOME/IP ServiceInstance on one VLAN and for ALL SOME/IP ServiceInstances on a different VLAN via a single RPortPrototype. But the middleware is only able to search for either ALL SOME/IP ServiceInstances or for one concrete SOME/IP ServiceInstance on the same VLAN via a single RPortPrototype.

Please note that the depicted scenario in Figure 11.22 is covered by [constr_5115].

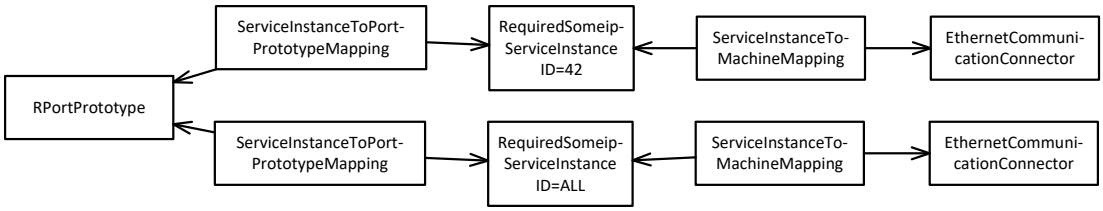


Figure 11.22: Depiction of allowed scenario

11.3.1.2.1 IP Configuration

In SOME/IP, the SubscribeEventGroup entry references IPv4 or IPv6 Endpoint options to indicate to the server where the client wants to receive the events of the SomeipEventGroup. Such an Endpoint contains the IP address of the client.

[TPS_MANI_03022]{DRAFT} **Context of RequiredSomeipServiceInstance** ⌈A RequiredSomeipServiceInstance can be mapped to a CommunicationConnector of a MachineDesign with the SomeipServiceInstanceToMachineMapping.

With this mapping an assignment of the RequiredSomeipServiceInstance to a unicast IP Address is established since the EthernetCommunicationConnector refers to a NetworkEndpoint in the role unicastNetworkEndpoint. The

unicastNetworkEndpoint defines the local IP address of the client.⌋(*RS_MANI_00009*, *RS_MANI_00024*)

11.3.1.2.2 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.

The modeling of the SomeipServiceInstanceToMachineMapping with TP and IP configuration is depicted in Figure D.1.

[TPS_MANI_03023]{DRAFT} **Udp Transport Protocol Configuration for RequiredSomeipServiceInstance** ⌈The SomeipServiceInstanceToMachineMapping.udpPort defines the Transport Protocol for a UDP communication in case that the server provides ServiceInterface content over UDP and the client wants to use it.⌋(*RS_MANI_00009*, *RS_MANI_00024*)

[TPS_MANI_03024]{DRAFT} **Tcp Transport Protocol Configuration for RequiredSomeipServiceInstance** ⌈The SomeipServiceInstanceToMachineMapping.tcpPort defines the Transport Protocol for a TCP communication in case that the server provides ServiceInterface content over TCP and the client wants to use it.⌋(*RS_MANI_00009*, *RS_MANI_00024*)

[TPS_MANI_03049]{DRAFT} **Tcp and Udp Transport Protocol Configuration for RequiredSomeipServiceInstance** ⌈It is allowed to set tcpPort and udpPort in the same SomeipServiceInstanceToMachineMapping. Such a setting shall be used in case that the server provides ServiceInterface content over Udp and Tcp and the client wants to use it.⌋(*RS_MANI_00009*, *RS_MANI_00024*)

[TPS_MANI_03237]{DRAFT} **Transport Protocol attributes defined for a RequiredSomeipServiceInstance** ⌈Each SomeipServiceInstanceToMachineMapping that is defined for a RequiredSomeipServiceInstance is allowed to have:

- a configured udpPort or
- a configured tcpPort or

- a configured udpPort and a tcpPort or
- no configured udpPort and tcpPort.

⌋(*RS_MANI_00009, RS_MANI_00024*)

A RequiredSomeipServiceInstance that is mapped by a SomeipServiceInstanceToMachineMapping that does not contain a udpPort and tcpPort is allowed to receive events over IP Multicast only.

In this case, it is not required for a client to have a unicast socket prepared if the server will always use the multicast transport. In such a case, the SubscribeEventGroup SOME/IP SD message that is sent from the ServiceConsumer to the ServiceProvider will not contain any Unicast Endpoint options.

In addition, the corresponding ServiceInterface is not allowed to have any Methods defined since the Request/Response communication pattern is restricted to IP Unicast only.

An additional prerequisite for such a setup is that the ProvidedSomeipServiceInstance is configured for IP multicast transmission only. In other words, the value of attribute multicastThreshold for all SomeipProvidedEventGroups needs to be set to 1.

[constr_5161]{DRAFT} **RequiredSomeipServiceInstance that is mapped by a SomeipServiceInstanceToMachineMapping without a configured tcpPort and udpPort** ⌈A RequiredSomeipServiceInstance that is mapped to a EthernetCommunicationConnector by a SomeipServiceInstanceToMachineMapping that does not have neither a udpPort nor a tcpPort is not allowed to reference a SomeipServiceInterfaceDeployment that includes SomeipMethodDeployments (directly or indirectly via ServiceFieldDeployment) **at the time when the creation of the manifest is finished**.⌋()

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].

11.3.1.2.3 Service Discovery Client Configuration

Service Discovery phases on the Client side allow minimizing the number of Service Discovery messages and allow a fast synchronization upon ECU start.

For every RequiredSomeipServiceInstance on a Client different phases are existing:

- Down
- Requested

- Initial Wait Phase
- Repetition Phase
- Main Phase

[TPS_MANI_03025]{DRAFT} **Client Timing configuration for a RequiredSomeipServiceInstance** ⌈The Client Timing is configurable with SomeipSdClientServiceInstanceConfig that is referenced in the role sdClientConfig by the RequiredSomeipServiceInstance for which the Timing is valid.⌋*(RS_MANI_00024)*

The number of SomeipSdClientServiceInstanceConfig elements determine how many timers shall actually be used by the middleware to keep the randomized times. Via the reference RequiredSomeipServiceInstance.sdClientConfig each RequiredSomeipServiceInstance defines to which timer it is assigned.

Note that it is possible to define several SomeipSdClientServiceInstanceConfig elements with identical timing specification values in order to request several timer handling in the middleware.

If several RequiredSomeipServiceInstance share the same timer then the expiration of that timer will lead a combined sending of service discovery messages.

[TPS_MANI_03231]{DRAFT} **Sharing timers for RequiredSomeipServiceInstance** ⌈If several RequiredSomeipServiceInstances point to the same SomeipSdClientServiceInstanceConfig in the role sdClientConfig then all of these RequiredSomeipServiceInstances will share the same timers for their timing behavior. This will lead to combining several service discovery entries in one service discovery message.⌋*(RS_MANI_00024)*

Class	**SomeipSdClientServiceInstanceConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
Note	Client specific settings that are relevant for the configuration of SOME/IP Service-Discovery. **Tags:** atp.recommendedPackage=SomeipSdTimingConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initialFind Behavior	InitialSdDelayConfig	0..1	aggr	Controls initial find behavior of clients.
priority	PositiveInteger	0..1	attr	This attribute defines the VLAN frame priority for Service Discovery messages that result from RequiredSomeip ServiceInstances that are referncing this SomeipSdClient ServiceInstanceConfig (Find, SubscribeEventGroup, Stop SubscribeEventgroup). Values from 0 (best effort) to 7 (highest) are allowed.

Table 11.51: SomeipSdClientServiceInstanceConfig

[TPS_MANI_03026]{DRAFT} **Initial Wait Phase configuration for a RequiredSomeipServiceInstance** ⌈The Initial Wait Phase for a RequiredSomeipServiceInstance is configured with the initialFindBehavior and the two attributes initialDelayMinValue and initialDelayMaxValue.

If a calculated random timer based on these min and max values expires the first FindService entry will be sent out.⌋*(RS_MANI_00024)*

When the calculated random timer expires and no OfferService is received the Repetition Phase will be entered.

[TPS_MANI_03027]{DRAFT} **Repetition Phase configuration for a RequiredSomeipServiceInstance** ⌈The Repetition 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.

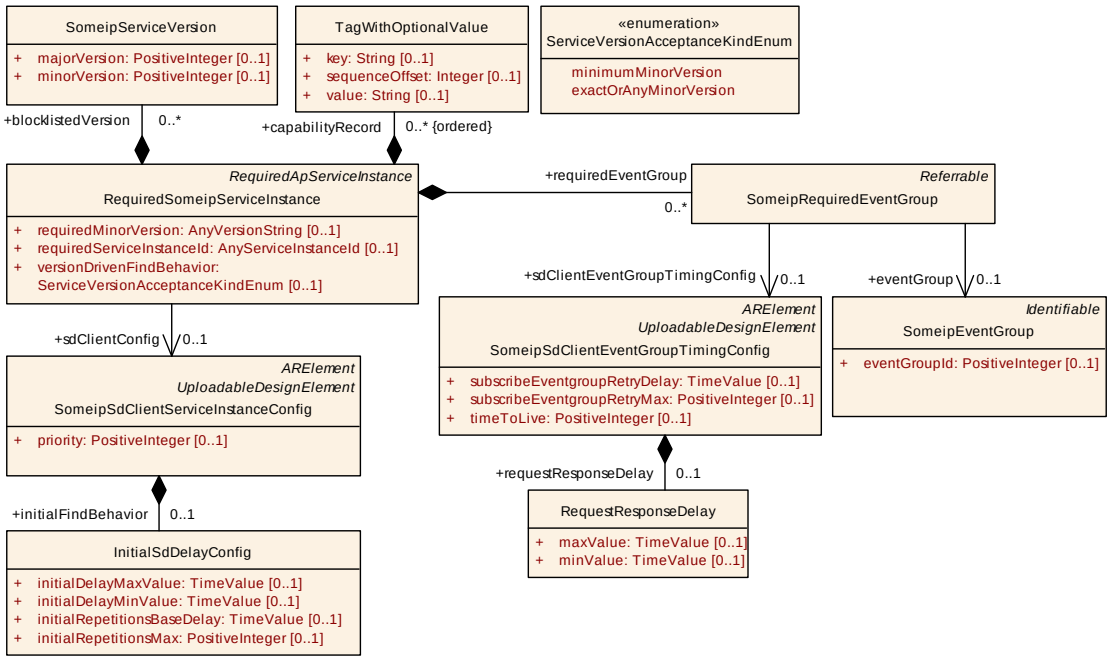


Figure 11.23: SOME/IP Service Discovery Client configuration settings

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 sent by the client.

SOME/IP allows for specifying 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.

Figure 11.24 shows an example of the different SOME/IP phases on the Client side.

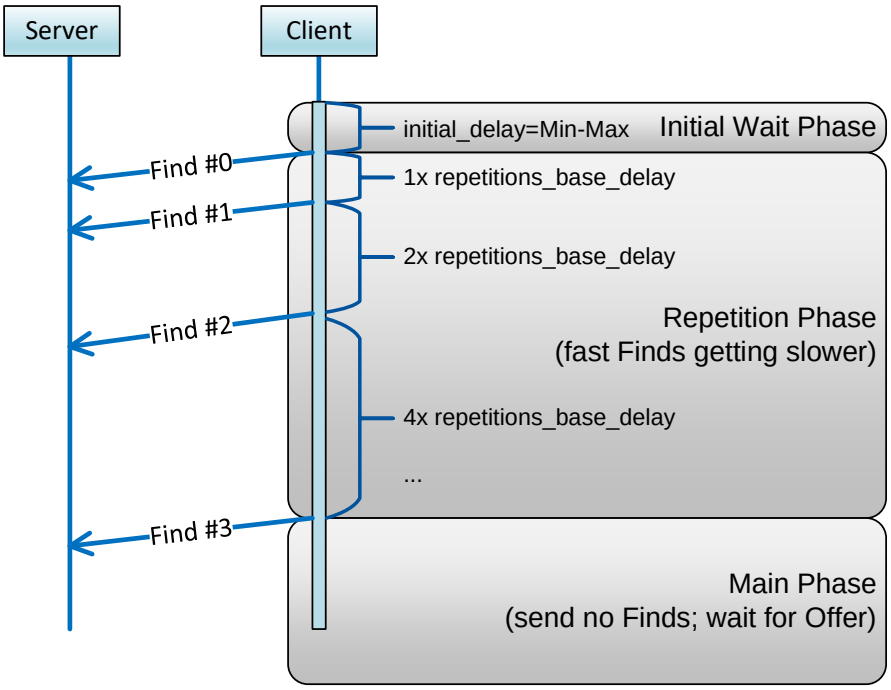


Figure 11.24: SOME/IP Client Timing example

[TPS_MANI_03029]{DRAFT} **Client Capability Records** ⌈A Capability Record (key/-value pair) on the Client side is configurable with the `capabilityRecord` and the two attributes `key` and `value`.⌋ *(RS_MANI_00024)*

11.3.1.2.4 Required Event Group

The `RequiredSomeipServiceInstance` aggregates a `SomeipRequiredEvent-Group` in the role `requiredEventGroup` that allows to define service instance specific configuration settings for a `SomeipEventGroup`.

Class	**SomeipRequiredEventGroup**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	The meta-class represents the ability to configure ServiceInstance related communication settings on the required side for each EventGroup separately.			
Base	*ARObject*, *Referrable*			
Aggregated by	RequiredSomeipServiceInstance.requiredEventGroup			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventGroup	SomeipEventGroup	0..1	ref	Reference to the SomeipEventGroup in the System Manifest for which the ServiceInstance related Event Group settings are valid.

▽

△

Class	SomeipRequiredEventGroup			
sdClientEvent GroupTiming Config	SomeipSdClientEvent GroupTimingConfig	0..1	ref	Client Timing configuration settings that are EventGroup specific.

Table 11.52: SomeipRequiredEventGroup

[constr_10222]{DRAFT} **Multiplicity of the reference in the role SomeipRequiredEventGroup.sdClientEventGroupTimingConfig** ⌈For each SomeipRequiredEventGroup, the reference in the role sdClientEventGroupTimingConfig shall exist **at the time when the creation of the manifest is finished**.⌋*()*

Class	SomeipSdClientEventGroupTimingConfig			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
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.recommendedPackage=SomeipSdTimingConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
request ResponseDelay	RequestResponseDelay	0..1	aggr	The Service Discovery shall delay answers to unicast messages triggered by multicast messages (e.g. Subscribe Eventgroup after Offer Service).
subscribe Eventgroup RetryDelay	TimeValue	0..1	attr	This attribute defines the interval in seconds to re-trigger a subscription to a Eventgroup, if a retry to subscribe to a Eventgroup is configured (subscribeEventgroupRetryMax > 0).
subscribe Eventgroup RetryMax	PositiveInteger	0..1	attr	This attribute define the maximum counts of retries to subscribe to an Eventgroup. If the value is set to 0 no retry shall be done. If the value is set to 255 the retry shall be done as along as the Eventgroup is requested and no SubscribeEventGroupAck was received.
timeToLive	PositiveInteger	0..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 11.53: SomeipSdClientEventGroupTimingConfig

[TPS_MANI_03030]{DRAFT} **SomeipSdClientEventGroupTimingConfig.timeToLive for SubscribeEventGroup Entries** ⌈The lifetime of an event subscription is configurable with the timeToLive attribute of SomeipSdClientEventGroupTimingConfig.

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

[TPS_MANI_03031]{DRAFT} **Clients RequestResponseDelay for received ServiceOffer entries** ⌈The Client will delay the SubscribeEventGroup answer to a received ServiceOffer message by the configured SomeipSdClientEventGroupTimingConfig.requestResponseDelay.

The actual delay will be randomly chosen between the maxValue and minValue.⌋ *(RS_MANI_00024)*

11.3.1.2.5 RequiredSomeipServiceInstance related method call properties

[TPS_MANI_03156]{DRAFT} **RequiredSomeipServiceInstance related configuration settings for methods** ⌈The class SomeipMethodProps that is aggregated by the RequiredSomeipServiceInstance in the role methodRequestProps allows specifying RequiredSomeipServiceInstance related configuration settings for a method request message. The method is defined in the SomeipServiceInterfaceDeployment referenced by the RequiredSomeipServiceInstance in the role serviceInterfaceDeployment.⌋*(RS_MANI_00024)*

[TPS_MANI_03159]{DRAFT} **Configuration of a data collection on a RequiredSomeipServiceInstance for transmission over UDP** ⌈The attributes udpCollectionBufferTimeout and udpCollectionTrigger support the configuration of a data collection of several messages for transmission over UDP. In the RequiredSomeipServiceInstance all method requests for which the udpCollectionTrigger is set to never will be collected in a buffer until a trigger arrives that starts the data transmission.

The following trigger options are supported:

- a message needs to be transmitted for which the udpCollectionTrigger is set to always.
- the udpCollectionBufferTimeout is reached for a message.
- the buffer size defined by the attribute udpCollectionBufferSizeThreshold is reached.

⌋*(RS_MANI_00024)*

The modeling of the RequiredSomeipServiceInstance-related event-properties and method-properties is depicted in Figure 11.20.

11.3.1.3 Provided Service Instance with static remote peers

[TPS_MANI_03312] Static configuration of remote peer addresses for a ProvidedSomeipServiceInstance ⌈AUTOSAR supports a static configuration of remote peers for a ProvidedSomeipServiceInstance that is mapped with a SomeipServiceInstanceToMachineMapping to an ApApplicationEndpoint with the remoteUnicastConfig.⌋*()*

If such SomeipRemoteUnicastConfig is referenced by the SomeipServiceInstanceToMachineMapping then the Service Discovery mechanism is not used and

the address of the remote peers is retrieved from the configuration. This allows to establish a communication connection between the local address (defined by the ApApplicationEndpoint and the NetworkEndpoint) and the configured remote address that is defined by the referenced SomeipRemoteUnicastConfig.

Please note that a collection of remote peers can be defined if the SomeipServiceInstanceToMachineMapping references a ProvidedSomeipServiceInstance in the role serviceInstance. In this case a communication connection to each of the configured remote peer addresses will be established. The ProvidedSomeipServiceInstance will transmit event and field notifications to the address that is defined in the SomeipRemoteUnicastConfig.

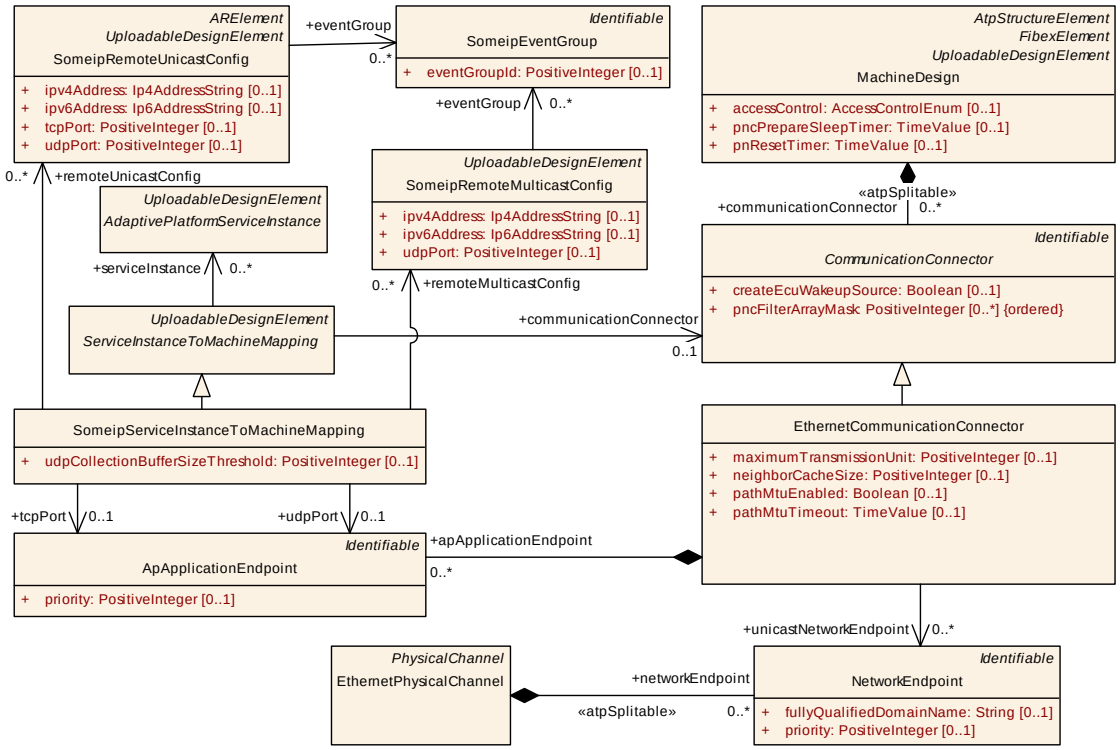


Figure 11.25: SomeipServiceInstanceToMachineMapping with static configured remote peers

Class	**SomeipRemoteUnicastConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class is used to statically configure the remote peer's unicast address in case that a static service connection is used and only a single remote peer exists. **Tags:** atp.Status=candidate atp.recommendedPackage=SomeipRemoteUnicastConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**SomeipRemoteUnicastConfig**			
eventGroup	SomeipEventGroup	*	ref	Reference to the SomeipEventGroups this Someip RemoteUnicatConfig applies to. **Tags:** atp.Status=candidate
ipv4Address	Ip4AddressString	0..1	attr	This attribute defines the IPv4 address of the remote peer to allow a static service connection between Service Provider and Service Consumer. **Tags:** atp.Status=candidate
ipv6Address	Ip6AddressString	0..1	attr	This attribute defines the IPv6 address of the remote peer to allow a static service connection between Service Provider and Service Consumer. **Tags:** atp.Status=candidate
tcpPort	PositiveInteger	0..1	attr	This attribute defines the tcpPort of the remote peer to allow a static service connection between Service Provider and Service Consumer. **Tags:** atp.Status=candidate
udpPort	PositiveInteger	0..1	attr	This attribute defines the udpPort of the remote peer to allow a static service connection between Service Provider and Service Consumer. **Tags:** atp.Status=candidate

Table 11.54: SomeipRemoteUnicastConfig

[constr_5355] **`SomeipServiceInstanceToMachineMapping` with configured remote peer addresses shall not mix `ProvidedSomeipServiceInstance`s and `RequiredSomeipServiceInstance`s** ⌈A `SomeipServiceInstanceToMachineMapping` that contains a reference to a `SomeipRemoteUnicastConfig` with the `remoteUnicastConfig` shall not reference:

- `ProvidedSomeipServiceInstance`s in the role `serviceInstance` and
- `RequiredSomeipServiceInstance`s in the role `serviceInstance` at the same time.

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

[TPS_MANI_03313] **Semantics of `SomeipRemoteUnicastConfig.eventGroup` reference** ⌈For each remote peer that is specified with the `SomeipRemoteUnicastConfig` it is possible to statically define the `SomeipEventGroup`s that the remote peer is interested in with the `SomeipRemoteUnicastConfig.eventGroup` reference.⌋*()*

With the `SomeipRemoteUnicastConfig.eventGroup` information each service provider has the knowledge from the configuration to which service consumers which event notifications need to be transmitted.

An example for the static configuration of a `ProvidedSomeipServiceInstance` with remote peers can be found in Appendix B.2.6.1.

11.3.1.4 Required Service Instance with static remote peers

[TPS_MANI_03314] Static configuration of a remote peer address for a `RequiredSomeipServiceInstance` ⌈AUTOSAR supports a static configuration of a remote peer for a `RequiredSomeipServiceInstance` that is mapped with a `SomeipServiceInstanceToMachineMapping` to an `ApApplicationEndpoint` with the `remoteUnicastConfig`.⌋*()*

With the static configuration of a remote peer the `RequiredSomeipServiceInstance` knows the address of the `ProvidedSomeipServiceInstance` and is able to call methods of the provided service.

[constr_5356] `RequiredSomeipServiceInstance` is allowed to have only a single statically configured remote peer as service provider ⌈A `SomeipServiceInstanceToMachineMapping` that contains references to a `RequiredSomeipServiceInstance` with the `serviceInstance` is allowed to reference only a single `SomeipRemoteUnicastConfig` in the role `remoteUnicastConfig` **at the time when the creation of the manifest is finished**.⌋*()*

Since the service provider may transport event notification via IP Multicast, the service consumer needs to know the IP Multicast groups it needs to join:

[TPS_MANI_03315] Semantics of `SomeipRemoteMulticastConfig` ⌈It is possible to statically define the IP Multicast addresses that the `RequiredSomeipServiceInstance`s shall listen to with the `SomeipRemoteMulticastConfig` element that is referenced in the `remoteMulticastConfig` role by the `SomeipServiceInstanceToMachineMapping` that references the `RequiredSomeipServiceInstance`s in the role `serviceInstance`.⌋*()*

Class	**SomeipRemoteMulticastConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class is used to statically configure the remote peer's multicast address. **Tags:** atp.Status=candidate atp.recommendedPackage=RemoteMulticastConfigs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventGroup	SomeipEventGroup	*	ref	Reference to the SomeipEventGroups this Someip RemoteMulticastConfig applies to. **Tags:** atp.Status=candidate
ipv4Address	Ip4AddressString	0..1	attr	This attribute defines the multicast IPv4 address to allow a static service connection between Service Provider and Service Consumers. **Tags:** atp.Status=candidate

▽

△

Class	SomeipRemoteMulticastConfig			
ipv6Address	Ip6AddressString	0..1	attr	This attribute defines the multicast IPv6 address to allow a static service connection between Service Provider and Service Consumers. **Tags:** atp.Status=candidate
udpPort	PositiveInteger	0..1	attr	This attribute defines the udpPort used for the multicast communication. **Tags:** atp.Status=candidate

Table 11.55: SomeipRemoteMulticastConfig

[TPS_MANI_03316] Semantics of `SomeipRemoteMulticastConfig.eventGroup` ⌈For each Multicast group that is specified with the `SomeipRemoteMulticastConfig`, it is possible to statically define the `eventGroup`s that include events that may be transmitted from the configured remote peer to the configured IP Multicast address.⌋*()*

[constr_5357] `SomeipRemoteMulticastConfig` shall only be used on required side ⌈Only a `SomeipServiceInstanceToMachineMapping` that contains references to one or several `RequiredSomeipServiceInstance`s with the `serviceInstance` role is allowed to reference one or several `SomeipRemoteMulticastConfig`s in the role `remoteUnicastConfig` **at the time when the creation of the manifest is finished**.⌋*()*

An example for the static configuration of a `RequiredSomeipServiceInstance` with remote peers can be found in Appendix B.2.6.2.

11.3.2 DDS Service Instance Deployment

In the case of DDS used as the transport layer the derived meta-classes are `DdsProvidedServiceInstance` or `DdsRequiredServiceInstance`. These meta-classes also carry attributes that apply for the service discovery on DDS.

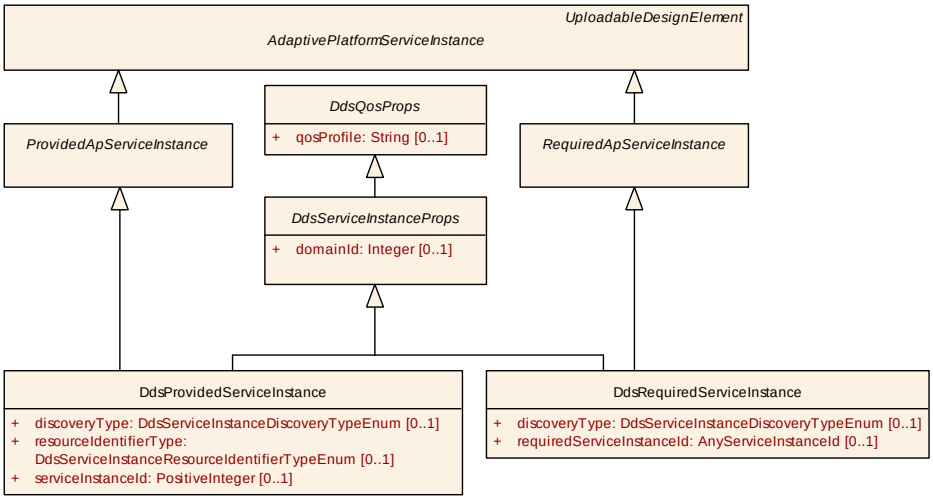


Figure 11.26: Dds Service Instances

Two discovery protocols are supported by the DDS Network binding:

- Service Instance announcement via purpose-specific formatting of the Domain Participant USER_DATA QoS policy
- Service Instance announcement through a purpose-specific discovery Topic.

For details on these two protocols and their implications, please refer to section 7.7.3 "DDS Network Binding" in [10].

Class	***DdsQosProps*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	QoS configuration properties for the DDS entities associated with an event, method, or field provided by or requested from a Service Instance using DDS as the underlying network binding.			
Base	*ARObject*			
Subclasses	DdsEventQosProps, DdsFieldQosProps, *DdsServiceInstanceProps*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
qosProfile	String	0..1	attr	Identifies a group of QoS Policies that apply to the DDS entities associated with the event, method, field, or the service instance.

Table 11.56: DdsQosProps

Class	***DdsServiceInstanceProps*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	Common configuration properties for the DDS entities provided by or requested from a Service Instance using DDS as the underlying network binding.			
Base	*ARObject*, *DdsQosProps*			
Subclasses	DdsProvidedServiceInstance, DdsRequiredServiceInstance			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
domainId	Integer	0..1	attr	This attribute identifies the DDS Domain the Service Instance shall join.

Table 11.57: DdsServiceInstanceProps

[constr_10223]{DRAFT} **Multiplicity of attribute `DdsServiceInstanceProps.domainId`** ⌈For each `DdsServiceInstanceProps`, the attribute `domainId` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

11.3.2.1 Provided DDS Service Instance

[TPS_MANI_03527]{DRAFT} **Definition of `DdsProvidedServiceInstance`** ⌈The `DdsProvidedServiceInstance` configures the Service to join a DDS Domain with the `domainId` attribute, and to instantiate the underlying DDS entities according to a QoS profile with the `qosProfile` attribute, using the discovery protocol defined by `discoveryType` and resource identification scheme defined by the `resourceIdentifierType` attribute. Moreover, it assigns an Instance ID to the Service for deployment with the `serviceInstanceId` attribute.⌋*(RS_MANI_00038)*

[TPS_MANI_03650]{DRAFT} **Definition of DdsServiceInstanceResourceIdentifierTypeEnum** ⌈The resource identification schemes enumerated by DdsServiceInstanceResourceIdentifierTypeEnum in DdsProvidedServiceInstance.resourceIdentifierType attribute define the various combinations of DDS features through which Provided Service Interface Instances can advertise, be bound, and communicate on a given DDS domain, as defined by section 7.7.3 "DDS Network Binding" of [10].⌋*(RS_MANI_00038)*

[constr_3528]{DRAFT} **Value range of DdsServiceInstanceProps.domainId** ⌈The value of attribute DdsServiceInstanceProps.domainId at DdsProvidedServiceInstance and domainId at DdsRequiredServiceInstance shall be in the range of a signed 32-bit integer **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3529]{DRAFT} **Value range of DdsProvidedServiceInstance.serviceInstanceId** ⌈The value of attribute DdsProvidedServiceInstance.serviceInstanceId shall be in the range of 0..65535 **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3541]{DRAFT} **qosProfile mandatory for DdsProvidedServiceInstance** ⌈The attribute qosProfile shall be defined for every DdsProvidedServiceInstance **at the time when manifest creation is finished at the time when the creation of the manifest is finished**.⌋*()*

[constr_3645]{DRAFT} **discoveryType mandatory for DdsProvidedServiceInstance** ⌈The attribute discoveryType shall be defined for every DdsProvidedServiceInstance **at the time when the creation of the manifest is finished**.⌋*()*

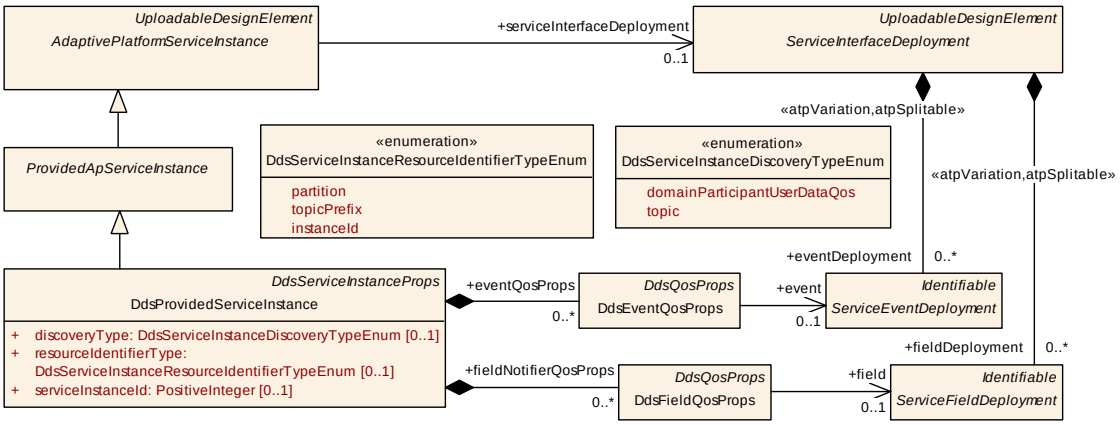


Figure 11.27: Provided Dds Service Instances

[constr_3646]{DRAFT} **resourceIdentifierType mandatory for DdsProvidedServiceInstance** ⌈The attribute resourceIdentifierType shall be defined for every DdsProvidedServiceInstance **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3647]{DRAFT} **resourceIdentifierType value for USER_DATA QoS-based discovery** ⌈If the value of discoveryType is domainParticipantUserDataQos, for a given DdsProvidedServiceInstance, the only valid value for attribute resourceIdentifierType is partition **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3564]{DRAFT} **Consistency between DDS Service Interface Deployment and Provided DDS Service Instance** ⌈Transport attributes DdsServiceInterfaceDeployment.transportProtocol and DdsEventDeployment.transportProtocol shall be consistent with DDS profiles generated and selected by the DdsQosProps component of DdsProvidedServiceInstance, DdsFieldQosProps, and DdsEventQosProps **at the time when the creation of the manifest is finished**.⌋*()*

Class	**DdsProvidedServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in a concrete implementation on top of DDS. **Tags:** atp.recommendedPackage=ServiceInstances			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *DdsQosProps*, *DdsServiceInstanceProps*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *ProvidedApServiceInstance*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
discoveryType	DdsServiceInstanceDiscoveryTypeEnum	0..1	attr	Discovery protocol.
eventQosProps	DdsEventQosProps	*	aggr	List of configuration properties for the Events that are provided by the Service Instance.
fieldNotifierQosProps	DdsFieldQosProps	*	aggr	List of configuration properties for Field notifiers that are provided by the Service Instance.
resourceIdentifierType	DdsServiceInstanceResourceIdentifierTypeEnum	0..1	attr	Type of resource identification scheme.
serviceInstanceId	PositiveInteger	0..1	attr	Identification number that is used by DDS to identify DomainParticipants associated with an instance of the service.

Table 11.58: DdsProvidedServiceInstance

Enumeration	**DdsServiceInstanceDiscoveryTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment
Note	Supported discovery schemes for DDS Service Instances.
Aggregated by	DdsProvidedServiceInstance.discoveryType, DdsRequiredServiceInstance.discoveryType
Literal	***Description***
domainParticipantUserDataQos	The USER_DATA QoS policy is used to advertise and discover available Service Instances hosted by each Domain Participant. **Tags:** atp.EnumerationLiteralIndex=0

▽

△

Enumeration	**DdsServiceInstanceDiscoveryTypeEnum**
topic	A purpose-specific Topic is used to convey availability of Service Instances and how to bind against them. **Tags:** atp.EnumerationLiteralIndex=1

Table 11.59: DdsServiceInstanceDiscoveryTypeEnum

Enumeration	**DdsServiceInstanceResourceIdentifierTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment
Note	Supported Resource Identification schemes for DDS Service Instances.
Aggregated by	DdsProvidedServiceInstance.resourceIdentifierType
Literal	***Description***
instanceId	In-band instance identification fields are used to discriminate samples related to specific Service Instances sharing the same DDS Topics • Partitions: - • Topics: ara.com://services/<InterfaceId>/<Major>.<<Minor>/<TopicName> **Tags:** atp.EnumerationLiteralIndex=2
partition	The DDS PARTITION QoS policy is used to isolate DDS Topics related to specific Service Instances • Partitions: ara.com://services/<InterfaceId>/<InstanceId> • Topics: ara.com://services/<InterfaceId>/<Major>.<Minor>/<TopicName> **Tags:** atp.EnumerationLiteralIndex=0
topicPrefix	Unique prefixes are assigned to DDS Topics related to specific Service Instances • Partitions: - • Topics: ara.com://services/<InterfaceId>/<InstanceId>/<TopicName> **Tags:** atp.EnumerationLiteralIndex=1

Table 11.60: DdsServiceInstanceResourceIdentifierTypeEnum

[TPS_MANI_03528]{DRAFT} **Definition of `DdsProvidedServiceInstance.eventQosProps`** ⌈The `DdsProvidedServiceInstance.eventQosProps` configures the DDS entities associated with the `event` according to a QoS Profile specified with the `qosProfile` attribute.⌋*(RS_MANI_00038)*

[TPS_MANI_03531]{DRAFT} **`qosProfile` of `DdsProvidedServiceInstance.eventQosProps` is optional** ⌈The attribute `qosProfile` of `DdsProvidedServiceInstance.eventQosProps` is optional; if `qosProfile` is not defined, the underlying DDS entities shall be configured according to the `qosProfile` attribute of the parent `DdsProvidedServiceInstance`.⌋*(RS_MANI_00038)*

[TPS_MANI_03561]{DRAFT} **Definition of `DdsProvidedServiceInstance.fieldNotifierQosProps`** ⌈The `DdsProvidedServiceInstance.fieldNotifierQosProps` configures the DDS entities associated with the `field` according to a QoS Profile specified with the `qosProfile` attribute.⌋*(RS_MANI_00038)*

Class	DdsEventQosProps			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	Configuration properties of the Event using DDS as the underlying network binding.			
Base	*ARObject*, *DdsQosProps*			
Aggregated by	DdsProvidedServiceInstance.eventQosProps, DdsRequiredServiceInstance.eventQosProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
event	ServiceEvent Deployment	0..1	ref	Reference to an event that is provided.

Table 11.61: DdsEventQosProps

[constr_10224]{DRAFT} **Multiplicity of reference in the role `DdsEventQosProps`. `event`** ⌈For each `DdsEventQosProps`, the reference in the role `event` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[TPS_MANI_03562]{DRAFT} **`qosProfile` of `DdsProvidedServiceInstance`. `fieldNotifierQosProps` is optional** ⌈The attribute `qosProfile` of `DdsProvidedServiceInstance.fieldNotifierQosProps` is optional; if `qosProfile` is not defined, the underlying DDS entities shall be configured according to the `qosProfile` attribute of the parent `DdsProvidedServiceInstance`.⌋*(RS_MANI_00038)*

Class	DdsFieldQosProps			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	Configuration properties of the Field interaction when using DDS as the underlying network binding.			
Base	*ARObject*, *DdsQosProps*			
Aggregated by	DdsProvidedServiceInstance.fieldNotifierQosProps, DdsRequiredServiceInstance.fieldNotifierQosProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
field	ServiceField Deployment	0..1	ref	Reference to the field.

Table 11.62: DdsFieldQosProps

[constr_10225]{DRAFT} **Multiplicity of reference in the role `DdsFieldQosProps`. `field`** ⌈For each `DdsFieldQosProps`, the reference in the role `field` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

11.3.2.2 Required DDS Service Instance

[TPS_MANI_03529]{DRAFT} **Definition of `DdsRequiredServiceInstance`** ⌈The `DdsRequiredServiceInstance` configures the Client to join a DDS Domain with the `domainId` attribute, and to instantiate the underlying DDS entities according to a QoS Profile with the `qosProfile` attribute, using the discovery protocol defined by `discoveryType`. Optionally, the `requiredServiceInstanceId` attribute allows a Client to search for a specific Instance ID of the serviceInterface.⌋*(RS_MANI_00038)*

[constr_3542]{DRAFT} **qosProfile mandatory for DdsRequiredServiceInstance** ⌈The attribute qosProfile shall be defined for every DdsRequiredServiceInstance **at the time when the creation of the manifest is finished.**⌋()

[constr_3648]{DRAFT} **discoveryType mandatory for DdsRequiredServiceInstance** ⌈The attribute discoveryType shall be defined for every DdsRequiredServiceInstance **at the time when the creation of the manifest is finished.**⌋()

[constr_3565]{DRAFT} **Consistency between DDS Service Interface Deployment and Required DDS Service Instance** ⌈Transport attributes DdsServiceInterfaceDeployment.transportProtocol and DdsEventDeployment.transportProtocol shall be consistent with DDS profiles generated and selected by the DdsQosProps component of DdsRequiredServiceInstance, DdsFieldQosProps, and DdsEventQosProps **at the time when the creation of the manifest is finished.**⌋()

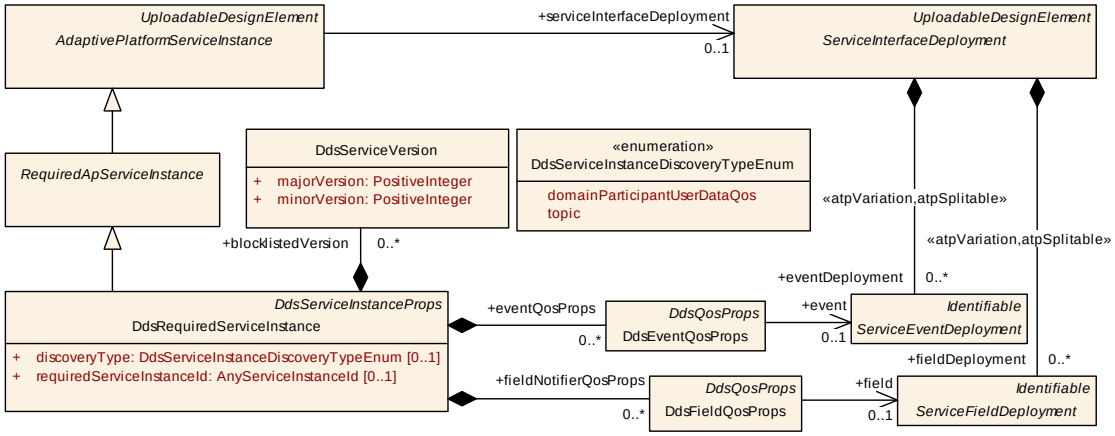


Figure 11.28: Required Dds Service Instances

Class	**DdsRequiredServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in a concrete implementation on top of DDS. **Tags:** atp.recommendedPackage=ServiceInstances			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *DdsQosProps*, *DdsServiceInstanceProps*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *RequiredApServiceInstance*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
blocklisted Version	DdsServiceVersion	*	aggr	Collection of blocklisted versions.
discoveryType	DdsServiceInstance DiscoveryTypeEnum	0..1	attr	Discovery protocol.
eventQosProps	DdsEventQosProps	*	aggr	List of configuration properties for the Events that are required by the Service Instance.

△

Class	**DdsRequiredServiceInstance**			
fieldNotifierQos Props	DdsFieldQosProps	*	aggr	List of configuration properties for Field notifiers that are required by the Service Instance.
requiredService InstanceId	AnyServiceInstanceId	0..1	attr	This attribute represents the ability to describe the required service instance ID.

Table 11.63: DdsRequiredServiceInstance

Class	**DdsServiceVersion**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	Definition of a Dds service version.			
Base	*ARObject*			
Aggregated by	DdsRequiredServiceInstance.blocklistedVersion			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
majorVersion	PositiveInteger	1	attr	Service major version. **Tags:** xml.sequenceOffset=10
minorVersion	PositiveInteger	1	attr	Service minor version. **Tags:** xml.sequenceOffset=20

Table 11.64: DdsServiceVersion

[TPS_MANI_03530]{DRAFT} **Definition of `DdsRequiredServiceInstance.eventQosProps`** ⌈The `DdsRequiredServiceInstance.eventQosProps` configures the DDS entities responsible for subscribing to an `event` according to a QoS Profile specified with the `qosProfile` attribute.⌋*(RS_MANI_00038)*

[TPS_MANI_03532]{DRAFT} **`qosProfile` of `DdsRequiredServiceInstance.eventQosProps` is optional** ⌈The attribute `qosProfile` of `DdsRequiredServiceInstance.eventQosProps` is optional; if `qosProfile` is not defined, the underlying DDS entities shall be configured according to the `qosProfile` attribute of the parent `DdsRequiredServiceInstance`.⌋*(RS_MANI_00038)*

[TPS_MANI_03567]{DRAFT} **Definition of `DdsRequiredServiceInstance.fieldNotifierQosProps`** ⌈The `DdsRequiredServiceInstance.fieldNotifierQosProps` configures the DDS entities associated with the `field` according to a QoS Profile specified with the `qosProfile` attribute.⌋*(RS_MANI_00038)*

[TPS_MANI_03568]{DRAFT} **`qosProfile` of `DdsRequiredServiceInstance.fieldNotifierQosProps` is optional** ⌈The attribute `qosProfile` of `DdsRequiredServiceInstance.fieldNotifierQosProps` is optional; if `qosProfile` is not defined, the underlying DDS entities shall be configured according to the `qosProfile` attribute of the parent `DdsRequiredServiceInstance`.⌋*(RS_MANI_00038)*

11.3.2.3 DDS Service Instance to Machine mapping

The DdsServiceInstanceToMachineMapping defines on which network / VLAN the DDS communication shall be deployed.

[TPS_MANI_03533]{DRAFT} **DdsServiceInstanceToMachineMapping** ⌈The DdsServiceInstanceToMachineMapping defines for a specific serviceInstance (either DdsProvidedServiceInstance or DdsRequiredServiceInstance) on which network the communication shall be done using the reference communicationConnector to CommunicationConnector.⌋*(RS_MANI_00038)*

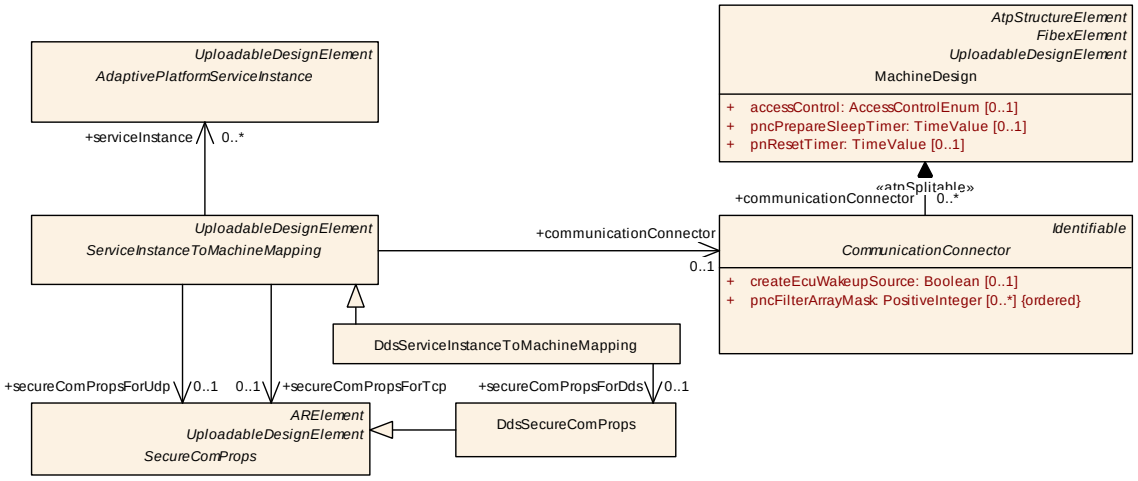


Figure 11.29: Dds Service Instance to Machine mapping

[constr_3684]{DRAFT} **Mutual exclusivity of Secure Communication Properties** ⌈The attributes ServiceInstanceToMachineMapping.secOcComPropsForMulticast and DdsServiceInstanceToMachineMapping.secureComPropsForDds are mutually exclusive, meaning zero or just one of them shall be set depending on whether no security, SecOC, or DDS Security is chosen as data-level security (optionally) above transport-level security **at the time when the creation of the manifest is finished**.⌋*()*

Class	**DdsServiceInstanceToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	This meta-class allows to map DdsServiceInstances to a CommunicationConnector of a Machine. **Tags:** atp.recommendedPackage=ServiceInstanceToMachineMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *ServiceInstanceToMachineMapping*, *UploadableDesignElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
secureCom PropsForDds	DdsSecureComProps	0..1	ref	Reference to SecureComProps applicable to the service instance.

Table 11.65: DdsServiceInstanceToMachineMapping

11.3.3 User Defined Service Instance Deployment

[TPS_MANI_03032]{DRAFT} **Description of middleware technologies not standardized by AUTOSAR** ⌈The elements `ProvidedUserDefinedServiceInstance` and `RequiredUserDefinedServiceInstance` can be used to describe alternative middleware technologies that are not standardized by AUTOSAR.⌋ *(RS_MANI_00014)*

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.

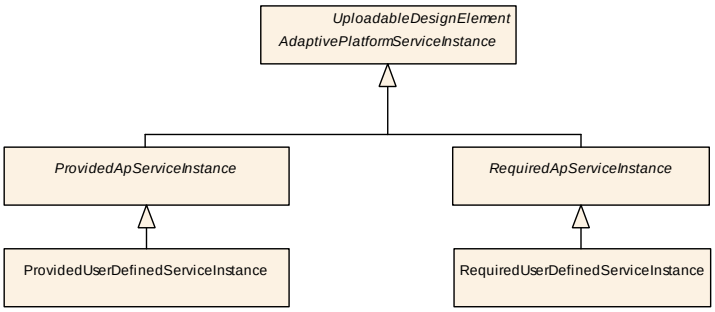


Figure 11.30: User Defined Service Instance Deployment

Class	**ProvidedUserDefinedServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a provided service instance in a concrete implementation that is not standardized by AUTOSAR. **Tags:** atp.recommendedPackage=ServiceInstances			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *ProvidedApServiceInstance*, *Referrable*, *Uploadable DesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 11.66: ProvidedUserDefinedServiceInstance

Class	**RequiredUserDefinedServiceInstance**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment			
Note	This meta-class represents the ability to describe the existence and configuration of a required service instance in a concrete implementation that is not standardized by AUTOSAR. **Tags:** atp.recommendedPackage=ServiceInstances			
Base	*ARElement*, *ARObject*, *AdaptivePlatformServiceInstance*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *RequiredApServiceInstance*, *Uploadable DesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 11.67: RequiredUserDefinedServiceInstance

11.3.4 Service Deployment Relations

The meta-model is set up in a way that it is in general possible to create a relation between different sub-classes of

- ServiceInterfaceDeployment
- AdaptivePlatformServiceInstance
- ServiceInstanceToMachineMapping

For example, from the formal point of view, it would be possible to let a UserDefinedServiceInstanceToMachineMapping reference a DdsProvidedServiceInstance that in turn references a SomeipServiceInterfaceDeployment. But it is obviously questionable whether such a combination makes sense in a real-world model.

The consequence of the modeling is therefore that the possible combinations of the three mentioned meta-classes need to be regulated by a constraint.

[constr_3641]{DRAFT} **Allowed combinations of ServiceInterfaceDeployment, AdaptivePlatformServiceInstance, ServiceInstanceToMachineMapping** ⌈

	DdsProvidedServiceInstance	ProvidedSomeipServiceInstance	ProvidedUserDefinedServiceInstance	DdsRequiredServiceInstance	RequiredSomeipServiceInstance	RequiredUserDefinedServiceInstance
DdsServiceInterfaceDeployment	Yes	No	Yes	Yes	No	Yes
SomeipServiceInterfaceDeployment	No	Yes	Yes	No	Yes	Yes
UserDefinedServiceInterfaceDeployment	No	No	Yes	No	No	Yes
DdsServiceInstanceToMachineMapping	Yes	No	No	Yes	No	No
SomeipServiceInstanceToMachineMapping	No	Yes	No	No	Yes	No
UserDefinedServiceInstanceToMachineMapping	No	Yes	Yes	No	Yes	Yes

This rule shall be imposed **at the time when the creation of the manifest is finished**

⌋()

11.4 EndToEndProtection

AUTOSAR supports the protection of events, methods, Field notifiers, Field get methods and Field set methods with E2E Profiles that are defined in the E2E Communication Protection Library [34].

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

Please note that the E2E protection of a field notifier is possible with the End2EndEventProtectionProps.event reference since each specific ServiceFieldDeployment element aggregates a ServiceEventDeployment in the role notifier. If such an aggregated ServiceEventDeployment is referenced with the End2EndEventProtectionProps.event reference the E2E protection settings are valid for the notifier that is embedded by the ServiceFieldDeployment.

Since the End2EndEventProtectionProps element is aggregated by the abstract AdaptivePlatformServiceInstance it can be used to describe the End-to-End protection on specific derived classes like ProvidedSomeipServiceInstance or RequiredSomeipServiceInstance that fit the underlying middleware.

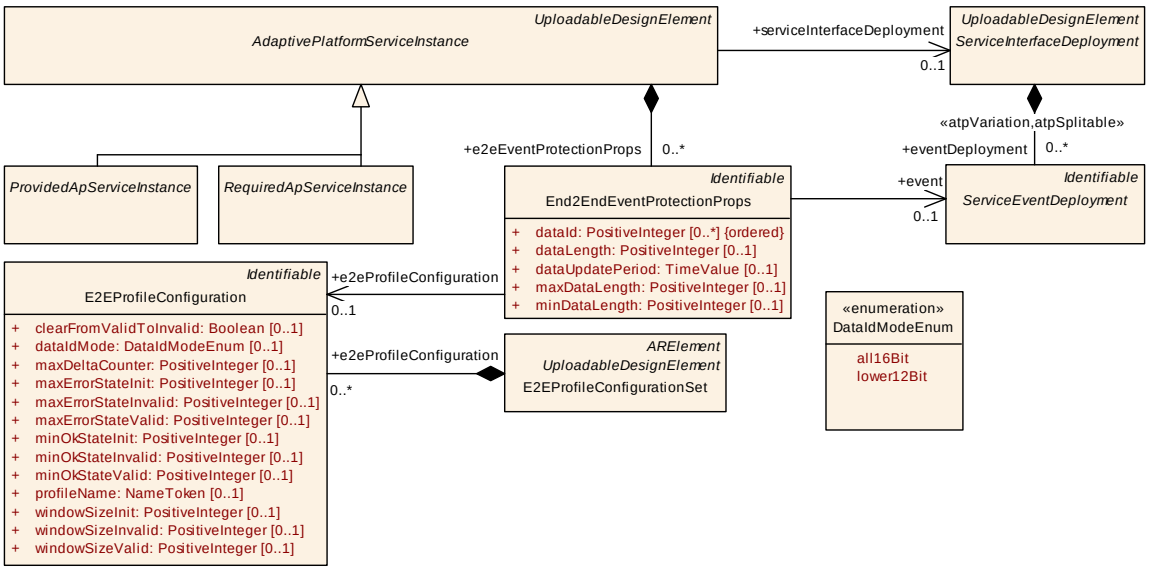


Figure 11.31: E2E EventProtection

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_03228]{DRAFT} **Usage of End2EndMethodProtectionProps** ⌈The End2EndMethodProtectionProps element is used to define method specific E2E configuration settings in the context of an AdaptivePlatformServiceInstance.⌋ *(RS_MANI_00028)*

Please note that the E2E protection of field `get` and `set` methods is possible with the `End2EndMethodProtectionProps.method` reference since each specific `ServiceFieldDeployment` element is allowed to aggregate a `ServiceMethodDeployment` in the role `get` and/or `set`.

If such an aggregated `ServiceMethodDeployment` is referenced with the `End2EndMethodProtectionProps.method` reference the E2E protection settings are valid for the `get` or `set` method that is embedded by the `ServiceFieldDeployment`.

[TPS_MANI_03129]{DRAFT} **E2E profile** ⌈The E2E profile is defined by `E2EProfileConfiguration.profileName`.⌋*(RS_MANI_00028)*

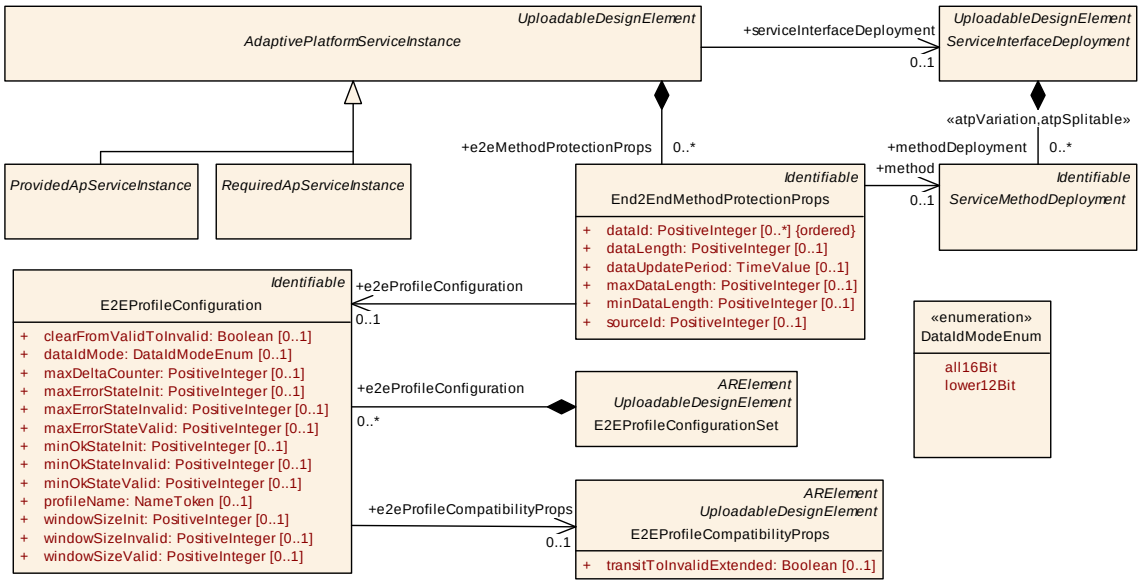


Figure 11.32: E2E MethodProtection

[TPS_MANI_03130]{DRAFT} **Standardized values of the attribute `E2EProfileConfiguration.profileName`** ⌈The `E2EProfileConfiguration.profileName` that is referenced by an `End2EndEventProtectionProps` or by an `End2EndMethodProtectionProps` can have the following values that are standardized by AUTOSAR: `PROFILE_04`, `PROFILE_05`, `PROFILE_06`, `PROFILE_07`, `PROFILE_08`, `PROFILE_11`, `PROFILE_22`, `PROFILE_04m`, `PROFILE_07m`, `PROFILE_44`, `PROFILE_08m`, and `PROFILE_44m`.⌋*(RS_MANI_00028)*

[TPS_MANI_03131]{DRAFT} **Non-Standardized values of attribute `E2EProfileConfiguration.profileName`** ⌈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 on the *AUTOSAR adaptive Platform*. In addition, it is positively possible to use other than the standardized values for the `profileName`.⌋*(RS_MANI_00028)*

[TPS_MANI_03128]{DRAFT} **Usage of same `End2EndEventProtectionProps.dataId` in case of Multi-Binding** ⌈In case of Multi-Binding, i.e. if different AdaptivePlatformServiceInstances exist that are mapped by ServiceInstanceToPortPrototypeMapping to the same PortPrototype, the different AdaptivePlatformServiceInstances may contain the same dataId for the same event.⌋ *(RS_MANI_00028)*

In other words, if a PortPrototype contains two transport layer bindings, e.g. a ProvidedSomeipServiceInstance and a ProvidedUserDefinedServiceInstance representing an IPC communication, then an event is allowed to be protected with the same dataId in both AdaptivePlatformServiceInstances because the two AdaptivePlatformServiceInstances effectively represent the identical piece of data.

[TPS_MANI_03229]{DRAFT} **Usage of same `End2EndMethodProtectionProps.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 method.⌋ *(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 a method is allowed to be protected with the same dataId in both AdaptivePlatformServiceInstances.

[TPS_MANI_03252]{DRAFT} **Usage of same `End2EndMethodProtectionProps.sourceId` 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 sourceId (for the same and even for a different method).⌋ *(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 a single as well as different methods are allowed to be protected with the same sourceId in both AdaptivePlatformServiceInstances.

Class	**End2EndEventProtectionProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::E2E			
Note	This element allows to protect an event or a field notifier with an E2E profile.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AdaptivePlatformServiceInstance*.e2eEventProtectionProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**End2EndEventProtectionProps**			
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 EndToEnd Protection.
dataLength	PositiveInteger	0..1	attr	Length of payload including E2E header in bits.
dataUpdate Period	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.
e2eProfile Configuration	E2EProfileConfiguration	0..1	ref	Reference to E2E profile configuration settings that are valid to protect the referenced event or field notifier.
event	ServiceEvent Deployment	0..1	ref	Reference to an event that is protected by the E2E profile.
maxDataLength	PositiveInteger	0..1	attr	Maximum length of payload including E2E header in bits.
minDataLength	PositiveInteger	0..1	attr	Minimum length of payload including E2E header in bits.

Table 11.68: End2EndEventProtectionProps

Class	**End2EndMethodProtectionProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::E2E			
Note	This element allows to protect a method, a field setter or a field getter with an E2E profile.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AdaptivePlatformServiceInstance*.e2eMethodProtectionProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataId (ordered)	PositiveInteger	*	attr	This represents a numerical identifier that is included in the CRC calculation. This dataId is used for call and response. Note: ID is used for protection against masquerading. The details concerning the maximum number of values (this information is specific for each E2E profile) applicable for this attribute are controlled by a semantic constraint that depends on the category of the EndToEnd Protection.
dataLength	PositiveInteger	0..1	attr	Length of payload including E2E header in bits.
dataUpdate Period	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.
e2eProfile Configuration	E2EProfileConfiguration	0..1	ref	Reference to E2E profile configuration settings that are valid to protect the referenced method, field getter or field setter.
maxDataLength	PositiveInteger	0..1	attr	Maximum length of payload including E2E header in bits.
method	ServiceMethod Deployment	0..1	ref	Reference to a method, a field getter or a field setter that is protected by the E2E profile.
minDataLength	PositiveInteger	0..1	attr	Minimum length of payload including E2E header in bits.

▽

△

Class	End2EndMethodProtectionProps			
sourceId	PositiveInteger	0..1	attr	This represents a unique numerical identifier identifying the source of a certain transmission. In case of C/S communication, this ID uniquely identifies the client. Note: ID is used for protection against masquerading. The details concerning the maximum number of values (this information is specific for each E2E profile) applicable for this attribute are controlled by a semantic constraint that depends on the category of the EndToEnd Protection.

Table 11.69: End2EndMethodProtectionProps

Class	E2EProfileConfigurationSet			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::E2E			
Note	This meta-class represents the ability to aggregate a collection of E2EProfileConfigurations. **Tags:** atp.recommendedPackage=E2EProfileConfigurationSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
e2eProfile Configuration	E2EProfileConfiguration	*	aggr	This represents the collection of E2EProfileConfigurations aggregated at the E2EProfileConfigurationSet.

Table 11.70: E2EProfileConfigurationSet

Class	E2EProfileConfiguration			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::E2E			
Note	This element holds E2E profile specific configuration settings.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	E2EProfileConfigurationSet.e2eProfileConfiguration			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
clearFromValid ToInvalid	Boolean	0..1	attr	Clear monitoring window on transition from state Valid to state Invalid.
dataIdMode	DataIdModeEnum	0..1	attr	This attribute describes the inclusion mode that is used to include the implicit Data ID in the one-byte CRC.
e2eProfile Compatibility Props	E2EProfileCompatibility Props	0..1	ref	Reference to additional settings for the E2E state machine.
maxDelta Counter	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 Max DeltaCounter is 3, then at the next reception the receiver can accept Counters with values 2, 3 or 4.
maxErrorState Init	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_INIT.
maxErrorState Invalid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_INVALID.

▽

△

Class	**E2EProfileConfiguration**			
maxErrorState Valid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_VALID.
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.
minOkState Invalid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INVALID.
minOkState Valid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_VALID.
profileName	NameToken	0..1	attr	Definition of the E2E profile.
windowSizeInit	PositiveInteger	0..1	attr	Size of the monitoring window of state Init for the E2E state machine.
windowSize Invalid	PositiveInteger	0..1	attr	Size of the monitoring window of state Invalid for the E2E state machine.
windowSize Valid	PositiveInteger	0..1	attr	Size of the monitoring window of state Valid for the E2E state machine.

Table 11.71: E2EProfileConfiguration

[constr_10226]{DRAFT} **Multiplicity of attribute `E2EProfileConfiguration.profileName`** ⌈For each `E2EProfileConfiguration`, the attribute `profileName` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

Enumeration	**DataIdModeEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer
Note	Supported inclusion modes to include the implicit two-byte Data ID in the one-byte CRC.
Aggregated by	E2EProfileConfiguration.dataIdMode, EndToEndTransformationDescription.dataIdMode
Literal	*Description*
all16Bit	Two bytes are included in the CRC (double ID configuration). **Tags:** atp.EnumerationLiteralIndex=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.EnumerationLiteralIndex=2

Table 11.72: DataIdModeEnum

Class	**E2EProfileCompatibilityProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	This meta-class collects settings for configuration of the E2E state machine. **Tags:** atp.recommendedPackage=E2EProfileCompatibilityPropsCollection			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

▽

△

Class	E2EProfileCompatibilityProps			
transitToInvalid Extended	Boolean	0..1	attr	E2E State machine behavior concerning transition from NODATA/INIT to INVALID value=0 (false): no direct transition from NODATA to INVALID, no transition from INIT to INVALID due to counter-related faults (Autosar R19-11 or former behavior) value=1 (true): direct transition from NODATA to INVALID covered, transition from INIT to INVALID due to counter-related faults covered (state machine extended)

Table 11.73: E2EProfileCompatibilityProps

It is possible to overwrite the E2E state machine configuration settings that are defined in `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 notifiers.

Likewise, it is possible to overwrite the E2E state machine configuration settings that are defined in `E2EProfileConfiguration` at the `RPortPrototype` of a `SwComponentType` with settings available in the `ClientComSpec` as described in [TPS_MANI_01324].

With this approach it is possible to define individual E2E settings for different callers of a `method`.

Finally, it is possible to overwrite the E2E state machine configuration settings that are defined in `E2EProfileConfiguration` at the `PPortPrototype` of a `SwComponentType` with settings available in the `ServerComSpec` as described in [TPS_MANI_01325].

[constr_3493]{DRAFT} **Applicable attributes for standardized E2E Profiles** ⌈

E2E Attributes	Root Element			Attribute Existence per Profile											
	End2EndEventProtectionProps	End2EndMethodProtectionProps	E2EProfileConfiguration	PROFILE_04	PROFILE_05	PROFILE_06	PROFILE_07	PROFILE_08	PROFILE_11	PROFILE_22	PROFILE_04m	PROFILE_07m	PROFILE_44	PROFILE_08m	PROFILE_44m
dataId	x	x		1	1	1	1	1	1	n	1	1	1	1	1
dataLength	x	x			x				x	x					
minDataLength	x	x		x		x	x	x			x	x	x	x	x

▽

△

E2E Attributes	Root Element			Attribute Existence per Profile											
	End2EndEventProtectionProps	End2EndMethodProtectionProps	E2EProfileConfiguration	PROFILE_04	PROFILE_05	PROFILE_06	PROFILE_07	PROFILE_08	PROFILE_11	PROFILE_22	PROFILE_04m	PROFILE_07m	PROFILE_44	PROFILE_08m	PROFILE_44m
maxDataLength	x	x		x		x	x	x			x	x	x	x	x
dataUpdatePeriod	x	x		x	x	x	x	x	x	x	x	x	x	x	x
sourceId		x									x	x		x	x
dataIdMode			x						x						
maxDeltaCounter			x	x	x	x	x	x	x	x	x	x	x	x	x
maxErrorStateInit			x	x	x	x	x	x	x	x	x	x	x	x	x
maxErrorStateInvalid			x	x	x	x	x	x	x	x	x	x	x	x	x
maxErrorStateValid			x	x	x	x	x	x	x	x	x	x	x	x	x
minOkStateInit			x	x	x	x	x	x	x	x	x	x	x	x	x
minOkStateInvalid			x	x	x	x	x	x	x	x	x	x	x	x	x
minOkStateValid			x	x	x	x	x	x	x	x	x	x	x	x	x
windowSizeValid			x	x	x	x	x	x	x	x	x	x	x	x	x
windowSizeInvalid			x	x	x	x	x	x	x	x	x	x	x	x	x
windowSizeInit			x	x	x	x	x	x	x	x	x	x	x	x	x
clearFromValidToInvalid			x	x	x	x	x	x	x	x	x	x	x	x	x

This rule shall be imposed **at the time when the creation of the manifest is finished**

⌋()

In PROFILE_22, the dataId is defined as a list of 16 dataId values, where a different value is transmitted depending on the counter value.

Please also note that the Classic Platform attributes counterOffset, crcOffset and dataIdNibbleOffset are not configurable on the *AUTOSAR adaptive Platform* and are set to fixed values by the AUTOSAR Standard.

[constr_5230]{DRAFT} **Attribute E2EProfileCompatibilityProps.transitToInvalidExtended shall exist for each E2EProfileConfiguration** ⌈For each E2EProfileConfiguration, a reference to E2EProfileCompatibilityProps in the role e2eProfileCompatibilityProps shall exist and the referenced E2EProfileCompatibilityProps shall define a value for the attribute transitToInvalidExtended **at the time when the creation of the manifest is finished**.⌋()

[constr_5250]{DRAFT} **Protection of `AdaptivePlatformServiceInstance`s of the same `ServiceInterfaceDeployment`** ⌈If several `AdaptivePlatformServiceInstance`s exist that are referencing the same `ServiceInterfaceDeployment` and these `AdaptivePlatformServiceInstance`s contain aggregated `End2EndMethodProtectionProps` and/or `End2EndEventProtectionProps` then the `E2EProfileConfiguration`s that are referenced by the `End2EndMethodProtectionProps` and `End2EndEventProtectionProps` shall have the same `profileName` defined **at the time when the creation of the manifest is finished**.⌋*()*

In other words it is not allowed to protect different `AdaptivePlatformServiceInstance`s of the same `ServiceInterfaceDeployment` with different E2E Profiles. Please note that the `End2EndMethodProtectionProps` and/or `End2EndEventProtectionProps` in the different `AdaptivePlatformServiceInstance`s are allowed to reference different `E2EProfileConfiguration`s since the same E2E Profile may be configured with different E2E settings.

11.5 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.

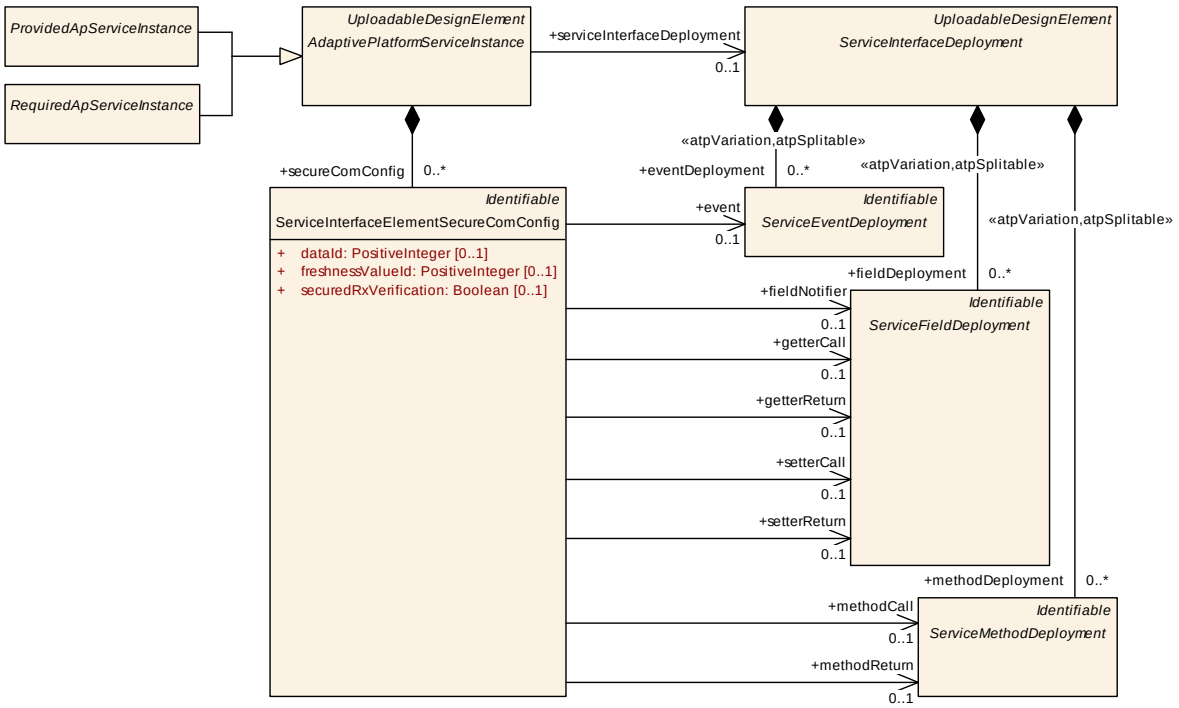


Figure 11.33: Secure Communication

[TPS_MANI_03133]{DRAFT} **Usage of `ServiceInterfaceElementSecureComConfig`** ⌈The `ServiceInterfaceElementSecureComConfig` element is used to define `ServiceInterface` element specific secure communication configuration settings in the context of an `AdaptivePlatformServiceInstance`.⌋*(RS_MANI_00036)*

The modeling allows protecting selected elements of a `ServiceInterface`, like particular `event`s or `method`s.

Since the `ServiceInterfaceElementSecureComConfig` meta-class is aggregated by the abstract `AdaptivePlatformServiceInstance` it can be used to configure the secure communication on specific derived classes like `ProvidedSomeipServiceInstance` or `RequiredSomeipServiceInstance` that fit the underlying middleware. With this approach it is possible to define different communication security protections for different used transport layer mechanisms in case of Multi-Binding.

Class	**ServiceInterfaceElementSecureComConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	This element allows to secure the communication of the referenced ServiceInterface element.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AdaptivePlatformServiceInstance*.secureComConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataId	PositiveInteger	0..1	attr	This attribute defines a unique numerical identifier for the referenced ServiceInterface element.
event	ServiceEvent Deployment	0..1	ref	Reference to an event that is protected by a security protocol.
fieldNotifier	ServiceField Deployment	0..1	ref	Reference to a field notifier that is protected by a security protocol.
freshnessValue Id	PositiveInteger	0..1	attr	This attribute defines the Id of the Freshness Value.
getterCall	ServiceField Deployment	0..1	ref	Reference to a field getter call message that is protected by a security protocol.
getterReturn	ServiceField Deployment	0..1	ref	Reference to a field getter return message that is protected by a security protocol.
methodCall	ServiceMethod Deployment	0..1	ref	Reference to a method call message that is protected by a security protocol.
methodReturn	ServiceMethod Deployment	0..1	ref	Reference to a method return message that is protected by a security protocol.
securedRx Verification	Boolean	0..1	attr	This attribute defines whether the ServiceInterface element shall verify its security credentials during reception.
setterCall	ServiceField Deployment	0..1	ref	Reference to a field setter call message that is protected by a security protocol.
setterReturn	ServiceField Deployment	0..1	ref	Reference to a field setter return message that is protected by a security protocol.

Table 11.74: ServiceInterfaceElementSecureComConfig

[constr_3391]{DRAFT} **ServiceInterfaceElementSecureComConfig references to ServiceInterfaceDeployment elements** ⌈ServiceInterfaceElementSecureComConfig element shall be defined for exactly one ServiceInterface element and shall therefore contain only one single reference to an element defined in the scope of a ServiceInterfaceDeployment **at the time when the creation of the manifest is finished**.⌋*()*

The attributes in the ServiceInterfaceElementSecureComConfig meta-class are defining security configuration settings that are specific for the referenced ServiceInterface element in the context of an AdaptivePlatformServiceInstance. The used security protocol is defined in the ServiceInstanceToMachineMapping.

[TPS_MANI_03199]{DRAFT} **Endpoint protection by SecureComProps** ⌈The ServiceInstanceToMachineMapping allows to assign a security protocol configuration settings that are defined in the referenced SecureComProps meta-class to protect endpoints that are defined by the Transport Protocol, Port and IP Address on which one or several AdaptivePlatformServiceInstances are provided or consumed.⌋ *(RS_MANI_00036)*

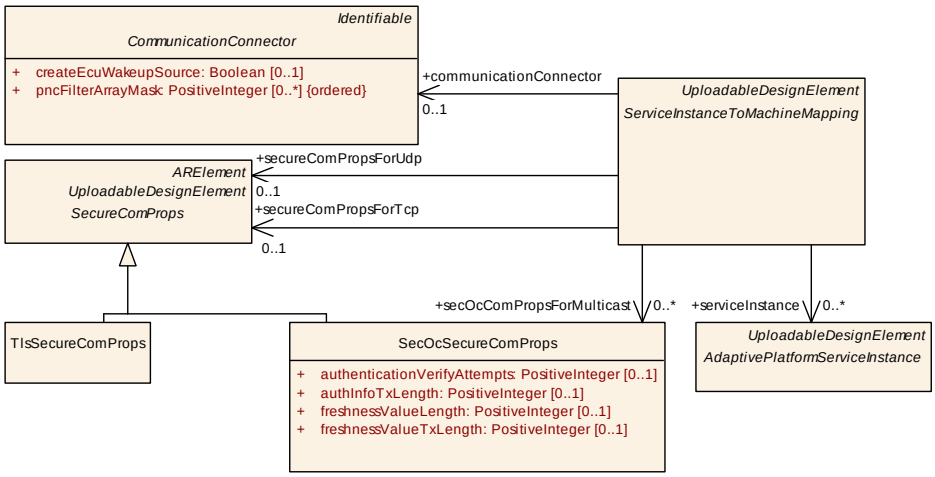


Figure 11.34: Security protocol configuration

[TPS_MANI_03200]{DRAFT} **SecureComProps for udp, tcp and multicast communication** ⌈The ServiceInstanceToMachineMapping allows to assign a security protocol configuration settings for:

- udp communication if the ServiceInstanceToMachineMapping refers to the SecureComProps in the role secureComPropsForUdp
- tcp communication in case the ServiceInstanceToMachineMapping refers to the SecureComProps in the role secureComPropsForTcp
- multicast communication in case the ServiceInstanceToMachineMapping refers to the SecOcSecureComProps in the role secOcComPropsForMulticast

⌋*(RS_MANI_00036)*

With this modeling approach it is possible to configure different security protocol settings for the communication over TCP, UDP and multicast. For example it is allowed to use different settings to protect the TCP communication of a AdaptivePlatformServiceInstance via TLS and UDP communication via DTLS.

Please note that protection of IP multicast traffic is only supported by SecOC and therefore the ServiceInstanceToMachineMapping refers directly the SecOcSecureComProps in the secOcComPropsForMulticast role.

Class	**SecureComProps** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	This meta-class defines a communication security protocol and its configuration settings.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	DdsSecureComProps, SecOcSecureComProps, TlsSecureComProps			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 11.75: SecureComProps

11.5.1 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.

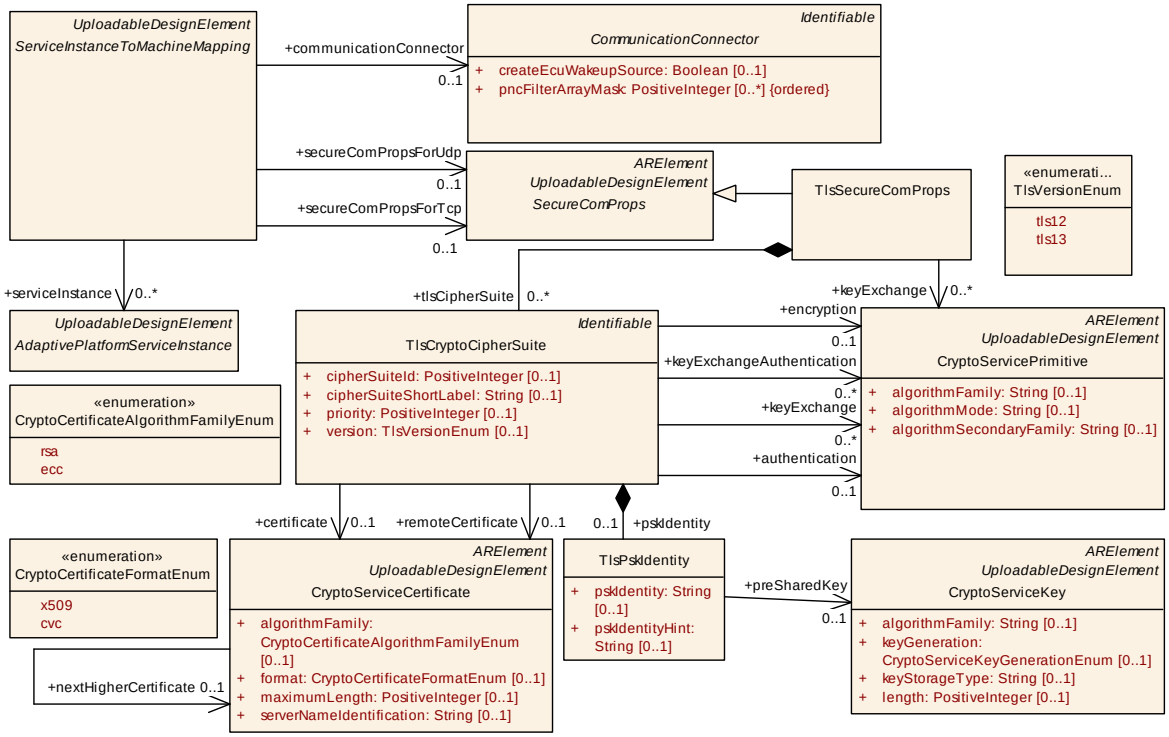


Figure 11.35: Secure Communication over TLS

It is a common use case that only one end of a TLS-based connection is actually modeled in an AUTOSAR model. It is therefore important that the modeling does not rely on or imply knowledge about both ends of such a TLS-based connection.

An AUTOSAR model that only describes one end of the communication is positively required to work, independently of the availability of a formal modeling of the other end.

Class	**TlsSecureComProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	Configuration of the Transport Layer Security protocol (TLS). **Tags:** atp.recommendedPackage=SecureComProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *SecureComProps*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
keyExchange	CryptoServicePrimitive	*	ref	This reference identifies the shared (i.e. applicable for each of the aggregated cipher suites) crypto service primitive for the execution of key exchange during the handshake phase.
tlsCipherSuite	TlsCryptoCipherSuite	*	aggr	Collection of supported cipher suites that are used to negotiate the security settings for a network connection defined by the ServiceInstanceToMachineMapping.

Table 11.76: TlsSecureComProps

Enumeration	**CryptoServiceKeyGenerationEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This enumeration shall be taken to express the handling of a crypto key in terms of whether it is obtained from e.g. a diagnostic tester or whether it is created by derivation from a master key.
Aggregated by	CryptoServiceKey.keyGeneration
Literal	***Description***
keyDerivation	This means that the crypto key is created by derivation from a master key. **Tags:** atp.EnumerationLiteralIndex=0
keyStorage	This means that the crypto key is obtained from an external entity, e.g. a diagnostic tester. **Tags:** atp.EnumerationLiteralIndex=1

Table 11.77: CryptoServiceKeyGenerationEnum

TLS is composed of the TLS Record Protocol and the TLS Handshake Protocol. The Record Protocol provides connection security and encrypts and authenticate packets. The record layer functions can be called at any time after the handshake process is finished, when there is need to receive or send data.

The Handshake Protocol allows the server and client to authenticate each other and to negotiate encryption algorithms and cryptographic keys before any data is exchanged.

In order to establish a cryptographically secure data channel, the communication partners in form of `ServiceInstanceToMachineMapping`s shall agree on ciphersuites and on keys that will be used to encrypt the data.

The client sends a list of supported ciphersuites to the server. The server decides on a ciphersuite from the list provided by the client, and continues with the handshake. Please note that the server and client roles cannot be swapped while the connection exists, i.e. a *server* remains the *server* for the full amount of time the connection exists.

[TPS_MANI_03213]{DRAFT} **Semantics of meta-class `TlsSecureComProps`** ⌈As a sub-class of `SecureComProps`, meta-class `TlsSecureComProps` has the ability to collect the TLS-related configuration aspects from either the perspective of the client or the server.

In the case of TLS, the collection boils down to the aggregation of meta-class `TlsCryptoCipherSuite` in the role `tlsCipherSuite` plus the ability (by means of the role `keyExchange`) to define handshake properties that are shared for each of the aggregated `tlsCipherSuite`.⌋*(RS_MANI_00036)*

[constr_5047]{DRAFT} **Supported values of `TlsSecureComProps.category`** ⌈The only supported values of attribute `TlsSecureComProps.category` are:

- **`TLS_SERVER`**: the `TlsSecureComProps` assumes the role of the *server* in the TLS connection.
- **`TLS_CLIENT`**: the `TlsSecureComProps` assumes the role of the *client* in the TLS connection.

This rule shall be imposed **at the time when the creation of the manifest is finished**.

⌋*()*

[TPS_MANI_03134]{DRAFT} **Configuration of supported TLS ciphersuites** ⌈The creation of a TLS connection requires the usage of a suite of cryptographic operations in specific roles, also known as a *cipher suite*.

Meta-class `TlsCryptoCipherSuite` represents a given cipher suite for a TLS connection. `TlsCryptoCipherSuite` references meta-class `CryptoServicePrimitive` in three dedicated roles that represent the steps of the creation of a TLS connection.

More specifically, the cryptographic operations for setting up a TLS connection involve the following steps:

- **Key exchange**: these `CryptoServicePrimitive`s may be used for the handshake phase of the TLS connection. Different alternatives exist for executing this phase and therefore the multiplicity of this reference is 0..*.
- **Authentication** of communication partners during the operational phase of the TLS connection. For this purpose a single `CryptoServicePrimitive` is used on each end of the communication.
- **Encryption** of content exchanged between the communication partners that have established the TLS connection. For this purpose a single `CryptoServicePrimitive` is used on each end of the communication.

⌋(*RS_MANI_00036*)

Please note that according to TLS each `TlsCryptoCipherSuite` may support multiple PSKs or certificates. The model restricts the multiplicity of references from `TlsCryptoCipherSuite` to `CryptoServiceCertificate` and `TlsPskIdentity` to 0..1. A TLS configuration with multiple PSKs or Certificates per CipherSuite can be achieved by having multiple `TlsCryptoCipherSuite`'s that share the same set of parameters but reference different `CryptoServiceCertificate`'s or `TlsPskIdentity`'s, respectively.

Class	**TlsCryptoCipherSuite**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class represents a cipher suite for describing cryptographic operations in the context of establishing a connection of ApplicationEndpoints that is protected by TLS.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	TlsCryptoServiceMapping.tlsCipherSuite, TlsSecureComProps.tlsCipherSuite			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authentication	CryptoServicePrimitive	0..1	ref	This reference identifies the crypto service primitive for the generation and verification of MACs.
certificate	CryptoService Certificate	0..1	ref	This reference identifies the applicable local certificate.
cipherSuiteId	PositiveInteger	0..1	attr	Identification of the CipherSuite according to the IANA assignments list.
cipherSuite ShortLabel	String	0..1	attr	Name of the CipherSuite according to the IANA assignments list.
ellipticCurve	CryptoEllipticCurve Props	*	ref	This references point to the properties of elliptic curves.
encryption	CryptoServicePrimitive	0..1	ref	This reference identifies the crypto service primitive for the execution of encryption.
keyExchange	CryptoServicePrimitive	*	ref	This reference identifies the individual (i.e. per cipher suite) crypto service primitive for the execution of key exchange during the handshake phase.
keyExchange Authentication	CryptoServicePrimitive	*	ref	This reference identifies the crypto service primitives for the generation and verification of signatures during the key exchange algorithm.
priority	PositiveInteger	0..1	attr	This attribute identifies the priority of the cipher suite. Range: 1..65535. Lower values represent higher priorities.
props	TlsCryptoCipherSuite Props	0..1	aggr	The aggregated TlsCryptoCipherSuiteProps provide details for the TLS Cipher Suite.
pskIdentity	TlsPskIdentity	0..1	aggr	Pre-shared key identity shared during the handshake among the communication parties, to establish a TLS connection if the handshake is based on the existence of a pre-shared key.
remote Certificate	CryptoService Certificate	0..1	ref	This reference identifies the applicable remote certificate.
signature Scheme	CryptoSignature Scheme	*	ref	This reference points to the properties of a TLS Signature Scheme.
version	TlsVersionEnum	0..1	attr	This attribute supports the definition of the applicable version of TLS.

Table 11.78: TlsCryptoCipherSuite

Class	CryptoServicePrimitive			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class has the ability to represent a crypto primitive. **Tags:** atp.recommendedPackage=CryptoPrimitives			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
algorithmFamily	String	0..1	attr	This attribute represents a description of the family (e.g. AES) of crypto algorithm implemented by the crypto primitive.
algorithmMode	String	0..1	attr	This attribute represents a description of the mode of the crypto algorithm implemented by the crypto primitive.
algorithm Secondary Family	String	0..1	attr	This attribute represents a further description of the secondary family of crypto algorithm implemented by the crypto primitive. The secondary family is needed for the specification of the hash algorithm for a signature check, e.g. using RSA.

Table 11.79: CryptoServicePrimitive

Class	CryptoEllipticCurveProps			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class provides attributes to specify the properties of elliptic curves. **Tags:** atp.recommendedPackage=CryptoEllipticCurveProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
namedCurveId	PositiveInteger	0..1	attr	Defines the value of one specific NamedCurve Id.

Table 11.80: CryptoEllipticCurveProps

Class	TlsCryptoCipherSuiteProps			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class provides attributes to specify details of TLS Cipher Suites.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	TlsCryptoCipherSuite.props			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
tcpIpTlsUse Security ExtensionForce EncryptThen Mac	Boolean	0..1	attr	Defines if the security extension according to IETF RFC 7366 shall be supported. This is useful for cipher suites using CBC mode.

Table 11.81: TlsCryptoCipherSuiteProps

Class	**CryptoSignatureScheme**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class provides attributes to specify the TLS Signature Scheme. **Tags:** atp.recommendedPackage=CryptoSignatureSchemas			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
signature SchemeId	PositiveInteger	0..1	attr	Defines the value of one specific TLS Signature Scheme.

Table 11.82: CryptoSignatureScheme

Enumeration	**TlsVersionEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This meta-class has the ability to identify a specific version of the transport-layer security (TLS) protocol.
Aggregated by	TlsCryptoCipherSuite.version
Literal	***Description***
tls12	TLS version 1.2 **Tags:** atp.EnumerationLiteralIndex=0
tls13	TLS version 1.3 **Tags:** atp.EnumerationLiteralIndex=2

Table 11.83: TlsVersionEnum

Class	**CryptoServiceKey**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class has the ability to represent a crypto key. **Tags:** atp.recommendedPackage=CryptoDevelopmentKeys			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
algorithmFamily	String	0..1	attr	This attribute represent the description of the family of the applicable crypto algorithm.
developmentValue	ValueSpecification	0..1	aggr	This aggregation represents the ability to assign a specific value to the crypto key as part of the system description. This value can then be taken for the development of the respective ECU.
keyGeneration	CryptoServiceKeyGenerationEnum	0..1	attr	This attribute describes how a the specific cryptographic key is created.
keyStorageType	String	0..1	attr	This attribute describes where the enclosing cryptographic key shall be stored. AUTOSAR reserves specific values for this attributes but it is possible to insert custom values as well.
length	PositiveInteger	0..1	attr	This attribute describes the length of the cryptographic key in bits.

Table 11.84: CryptoServiceKey

[TPS_MANI_03214]{DRAFT} **Existence of `TlsCryptoCipherSuite.keyExchange` vs. `TlsSecureComProps.keyExchange`** ⌈The role `TlsSecureComProps.keyExchange` has been introduced as an optimization.

It is assumed that the references for key exchange look pretty similar if not identical for many concrete `TlsCryptoCipherSuite`s.

Adding these references in an identical form to a bunch of `TlsCryptoCipherSuite`s does not really make sense. Therefore, `TlsSecureComProps` allows to define these references as well with the intention to make them valid for all `TlsSecureComProps.tlsCipherSuite`s.

A mixture of references in the role `TlsCryptoCipherSuite.keyExchange` and `TlsSecureComProps.keyExchange` is supported.⌋*(RS_MANI_00036)*

[TPS_MANI_03215]{DRAFT} **Semantics of `CryptoServiceCertificate`** ⌈Meta-class `CryptoServiceCertificate` represents a cryptographic certificate needed for the creation of a TLS connection between *server* and *client*.⌋*(RS_MANI_00036)*

Class	**CryptoServiceCertificate**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class represents the ability to model a cryptographic certificate. **Tags:** atp.recommendedPackage=CryptoServiceCertificates			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
algorithmFamily	CryptoCertificate AlgorithmFamilyEnum	0..1	attr	This attribute represents a description of the family of crypto algorithm used to generate public key and signature of the cryptographic certificate.
format	CryptoCertificateFormat Enum	0..1	attr	This attribute can be used to provide information about the format used to create the certificate
maximum Length	PositiveInteger	0..1	attr	This attribute represents the ability to define the maximum length of the certificate in bytes.
nextHigher Certificate	CryptoService Certificate	0..1	ref	The reference identifies the next higher certificate in the certificate chain.
serverName Identification	String	0..1	attr	Server Name Indication (SNI) is needed if the IP address hosts multiple servers (on the same port), each of them using a different certificate. If the client sends the SNI to the Server in the client hello, the server looks the SNI up in its certificate list and uses the certificate identified by the SNI.

Table 11.85: CryptoServiceCertificate

Enumeration	**CryptoCertificateAlgorithmFamilyEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This meta-class defies possible cryptographic algorithm families used to create public keys and signatures within the certificate.
Aggregated by	CryptoServiceCertificate.algorithmFamily
Literal	***Description***

▽

△

Enumeration	**CryptoCertificateAlgorithmFamilyEnum**
ecc	The cryptographic operations in the certificate are executed using elliptic curves (ecc) **Tags:** atp.EnumerationLiteralIndex=2
rsa	The cryptographic operations in the certificate are executed using the RSA approach. **Tags:** atp.EnumerationLiteralIndex=1

Table 11.86: CryptoCertificateAlgorithmFamilyEnum

Enumeration	**CryptoCertificateFormatEnum**
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This meta-class defines possible formats of cryptographic certificates.
Aggregated by	CryptoServiceCertificate.format
Literal	***Description***
cvc	The certificate has been created in Card Verifiable Certificate (CVC) format **Tags:** atp.EnumerationLiteralIndex=2
x509	The certificate is created in X.509 format. **Tags:** atp.EnumerationLiteralIndex=1

Table 11.87: CryptoCertificateFormatEnum

[constr_5048]{DRAFT} **Existence of `TlsCryptoCipherSuite.certificate` and `TlsCryptoCipherSuite.pskIdentity` in the *server* role** ⌈**Either**

- the reference to `CryptoServiceCertificate` in the role `TlsCryptoCipherSuite.certificate`
- the aggregation of `TlsPskIdentity` in the role `TlsCryptoCipherSuite.pskIdentity`

shall exist if the `TlsCryptoCipherSuite` is aggregated by `TlsSecureComProps` that has the attribute `category` set to the value `TLS_SERVER` **`at the time when the creation of the manifest is finished`**.⌋*()*

In other words two different approaches are supported by TLS for the handling of key compromise: Pre-shared secret and certificate.

The server may optionally request a certificate from the *client*. If this option is not used then other documented approaches for completing the handshake phase is foreseen for the specific case.

[TPS_MANI_03216]{DRAFT} **Existence of `TlsCryptoCipherSuite.certificate` and `TlsCryptoCipherSuite.pskIdentity` in the *client* role** ⌈The client (`TlsSecureComProps` has set the value of attribute `category` to `TLS_CLIENT`) has the following authentication options:

- the reference to `CryptoServiceCertificate` in the role `TlsCryptoCipherSuite.certificate` exists,
- the aggregation of `TlsPskIdentity` in the role `TlsCryptoCipherSuite.pskIdentity` exists,

- neither one nor the other exists. In this case the handshake is provided on the basis of the server certificate only.

⌋(*RS_MANI_00036*)

In the pre-shared Key approach the client indicates which key to use by including a `pskIdentity` in the ClientKeyExchange message. To help the client in selecting which identity to use, the server can provide a `pskIdentityHint` in the ServerKeyExchange message. Please note that the usage of `pskIdentityHint`s is restricted for usage with TLS 1.2.

[TPS_MANI_03137]{DRAFT} **`ServiceInterfaceElementSecureComConfig` is not relevant in case of TLS communication** ⌈The element `ServiceInterfaceElementSecureComConfig` is not relevant in case of TLS communication.⌋ *(RS_MANI_00036)*

[constr_3485]{DRAFT} **UDP endpoint using DTLS SERVER role can only serve provided service instances** ⌈A `ServiceInstanceToMachineMapping` that refers to `TlsSecureComProps` in the role `secureComPropsForUdp` is only allowed to reference `ProvidedApServiceInstance`s in the role `serviceInstance` if the `TlsSecureComProps` has the category `TLS_SERVER` **at the time when the creation of the manifest is finished**.⌋*()*

[constr_3486]{DRAFT} **TCP endpoint using TLS SERVER role can only serve provided service instances** ⌈A `ServiceInstanceToMachineMapping` that refers to `TlsSecureComProps` in the role `secureComPropsForTcp` is only allowed to reference `ProvidedApServiceInstance`s in the role `serviceInstance` if the `TlsSecureComProps` has the category `TLS_SERVER` **at the time when the creation of the manifest is finished**.⌋*()*

[constr_5260]{DRAFT} **UDP endpoint using DTLS CLIENT role can only serve required service instances** ⌈A `ServiceInstanceToMachineMapping` that refers to `TlsSecureComProps` in the role `secureComPropsForUdp` is only allowed to reference `RequiredApServiceInstance`s in the role `serviceInstance` if the `TlsSecureComProps` has the category `TLS_CLIENT` **at the time when the creation of the manifest is finished**.⌋*()*

[constr_5261]{DRAFT} **TCP endpoint using TLS CLIENT role can only serve required service instances** ⌈A `ServiceInstanceToMachineMapping` that refers to `TlsSecureComProps` in the role `secureComPropsForTcp` is only allowed to reference `RequiredApServiceInstance`s in the role `serviceInstance` if the `TlsSecureComProps` has the category `TLS_CLIENT` **at the time when the creation of the manifest is finished**.⌋*()*

The reason for [constr_3485], [constr_3486], [constr_5260], [constr_5261] is that the (D)TLS client needs to establish the (D)TLS connection and a TCP/UDP endpoint that is described by the `ServiceInstanceToMachineMapping` can only take one role: (D)TLS client or (D)TLS server. If a `ServiceInstanceToMachineMapping` would act as (D)TLS client and would refer to a `ProvidedApServiceInstance` then this (D)TLS client would need to establish the (D)TLS connection. But in this case

the (D)TLS client would not know to which remote service client a connection needs to be established since different RequiredApServiceInstances may directly call methods of the ProvidedApServiceInstance without any registration.

The same issue exists if the ServiceInstanceToMachineMapping acts as (D)TLS server and refers to RequiredApServiceInstances. The (D)TLS client needs to establish the (D)TLS connection before any messages are exchanged. But the remote service provider has no knowledge that this service consumer wants to call methods over a (D)TLS connection.

11.5.2 Secure Communication over SecOC

AUTOSAR Secure Onboard Communication (SecOC) supports symmetric and asymmetric authentication approaches. To configure the SecOC secure protection of a message by a MAC or Signature the ServiceInterfaceElementSecureComConfig needs to be defined. This element contains the configuration settings for the individual ServiceInterface elements. In addition, the ServiceInstanceToMachineMapping needs to point to SecOcSecureComProps to configure the endpoint protection that is defined by the Transport Protocol, Port and IP Address.

[constr_3392]{DRAFT} **ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are mandatory in case of SecOC communication** ⌈The attributes ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are mandatory in case of SecOC communication **at the time when the creation of the manifest is finished**.⌋ *()*

[TPS_MANI_03664]{DRAFT} **ServiceInterfaceElementSecureComConfig.securedRxVerification** ⌈The attribute ServiceInterfaceElementSecureComConfig.securedRxVerification defines whether a received SecOC message shall be verified with respect to the security credentials or not.

If the attribute ServiceInterfaceElementSecureComConfig.securedRxVerification is set to false, then security verification shall not be applied and the message payload shall be forwarded to the receivers.

If the attribute ServiceInterfaceElementSecureComConfig.securedRxVerification is set to true or is not defined, then security verification shall be applied.⌋ *(RS_MANI_00036)*

[constr_3691]{DRAFT} **Existence of ServiceInterfaceElementSecureComConfig.securedRxVerification** ⌈The attribute ServiceInterfaceElementSecureComConfig.securedRxVerification shall only be defined for a ServiceInterfaceElementSecureComConfig with the following definitions:

- The ServiceInterfaceElementSecureComConfig is aggregated by a RequiredApServiceInstance and defines at least one of the following roles:

  - ServiceInterfaceElementSecureComConfig.event
  - ServiceInterfaceElementSecureComConfig.fieldNotifier
  - ServiceInterfaceElementSecureComConfig.getterReturn
  - ServiceInterfaceElementSecureComConfig.setterReturn
  - ServiceInterfaceElementSecureComConfig.methodReturn
- The ServiceInterfaceElementSecureComConfig is aggregated by a ProvidedApServiceInstance and defines at least one of the following roles:
  - ServiceInterfaceElementSecureComConfig.getterCall
  - ServiceInterfaceElementSecureComConfig.setterCall
  - ServiceInterfaceElementSecureComConfig.methodCall

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

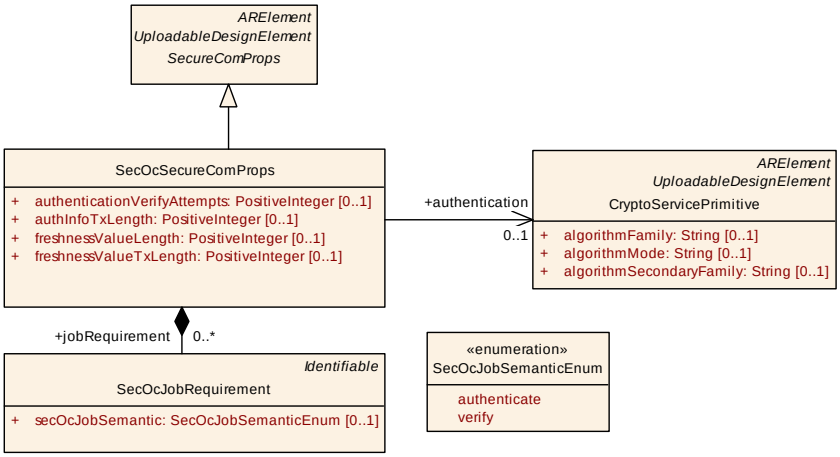


Figure 11.36: Secure Communication over SecOC

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

Class	**SecOcSecureComProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	Configuration of AUTOSAR SecOC. **Tags:** atp.recommendedPackage=SecureComProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *SecureComProps*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authentication	CryptoServicePrimitive	0..1	ref	This reference defines the authentication algorithm used for MAC generation and verification.

△

Class	**SecOcSecureComProps**			
authentication VerifyAttempts	PositiveInteger	0..1	attr	This attribute defines the additional number of authentication attempts that are to be carried out when the generation of the authentication information failed for a given message. If zero is set than only one authentication attempt is done.
authInfoTx Length	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.
freshnessValue Length	PositiveInteger	0..1	attr	This attribute defines the complete length in bits of the Freshness Value.
freshnessValue TxLength	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.
jobRequirement	SecOcJobRequirement	*	aggr	Collection of cryptographic job requirements.

Table 11.88: SecOcSecureComProps

[TPS_MANI_03139]{DRAFT} **Standardized SecOC Security Profiles** ⌈The SecOC security profile that is defined by `SecOcSecureComProps.category` can have the following values that are standardized by AUTOSAR: `PROFILE_01`, `PROFILE_02`, `PROFILE_03`.⌋*(RS_MANI_00036)*

The attribute values for the predefined categories mentioned in [TPS_MANI_03139] are defined in [constr_3325] in [18].

[constr_5347]{DRAFT} **Supported value range for attribute `SecOcSecureComProps.authenticationVerifyAttempts`** ⌈The supported value range of attribute `SecOcSecureComProps.authenticationVerifyAttempts` is limited to the interval [0..65535] **`at the time when the creation of the manifest is finished`**.⌋*()*

[TPS_MANI_03140]{DRAFT} **Non-Standardized SecOC Security Profiles** ⌈The values for the `SecOcSecureComProps.category` mentioned in [TPS_MANI_03139] are standardized and reserved for being used in the way the AUTOSAR standard foresees. In addition, it is positively possible to use other than the standardized values for the `SecOcSecureComProps.category`.⌋*(RS_MANI_00036)*

With the `SecOcJobRequirement` the cryptographic routines can be selected that need to be supported. In case of SecOC it can be selected whether the symmetric and/or asymmetric authentication approach is needed.

Class	**SecOcJobRequirement**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	Requirements for the cryptographic job that need to be executed.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SecOcSecureComProps.jobRequirement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
secOcJob Semantic	SecOcJobSemantic Enum	0..1	attr	This attribute defines the cryptographic algorithm that needs to be supported.

Table 11.89: SecOcJobRequirement

[constr_10227]{DRAFT} **Multiplicity of attribute SecOcJobRequirement.secOcJobSemantic** ⌈For each SecOcJobRequirement, the attribute secOcJobSemantic shall exist **at the time when the creation of the manifest is finished**.⌋*()*

Enumeration	**SecOcJobSemanticEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication
Note	List of cryptographic routines supported by SecOC.
Aggregated by	SecOcJobRequirement.secOcJobSemantic
Literal	***Description***
authenticate	Authentication algorithm for Authenticator generation/verification. **Tags:** atp.EnumerationLiteralIndex=0
verify	Asymmetric cryptographic algorithm to generate/verify a signature **Tags:** atp.EnumerationLiteralIndex=1

Table 11.90: SecOcJobSemanticEnum

11.5.3 Secure Communication over DDS

The configuration of participant identities for the DDS Security [35] plugins is supported by the DdsSecureComProps meta-class, which is a specialization of SecureComProps.

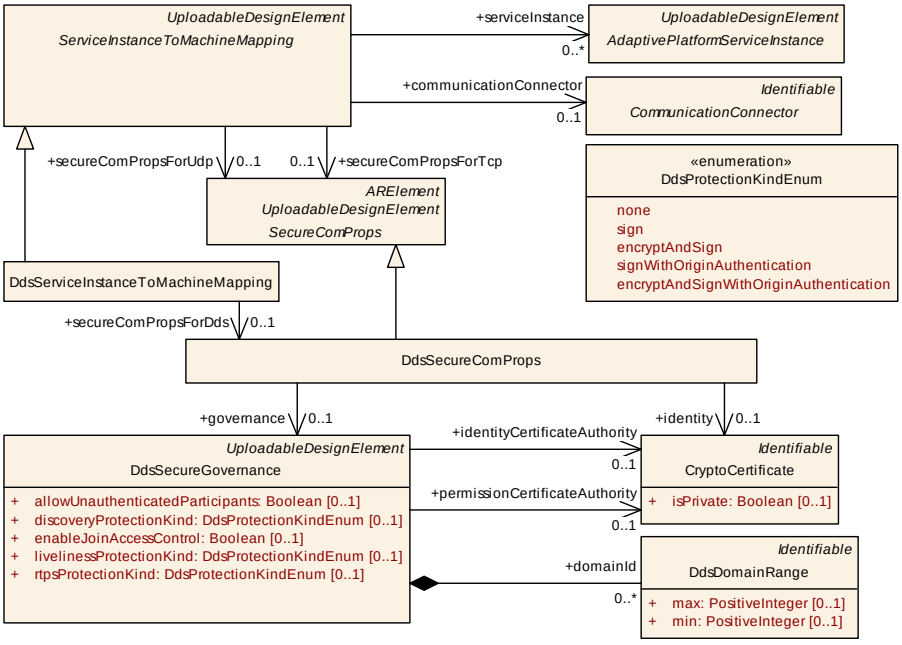


Figure 11.37: DdsSecureComProps at the DdsServiceInstanceToMachineMapping

Please note the following classes and constraints depict a structural specification of DDS Security deployment configuration. For functional details please refer to section "Secure Communication" of SWS_CommunicationManagement [10].

These modeling elements affect only DDS Security (as defined by [35]) deployments via DdsServiceInstanceToMachineMapping.secureComPropsForDds. For DDS Transport Security over TCP (TLS) and UDP (DTLS) configuration please refer to ServiceInstanceToMachineMapping.secureComPropsForTcp and ServiceInstanceToMachineMapping.secureComPropsForUdp, respectively.

Class	**DdsSecureComProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceMapping			
Note	Identity and governance information of participants in case of DDS Security. **Tags:** atp.recommendedPackage=SecureComProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *SecureComProps*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
governance	DdsSecureGovernance	0..1	ref	This attribute defines general DDS Security communication properties applicable to the DDS domain(s) in which the subject operates. **Tags:** atp.Status=candidate
identity	CryptoCertificate	0..1	ref	This attribute defines the cryptographic identity of the subject.

Table 11.91: DdsSecureComProps

[constr_3678]{DRAFT} **Existence of attributes for DdsSecureComProps** ⌈The following attributes of DdsSecureComProps shall exist **at the time when the creation of the manifest is finished**

- identity
- governance

⌋()

[TPS_MANI_03661]{DRAFT} **Configuration of Governance in DDS Security** ⌈The DdsSecureGovernance meta-class defines domain-wide rules determining relevant certificate authorities (for participant identities and permission manifests), constraints related to how participants may join such domain(s) and protection levels associated to each kind of communication (none, sign, encrypt+sign, with or without origin authentication).⌋*(RS_MANI_00036)*

Class	**DdsSecureGovernance**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication
Note	Configuration of DDS Security for all applications joining a specific set of DDS Domains. **Tags:** atp.Status=candidate atp.recommendedPackage=DdsSecureGovernances
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*
Aggregated by	ARPackage.element

▽

△

Class	**DdsSecureGovernance**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allowUnauthen-ticated Participants	Boolean	0..1	attr	Defines whether unauthenticated participants can join this domain. **Tags:** atp.Status=candidate
discovery ProtectionKind	DdsProtectionKind Enum	0..1	attr	Defines the kind of cryptographic transformation to apply in DDS discovery communication. **Tags:** atp.Status=candidate
domainId	DdsDomainRange	*	aggr	Set of domains to be covered by this property set. **Tags:** atp.Status=candidate
enableJoin AccessControl	Boolean	0..1	attr	Defines whether access control is to be enforced upon joining this domain. **Tags:** atp.Status=candidate
identity Certificate Authority	CryptoCertificate	0..1	ref	Certificate representing the identity certificate authority applicable to the domain(s) specified by domainsIds. **Tags:** atp.Status=candidate
liveliness ProtectionKind	DdsProtectionKind Enum	0..1	attr	Defines the kind of cryptographic transformation to apply in DDS liveliness communication. **Tags:** atp.Status=candidate
permission Certificate Authority	CryptoCertificate	0..1	ref	Certificate representing the permissions certificate authority applicable to the domain(s) specified by domainsIds. **Tags:** atp.Status=candidate
rtpsProtection Kind	DdsProtectionKind Enum	0..1	attr	Defines the kind of cryptographic transformation to apply to whole DDS RTPS. **Tags:** atp.Status=candidate

Table 11.92: DdsSecureGovernance

Class	**DdsDomainRange**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication			
Note	DDS Domain ID range.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DdsSecureGovernance.domainId			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
max	PositiveInteger	0..1	attr	Upper bound of the DdsDomainRange.
min	PositiveInteger	0..1	attr	Lower bound of the DdsDomainRange.

Table 11.93: DdsDomainRange

Enumeration	**DdsProtectionKindEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::SecureCommunication
Note	Supported cryptographic transformations (extended).
Aggregated by	DdsSecureGovernance.discoveryProtectionKind, DdsSecureGovernance.livelinessProtectionKind, DdsSecureGovernance.rtpsProtectionKind, DdsTopicAccessRule.dataProtectionKind, DdsTopic AccessRule.metadataProtectionKind
Literal	***Description***

▽

△

Enumeration	DdsProtectionKindEnum
encryptAndSign	encryption and MAC transformations (in that precise order) are applied **Tags:** atp.EnumerationLiteralIndex=2
encryptAndSign WithOrigin Authentication	similar to "EncryptAndSign" but with additional authentication codes produced under different secret keys, which prevents receiving peers from impersonating a specific sender **Tags:** atp.EnumerationLiteralIndex=4
none	no transformation is applied **Tags:** atp.EnumerationLiteralIndex=0
sign	Message Authentication Code (MAC) is applied, no encryption **Tags:** atp.EnumerationLiteralIndex=1
signWithOrigin Authentication	similar to "sign" but with additional authentication codes produced under different secret keys, which prevents receiving peers from impersonating a specific sender **Tags:** atp.EnumerationLiteralIndex=3

Table 11.94: DdsProtectionKindEnum

[constr_3679]{DRAFT} **Existence of attributes for `DdsSecureGovernance`** ⌈The following attributes of `DdsSecureGovernance` shall exist **`at the time when the creation of the manifest is finished`**

- at least one `domainId`
- `identityCertificateAuthority`
- `permissionCertificateAuthority`
- `allowUnauthenticatedParticipants`
- `enableJoinAccessControl`
- `discoveryProtectionKind`
- `livelinessProtectionKind`
- `rtpsProtectionKind`

⌋()

[constr_3682]{DRAFT} **Values of `DdsDomainRange.min` and `DdsDomainRange.max`** ⌈The value of `DdsDomainRange.min` shall be less than or equal to the value of `DdsDomainRange.max` **`at the time when the creation of the manifest is finished`**.⌋()

12 Raw Data Stream Manifest

12.1 Raw Data Stream Deployment

[TPS_MANI_01285]{DRAFT} **Purpose of meta-class `RawDataStreamDeployment`** ⌈Meta-class `RawDataStreamDeployment` has the ability to further qualify an existing `AbstractRawDataStreamInterface` on deployment level.⌋*(RS_MANI_00067)*

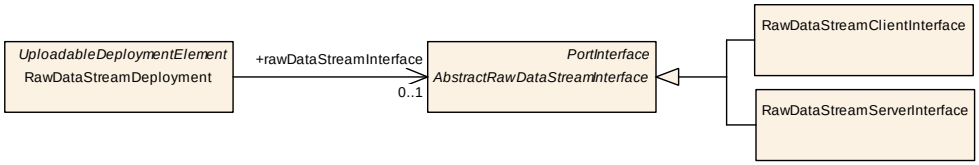


Figure 12.1: Modeling of the `RawDataStreamDeployment`

Class	**RawDataStreamDeployment**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::RawDataStreamMapping			
Note	This meta-class represents the ability to model deployment-level information for a raw data stream **Tags:** atp.recommendedPackage=RawDataStreamDeployments			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
rawDataStream Interface	AbstractRawData StreamInterface	0..1	ref	This reference identifies the corresponding RawData StreamInterface,

Table 12.1: RawDataStreamDeployment

12.2 Raw Data Stream Mapping

[TPS_MANI_01287]{DRAFT} **Semantics of `RawDataStreamMapping`** ⌈On the deployment side, the access to a raw data stream requires the provision of actual transport for the raw data.

In principle, it would be possible to implement the transport on top of various technologies.

Therefore, abstract meta-class `RawDataStreamMapping` exists to provide the principle ability to map to an `RPortPrototype` and to a `Process`.

The mapping to a concrete transport technology is left to sub-classes of `RawDataStreamMapping`.⌋*(RS_MANI_00067)*

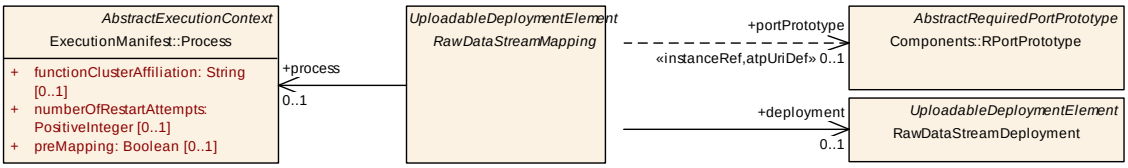


Figure 12.2: Modeling of the `RawDataStreamMapping`

Class	***RawDataStreamMapping*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::RawDataStreamMapping			
Note	This meta-class acts as an abstract base class for mapping raw data streams to the application software.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	*EthernetRawDataStreamMapping*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
deployment	RawDataStream Deployment	0..1	ref	This reference identifies the applicable RawDataStream Deployment.
portPrototype	RPortPrototype	0..1	iref	Reference to a specific PortPrototype that represents the raw data stream to the application. **Stereotypes:** atpUriDef **InstanceRef implemented by:** RPortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	Reference to the Process in which the Executable that contains the SoftwareComponent and the referenced Port Prototype is executed.

Table 12.2: RawDataStreamMapping

12.3 Raw Data Streams over Ethernet

The configuration of raw data streams over Ethernet is done by means of the sub-classes of meta-class `AbstractRawDataStreamEthernetCredentials` as well as meta-class `EthernetRawDataStreamMapping`.

The obvious requirement to the configuration of the Ethernet credentials is that both ends of the communication have to be configured sufficiently, such that a connection between the two ends can be established.

The modeling of the configuration of Ethernet credentials uses different concepts for the description of the **local** end vs. the description of the **remote** end, see [TPS_MANI_01354].

[TPS_MANI_01354]{DRAFT} **Rationale for the existence of meta-class `AbstractRawDataStreamEthernetCredentials`** ⌈On the *AUTOSAR adaptive platform*, the configuration of Ethernet credentials is typically modeled by means of the combination of `NetworkEndpoint` and `ApApplicationEndpoint`.

This approach is also used for the unicast raw data stream communication, but only for the **local** configuration of unicast Ethernet credentials.

The credentials of the **remote** end (unicast and multicast) are configured in a simplified way by means of sub-classes of meta-class AbstractRawDataStreamEthernet-Credentials.

This approach also supports the configuration of credentials for a remote end that is not running an AUTOSAR stack and that is therefore not contributing it's credentials inside an AUTOSAR model in a more approachable way.⌋ *(RS_MANI_00067)*

Please note that, from the point of view of the application software and in the case of raw data stream communication both communication ends, i.e. the client side **and** the server side are implemented using an RPortPrototype.

This (from an AUTOSAR perspective) seemingly counter-intuitive aspect is also depicted in Figure 12.4, i.e. RawDataStreamMapping does only define a reference to an RPortPrototype.

Meta-class AbstractRawDataStreamEthernetCredentials and its subclasses (see Figure 12.3) have the ability to define IP addresses (either V4 or V6) and transport protocol (UDP and/or TCP).

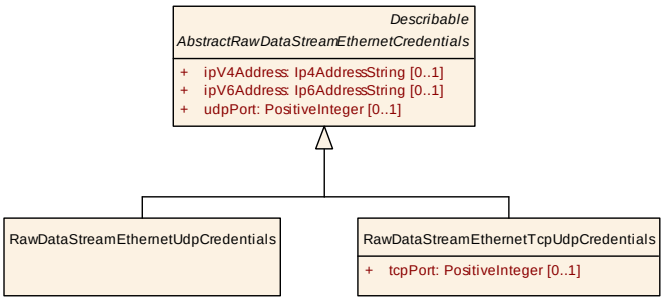


Figure 12.3: Modeling of the AbstractRawDataStreamEthernetCredentials

[constr_10076]{DRAFT} **Existence of RawDataStreamEthernetUdpCredentials.udpPort** ⌈In the context of RawDataStreamEthernetUdpCredentials, the attribute udpPort shall exists **at the time when the creation of the manifest is finished**.⌋ *()*

Class	***AbstractRawDataStreamEthernetCredentials*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::RawDataStreamMapping			
Note	This meta-class serves as an abstract base class for the configuration of network credentials.			
Base	*ARObject*, *Describable*			
Subclasses	RawDataStreamEthernetTcpUdpCredentials, RawDataStreamEthernetUdpCredentials			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ipV4Address	Ip4AddressString	0..1	attr	This attribute describes the IP V4 address of the remote server.
ipV6Address	Ip6AddressString	0..1	attr	This attribute describes the IP V6 address of the remote server.
udpPort	PositiveInteger	0..1	attr	This attribute represents the configuration of a UDP port number.

Table 12.3: AbstractRawDataStreamEthernetCredentials

Class	**RawDataStreamEthernetUdpCredentials**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::RawDataStreamMapping			
Note	This-meta-class represents the ability to create a configuration of network credentials for a raw data stream connection over UDP.			
Base	*ARObject*, *AbstractRawDataStreamEthernetCredentials*, *Describable*			
Aggregated by	EthernetRawDataStreamRemoteClientConfig.multicastCredentials, EthernetRawDataStreamRemote ClientConfig.unicastUdpCredentials, EthernetRawDataStreamRemoteServerConfig.multicastCredentials			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table 12.4: RawDataStreamEthernetUdpCredentials

Class	**RawDataStreamEthernetTcpUdpCredentials**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::RawDataStreamMapping			
Note	This-meta-class represents the ability to create a configuration of network credentials for a raw data stream connection over TCP and UDP (inherited from base class).			
Base	*ARObject*, *AbstractRawDataStreamEthernetCredentials*, *Describable*			
Aggregated by	EthernetRawDataStreamRemoteServerConfig.unicastCredentials			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
tcpPort	PositiveInteger	0..1	attr	This attribute represents the configuration of a TCP port number.

Table 12.5: RawDataStreamEthernetTcpUdpCredentials

[constr_10077]{DRAFT} **Existence of `ipV4Address` and `ipV6Address` within `AbstractRawDataStreamEthernetCredentials`** ⌈Within the context of a `AbstractRawDataStreamEthernetCredentials`, either the attribute `ipV4Address` or the attribute `ipV6Address` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10078]{DRAFT} **Existence of `RawDataStreamEthernetTcpUdpCredentials.tcpPort` and `udpPort`** ⌈In the context of a `RawDataStreamEthernetTcpUdpCredentials`, either the attribute `tcpPort` **or** `udpPort` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[TPS_MANI_01355]{DRAFT} **Definition of local Ethernet credentials** ⌈The definition of the **local** Ethernet credentials is done by means of the aggregation of meta-class `EthernetRawDataStreamLocalEndpointConfig` in the role `localEndpointConfig`.

`EthernetRawDataStreamLocalEndpointConfig`, in turn, defines the following aggregations for the configuration of local Ethernet credentials:

- a reference to `EthernetCommunicationConnector` in the role `localCommConnector`, that in turn refers to `NetworkEndpoint`.
- references to `ApApplicationEndpoint` in the roles
  - `EthernetRawDataStreamLocalEndpointConfig.localUdpPort`
  - `EthernetRawDataStreamLocalEndpointConfig.localTcpPort`

⌋*(RS_MANI_00067)*

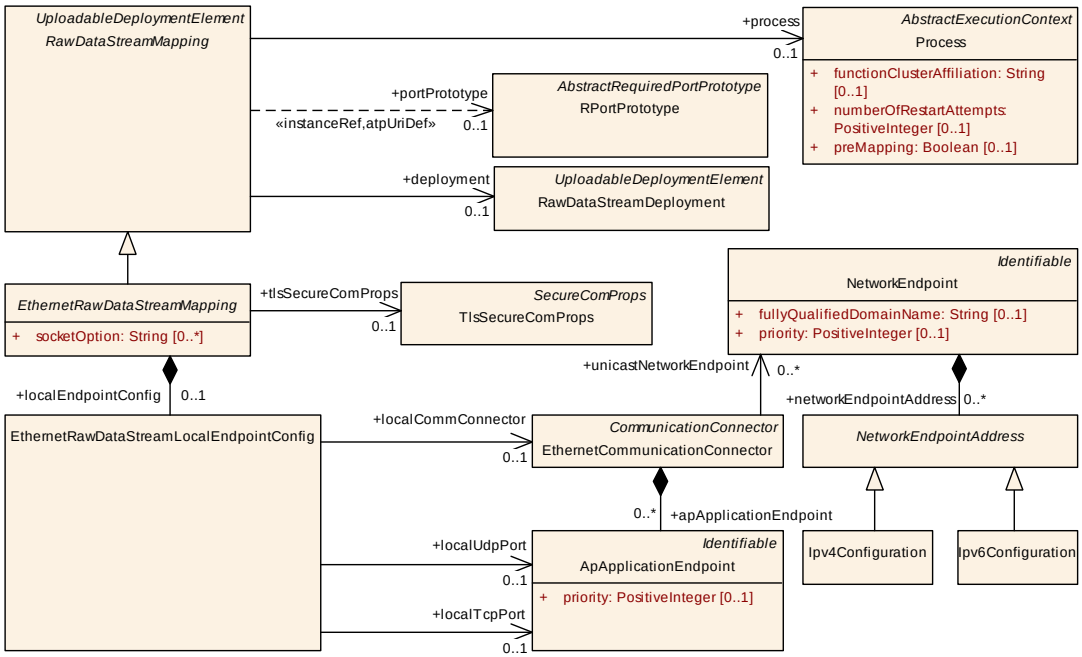


Figure 12.4: Modeling of the `EthernetRawDataStreamMapping`

Please note that the configuration of the corresponding remote end depends on the question whether the remote end represents a server (see [TPS_MANI_01356] in section 12.3.1) or a client (see [TPS_MANI_01357] in section 12.3.2).

Class	***EthernetRawDataStreamMapping*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::RawDataStreamMapping			
Note	This meta-class serves as the abstract bases class for the ability to map a PortPrototype to a Ethernet-based communication channel.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	EthernetRawDataStreamClientMapping, EthernetRawDataStreamServerMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
localEndpoint Config	EthernetRawData StreamLocalEndpoint Config	0..1	aggr	This aggregation is used to configure the credentials of the endpoint.
socketOption	String	*	attr	This attribute represents the ability to specify non-formal socket options that might only be valid for specific platforms. AUTOSAR does not define a standardized meaning for the possible values of this attribute.
tlsSecureCom Props	TlsSecureComProps	0..1	ref	This reference provides the ability to define TLS-related properties for the enclosing SocketRawDataStream Mapping.

Table 12.6: EthernetRawDataStreamMapping

Class	**EthernetRawDataStreamLocalEndpointConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::RawDataStreamMapping			
Note	This meta-class has the ability to act as a wrapper for the configuration of the remote endpoint in the context of a raw data stream mapping.			
Base	*ARObject*			
Aggregated by	*EthernetRawDataStreamMapping*.localEndpointConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
localComm Connector	EthernetCommunication Connector	0..1	ref	This attribute represents the CommunicationConnector taken for socket-based data communication.
localTcpPort	ApApplicationEndpoint	0..1	ref	This aggregation represents the configuration of a local TCP port number.
localUdpPort	ApApplicationEndpoint	0..1	ref	This aggregation represents the configuration of a local unicast UDP port number.

Table 12.7: EthernetRawDataStreamLocalEndpointConfig

[constr_10079]{DRAFT} **Existence of `EthernetRawDataStreamMapping.localTcpPort` and `localUdpPort`** ⌈In the context of a `EthernetRawDataStreamMapping.localCommConnector`, only **one** attribute out of

- `localTcpPort`
- `localUdpPort`

shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

12.3.1 Ethernet Raw Data Stream Client Mapping

A raw data stream client may be prepared to receive data via unicast or multicast communication, but it shall only send requests via unicast to exactly one server.

[TPS_MANI_01356]{DRAFT} **Definition of remote server's Ethernet credentials** ⌈The definition of the **remote** server's Ethernet credentials is done by means of attributes of meta-class `EthernetRawDataStreamRemoteServerConfig`:

multicast If multicast configuration shall be supported, then the aggregation `EthernetRawDataStreamRemoteServerConfig.multicastCredentials` shall exist and the attribute `udpPort` shall be configured.

unicast If unicast configuration shall be supported, then the aggregation `EthernetRawDataStreamRemoteServerConfig.unicastUdpCredentials` shall exist and either the attribute `udpPort` **or** attribute `tcpPort` shall be configured.

⌋*(RS_MANI_00067)*

The mutual existence of attributes `udpPort` and `tcpPort` is also addressed by [constr_10078].

Note that it is also possible for the client to use ephemeral ports for the communication. In this case, the EthernetRawDataStreamLocalEndpointConfig.localTcpPort – by convention – would be configured to value 0.

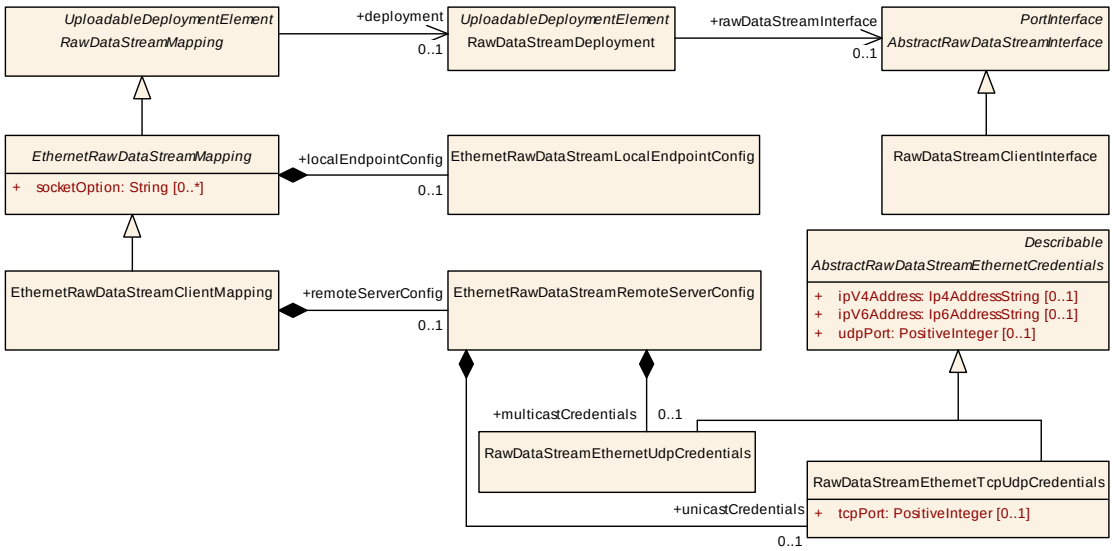


Figure 12.5: Modeling of the EthernetRawDataStreamClientMapping

Class	**EthernetRawDataStreamClientMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::RawDataStreamMapping			
Note	This meta-class represents the ability to map a client PortPrototype to a Ethernet-based communication channel. **Tags:** atp.recommendedPackage=RawDataStreamingMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *EthernetRawDataStreamMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *Uploadable DeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
remoteServer Config	EthernetRawData StreamRemoteServer Config	0..1	aggr	This aggregation is used to configure the credentials of the remote server.

Table 12.8: EthernetRawDataStreamClientMapping

Class	**EthernetRawDataStreamRemoteServerConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::RawDataStreamMapping			
Note	This meta-class has the ability to act as a wrapper for the configuration of the remote server in the context of a raw data stream client mapping.			
Base	*ARObject*			
Aggregated by	EthernetRawDataStreamClientMapping.remoteServerConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
multicast Credentials	RawDataStream EthernetUdpCredentials	0..1	aggr	This aggregation represents the configuration of multicast credentials for communication with a remote raw data stream server.

△

Class	**EthernetRawDataStreamRemoteServerConfig**			
unicast Credentials	RawDataStream EthernetTcpUdp Credentials	0..1	aggr	This meta-class represents the ability to map a server PortPrototype to a Ethernet-based communication channel.

Table 12.9: EthernetRawDataStreamRemoteServerConfig

12.3.2 Ethernet Raw Data Stream Server Mapping

A raw data stream server shall *either* communicate via unicast *or* via multicast, as expressed in [constr_10086]. A switch between the two approaches depending on the evaluation of conditions at runtime, as possible in the SOME/IP communication, is not supported for raw data streams.

[TPS_MANI_01357]{DRAFT} **Definition of remote client's Ethernet credentials** ⌈The definition of the **remote** client's Ethernet credentials is done by means of attributes of meta-class `EthernetRawDataStreamRemoteClientConfig`:

multicast If multicast configuration shall be supported, then the aggregation `EthernetRawDataStreamRemoteClientConfig.multicastCredentials` shall exist and the attribute `udpPort` shall be configured.

unicast If unicast configuration shall be supported, then the aggregation `EthernetRawDataStreamRemoteClientConfig.unicastUdpCredentials` shall exist and attribute `udpPort` shall be configured.

⌋*(RS_MANI_00067)*

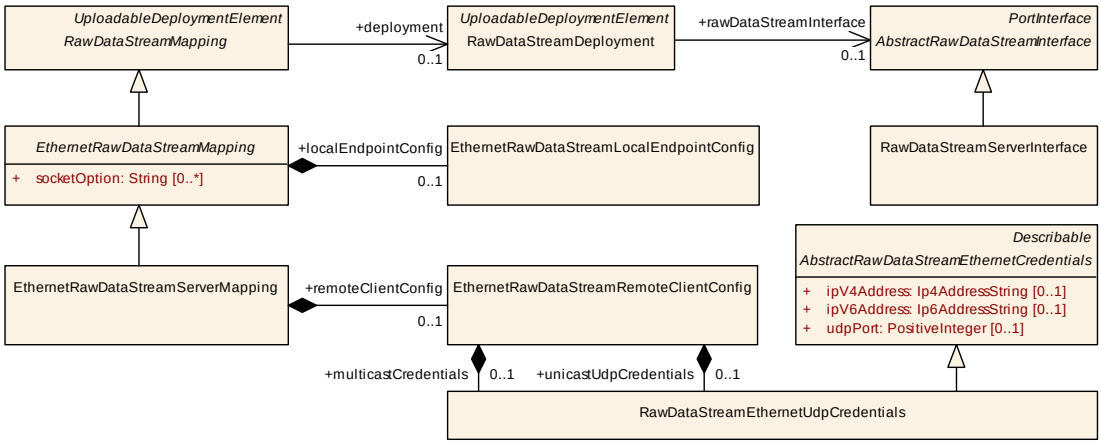


Figure 12.6: Modeling of the `EthernetRawDataStreamServerMapping`

Please note that a raw data stream server does not need to configure a TCP port of the client because the information about the TCP port of the client is conveyed as part of the TCP protocol.

[constr_10086]{DRAFT} **Existence of `unicastUdpCredentials` and `multicastCredentials` in the context of a `EthernetRawDataStreamServerMapping`** ⌈In

the context of a `EthernetRawDataStreamServerMapping`, only one aggregation out of

- `remoteClientConfig.multicastCredentials`
- `remoteClientConfig.unicastUdpCredentials`

shall exist **`at the time when the creation of the manifest is finished.`**⌋()

Class	**EthernetRawDataStreamServerMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::RawDataStreamMapping			
Note	This meta-class represents the ability to map a server PortPrototype to a Ethernet-based communication channel. **Tags:** atp.recommendedPackage=RawDataStreamingMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *EthernetRawDataStreamMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *RawDataStreamMapping*, *Referrable*, *Uploadable DeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
remoteClient Config	EthernetRawData StreamRemoteClient Config	0..1	aggr	This aggregation is used to configure the credentials of the remote client.

Table 12.10: EthernetRawDataStreamServerMapping

Class	**EthernetRawDataStreamRemoteClientConfig**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::RawDataStreamMapping			
Note	This meta-class has the ability to act as a wrapper for the configuration of the remote server in the context of a raw data stream client mapping.			
Base	*ARObject*			
Aggregated by	EthernetRawDataStreamServerMapping.remoteClientConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
multicast Credentials	RawDataStream EthernetUdpCredentials	0..1	aggr	This aggregation represents the configuration of multicast credentials for communication with a remote raw data stream client.
unicastUdp Credentials	RawDataStream EthernetUdpCredentials	0..1	aggr	This aggregation represents the configuration of a remote raw data stream client that communicates via unicast over UDP.

Table 12.11: EthernetRawDataStreamRemoteClientConfig

Examples for the configuration of raw data streams can be found in Appendix B.9.

13 Signal-based communication

13.1 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 using signal-based communication over, e.g., CAN or FlexRay, the translation of the signal-based content into `ServiceInterface`s needs to be established. The preconditions for this use-case are defined in section 13.2.

Such a `signal/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. This approach is defined in the System Template specification for the Classic platform [18]. It is up to the vehicle architecture design to choose whether the `signal/service translation` shall be implemented on a Classic platform ECU or on an Adaptive platform Machine.

Another alternative for this translation is to happen directly on the `Machine` on the *AUTOSAR adaptive platform* by an Application that is running in the Process, as sketched in Figure 13.1.

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/service translation 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 13.1, the *AUTOSAR classic platform* System Description would also contain a Pdu or Signal Gateway configuration between the Ethernet and the CAN network to forward the PDUs between the networks.

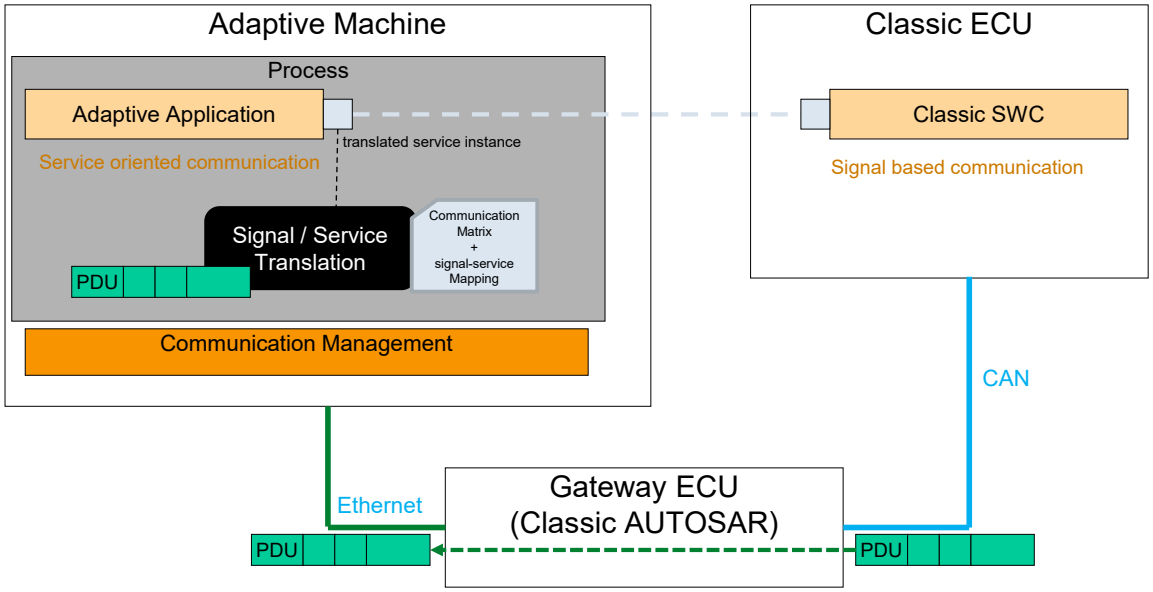


Figure 13.1: Signal-service-translation in Application on Adaptive Machine

The *translated service instance* is accessed from the application software using the ara::com API. The translation is designed as a network binding similar to the binding of SOME/IP. So the communication direction is

- if the signal-based payload is received by the Machine then the application software has a RPortPrototype
- if the signal-based payload is sent by the Machine then the application software has a PPortPrototype.

Another approach for the usage of signal/service translation is the interaction of a Classic AUTOSAR Instance and an Adaptive AUTOSAR Instance running on the same virtualization technology.

In such a setup, the IPC technology is able to transport ISignalIPdus between the two platform instances. It is not defined whether the IPC technology requires any Message Header (according to section 13.2.1) being part of the payload. Thus, no dedicated configuration for this Message Header (specifically the configuration of the Header ID) is required in case this information can already be derived from the protocol information of the underlying IPC technology (e.g. dedicated IPC channels for different messages).

Figure 13.2 sketches an example setup where a virtualization technology enables the transport of messages between a Classic and an Adaptive Platform instance.

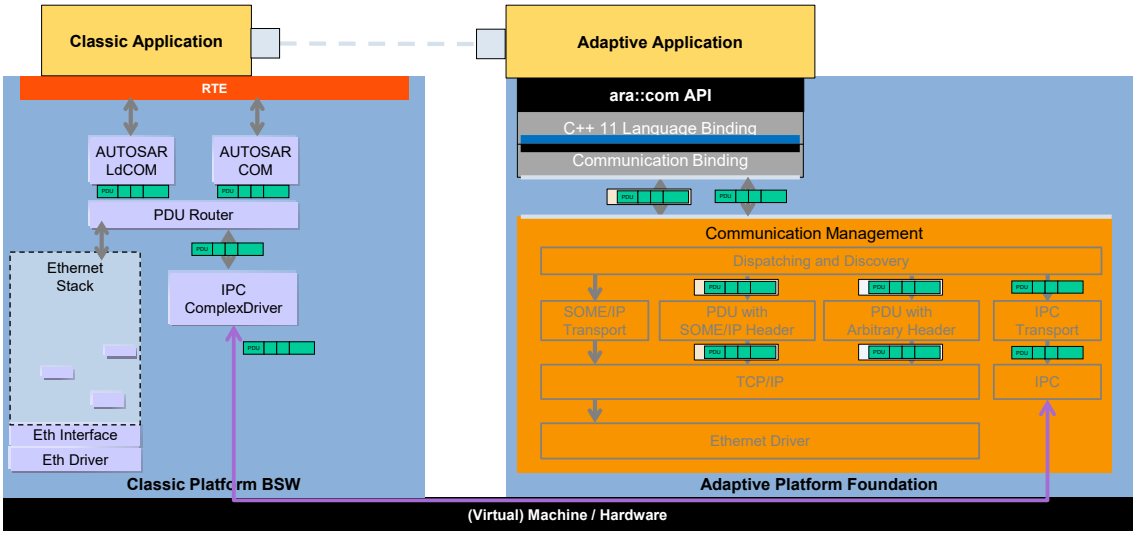


Figure 13.2: Signal/service translation using IPC technology

Based on the `signal/service translation` network binding there are several approaches for further processing the translated information (illustrated in figure 13.3):

- The application software on the *AUTOSAR adaptive platform* directly consumes/produces[1] the translated service (1)
  This is the typical approach if there is only one piece of application software on the *AUTOSAR adaptive platform* interested in the translated data.
- The application software on the *AUTOSAR adaptive platform* manages and performs a functional routing (2)
  This is the approach if the translated data shall be available as a service again for further processing. Here it is up to the implementation of the application software on the *AUTOSAR adaptive platform* how the translated data is routed to the secondary service, especially whether data combination and data conversion are applied.
- Pass Through Connectors: The translated service is passed through to be available to further application software on the *AUTOSAR adaptive platform*. This is the approach if there exist several further application software on the *AUTOSAR adaptive platform* which are using the translated service. In this case the translation only has be performed once. Two cases can be distinguished:
  - The translated service and the secondary service use the same `ServiceInterface` (3)

[1] In the explanations the direction `signal-service-translation` is usually used, the `service-signal-translation` direction is supported as well. For simplicity this is only mentioned explicitly if the mapping behavior is not symmetrical.

– The translated service and the secondary service use different (but compatible) ServiceInterfaces (4)

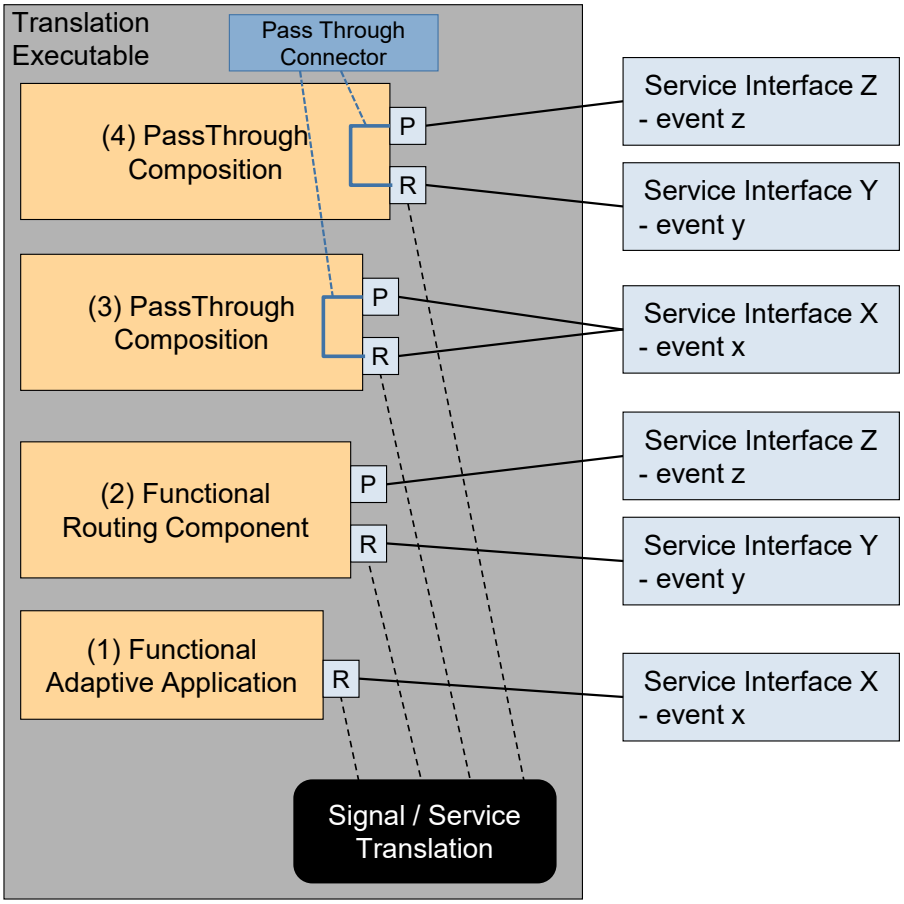


Figure 13.3: Signal/service translation and PassThroughConnector

The methodological approach is that out of the signal/service translation mapping descriptions it is possible to automatically generate the network binding code for the signal/service translation.

If there is a CompositionSwComponentType defined with PassThroughSwConnectors then also the AdaptiveApplicationSwComponentType implementing the pass through behavior can be generated (see section 13.7).

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/service translation 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/service translation mapping is built off-board and the application executable gets uploaded to the target Machine in response to changes in the communication matrix.

The Executable utilizing the signal/service translation and/or implementing the PassThroughSwConnectors is considered to belong to the APPLICATION_LEVEL domain as defined in [constr_1605]. This also applies to cases where the code implementing the PassThroughSwConnectors has been generated automatically by some tooling.

[TPS_MANI_03643]{DRAFT} **Translation-Executable category** ⌈The category value of the Executable utilizing the signal/service translation and/or implementing the PassThroughSwConnectors shall be set to APPLICATION_LEVEL.⌋ *(RS_MANI_00063)*

The following terminology is used in the context of signal/service translation:

Signal/service translation defines the feature this chapter is concerned with. It does not prescribe a specific translation direction.

Signal-service-translation defines the translation direction from a signal-based to a service oriented representation.

Service-signal-translation defines the translation direction from a service oriented to a signal-based representation.

13.2 Signal-based prerequisites

The identification of the received and sent messages and their actual length needs to be determined for each individual Pdu. Depending on the transport technology and configuration, a Message Header might be required. If the transport technology can determine the kind of Pdu and the length out of its protocol information, then the Message Header may be omitted.

If a custom transport technology is used (e.g. IPC between a Classic Platform instance and an Adaptive Platform instance), it depends on that specific transport technology whether a Message Header and length information is required or not.

As the sole communication network currently supported is Ethernet, the Pdus have to be transported on that network.

Although there is in theory the possibility to directly put the Pdu on the Ethernet, this approach would require an individual Socket per Pdu. This is an approach where especially Classic platform ECUs do not have enough resources available to allow individual Sockets per Pdu. Nevertheless, there is also support by the signal/service translation to have no Message Header defined for Ethernet transport.

In case of Ethernet transport and the approach of several Pdus sharing a Socket, there is a need to use a dedicated Message Header per Pdu. This is illustrated in figure 13.4. The Classic Platform ECU routes the Pdus from a CAN network to the Ethernet.

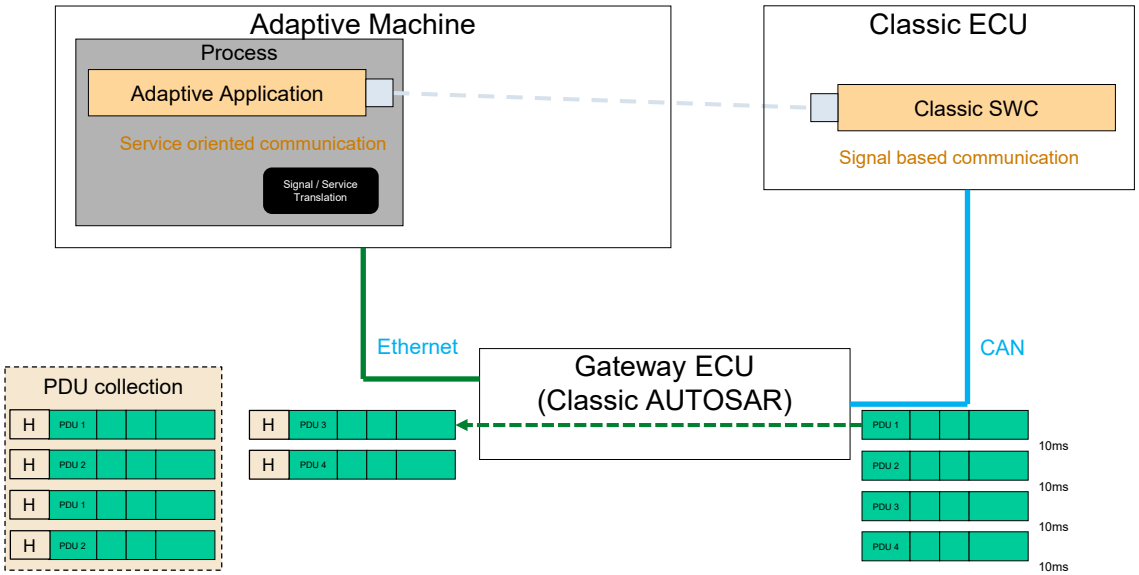


Figure 13.4: Network architecture for signal-service-translation

In order to have efficient transport of many Pdus on Ethernet additionally the SOME/-IP collection mechanism may be used (i.e. definition of pduCollectionTrigger, pduCollectionSemantics, pduCollectionPduTimeout).

13.2.1 Message Header

In case of Ethernet transport and the approach of several Pdus sharing a Socket, the identification of the received and sent messages is done using a 64 bit wide Message Header. This Message Header is composed of a 32 bit wide Message ID and a 32 bit wide Message Length field.

[TPS_MANI_03652]{DRAFT} **Signal/service translation header format** ⌈If a Message Header is used then the message-transporting signal-based Pdus shall have a 32 bit wide Message ID field and a 32 bit wide Message Length field.⌋*(RS_MANI_-00063)*

[TPS_MANI_03653]{DRAFT} **Signal/service translation header endianness** ⌈The header of a message-transporting signal-based Pdus shall be encoded in big endian.⌋ *(RS_MANI_00063)*

There are three variants supported by the signal/service translation approach how the Message Header is interpreted:

Partial SOME/IP Header : 32 bit wide Message ID field and 32 bit wide Length field (just Message ID and Length, then signal-based payload. Message ID according to the rules of SOME/IP: ServiceId and MethodId [PRS_SOMEIP_-00245])

Arbitrary Message Header : 32 bit wide Message ID field and 32 bit wide Length field (e.g. CAN-Id taken as Message Id) with signal-based payload

No Message Header : No standardized Message Header is defined.

In figure 13.5, the *Full SOME/IP Header* is shown for illustration purposes. In the context of signal/service translation the *Full SOME/IP Header* is not used. The *Signal-Based Header* can either be configured according to the Partial SOME/-IP Header, the Arbitrary Message Header, or the No Message Header approach.

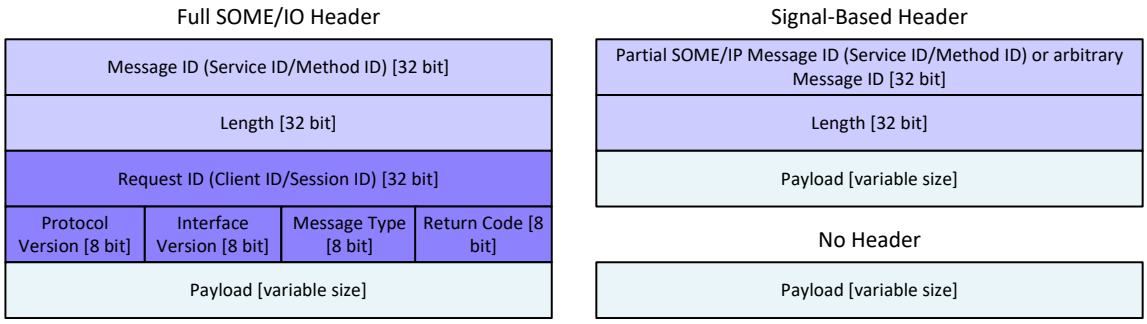


Figure 13.5: SOME/IP Message Header Format

Which Message Header variant is used for a dedicated message is defined according to the input configuration:

[TPS_MANI_03654]{DRAFT} **Definition of Partial SOME/IP Header ID** ⌈If there is a RequiredSomeipServiceInstance of category SIGNAL-BASED_WITH_HEADER defined for the signal-service-translation (resp. ProvidedSomeipServiceInstance for the service-signal-translation), then the Message ID shall be interpreted according to the SOME/IP Message ID rules [PRS_SOMEIP_00245]. Namely the Message ID is composed of the SomeipServiceInterfaceDeployment.serviceInterfaceId and the SomeipEventDeployment.eventId + 0x8000.⌋*(RS_MANI_00063)*

[TPS_MANI_03577]{DRAFT} **Definition of Arbitrary Message Header ID** ⌈If there is a RequiredUserDefinedServiceInstance of category SIGNAL-BASED_WITH_HEADER defined for the signal-service-translation (resp. ProvidedUserDefinedServiceInstance for the service-signal-translation), then the Message ID shall be taken from the SocketConnectionIpduIdentifier.headerId attribute.⌋*(RS_MANI_00063)*

[constr_3650]{DRAFT} **headerId required in case of Arbitrary Message Header** ⌈If [TPS_MANI_03577] applies, then the respective SocketConnectionIpduIdentifier.headerId shall be defined **at the time when the creation of the manifest is finished**.⌋*()*

There is also the possibility to define the No Message Header option:

[TPS_MANI_03655]{DRAFT} **Definition of `No Message Header` - implicit SOME/IP Message Identification** ⌈If there is a `RequiredSomeipServiceInstance` of `category` `SIGNALBASED_NO_HEADER` defined for the `signal-service-translation` (resp. `ProvidedSomeipServiceInstance` for the `service-signal-translation`), then no Message Header shall be used.

In this case it is either possible to derive the Message identification from the assignment to unambiguous communication channels or the Message identification is derived from protocol information of the underlying IPC technology.⌋*(RS_MANI_00063)*

[TPS_MANI_03656]{DRAFT} **Definition of `No Message Header` - implicit Arbitrary Message Identification** ⌈If there is a `RequiredUserDefinedServiceInstance` of `category` `SIGNALBASED_NO_HEADER` defined for the `signal-service-translation` (resp. `ProvidedUserDefinedServiceInstance` for the `service-signal-translation`), then no Message Header shall be used.

In this case it is either possible to derive the Message identification from the assignment to unambiguous communication channels or the Message identification is derived from protocol information of the underlying IPC technology.⌋*(RS_MANI_00063)*

13.3 Signal-based Deployment

The signal-service-translation is embedded in the `SOME/IP` deployment using `SomeipServiceInterfaceDeployment`. The attribute `SomeipEventDeployment.serializer` defines whether the `someip` or the `signalBased` serialization shall be used for a specific `event` (see also figure 11.4).

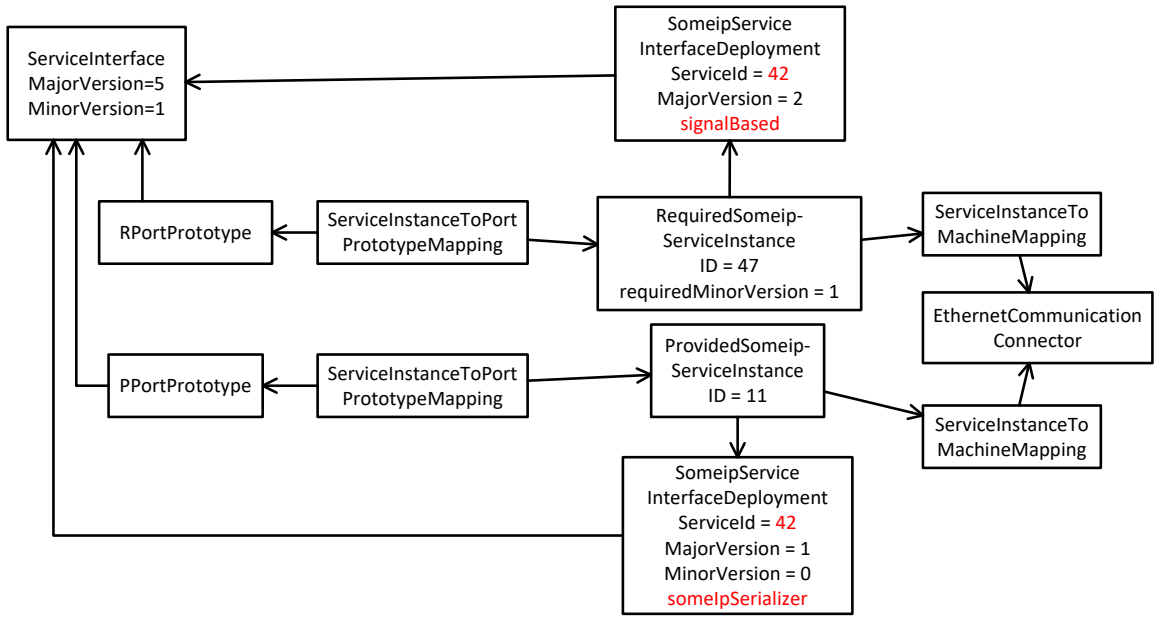


Figure 13.6: Example of 1 `PPortPrototype` mapped to one VLAN

[TPS_MANI_03578]{DRAFT} **Signal-based `ServiceInterface` binding over `Ethernet`** ⌈In case of signal-based communication over `Ethernet` the `SomeipServiceInterfaceDeployment` is used to define the `Pdu` transport over the network.⌋ *(RS_MANI_00063)*

This aspect is described in section 13.2.

[TPS_MANI_03579]{DRAFT} **Signal-based `ServiceEventDeployment` over `Ethernet`** ⌈If the attribute `SomeipEventDeployment.serializer` equals `signalBased` then the `event` referenced by `ServiceEventDeployment.event` shall be handled using signal-service-translation.⌋ *(RS_MANI_00063)*

Figure 13.6 illustrates a use-case where a `ServiceInterface` has two `SomeipServiceInterfaceDeployment`s, one with `signalBased serializer` technology and one with `someip`.

The translation use-case is defined by having one `RPortPrototype` receiving the `signalBased` serialized messages and one `PPortPrototype` providing the `someip` serialized service.

13.4 Signal/Service Translation Mapping

This chapter describes the mapping of `ServiceInterface` elements of a specific `AdaptivePlatformServiceInstance` to `ISignalTriggering`s.

Note that according to [TPS_MANI_03555] the same Ethernet socket (via `ISignalTriggering`, `PduTriggering`, and `SocketConnection`) may be used for signal-based and service-oriented communication at the same time.

This allows to define one service instance which consists of events with different serialization technologies (i.e. `someip` and `signalBased`).

[TPS_MANI_03627]{DRAFT} **No `signal/service translation` for methods** ⌈As Methods and Field getter/setter are already serialized using the `someip` serialization there is no need for `signal/service translation` for them.⌋ *(RS_MANI_00029)*

Therefore, Methods and Field getter/setter are directly accessible using ara::com via the *Signal Based Network Binding* as defined in Communication Management [10]. For passing methods through refer to section 13.7.

A `ServiceInstanceToSignalMapping` refers to an `AdaptivePlatformServiceInstance` in the role `ServiceInstanceToSignalMapping.serviceInstance`. While it would be possible to define the mappings for all the `event`s, `field`s, and `trigger`s of an `AdaptivePlatformServiceInstance` in the scope of one `ServiceInstanceToSignalMapping`, it is rather advised to create several `ServiceInstanceToSignalMapping`s referring to the same `AdaptivePlatformServiceInstance`, one for each service member.

[advisory_02000]{DRAFT} **Define a `ServiceInstanceToSignalMapping` per member of an `AdaptivePlatformServiceInstance`** ⌈If an AdaptivePlatformServiceInstance is referenced by a ServiceInstanceToSignalMapping in the role ServiceInstanceToSignalMapping.serviceInstance, then there should be one instance of ServiceInstanceToSignalMapping defined for every member of the corresponding ServiceInterface (event, field, trigger) **at the time when the creation of the manifest is finished**.⌋*()*

13.4.1 Handling of `ISignalGroup`s

ISignalGroups are used in the definition of ISignalIPdus to express that certain ISignals belong together and form a consistent set of data. The definition of ISignalGroups is part of the signal based communication matrix definition.

ISignalGroups do not need to be explicitly mapped when defining the ServiceInstanceToSignalMappings, they are implicitly covered in the mapping of the individual ISignals belonging to the ISignalGroup.

[advisory_02001]{DRAFT} **No explicit `signal/service translation` for `ISignalGroup`s** ⌈The mapping elements of a ServiceInstanceToSignalMapping:

- SignalBasedEventElementToISignalTriggeringMapping.iSignalTriggering
- SignalBasedTriggerToISignalTriggeringMapping.iSignalTriggering
- SignalBasedFieldToISignalTriggeringMapping.notifierSignalTriggering

should not refer to an ISignalTriggering that in turn refers to an ISignalGroup **at the time when the creation of the manifest is finished**.⌋*()*

13.4.2 Direction of Signal/Service Translation Mapping

The definition of the translation mappings has no explicitly modeled direction (has no source or target point of view). Each individual mapping (AbstractSignalBasedToISignalTriggeringMapping) takes one element from a ServiceInterface and one ISignalTriggering.

[TPS_MANI_03635]{DRAFT} **Determination of translation direction** ⌈The translation direction is determined by

- the communicationDirection of the referenced ISignalPort which is referenced by the ISignalTriggering

- the kind of service instance which is referenced from the ServiceInstanceToSignalMapping in the role serviceInstance. It can be either a ProvidedApServiceInstance or a RequiredApServiceInstance.

⌋(*RS_MANI_00029*)

[constr_3636]{DRAFT} **Consistent ISignal communication direction in and RequiredApServiceInstance** ⌈If the ServiceInstanceToSignalMapping.serviceInstance refers to a RequiredApServiceInstance then any ServiceInstanceToSignalMapping.eventElementMapping (respectively ServiceInstanceToSignalMapping.fieldMapping) shall refer to an ISignalTriggering which in turn refers to an ISignalPort with communicationDirection equal to in **at the time when the creation of the manifest is finished**.⌋()

[constr_3637]{DRAFT} **Consistent ISignal communication direction out and ProvidedApServiceInstance** ⌈If the ServiceInstanceToSignalMapping.serviceInstance refers to a ProvidedApServiceInstance then any ServiceInstanceToSignalMapping.eventElementMapping (respectively ServiceInstanceToSignalMapping.fieldMapping) shall refer to an ISignalTriggering which in turn refers to an ISignalPort with communicationDirection equal to out **at the time when the creation of the manifest is finished**.⌋()

Class	**ServiceInstanceToSignalMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class is defined for a specific ServiceInstance and contains the mappings of elements of a ServiceInterface for which the ServiceInstance is defined to individual ISignalTriggerings. **Tags:** atp.Status=candidate atp.recommendedPackage=ServiceInstanceToSignalMapping			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventElement Mapping	SignalBasedEvent ElementToISignal TriggeringMapping	*	aggr	Mapping of an event or an element inside of the event to an ISignalTriggering. **Tags:** atp.Status=candidate
fieldMapping	SignalBasedFieldToI SignalTriggering Mapping	*	aggr	Mapping of a field to ISignalTriggerings. **Tags:** atp.Status=candidate
fireAndForget MethodMapping	SignalBasedFireAnd ForgetMethodToISignal TriggeringMapping	*	aggr	Mapping of an ISignalTriggering being part of a fire and forget message to a ClientServerOperation. **Tags:** atp.Status=candidate
serviceInstance	AdaptivePlatform ServiceInstance	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=candidate
triggerMapping	SignalBasedTriggerToI SignalTriggering Mapping	*	aggr	Mapping of a trigger to an ISignalTriggering. **Tags:** atp.Status=candidate

Table 13.1: ServiceInstanceToSignalMapping

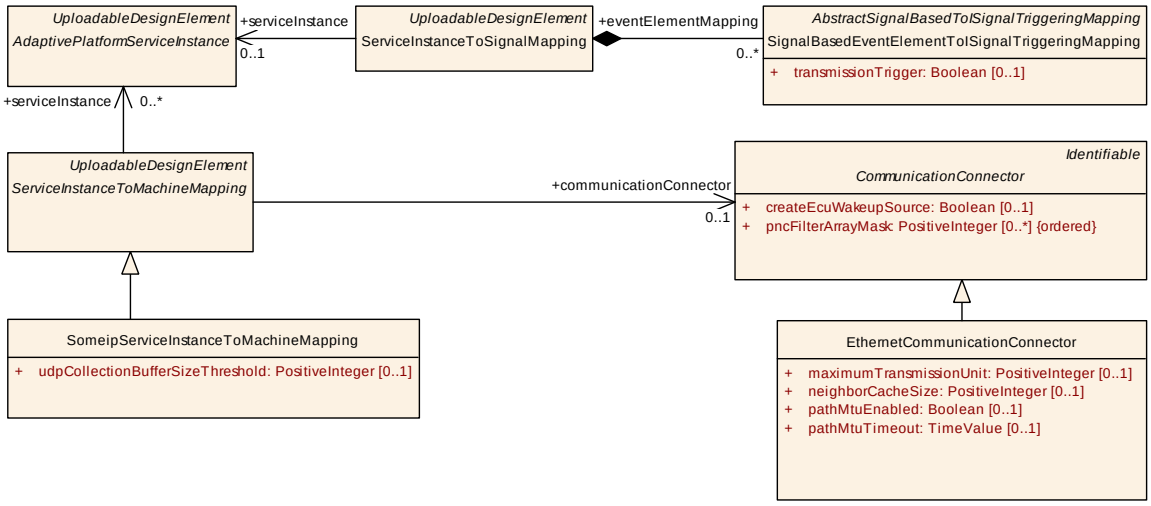


Figure 13.7: Relation of `ServiceInstanceToSignalMapping` and `CommunicationConnector`

The `ServiceInstanceToSignalMapping` refers to an `AdaptivePlatformServiceInstance` and thereby defines which `serviceInterfaceDeployment` elements will be mapped by the aggregated `eventElementMapping`, and/or `fieldMapping` to `ISignalTriggering`s. This is described in detail in the following chapters.

The `ServiceInstanceToMachineMapping` which refers to the `AdaptivePlatformServiceInstance` (which in turn is referenced by the `ServiceInstanceToSignalMapping`) defines on which `CommunicationConnector` (i.e. network / VLAN) the signal based communication shall be performed.

[TPS_MANI_03629]{DRAFT} **Relation of `ServiceInstanceToSignalMapping` and `CommunicationConnector`** ⌈The `ServiceInstanceToMachineMapping` referring to the `AdaptivePlatformServiceInstance` defines on which `CommunicationConnector` the `AdaptivePlatformServiceInstance` shall be communicated. If a `ServiceInstanceToSignalMapping` refers to the same `AdaptivePlatformServiceInstance` then the signal based communication shall be performed on the referenced `CommunicationConnector`.⌋ *(RS_MANI_00029)*

13.4.3 SignalBasedEvent Mapping

It is required that every `event` with `signalBased` serialization has a `ServiceInstanceToSignalMapping` defined.

[constr_3550]{DRAFT} **Existence of `ServiceInstanceToSignalMapping` for an `event` with `signalBased` serialization** ⌈If

- an `event` is referenced by a `SomeipEventDeployment` in the role `event` and
- the attribute `SomeipEventDeployment.serializer` is set to `signalBased`,

then a ServiceInstanceToSignalMapping shall exist with eventElementMapping referring to the event in the role dataPrototypeInServiceInterfaceRef **at the time when the creation of the manifest is finished.** ⌋()

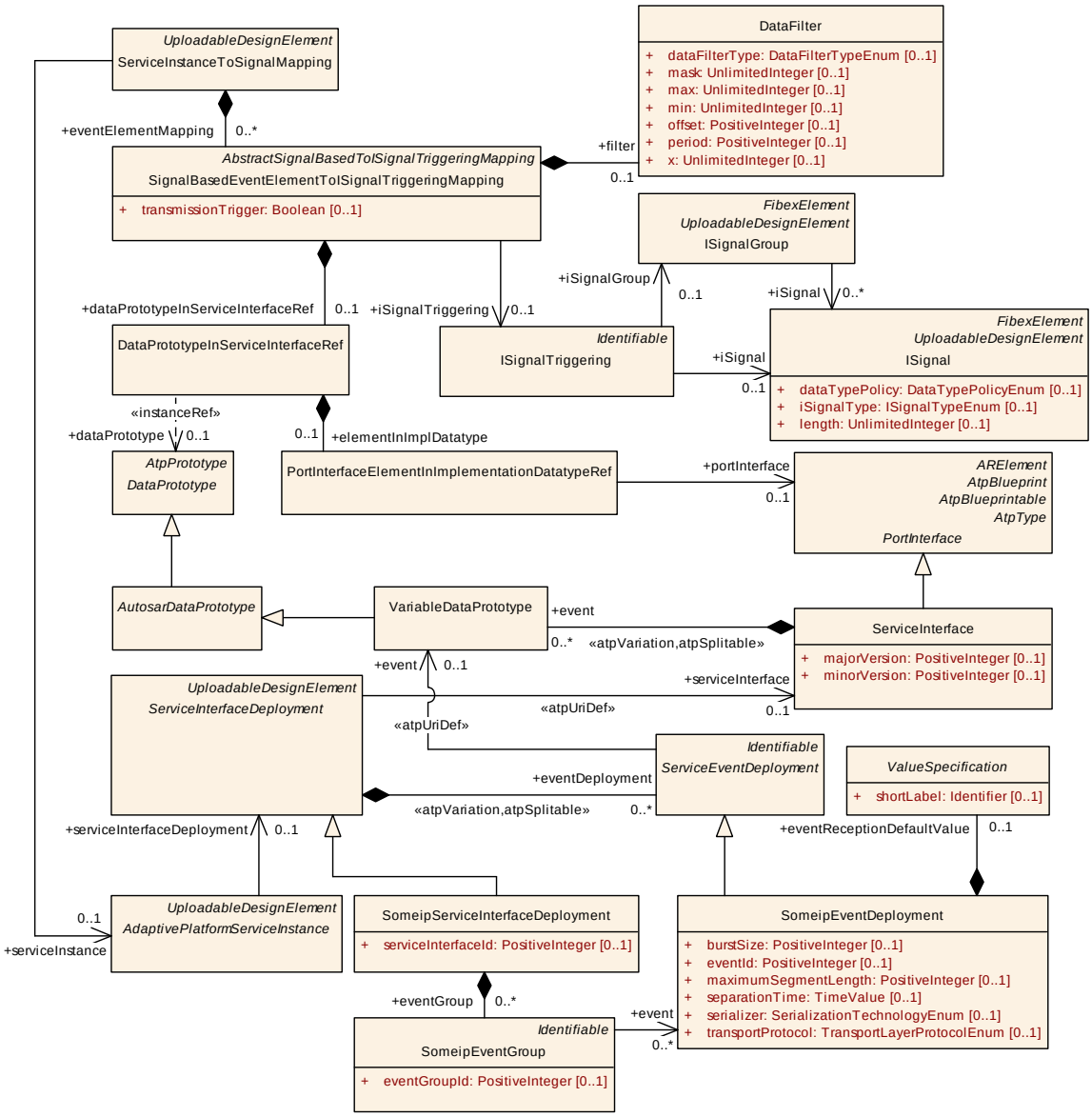


Figure 13.8: Mapping of Event elements to ISignals

In case of composite payload it is important that the mapping definition is complete from the target point of view:

[constr_3551]{DRAFT} **Full mapping of target ISignalGroup** ⌈If

- an ISignalTriggering is part of a ServiceInstanceToSignalMapping and
- the ISignalTriggering refers to an ISignalPort with communicationDirection equals out and

- the ISignalTriggering refers to an ISignalGroup in the role iSignalGroup,

then a SignalBasedEventElementToISignalTriggeringMapping shall exist for every ISignal referenced by the ISignalGroup in the role iSignal **at the time when the creation of the manifest is finished**.⌋()

[constr_3552]{DRAFT} **Full mapping of target event** ⌈If

- the ServiceInstanceToSignalMapping refers to a ProvidedSomeipServiceInstance and
- the dataPrototypeInServiceInterfaceRef refers to a DataPrototype which is part of a composite data type,

then a SignalBasedEventElementToISignalTriggeringMapping shall exist for every DataPrototype that is part of the composite data type **at the time when the creation of the manifest is finished**.⌋()

[TPS_MANI_03124]{DRAFT} **ServiceInterface.event to ISignalTriggering mapping** ⌈The SignalBasedEventElementToISignalTriggeringMapping meta-class provides the ability to map a DataPrototype defined in the context of a ServiceInterface to one ISignalTriggering of the ISignal or ISignalGroup.⌋(*RS_MANI_00029*)

Class	**SignalBasedEventElementToISignalTriggeringMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class defines the mapping of a ServiceInterface event or an element that is defined inside of the event in case that the datatype is composite to an ISignalTriggering. **Tags:** atp.Status=candidate			
Base	*ARObject*, *AbstractSignalBasedToISignalTriggeringMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ServiceInstanceToSignalMapping.eventElementMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataPrototypeIn ServiceInterface Ref	DataPrototypeInService InterfaceRef	0..1	aggr	Reference to a DataPrototype or to an internal structure of a DataPrototype in the context of a ServiceInterface. **Tags:** atp.Status=candidate
filter	DataFilter	0..1	aggr	Defines an optional filter to be applied during translation. **Tags:** atp.Status=candidate
iSignal Triggering	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=candidate
transmission Trigger	Boolean	0..1	attr	Defines whether the source element triggers the sending of the respective payload. **Tags:** atp.Status=candidate

Table 13.2: SignalBasedEventElementToISignalTriggeringMapping

[constr_10252]{DRAFT} **Multiplicity of attribute SignalBasedEventElementToISignalTriggeringMapping.dataPrototypeInServiceInterfaceRef**

⌈For each SignalBasedEventElementToISignalTriggeringMapping, the attribute dataPrototypeInServiceInterfaceRef shall exist **at the time when the creation of the manifest is finished**.⌋()

Class	**AbstractSignalBasedToISignalTriggeringMapping** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class is the common class for all SIgnalBased to ISignalTRiggering mappings. **Tags:** atp.Status=candidate			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	SignalBasedEventElementToISignalTriggeringMapping, SignalBasedFieldToISignalTriggeringMapping, SignalBasedFireAndForgetMethodToISignalTriggeringMapping, SignalBasedTriggerToISignalTriggering Mapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 13.3: AbstractSignalBasedToISignalTriggeringMapping

A ServiceInstanceToSignalMapping with several SignalBasedEventElementToISignalTriggeringMappings is used to map the data leaves of *TestEvent* to the corresponding ISignalTriggerings.

An example of a mapping of event content to a signal is described in Appendix B.10.1.1.

13.4.4 SignalBasedTrigger Mapping

It is required that every trigger with signalBased serialization has a ServiceInstanceToSignalMapping defined:

[constr_5318]{DRAFT} **Existence of ServiceInstanceToSignalMapping for an trigger with signalBased serialization** ⌈If a trigger is referenced by a SomeipEventDeployment in the role trigger and the attribute SomeipEventDeployment.serializer is set to signalBased then a ServiceInstanceToSignalMapping shall exist with triggerMapping referring to the trigger in the role trigger **at the time when the creation of the manifest is finished**.⌋()

[TPS_MANI_03287]{DRAFT} **ServiceInterface.trigger to ISignalTriggering mapping** ⌈The SignalBasedTriggerToISignalTriggeringMapping meta-class provides the ability to map a trigger defined in the context of a ServiceInterface to one ISignalTriggering of the ISignal.⌋(*RS_MANI_00063*)

Please note that [constr_1198] and [constr_1199] that are defined in the System Template specification [18] apply for the ISignal to which the trigger is mapped via the ISignalTriggering by the SignalBasedTriggerToISignalTriggeringMapping. This means that the length of the ISignal shall be 0 and an update bit shall be defined for the ISignal in case that no DataTransformation is used.

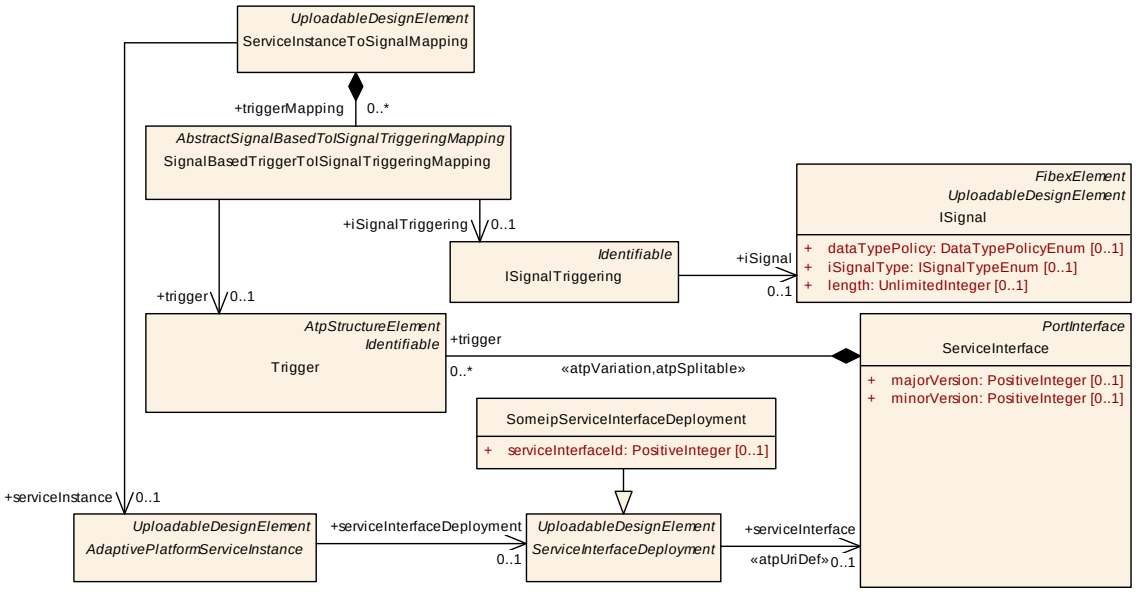


Figure 13.9: Mapping of a Trigger to an ISignal

Class	**SignalBasedTriggerToISignalTriggeringMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class defines the mapping of a ServiceInterface trigger to an ISignalTriggering. **Tags:** atp.Status=candidate			
Base	*ARObject*, *AbstractSignalBasedToISignalTriggeringMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ServiceInstanceToSignalMapping.triggerMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iSignal Triggering	ISignalTriggering	0..1	ref	Reference to the ISignalTriggering that is used to transport the trigger that is defined in a ServiceInterface in a signal-based way over a communication channel. **Tags:** atp.Status=candidate
trigger	Trigger	0..1	ref	Reference to a trigger defined in the context of a Service Interface. **Tags:** atp.Status=candidate

Table 13.4: SignalBasedTriggerToISignalTriggeringMapping

13.4.5 SignalBasedField Mapping

[advisory_02002]{DRAFT} **`ServiceInterface.field` mapping to `ISignalTriggering`s for primitive data type `field`s** ⌈If `Field.hasNotifier` equals true, then one `SignalBasedFieldToISignalTriggeringMapping` should be used to map a primitive data type `field` to one `ISignalTriggering` for the `ISignal` representing the primitive Notifier element using the role `notifierSignalTriggering`.

This rule shall be imposed **`at the time when the creation of the manifest is finished`**.⌋*()*

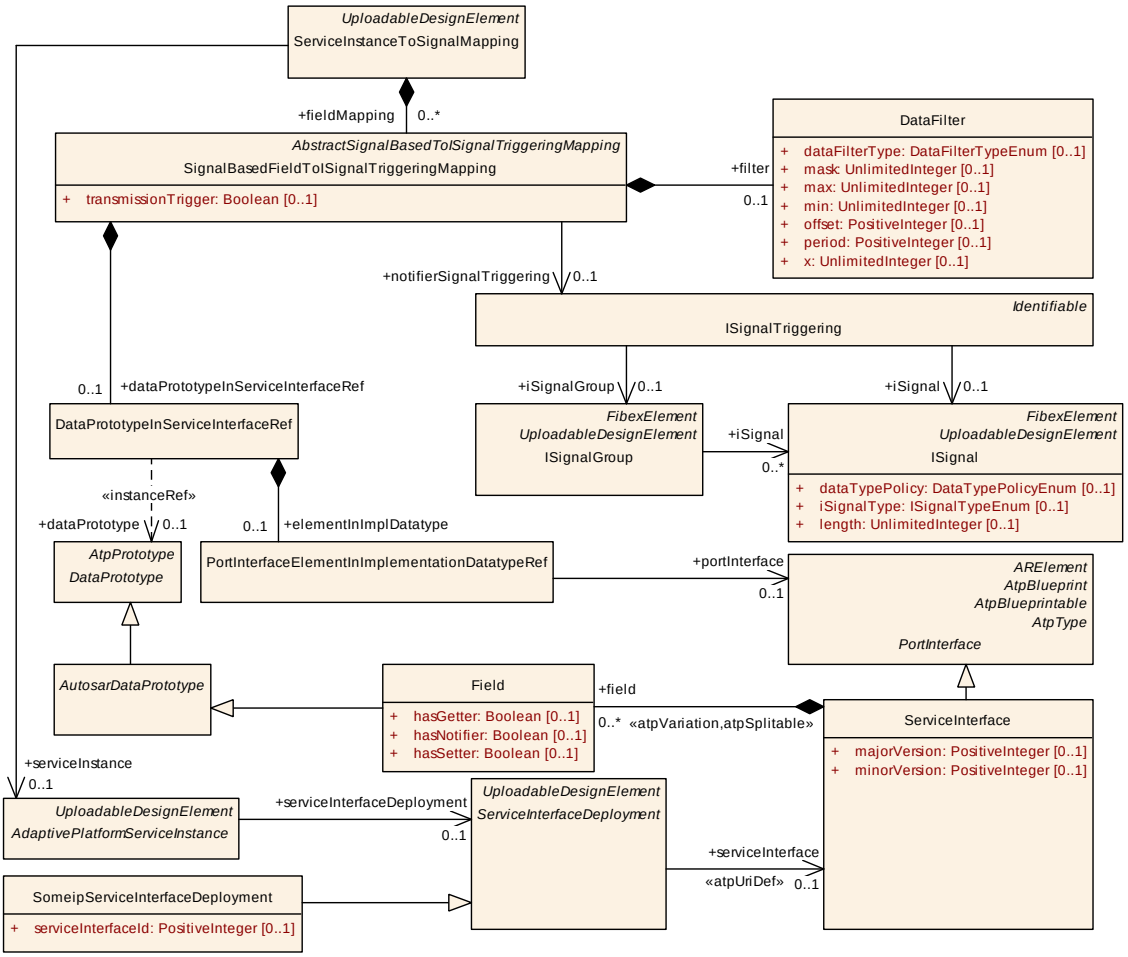


Figure 13.10: Mapping of Fields to ISignals

[advisory_02003]{DRAFT} **`ServiceInterface`.`field` mapping to `ISignalTriggering`s for composite data type `fields`** ⌈If `Field.hasNotifier` equals true, then a separate `SignalBasedFieldToISignalTriggeringMapping` should be used to map each `ISignal` belonging to the `ISignalGroup` used for the transport of the notifier using the role `notifierSignalTriggering`.

This rule shall be imposed **`at the time when the creation of the manifest is finished`**.⌋*()*

It means that several `SignalBasedFieldToISignalTriggeringMapping`s may be necessary to map a `field` to the corresponding `ISignalTriggering`s.

It is required that every `field` using a `SomeipFieldDeployment.notifier` with `signalBased` serialization has a `ServiceInstanceToSignalMapping` defined:

[constr_3553]{DRAFT} **Existence of `ServiceInstanceToSignalMapping` for an `field` with `signalBased` serialization** ⌈If a `field` is referenced by a `SomeipFieldDeployment` in the role `field` and that `SomeipFieldDeployment` aggregates a `SomeipEventDeployment` in the role `notifier` and the `SomeipEventDeployment` has an attribute `SomeipEventDeployment.serializer` set to `signalBased` then there shall exist a `ServiceInstanceToSignalMapping` with a

fieldMapping referring to the field in the role dataPrototypeInServiceInterfaceRef and the SignalBasedFieldToISignalTriggeringMapping shall refer to a ISignalTriggering in the role notifierSignalTriggering **at the time when the creation of the manifest is finished**.⌋()

[constr_10228]{DRAFT} **Multiplicity of attribute SignalBasedFieldToISignalTriggeringMapping.dataPrototypeInServiceInterfaceRef** ⌈For each SignalBasedFieldToISignalTriggeringMapping, the attribute dataPrototypeInServiceInterfaceRef shall exist **at the time when the creation of the manifest is finished**.⌋()

Class	**SignalBasedFieldToISignalTriggeringMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class defines the mapping of a ServiceInterface field to ISignalTriggerings that represent the notifier elements on a signal-based communication channel. **Tags:** atp.Status=candidate			
Base	*ARObject*, *AbstractSignalBasedToISignalTriggeringMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ServiceInstanceToSignalMapping.fieldMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataPrototypeIn ServiceInterface Ref	DataPrototypeInService InterfaceRef	0..1	aggr	Reference to a DataPrototype or to an internal structure of a DataPrototype in the context of a ServiceInterface. **Tags:** atp.Status=candidate
filter	DataFilter	0..1	aggr	Defines an optional filter to be applied during translation. **Tags:** atp.Status=candidate
notifierSignal Triggering	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=candidate
transmission Trigger	Boolean	0..1	attr	Defines whether the source notifier element triggers the sending of the respective payload. **Tags:** atp.Status=candidate

Table 13.5: SignalBasedFieldToISignalTriggeringMapping

An example of a mapping of a field to a signals is described in Appendix B.10.1.2.

13.4.6 SignalBased Fire and Forget Method Mapping

[TPS_MANI_03665]{DRAFT} **ServiceInterface.method with fireAndForget equals true to ISignalTriggerings mapping** ⌈The SignalBasedFireAndForgetMethodToISignalTriggeringMapping meta-class provides the ability to map a part of a method with ClientServerOperation.fireAndForget equals true to one ISignalTriggering for the ISignal representing one part of the “fire & forget” method message.⌋(*RS_MANI_00029*)

[constr_3692]{DRAFT} **DataPrototypeInServiceInterfaceInstanceRef.targetDataPrototype in the context of a SignalBasedFireAndForgetMethodToISignalTriggeringMapping** ⌈If a DataPrototypeInServiceInterfaceInstanceRef is aggregated by a SignalBasedFireAndForgetMethodToISignalTriggeringMapping in the role dataPrototypeInMethodArgumentInstanceRef, then the reference DataPrototypeInServiceInterfaceInstanceRef.targetDataPrototype shall refer to an ArgumentDataPrototype **at the time when the creation of the manifest is finished.**⌋ *()*

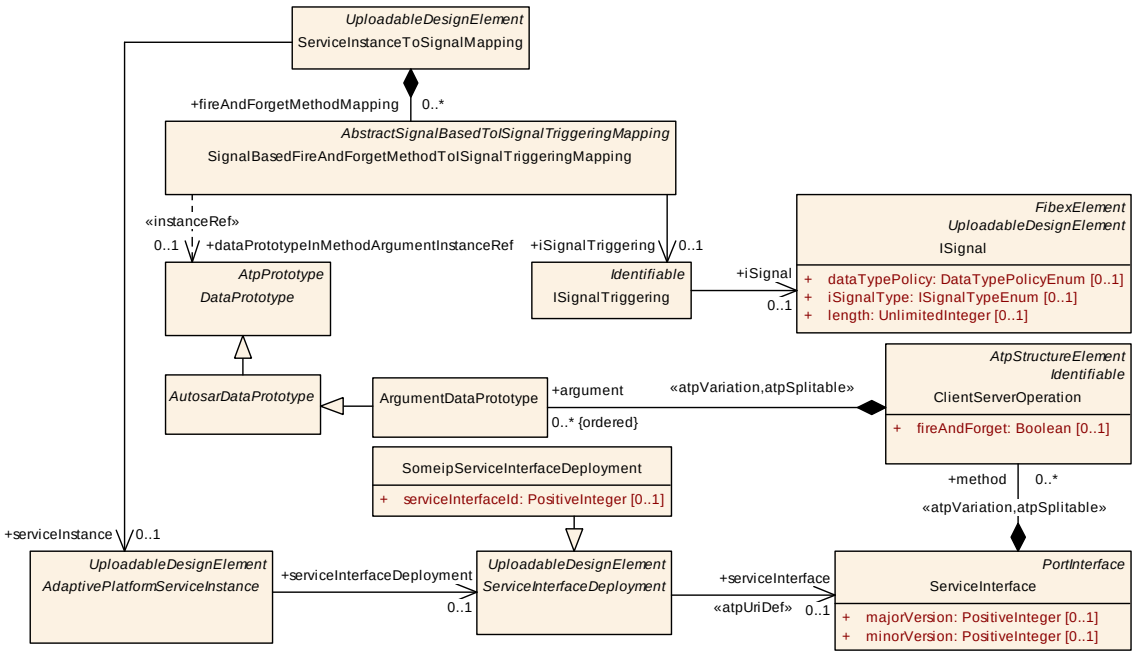


Figure 13.11: Mapping of "fire & forget" methods to ISignals

Class	**SignalBasedFireAndForgetMethodToISignalTriggeringMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SignalBasedCommunication			
Note	This meta-class defines the mapping of a ServiceInterface fire and forget method part to an ISignal Triggering. **Tags:** atp.Status=candidate			
Base	*ARObject*, *AbstractSignalBasedToISignalTriggeringMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ServiceInstanceToSignalMapping.fireAndForgetMethodMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataPrototypeIn Method Argument InstanceRef	DataPrototype	0..1	iref	Instance reference to a (potentially structured) member of a ClientServerOperation. **Tags:** atp.Status=candidate **InstanceRef implemented by:** DataPrototypeInService InterfaceInstanceRef
iSignal Triggering	ISignalTriggering	0..1	ref	Reference to an ISignalTriggering being part of a fire and forget message. **Tags:** atp.Status=candidate

Table 13.6: SignalBasedFireAndForgetMethodToISignalTriggeringMapping

13.5 Service Discovery and Static Communication Configuration

Although the ISignalIPdus, which are candidate for signal/service translation, may be communicated to/from CAN or FlexRay networks (where there is no service discovery support available) the messages still might be candidate for service discovery and event subscription on the Ethernet network. In such cases the gateway ECU connecting the Can/FlexRay networks to the Ethernet network is responsible to provide service discovery and event subscription support as part of the gateway operation.

Only SOME/IP supports service discovery in combination with signal/service translation. Whether or not service discovery shall be done in case of the Partial SOME/IP Header approach is defined by the category of the RequiredSomeipServiceInstance or ProvidedSomeipServiceInstance.

Even in cases where SOME/IP service discovery defined by the category of the RequiredSomeipServiceInstance or ProvidedSomeipServiceInstance is configured there may still be a static configurations applied as defined in [TPS_MANI_03295] and [TPS_MANI_03298].

If no SOME/IP service discovery is used, then there is the possibility to define a static configuration as described in section 13.5.4.

[TPS_MANI_03657]{DRAFT} **Signal-service-translation SOME/IP Service Discovery Find** ⌈If there is a RequiredSomeipServiceInstance of category SIGNAL-BASED_WITH_HEADER defined for the signal-service-translation, then this RequiredSomeipServiceInstance shall be handled using SOME/IP service discovery.⌋ *(RS_MANI_00063)*

[TPS_MANI_03658]{DRAFT} **Service-signal-translation SOME/IP Service Discovery Offer** ⌈If there is a ProvidedSomeipServiceInstance of category SIGNAL-BASED_WITH_HEADER defined for the service-signal-translation, then this ProvidedSomeipServiceInstance shall be handled using SOME/IP service discovery.⌋ *(RS_MANI_00063)*

If the source ISignalIPdu in a signal-service-translation scenario is composed of several parts (ISignalGroups and/or individual ISignals) and these parts shall be mapped to several events or field notifiers (according to [TPS_MANI_03667]) and a service discovery shall be performed, then a specific configuration of the SomeipServiceInterfaceDeployment and SomeipEventDeployment is required.

As in case of [TPS_MANI_03654] the SOME/IP Message ID is derived from the SomeipServiceInterfaceDeployment.serviceInterfaceId and the SomeipEventDeployment.eventId, all the events mapped from one ISignalIPdu shall have identical SomeipServiceInterfaceDeployment.serviceInterfaceIds and SomeipEventDeployment.eventIds.

[TPS_MANI_03666]{DRAFT} **Service Discovery for multi-event ISignalIPdus** ⌈If the ISignals of one ISignalIPdu are mapped to more than one events

(using SignalBasedEventElementToISignalTriggeringMapping.iSignalTriggering) or field notifiers (using SignalBasedFieldToISignalTriggeringMapping.notifierSignalTriggering),

then each of the SignalBasedEventElementToISignalTriggeringMappings and SignalBasedFieldToISignalTriggeringMappings referring to ISignals of that ISignalIPdu shall be aggregated by one or more ServiceInstanceToSignalMappings where the reference ServiceInstanceToSignalMapping.serviceInstance refers to a RequiredSomeipServiceInstance

and that RequiredSomeipServiceInstance shall refer to a SomeipServiceInterfaceDeployment where

- each of the referenced SomeipServiceInterfaceDeployments has the identical value for SomeipServiceInterfaceDeployment.serviceInterfaceId defined
- every event taking part in that signal-service-translation has the attribute SomeipEventDeployment.serializer set to SerializationTechnologyEnum.signalBased and has the identical value for:
  - SomeipEventDeployment.eventId
  - SomeipEventDeployment.transportProtocol

⌋(*RS_MANI_00063*)

[TPS_MANI_03659]{DRAFT} **No signal-service-translation SOME/IP Service Discovery Find** ⌈If there is a RequiredSomeipServiceInstance of category SIGNALBASED_NO_HEADER defined for the signal-service-translation, then this RequiredSomeipServiceInstance shall not be handled using SOME/IP service discovery.⌋(*RS_MANI_00063*)

[TPS_MANI_03660]{DRAFT} **No signal-service-translation SOME/IP Service Discovery Offer** ⌈If there is a ProvidedSomeipServiceInstance of category SIGNALBASED_NO_HEADER defined for the service-signal-translation, then this ProvidedSomeipServiceInstance shall not be handled using SOME/IP service discovery.⌋(*RS_MANI_00063*)

13.5.1 Service discovery control

The signal/service translation needs to determine when the translated service (service which originates in signal-based messages) shall actually be offered/subscribed.

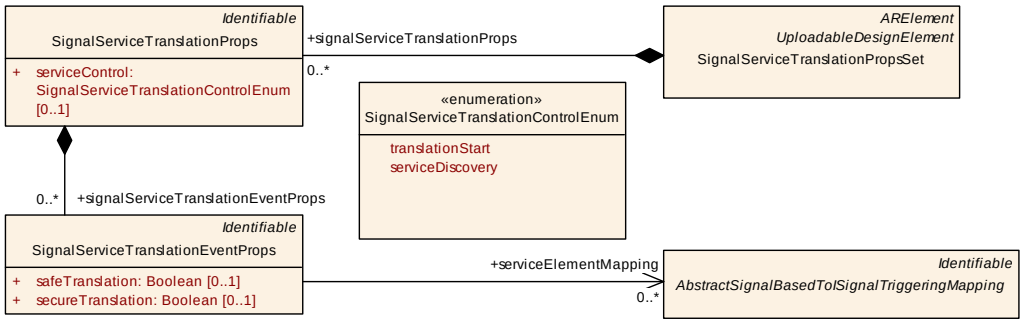


Figure 13.12: Signal/service translation properties

Attributes defining the behavior of `signal/service translation` are available at the `SignalServiceTranslationProps`. The reference `serviceElementMapping` determines to which service instance these settings apply.

Class	**SignalServiceTranslationPropsSet**			
Package	M2::AUTOSARTemplates::CommonStructure::SignalServiceTranslation			
Note	Collection of SignalServiceTranslationProps. **Tags:** atp.recommendedPackage=SignalServiceTranslationProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
signalService TranslationProps	SignalService TranslationProps	*	aggr	Collection of SignalServiceTranslationProps.

Table 13.7: SignalServiceTranslationPropsSet

Class	**SignalServiceTranslationProps**			
Package	M2::AUTOSARTemplates::CommonStructure::SignalServiceTranslation			
Note	This element allows to define the properties which are applicable for the signal/service translation service.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SignalServiceTranslationPropsSet.signalServiceTranslationProps			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
serviceControl	SignalService TranslationControlEnum	0..1	attr	Defines how the service instance control shall behave.
signalService TranslationEventProps	SignalService TranslationEventProps	*	aggr	Defines properties for a single translated event.

Table 13.8: SignalServiceTranslationProps

[constr_10255]{DRAFT} **Multiplicity of attribute `SignalServiceTranslationProps.serviceControl`** ⌈For each `SignalServiceTranslationProps`, the attribute `serviceControl` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

Class	**SignalServiceTranslationEventProps**			
Package	M2::AUTOSARTemplates::CommonStructure::SignalServiceTranslation			
Note	This element allows to define the properties which are applicable for the signal/service translation event.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SignalServiceTranslationProps.signalServiceTranslationEventProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
safeTranslation	Boolean	0..1	attr	Defined whether the translation shall happen in a safe way.
secure Translation	Boolean	0..1	attr	Defined whether the translation shall happen in a secure way.
serviceElement Mapping	AbstractSignalBasedToI SignalTriggering Mapping	*	ref	Reference to the collection of SignalBased to ISignal Triggerung mappings the properties apply to. **Tags:** atp.Status=candidate

Table 13.9: SignalServiceTranslationEventProps

[constr_10230]{DRAFT} **Multiplicity of attribute `SignalServiceTranslationEventProps.safeTranslation`** ⌈For each `SignalServiceTranslationEventProps`, the attribute `safeTranslation` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10231]{DRAFT} **Multiplicity of attribute `SignalServiceTranslationEventProps.secureTranslation`** ⌈For each `SignalServiceTranslationEventProps`, the attribute `secureTranslation` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

At which point in time a specific service instance (originating from `signal/service translation`) is actually offered / subscribed at the service discovery can be defined per service instance:

Possible approaches for service availability/subscription are:

- `translationStart` - right after translation software is started
- `serviceDiscovery` - availability of related service instance

The attribute `SignalServiceTranslationProps.serviceControl` defines the service instance control behavior.

Enumeration	**SignalServiceTranslationControlEnum**
Package	M2::AUTOSARTemplates::CommonStructure::SignalServiceTranslation
Note	This enumeration allows to define how the service instance offer/subscribe control shall behave.
Aggregated by	SignalServiceTranslationProps.serviceControl
Literal	***Description***
serviceDiscovery	Defines the start of service control when other service is available. **Tags:** atp.EnumerationLiteralIndex=2
translationStart	Defines the start of service control at translation start. **Tags:** atp.EnumerationLiteralIndex=0

Table 13.10: SignalServiceTranslationControlEnum

13.5.2 Service control right after translation start

If the availability of the signal-based PDUs is not controlled then the respective translated services offers/subscriptions may be activated immediately at start of the translation software.

[TPS_MANI_03580]{DRAFT} **Service offer at startup** ⌈For a provided translated service instance, if the `SignalServiceTranslationProps.serviceControl` equals `translationStart` then the translation software shall - right after translation software start - *offer* the respective translated service instance.⌋*(RS_MANI_00063)*

[TPS_MANI_03581]{DRAFT} **Service find at startup** ⌈For a required translated service instance, if the `SignalServiceTranslationProps.serviceControl` equals `translationStart` then the translation shall right after translation software start issue the *find* of the respective translated service instance and *subscribe* to its event groups.⌋*(RS_MANI_00063)*

13.5.3 Service control due to availability of related service instance

There are scenarios where the signal-based PDUs are actually controlled using SOME/IP Service Discovery. So there are services defined using events/methods/-fields and the service instances are offered / subscribed using the SOME/IP Service Discovery, just the payload of such services is not serialized according to the SOME/IP transformer rules (or a subset of events uses signal-based serialization). Therefore, a `signal/service translation` is required for the payload.

13.5.3.1 Signal-service-translation

[TPS_MANI_03582]{DRAFT} **Service find for required signal** ⌈For a required translated service instance, if the `SignalServiceTranslationProps.serviceControl` equals `serviceDiscovery` then upon startup the translation software component shall issue a *service find* for the `RequiredSomeipServiceInstance` which is referenced by the `ServiceInstanceToSignalMapping` in the role `serviceInstance`.⌋*(RS_MANI_00063)*

[TPS_MANI_03583]{DRAFT} **Service subscribe for required signal** ⌈For a required translated service instance, if the `SignalServiceTranslationProps.serviceControl` equals `serviceDiscovery` and the *service find* of [TPS_MANI_03582] was successful then the translation software component shall issue a *subscribe* to all `SomeipServiceInterfaceDeployment.eventGroup`s.⌋*(RS_MANI_00063)*

13.5.3.2 Service-signal-translation

[TPS_MANI_03606]{DRAFT} **Service offer for provided signal** ⌈For a provided translated service instance, if the `SignalServiceTranslationProps.serviceControl` equals `serviceDiscovery` then upon startup the translation software component shall issue an *offer* for the `ProvidedSomeipServiceInstance` which is referenced by the `ServiceInstanceToSignalMapping` in the role `serviceInstance`.⌋*(RS_MANI_00063)*

13.5.4 Static Communication Configuration

In case no service discovery shall be applied, it is possible to define a static configuration for non SOME/IP communication. The actual representation of the required information is not standardized, thus the information can be provided using the model extension mechanism, as exemplified in appendix C.

13.5.4.1 Static Communication Configuration for the Server

If a `ProvidedUserDefinedServiceInstance` shall be statically configured for communication on Ethernet, then several elements need to be defined:

- `localUdpPortNumber`: a port number for the local UDP port needs to be defined on which the service will be provided using UDP protocol.
- `localTcpPortNumber`: a port number for the local TCP port needs to be defined on which the service will be provided using TCP protocol.
- `UserDefinedServiceInstanceToMachineMapping`: a mapping needs to be defined (for the local connection where the service will be provided) that associates the `ProvidedUserDefinedServiceInstance` to an `EthernetCommunicationConnector` (VLAN).
- `remotePeers`: a list of remote IP-Address and Port combinations needs to be defined where the messages will be sent to. A mix of unicast and multicast IP-Addresses is supported.
- `udpCollectionBufferTimeout`: Maximum time, an outgoing message may be delayed, due to data collection.
- `udpCollectionTrigger`: Defines whether the element contributes to the triggering of the UDP data transmission if data collection is enabled.
- `udpCollectionBufferSizeThreshold`: Specifies the amount of data in bytes that shall be buffered for data transmission over the UDP connection in case data collection is enabled.

13.5.4.2 Static Communication Configuration for the Client

If a RequiredUserDefinedServiceInstance shall be statically configured for communication on Ethernet, then several elements need to be defined:

- localUdpPortNumber: a port number for the local UDP port needs to be defined on which the service will be consumed using UDP protocol.
- localTcpPortNumber: a port number for the local TCP port needs to be defined on which the service will be consumed using TCP protocol.
- UserDefinedServiceInstanceToMachineMapping: a mapping needs to be defined (for the local connection where the service will be consumed) that associates the RequiredUserDefinedServiceInstance to an EthernetCommunicationConnector (VLAN).
- eventMulticastUdpPort: a port number for the local UDP port in case of multicast reception may be defined.
- multicastIpAddress: an IP-Address where the events will be consumed in case of multicast reception.
- remotePeer: an IP-Address and Port combination needs to be defined for the remote provider of the service.

13.6 Translation behavior

The signal/service translation is defined as a network binding for the Communication Management [10] and it is not specified at which exact point in time or in which context the data transformation (i.e. signal-service-translation or service-signal-translation) will be executed.

The behavior of signal-service-translation is governed by the application software which calls ara::com-APIs to interact with the communication management.

13.6.1 ServiceInterface representation of translation data

Due to update-bits (see section 13.6.2) and data filtering (see section 13.6.3), the ServiceInterface – which is used to represent the signal/service translated information – needs to be able to represent partially available information. For such cases, the AUTOSAR data type systems has defined the isOptional feature of structured data (see also the Software Component Template [1] for details).

For ApplicationDataTypes, the optionality is configured at the ApplicationRecordElement.isOptional attribute.

For CppImplementationDataTypes, the optionality is configured at the CppImplementationDataTypeElement.isOptional attribute.

The goal of the signal/service translation mapping is to define the proper ServiceInterface representation for the corresponding ISignalIPdu, respecting update bits and filtering.

13.6.1.1 Optional elements in case signal-service-translation

The approach for the signal-service-translation direction is: if a received ISignal or ISignalGroup has an update-bit defined (see section 13.6.2) or the reception is guarded by a reception-filter (see section 13.6.3), then the corresponding element in the ServiceInterface event shall be defined as ApplicationRecordElement.isOptional (resp. CppImplementationDataTypeElement.isOptional).

Please find an example for the handling of optional elements in signal/service translation in Appendix B.10.2.1.

13.6.1.2 Optional elements in case service-signal-translation

The approach for the service-signal-translation direction is: if the ServiceInterface event has a member with CppImplementationDataTypeElement.isOptional, then the corresponding ISignal or ISignalGroup shall have an update-bit defined. During runtime, it will be checked which optional elements are actually defined in the event / field and the corresponding values are put into the ISignalIPdu. Also the update-bits are set accordingly.

Please find an example for the handling of optional elements in signal/service translation in Appendix B.10.2.2.

13.6.2 Event and update-bit semantics

A variety of sources may exist where the ISignalIPdus originate that are candidates for signal/service translation. And the ISignalIPdus produced by the signal/service translation may be routed to many different targets in the vehicle network. It is essential to understand the semantics of the received and sent data in the scope of the signal/service translation.

Often ISignalIPdus are transported using periodic transmission in order to not only transport data but also have a liveliness indication of the information producer and the communication path. If the sender would just transmit an ISignalIPdu when the value changes, a receiver would not be able to detect whether the sender just did not send anything because there is no new information or whether the sender (or the transport path) has a malfunction and therefore an intended sent message was not able to get through.

In the simplest case, the arrival of one ISignalIPdu actually represents the functional reception of all the included information. This means that the resulting event (resp. field) for the application software can be derived from the reception of that ISignalIPdu. In the opposite direction, the notification of an event (resp. field) from the application software results in the sending of the signal/service translation ISignalIPdu.

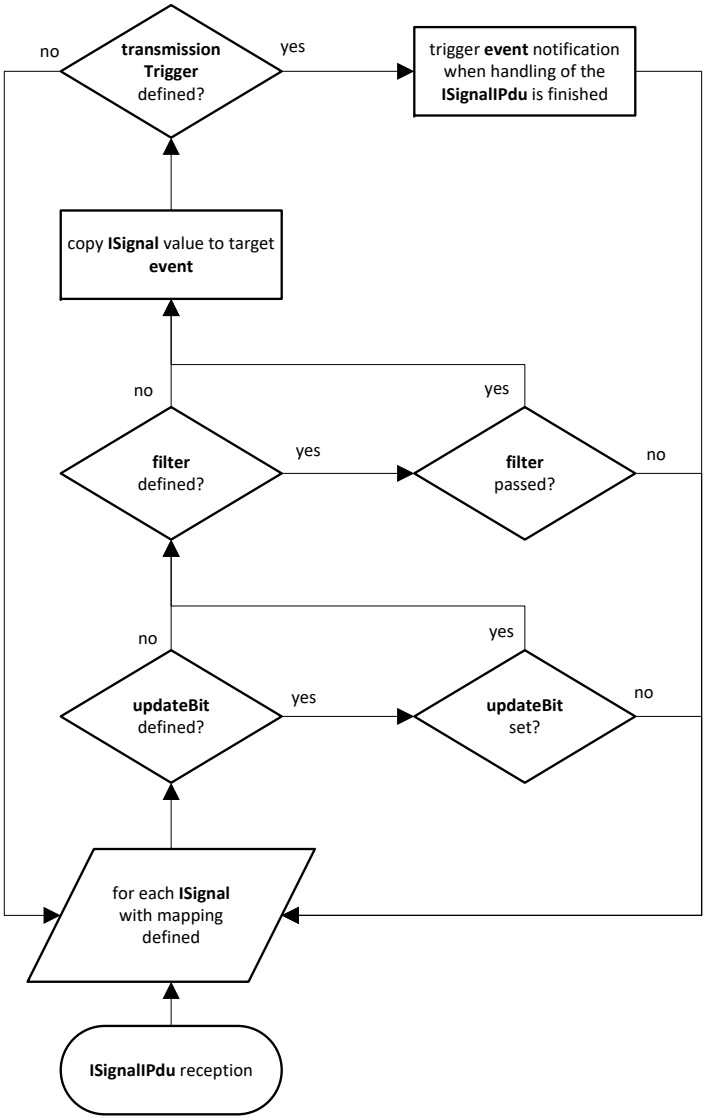


Figure 13.13: Processing order reception

But there are scenarios where the layout of the ISignalIPdu is composed of information from different sources or are produced at different points in time. In order to indicate which parts of the ISignalIPdu actually have been updated and which parts have just been send because they are part of the ISignalIPdu, the *update-bits* are

used. These update-bits are part of the ISignalIPdu payload and indicate whether the associated information has actually been sent intentionally or just has been sent because the ISignalIPdu was sent as a whole.

If no update-bits are used, then every arrival of an ISignalIPdu has to be considered as being updated by the producer as a whole.

If update-bits are used, then not all of the included information may actually have been produced by the sender. In an extreme case – if all information of the ISignalIPdu has associated update-bits – all of the update-bits might not be set, resulting in reception of an ISignalIPdu but without any effect to the signal-service-translation.

Another way to influence the translation behavior is the usage of reception data filters during signal-service-translation (see also section 13.6.3). This also may lead to the situation that the reception of an ISignalIPdu only triggers a subset of the contained information (or none at all).

The third check during signal-service-translation is the evaluation of the SignalBasedEventElementToISignalTriggeringMapping.transmissionTrigger (resp. SignalBasedFieldToISignalTriggeringMapping.transmissionTrigger). If an ISignal was actually taken to update the resulting event, the transmissionTrigger defines whether that update actually causes the notification of the mapped event.

One use-case may be to define which signal-service-translation elements actually cause the sending of an event. So if – due to update-bits and filters – only least important elements have been translated, it can be defined that in such cases the event will not be triggered.

[constr_3644]{DRAFT} **No transmissionTrigger support for service-signal-translation direction** ⌈If a ServiceInstanceToSignalMapping.serviceInstance refers to a ProvidedApServiceInstance then

- every SignalBasedEventElementToISignalTriggeringMapping aggregated in the role eventElementMapping and
- every SignalBasedFieldToISignalTriggeringMapping aggregated in the role fieldMapping

shall not have a SignalBasedEventElementToISignalTriggeringMapping.transmissionTrigger (respectively SignalBasedFieldToISignalTriggeringMapping.transmissionTrigger) defined **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_03646]{DRAFT} **Definition of transmission triggers for signal-service-translation** ⌈The attribute SignalBasedEventElementToISignalTriggeringMapping.transmissionTrigger, respectively SignalBasedFieldToISignalTriggeringMapping.transmissionTrigger, defines which translation parts contribute to the transmission triggering for the mapped payload.⌋ *(RS_MANI_00063)*

[TPS_MANI_03647]{DRAFT} **Full translation before transmission triggering** ⌈In case there has been a transmission trigger caused by a source signal, the signal-service-translation shall first process all other mapped source signals for that target before sending out the target.⌋*(RS_MANI_00063)*

[TPS_MANI_03648]{DRAFT} **Transmission trigger for signal-service-translation** ⌈If the attribute SignalBasedEventElementToISignalTriggeringMapping.transmissionTrigger, respectively SignalBasedFieldToISignalTriggeringMapping.transmissionTrigger, equals *true* then the translation of the respective source signal causes the sending of the target (after all mapped sources have been processed, see [TPS_MANI_03647]).⌋*(RS_MANI_00063)*

[TPS_MANI_03649]{DRAFT} **No transmission trigger for signal-service-translation** ⌈If the attribute SignalBasedEventElementToISignalTriggeringMapping.transmissionTrigger, respectively SignalBasedFieldToISignalTriggeringMapping.transmissionTrigger, is not defined or has the value *false*, then the translation of that respective source signal does not cause the sending of the target.⌋*(RS_MANI_00063)*

Note that in case none of the SignalBasedEventElementToISignalTriggeringMapping or SignalBasedFieldToISignalTriggeringMapping contributing to the target of a signal-service-translation has the attribute SignalBasedEventElementToISignalTriggeringMapping.transmissionTrigger (respectively SignalBasedFieldToISignalTriggeringMapping.transmissionTrigger) set to *true* then the translation target will not be sent.

The combination of update-bits, filters, and transmission triggers may be configured for one ISignal – resulting in a sequential processing according to figure 13.13.

13.6.3 Data filtering

Filtering can be used to potentially reduce the amount of generated event notifications in case of signal-service-translation direction. There is no filtering support available for the service-signal-translation direction, although the transmission mode selection can be used to influence the amount of sent ISignalIPdus (see section 13.8.5).

[constr_3643]{DRAFT} **No filter support for service-signal-translation direction** ⌈If a ServiceInstanceToSignalMapping.serviceInstance refers to a ProvidedApServiceInstance then

- every SignalBasedEventElementToISignalTriggeringMapping aggregated in the role eventElementMapping and
- every SignalBasedFieldToISignalTriggeringMapping aggregated in the role fieldMapping

shall not have a SignalBasedEventElementToISignalTriggeringMapping.filter (resp. SignalBasedFieldToISignalTriggeringMapping.filter) defined **at the time when the creation of the manifest is finished**.⌋ *()*

If a data filtering shall be applied during the signal-service-translation then it is possible to define a DataFilter at the SignalBasedEventElementToISignalTriggeringMapping and SignalBasedFieldToISignalTriggeringMapping in the role filter.

[TPS_MANI_03621]{DRAFT} **Data filtering during the signal-service-translation** ⌈If there is a SignalBasedEventElementToISignalTriggeringMapping.filter (resp. SignalBasedFieldToISignalTriggeringMapping.filter) defined this filtering shall be implemented during the signal-service-translation.

This is the only supported application of filters for signal-service-translation.⌋*(RS_MANI_00063)*

[TPS_MANI_03644]{DRAFT} **Supported reception data filters** ⌈The following DataFilter settings are supported by the signal-service-translation:

- always
- maskedNewEqualsX
- maskedNewDiffersX
- maskedNewDiffersMaskedOld

⌋*(RS_MANI_00063)*

Note that the setting of maskedNewDiffersMaskedOld requires the translation software to keep the previous (*old*) value for comparison.

The possibility to define an always filter allows to specify a filter for each mapped signal, even if that particular signal shall not contribute to the filter result - as it always yields *passed*. This allows to define a full set of filters and thus gives the possibility to check for complete filter definitions.

The usage of a *never* filter is not supported by the signal-service-translation. Such a filter would always yield *not passed*, which is equivalent to not having a SignalBasedEventElementToISignalTriggeringMapping defined at all for this specific ISignal.

Potentially there could be several SignalBasedEventElementToISignalTriggeringMapping.filters (resp. SignalBasedFieldToISignalTriggeringMapping.filters) defined in the context of one target element (in the example figure 13.14 the *event xyz* the *x1* and *z*).

In such a case ALL of the defined filters for the scoped filtered element have to pass in order to let the signal/service translation actually get effective. For

the optional element *x1* to actually be set both, the *filter (a)* and *filter (c)*, have to pass. The optional element *z* will only be set if the *filter (z)* passes.

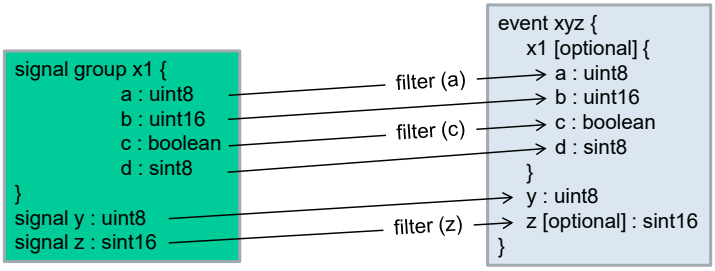


Figure 13.14: Filter example during `signal-service-translation`

In case the target event of the `signal-service-translation` has no optional elements defined and at least one filter does not pass then the whole event is not sent. This is illustrated in figure 13.15, where the *event xyz* is considered to be the scoped element.

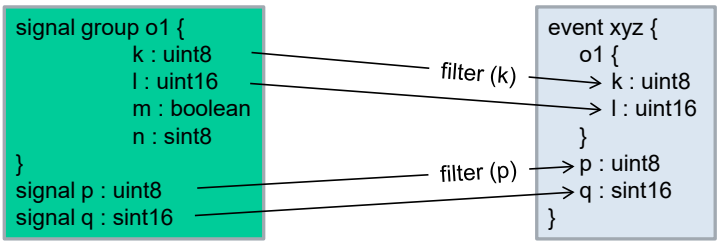


Figure 13.15: Filter example during `signal-service-translation` with no optional target elements

The `scoped filtered element` is defined as either

- the filtered element, if it is defined as *optional*
- the closest *optional* element traversing from the filtered element and going towards the root element
- the root element itself, if there is no *optional* element traversing from the filtered element and going towards the root element.

[TPS_MANI_03645]{DRAFT} **Applicability of a filter to the `scoped filtered element`** ⌈If a filter does not pass then the `scoped filtered element` shall not be considered for `signal-service-translation`.⌋ *(RS_MANI_00063)*

13.6.4 Translation from one source

The simplest translation approach is a one source translation. If the translation target is primitive then it has by nature only one source. If the translation target is a composite data then it depends on the structure of the sources whether a one source translation is possible.

[TPS_MANI_03620]{DRAFT} **Service discovery control** ⌈If the service discovery control is enabled for a specific service instance then any payload coming / going to that service instance shall be translated and forwarded to the mapped output(s).⌋ *(RS_MANI_00063)*

Every time an input signal or event arrives it is translated to the mapped output path. Since this is a one source translation the arrival of the input signal or event is the trigger for the translation. Therefore, if all the translation mappings have been performed the target payload can be sent out.

The example in figure 13.16 illustrates the full approach of a `signal-service-translation` with application component forwarding of the payload.

On the reception side, the `ISignalIPdu` *uv1* is the source of the `signal-service-translation`. The payload of the `ISignalIPdu` is translated into an instance of the `event` *uv2*. The `event` is part of a `ServiceInterface` that types the `RPort-Prototype` which defines the ara::com API at that `RPortPrototype`.

It is the job of the application component code to use the ara::com API and fetch the information from the `RPortPrototype` and forward it to the `PPortPrototype`. In this example, it is not further defined how that forwarding is implemented, but for such a simple use-case the usage of a `PassThroughSwConnector` could be used to express the desired forwarding. For further details on this approach, refer to section 13.7. Of course, also a custom code implementation can be used to fetch the data and forward it.

At the `PPortPrototype` a `ServiceInterface` with an `event` *uv3* is defined. This is the basis to provide the information is a service-oriented manner to further software components. What kind of network binding is used for the transport of the `PPort-Prototype` payload is not in scope of this example. However, the `PPortPrototype` could be mapped to any available kind of network binding available (SOME/IP, DDS, IPC, or even another `service-signal-translation`).

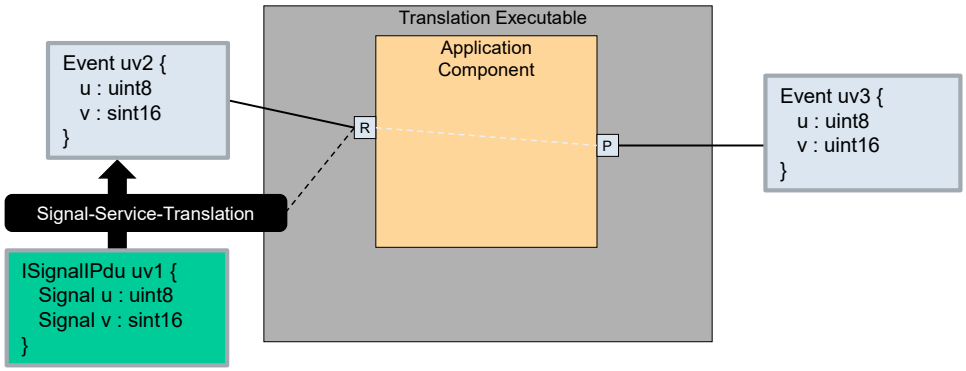


Figure 13.16: Example `signal-service-translation` and forwarding from one source

13.6.5 Translation from several sources

There is also the use-case to support the composition of data from different sources. However, because the sources will most likely not be received in a coordinated way, it would be hard to define an automatic behavior based on the usage of PassThroughSwConnectors for this setup. Thus, for the usage of PassThroughSwConnectors, it is required by the signal/service translation that the input for the signal/service translation comes from one source.

The merging of data from several sources can be implemented on application component level and still utilize the signal/service translation features to interface with ISignals and ISignalIPdus. The code, however, which reads several source data and produces a consistent set of information is considered to be custom code.

An example for signal/service translation from different sources can be found in Appendix B.10.2.3.1. Conversely, an example for service/signal translation from different sources is documented in Appendix B.10.2.3.2

13.6.6 Signal-Service-Translation to several targets

If the source ISignalIPdu in a signal-service-translation scenario is composed of several parts (ISignalGroups and/or individual ISignals), then it is possible to map the parts of that ISignalIPdu to several targets (events or field notifiers).

[TPS_MANI_03667]{DRAFT} **Mapping from one ISignalIPdu to several events in case of signal-service-translation** ⌈The ISignals of one ISignalIPdu can be mapped to one or more events (using SignalBasedEventElementToISignalTriggeringMapping.iSignalTriggering) or field notifiers (using SignalBasedFieldToISignalTriggeringMapping.notifierSignalTriggering).⌋ *(RS_MANI_00063)*

The individual SignalBasedEventElementToISignalTriggeringMappings and SignalBasedFieldToISignalTriggeringMappings will define in their DataPrototypeInServiceInterfaceRef the target of the signal-service-translation.

One scenario mandates the mapping of parts of an ISignalIPdu to different events: If the source ISignalIPdu is composed of several parts (ISignalGroups and/or individual ISignals) and at least one of the ISignalGroups is end-to-end protected.

In such cases it is required to get the end-to-end status per protected ISignalGroup, thus merging the whole ISignalIPdu's content into one event does not suffice, as only a part of the ISignalIPdu is actually end-to-end protected, and the ara::com status APIs are available on an event level.

The modeling approach according to section 13.4.3 results in one or more ServiceInstanceToSignalMappings which distribute the content of one ISignalIPdu to several events.

An example for the signal/service translation to several targets can be found in Appendix B.10.2.4.

13.7 Translation pass-through composition

In case a *Pass-through Composition* is defined (see figure 13.3) the behavior for the defined pass-through definition shall be implemented. It is well possible to automatically generate the implementation of such a pass-through behavior out of the information given in the *Pass-through Composition*.

The mapping approach is already used in section 3.3.2 where the `PassThroughSwConnector` is used to define the matching `PortPrototype`s for the *facade* use-case. Specifically [constr_5056] applies as well for the `signal/service translation`.

SOME/IP services may consist of a mixture of events, fields and methods. While events and field notifiers are (potentially) subject to `signal/service translation` it is well-defined that methods (methods, getter- and setter-methods, and fire-and-forget methods) are serialized according to the SOME/IP serialization rules.

However, the mixed nature of a specific service instance makes it necessary that methods also have to be considered in the *Pass-through Composition*. It is not so much about the translation of the payload serialization, but the general wrapping of events, fields and methods in one service instance makes it necessary that methods get *passed through* as well.

13.8 Expected features of Classic platform

Classic AUTOSAR SWS-COM Specification provides a variety of possibilities to pack signals into PDUs - from a structural as well as from a behavioral perspective. Also, the further layers in the COM-Stack may influence the PDUs layout.

The goal of this section is to specify which of these Classic COM-Stack features shall also be available on the Adaptive Platform `signal/service translation`.

13.8.1 Processing order

For the features of the Classic platform COM-Stack there is a well-defined processing order in which the actions are performed upon Tx and Rx of data. It is required for the `signal/service translation` to ensure the same processing order.

[TPS_MANI_03585]{DRAFT} **Processing order of COM-Stack features** ⌈For the COM-Stack features the `signal/service translation` supports the processing order shall be the same as on the Classic platform.⌋ *(RS_MANI_00063)*

13.8.2 Reception data filter

The reception filter support for signal-service-translation is defined in section 13.6.3.

The potentially defined reception filters at an ISignalPort shall be ignored, as [TPS_MANI_03621] is the only supported way to define filters for signal-service-translation.

[TPS_MANI_03589]{DRAFT} **Reception data filter of COM-Stack** ⌈If the ISignal-Triggering refers to an ISignalPort and that ISignalPort has a dataFilter defined then the signal-service-translation shall ignore the filter definition.⌋ *(RS_MANI_00063)*

13.8.3 Reception of invalid signal

[TPS_MANI_03592]{DRAFT} **ISignal invalidation of COM-Stack** ⌈If the ISignal-Triggering refers to an ISignalPort with communicationDirection equal in then the following values for ISignalPort.handleInvalid shall be supported by the signal/service translation:

- dontInvalidate
- replace

⌋*(RS_MANI_00063)*

[TPS_MANI_03593]{DRAFT} **handleInvalid = dontInvalidate behavior of COM-Stack** ⌈If the ISignalTriggering refers to an ISignalPort with communi-cationDirection equal in and the ISignalPort.handleInvalid equals don-tInvalidate then the signal/service translation shall not perform any in-validation handling.⌋*(RS_MANI_00063)*

[TPS_MANI_03594]{DRAFT} **handleInvalid = replace behavior of COM-Stack** ⌈If the ISignalTriggering refers to an ISignalPort with communicationDi-rection equal in and the ISignalPort.handleInvalid equals replace and the received value of the ISignal equals the ISignal.networkRepresentation-Props.invalidValue then the signal/service translation shall *replace* the invalidValue with the ISignal.initValue.⌋*(RS_MANI_00063)*

13.8.4 Update Bit handling

For details on the handling of update bits refer to section 13.6.

[TPS_MANI_03595]{DRAFT} **Update Bit support for ISignal** ⌈If the ISignal-Triggering refers to an ISignalPort with communicationDirection set to the value in and the ISignalTriggering refers to an ISignal and that ISignal is

mapped into an ISignalIPdu with a ISignalToIPduMapping.updateIndicationBitPosition defined, then the following rules apply:

- If the received *update bit* is *true*, then the respective ISignal shall be considered for reception.
- If the received *update bit* is *false*, then the respective ISignal shall not be considered for reception.

⌋(*RS_MANI_00063*)

13.8.5 Transfer properties and transmission modes for Service-Signal-Translation

[TPS_MANI_03590]{DRAFT} **Transfer properties and transmission modes of COM-Stack** ⌈If the ISignalTriggering refers to an ISignalPort with communicationDirection equal out and the ISignalTriggering refers to an ISignal and that ISignal is mapped into an ISignalIPdu with an iPduTimingSpecification and the IPduTiming has a transmissionModeDeclaration defined and the TransmissionModeDeclaration has a transmissionModeCondition defined and the following values for TransmissionModeCondition.dataFilter shall be supported by the service-signal-translation:

- always
- never
- maskedNewEqualsX
- maskedNewDiffersX

⌋(*RS_MANI_00063*)

Class	**TransmissionModeCondition**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication::Timing			
Note	Possibility to attach a condition to each signal within an I-PDU. If at least one condition evaluates to true, TRANSMISSION MODE True shall be used for this I-Pdu. In all other cases, the TRANSMISSION MODE FALSE shall be used.			
Base	*ARObject*			
Aggregated by	TransmissionModeDeclaration.transmissionModeCondition			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataFilter	DataFilter	0..1	aggr	Possibilities to define conditions
iSignalInIPdu	ISignalToIPduMapping	0..1	ref	Reference to a signal to which a condition is attached.

Table 13.11: TransmissionModeCondition

Class	**TransmissionModeDeclaration**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication::Timing			
Note	AUTOSAR COM provides the possibility to define two different TRANSMISSION MODES (True and False) for each I-PDU. As TransmissionMode selector the signal content can be evaluated via transmissionModeCondition (implemented directly in the COM module) or mode conditions can be defined with the modeDrivenTrue Condition or modeDrivenFalseCondition (evaluated by BswM and invoking Com_SwitchIpduTxMode COM API). If modeDrivenTrueCondition and modeDrivenFalseCondition are defined they shall never evaluate to true both at the same time. The mixing of Transmission Mode Switch via API and signal value is not allowed.			
Base	*ARObject*			
Aggregated by	IPduTiming.transmissionModeDeclaration			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
modeDriven FalseCondition	ModeDriven TransmissionMode Condition	*	aggr	Defines the trigger for the Com_SwitchIpduTxMode Transmission Mode switch. Only if all defined modeDriven FalseConditions evaluate to true (AND associated) the transmissionModeFalseTiming shall be activated. mode DrivenTrueCondition and modeDrivenFalseCondition shall never evaluate to true both at the same time.
modeDriven TrueCondition	ModeDriven TransmissionMode Condition	*	aggr	Defines the trigger for the Com_SwitchIpduTxMode Transmission Mode switch. Only if all defined modeDriven TrueConditions evaluate to true (AND associated) the transmissionModeTrueTiming shall be activated. mode DrivenTrueCondition and modeDrivenFalseCondition shall never evaluate to true both at the same time.
transmission ModeCondition	TransmissionMode Condition	*	aggr	The Transmission Mode Selector evaluates the conditions for a subset of signals and decides which transmission mode should be used. In case only one transmission mode is used there is no need for the "TransmissionMode Condition" and its sub-structure. In case the transmission mode shall be switched using the COM-API "Com_Switch IpduTxMode" there is no need for the "TransmissionMode Condition" and its sub-structure.
transmission ModeFalse Timing	TransmissionMode Timing	0..1	aggr	Timing Specification if the COM Transmission Mode is false. The Transmission Mode Selector is defined to be false, if all Conditions evaluate to false.
transmission ModeTrue Timing	TransmissionMode Timing	0..1	aggr	Timing Specification if the COM Transmission Mode is true. The Transmission Mode Selector is defined to be true, if at least one Condition evaluates to true.

Table 13.12: TransmissionModeDeclaration

13.8.6 Deadline monitoring

The application software is responsible for the deadline monitoring.

13.8.7 Signal and IPdu Transmission

The application software is responsible for the periodic production of data and triggering of sending.

13.8.8 IPdu multiplexing

On low-payload networks there is the usage of IPdu Multiplexing to handle the limited number of identifiers for the respective network transport (e.g. CAN-Ids). The potential alternatives of a `MultiplexedIPdu` shall be represented as optional alternative `CppImplementationDataType` of `category STRUCTURE` in the `ServiceInterface` event or field.

[TPS_MANI_03597]{DRAFT} **Support for `MultiplexedIPdu`** ⌈The `signal/service translation` shall support the handling of `MultiplexedIPdu` defined payload. The support shall be available for sending and receiving of `MultiplexedIPdu`s.⌋*(RS_MANI_00063)*

[TPS_MANI_03638]{DRAFT} **Mapping of `MultiplexedIPdu`** ⌈The content of the `MultiplexedIPdu` shall be mapped to a `CppImplementationDataType` of `category STRUCTURE` consisting of the elements:

- `MultiplexedIPdu` static part: directly mapped into the top-level structure
- `MultiplexedIPdu` dynamic part: each dynamic part alternative shall be mapped inside a `CppImplementationDataTypeElement` of `category STRUCTURE`

⌋*(RS_MANI_00063)*

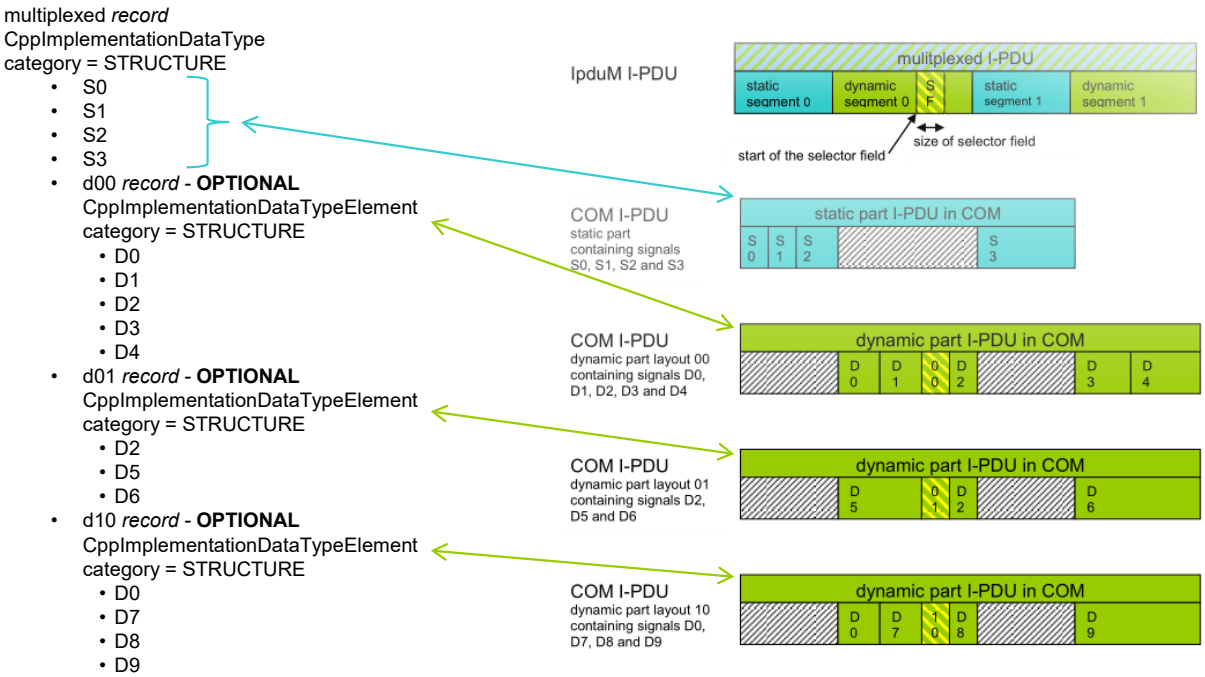


Figure 13.17: Example mapping for `MultiplexedIPdu`

[TPS_MANI_03639]{DRAFT} **Mapping of `MultiplexedIPdu` static part** ⌈The content of the `MultiplexedIPdu` static part shall be mapped to the a `CppImplementationDataType` of `category STRUCTURE` defined for the whole `MultiplexedIPdu` in [TPS_MANI_03638].⌋*(RS_MANI_00063)*

[TPS_MANI_03640]{DRAFT} **Mapping of `MultiplexedIPdu` dynamic part** ⌈The content of each `MultiplexedIPdu` dynamic part shall be mapped to one of the optional alternatives of the `CppImplementationDataTypeElement` of `category STRUCTURE` with `CppImplementationDataTypeElement.isOptional` set to *true* for each dynamic part in [TPS_MANI_03638].⌋*(RS_MANI_00063)*

Note that even in cases where there is just one member mapped from a `MultiplexedIPdu` dynamic part there shall still be a wrapping `CppImplementationDataTypeElement` of `category STRUCTURE` defined according to [TPS_MANI_03640].

[TPS_MANI_03641]{DRAFT} **No mapping of `MultiplexedIPdu` selector field value** ⌈The selector field value of a `MultiplexedIPdu` shall not be mapped in the context of `signal-service-translation` (or `service-signal-translation`).⌋*(RS_MANI_00063)*

In order to keep the event or field of the `ServiceInterface` status consistent there shall only be exactly one of the optional alternatives used.

[TPS_MANI_03642]{DRAFT} **Exactly one alternative defined for mapped event / field during runtime** ⌈Exactly one option (as defined in [TPS_MANI_03640] for the dynamic part) shall be defined in the event / field representing the `MultiplexedIPdu` during runtime.⌋*(RS_MANI_00063)*

In figure 13.17 an example for the mapping between a `MultiplexedIPdu` and the composite data type is illustrated.

13.9 End-to-End considerations

The `signal/service translation` on Adaptive platform follows a single-sided approach on ara::com. To the application software there is no difference whether the service is communicated using SOME/IP or DDS, or whether the service originates in a `signal/service translation`. As the translation is running in the same `Process` as the application software itself, the application software has direct access to the translated service payload.

In case there is an E2E header attached and/or a secure communication defined the translation software needs to check the validity (for reception) or calculate the CRC/-MAC (for sending) of signal-based payload.

The information whether and which E2E / Security profile is configured is defined in the System Template [18] part belonging to the mapping information of sections 13.4.3 and 13.4.5.

The assignment of the `ISignal`s to `ISignalIPdu`s which in turn are assigned to `SecuredIPdu`s defines the security aspects.

The assignment of `ISignal`s to `EndToEndTransformationDescription`s define the safety aspects.

13.9.1 Safety

The attribute `SignalServiceTranslationEventProps.safeTranslation` is used to explicitly require that the translation shall be configured in a safe transport and that the translation software shall handle the translation activity in an end-to-end preserving way.

[TPS_MANI_03607]{DRAFT} **Handling of safe `signal/service translation` in one `Executable`** ⌈It is required that the `signal-service-translation` (and `service-signal-translation`) of one Service/SignalGroup pair which are mapped to each other, shall be handled in one `Executable` to also cover a closed mapping from one E2E profile to another, if necessary.

The `signal/service translation` of different (independent) Services/SignalGroups may be handled by different `Executable`s.⌋*(RS_MANI_00063)*

[TPS_MANI_03608]{DRAFT} **Support for safe `signal/service translation`** ⌈The translation of E2E protected data shall be supported in both directions, `signal-service-translation` and `service-signal-translation`.⌋*(RS_MANI_00063)*

[TPS_MANI_03609]{DRAFT} **Support for safe `signal/service translation` with same or different E2E profiles** ⌈The translation of E2E protected data shall support the occurrence of

- the same E2E profile on both sides of the communication and
- different E2E profiles on each side of the communication.

⌋*(RS_MANI_00063)*

[TPS_MANI_03610]{DRAFT} **1:n mapping for E2E protected data** ⌈It shall be possible to map the same E2E protected source data to several E2E protected target data (1:n).⌋*(RS_MANI_00063)*

[TPS_MANI_03611]{DRAFT} **E2E protected target out of E2E protected sources** ⌈The content of one E2E protected target shall only be composed out of data from E2E protected sources.⌋*(RS_MANI_00063)*

The rationale for [TPS_MANI_03611] is to support the use-case where target data shall be E2E protected and it is composed of several sources.

[TPS_MANI_03614]{DRAFT} **No translation of not OK E2E protected data out of several sources** ⌈If a E2E protected source data is mapped into a composed E2E protected target data (according to [TPS_MANI_03611]) and
if the E2E-Check for the source data returns any E2E error (not *E_OK*) then
this source data shall not be forwarded to the respective target data and (if applicable) shall not trigger the transmission of the target.⌋*(RS_MANI_00063)*

If source data is not verified as *E_OK* it is not translated. If the translated E2E protected data comes from several sources there may occur correlation and synchronicity issues during translation.

[TPS_MANI_03612]{DRAFT} **Sufficient ASIL level of translation software** ⌈If the SignalServiceTranslationEventProps.safeTranslation equals true then the implementation of the translation software shall fulfill a sufficient ASIL.⌋*(RS_MANI_00063)*

[constr_3554]{DRAFT} **E2E protection configuration check** ⌈If the SignalServiceTranslationEventProps.safeTranslation equals true then the signal-based payload shall have an EndToEnd profile defined **at the time when the creation of the manifest is finished**.⌋*()*

The current EndToEnd profiles for Classic platform rely on a periodic communication paradigm. For the translation software of the Adaptive platform this requires to know the specified period the payload has to be updated / checked for.

[TPS_MANI_03598]{DRAFT} **Expected check period of E2E-Protected payload** ⌈If the RPortPrototype has a ReceiverComSpec.receptionProps.dataUpdatePeriod defined for an event then the application software calling the ara::com APIs shall check periodically for updates using the specified period.⌋*(RS_MANI_00063)*

[TPS_MANI_03599]{DRAFT} **Expected update period of E2E-Protected payload** ⌈If the PPortPrototype has a SenderComSpec.transmissionProps.dataUpdatePeriod defined for an event then the application software calling the ara::com APIs shall periodically update the event using the specified period.⌋*(RS_MANI_00063)*

Class	**ReceptionComSpecProps**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	This meta-class defines a set of reception attributes which the application software is assumed to implement.			
Base	*ARObject*			
Aggregated by	*ReceiverComSpec*.receptionProps			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dataUpdatePeriod	TimeValue	0..1	attr	This attribute defines the period in which the application shall check for updated data. This attribute is used for the configuration of the E2E protection, but may also indicate a general data reception period.
timeout	TimeValue	0..1	attr	This attribute defines the time interval after which the application shall assume that the to be received data reception has timed out, i.e. the respective data has not been received for that amount of time.

Table 13.13: ReceptionComSpecProps

Class	**TransmissionComSpecProps**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	This meta-class defines a set of transmission attributes which the application software is assumed to implement.			
Base	*ARObject*			
Aggregated by	*SenderComSpec*.transmissionProps			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

▽

△

Class	**TransmissionComSpecProps**			
dataUpdate Period	TimeValue	0..1	attr	This attribute defines the period in which the application is assumed to transmit the respective data.
minimumSend Interval	TimeValue	0..1	attr	This attribute defines the minimum interval between two consecutive transmissions of the respective data the application is assumed to ensure.
transmission Mode	TransmissionMode DefinitionEnum	0..1	attr	The attribute defines the mode in which the application is assumed to transmit the respective data.

Table 13.14: TransmissionComSpecProps

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

Table 13.15: TransmissionModeDefinitionEnum

13.9.1.1 Signal-service-translation

[TPS_MANI_03600]{DRAFT} **Signal-service-translation of E2E protected payload** ⌈Signal-service-translation shall check the end-to-end status of every received signal-based payload. If the E2E-Check returns *E_OK* for the received payload then the data shall be forwarded to the respective sending of the translation software.⌋*(RS_MANI_00063)*

Error handling:

[TPS_MANI_03601]{DRAFT} **Signal-service-translation of E2E protected payload - timeout handling** ⌈If no message is received within the specified message cycle time (timeout is detected), then no data shall be transmitted to the service-based part.⌋ *(RS_MANI_00063)*

[TPS_MANI_03602]{DRAFT} **Signal-service-translation of E2E protected payload - error handling** ⌈If the E2E-Check returns any E2E error (not *E_OK*), then the service-based message shall reflect that E2E error.⌋*(RS_MANI_00063)*

Note: This is necessary to provide E2E information to the application software and support an End-to-End view on the data exchange from sender to receiver / provider to consumer.

13.9.1.2 Service-signal-translation

[TPS_MANI_03603]{DRAFT} **Service-signal-translation of E2E protected payload** ⌈Service-signal-translation shall check the end-to-end status of every received service-oriented payload. If the E2E-Check returns *E_OK* for the received payload then the data can be forwarded to the respective sending of the translation software.⌋*(RS_MANI_00063)*

Error handling:

[TPS_MANI_03604]{DRAFT} **Service-signal-translation of E2E protected payload - timeout handling** ⌈If no message is received within the specified message cycle time (timeout is detected), then no data shall be transmitted to the signal-based part.⌋ *(RS_MANI_00063)*

[TPS_MANI_03605]{DRAFT} **Service-signal-translation of E2E protected payload - error handling** ⌈If the service-oriented payload is handed over with any E2E error (not *E_OK*), then the newly created signal-based E2E protected message shall reflect that E2E error.⌋*(RS_MANI_00063)*

Note: This is necessary to provide E2E information to the receiving application and support an End-to-End view on the data exchange from sender to receiver/provider to consumer.

13.9.2 Security

In the context of Signal Service Translation the Secure Onboard Communication *SecOC* [36] is the major security technology. Further technologies (like *IPSec* or *TLS*) have not been included in the considerations for `signal/service translation`.

The configuration of *SecOC* on the signal-based communication is defined by having the `ISignalTriggering` used in one of the `signal/service translation` mappings refer to an `ISignal` and that `ISignal` is part of an `ISignalIPdu`. A `PduTriggering` of this `ISignalIPdu` is referenced by a `SecuredIPdu` in the role `payload`.

The `SecuredIPdu` defines all the details which are required to *generate* / *verify* the cryptographic information. The description of the configuration is provided in the System Template of Classic platform [18].

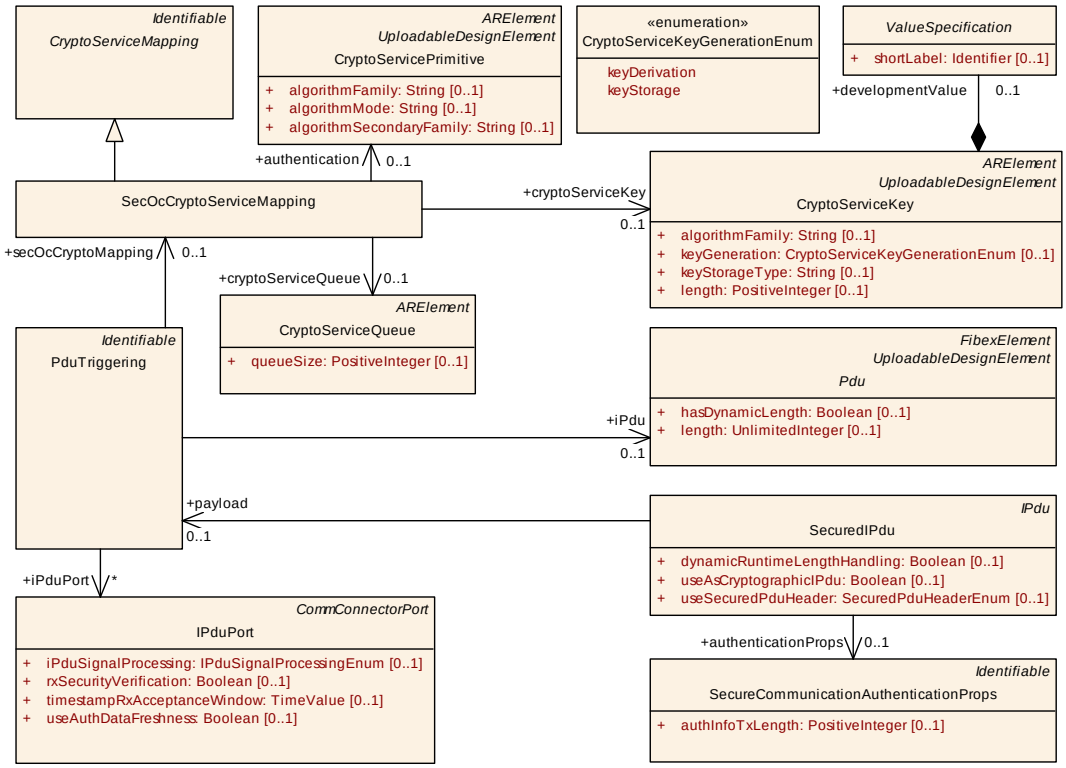


Figure 13.18: SecOC configuration

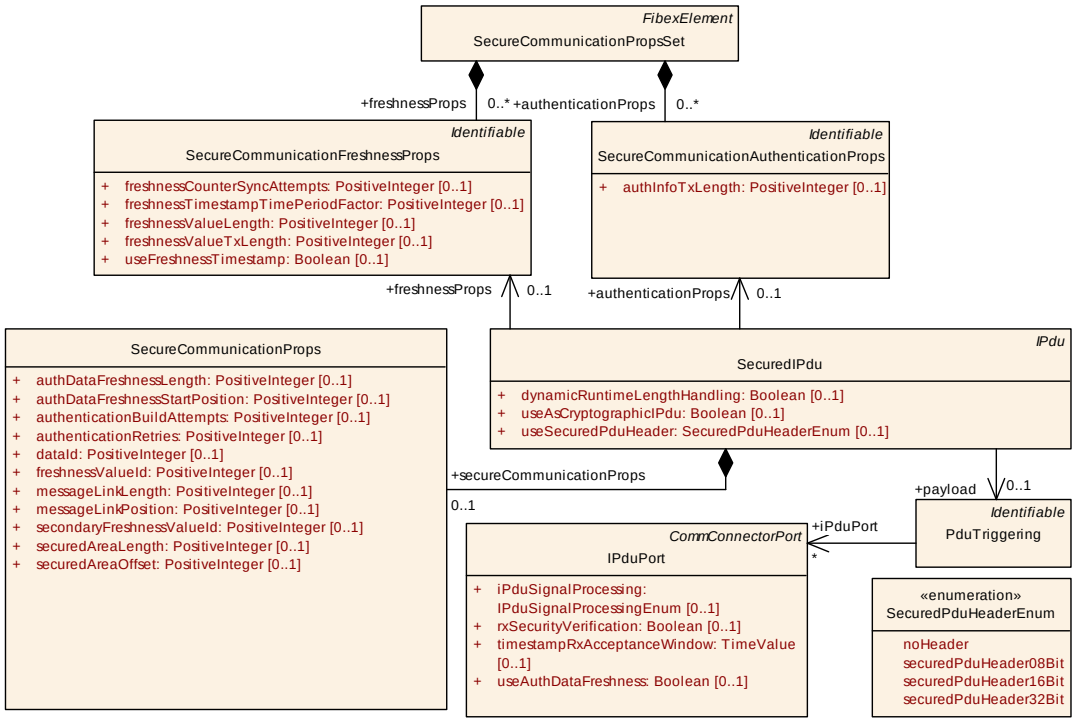


Figure 13.19: Details on `SecuredIPdu`

As the *SecOC* is highly embedded into the Classic platform architecture the `signal/-service translation` approach on security is to use the same architecture for its specification.

13.9.2.1 Restrictions on Classic Platform SecOC

[constr_3555]{DRAFT} **No support for `SecuredIPdu.useAsCryptographicIPdu` set to `true`** ⌈If a `PduTriggering` is referencing a `SecuredIPdu` where the attribute `useAsCryptographicIPdu` is set to the value `true`, then no subclass of `AbstractSignalBasedToISignalTriggeringMapping` shall refer to an `ISignalTriggering` that in turn refers to an `ISignal` that is referenced in the role `iSignal` by any `ISignalToIPduMapping` that is owned by the `ISignalIPdu` referenced by the `SecuredIPdu` in the role `payload` **`at the time when the creation of the manifest is finished`**.⌋*()*

The rationale for [constr_3555] is that the separate handling of two PDUs for the transport of one secured message is not supported by the SOME/IP protocol. In such cases the `signal/service translation` has to be performed on a Classic platform gateway ECU.

13.9.2.2 Link between Classic Platform SecOC and `CryptoKeySlot`

While most of the information how to configure the SecOC behavior are available in the Classic platform model, there is also the need to define which `CryptoKeySlot` shall be used for the processing of the SecOC message.

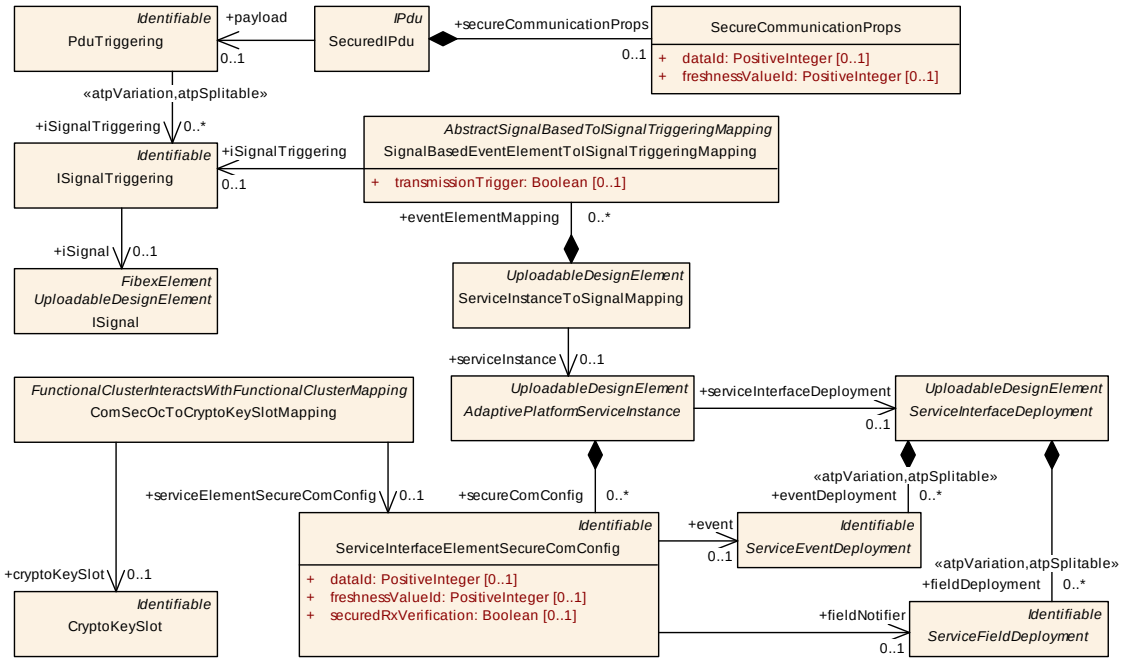


Figure 13.20: Classic SecOC and Adaptive `CryptoKeySlot`

[TPS_MANI_03636]{DRAFT} **Definition of `ServiceInterfaceElementSecureComConfig` in the context of `signal/service translation`** ⌈A `ServiceInterfaceElementSecureComConfig` at the `AdaptivePlatformServiceInstance` shall be created for each Event or FieldNotifier if the following conditions apply:

- the `AdaptivePlatformServiceInstance` is target of a `ServiceInstanceToSignalMapping` in the role `serviceInstance`
- the `ServiceInstanceToSignalMapping` has a `SignalBasedEventElementToISignalTriggeringMapping` (resp. `SignalBasedFieldToISignalTriggeringMapping`) that refers to an `ISignalTriggering` that are part of an `ISignalIPdu`
- that `ISignalIPdu` is referenced by a `PduTriggering` which in turn is referenced by a `SecuredIPdu` in the role `payload`
- the `SecuredIPdu` is referenced by a `PduTriggering` and that `PduTriggering` refers to an `IPduPort` that has the attribute `rxSecurityVerification` set to true.

⌋*(RS_MANI_00063)*

[TPS_MANI_03637]{DRAFT} **Ignored attributes of `ServiceInterfaceElementSecureComConfig` in the context of `signal/service translation`** ⌈If a `ServiceInterfaceElementSecureComConfig` qualifies according to [TPS_MANI_03636] then any attributes (e.g. `dataId` and `freshnessValueId`) - except the reference in the role `event` or `fieldNotifier` - shall be ignored. The information how to configure the SecOC shall be taken from the `SecuredIPdu`.⌋*(RS_MANI_00063)*

[TPS_MANI_03663]{DRAFT} **Ignored references of `ServiceInstanceToMachineMapping` in the context of signal/service translation** ⌈If an `AdaptivePlatformServiceInstance` qualifies according to [TPS_MANI_03636] and a `ServiceInstanceToMachineMapping` exists that refers to that `AdaptivePlatformServiceInstance` in the role `serviceInstance` and in the context of that `ServiceInstanceToMachineMapping` any of the following references exist:

- `ServiceInstanceToMachineMapping.secureComPropsForUdp`
- `ServiceInstanceToMachineMapping.secureComPropsForTcp`
- `ServiceInstanceToMachineMapping.secOcComPropsForMulticast`,

they shall be ignored.⌋*(RS_MANI_00063)*

The information of `SecOcSecureComProps` is available from the `SecuredIPdu` description in:

- `SecureCommunicationAuthenticationProps.authInfoTxLength`
- `SecureCommunicationFreshnessProps.freshnessValueLength`

- SecureCommunicationFreshnessProps.freshnessValueTxLength
- SecOcCryptoServiceMapping.authentication.

14 Cross-FunctionalCluster interaction

[TPS_MANI_03268]{DRAFT} **Semantics of `FunctionalClusterInteractsWithFunctionalClusterMapping`** ⌈Abstract meta-class `FunctionalClusterInteractsWithFunctionalClusterMapping` provides an anchor for the specification of interaction between two functional clusters.

The identification of the interaction use case towards the functional cluster implementation is done by using an `InstanceSpecifier` of the concrete subclass of `FunctionalClusterInteractsWithFunctionalClusterMapping` that is used to define the concrete interaction.⌋ *(RS_MANI_00023)*

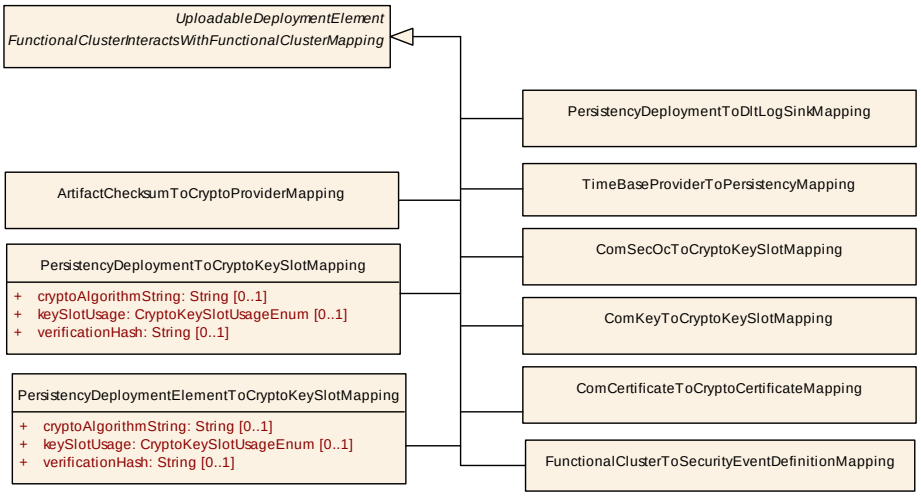


Figure 14.1: FunctionalClusterInteractsWithFunctionalClusterMapping Overview

Class	***FunctionalClusterInteractsWithFunctionalClusterMapping*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::FunctionalClusterInteractsWithFunctionalClusterMapping			
Note	This meta-class identifies a relation between functional clusters on the adaptive platform such one functional cluster can call APIs of the other functional cluster.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	ArtifactChecksumToCryptoProviderMapping, ComCertificateToCryptoCertificateMapping, ComKeyToCryptoKeySlotMapping, ComSecOcToCryptoKeySlotMapping, FunctionalClusterInteractsWithPersistencyDeploymentMapping, FunctionalClusterToSecurityEventDefinitionMapping, NmInteractsWithSmMapping, PersistencyDeploymentElementToCryptoKeySlotMapping, PersistencyDeploymentToCryptoKeySlotMapping, PersistencyDeploymentToDltLogSinkMapping, SmInteractsWithNmMapping, TimeBaseProviderToPersistencyMapping, UcmToTimeBaseResourceMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 14.1: FunctionalClusterInteractsWithFunctionalClusterMapping

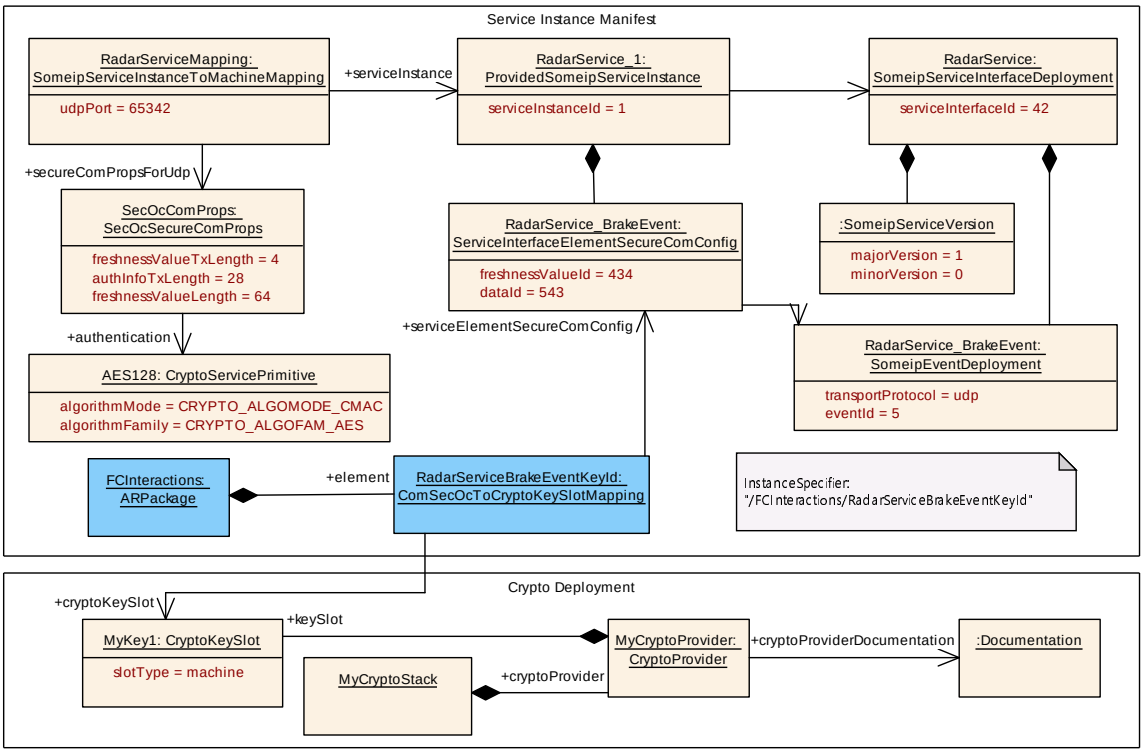


Figure 14.2: Example for the usage of `FunctionalClusterInteractsWithFunctionalClusterMapping`

In case of an application the model-path to an `PortPrototype` that is referencing a specific `PortInterface` defines the `InstanceSpecifier` that is used as identification towards the functional cluster implementation.

In case of a FunctionalCluster interaction the `PortPrototype` is not available. Therefore the path to a mapping element that is derived from `FunctionalClusterInteractsWithFunctionalClusterMapping` is used to define the `InstanceSpecifier` that is used in the API call.

The following figure shows the supported `FunctionalClusterInteractsWithFunctionalClusterMapping` subclasses that are available in the model.

The following Figure 14.2 shows an example for the usage of `FunctionalClusterInteractsWithFunctionalClusterMapping`. In this example, the Service Instance Manifest describes a configuration for a `ProvidedSomeipServiceInstance` that contains an `Event` that is protected by SecOC.

Inside of the Service Instance Manifest the “RadarServiceBrakeEventKeyId” represents the `FunctionalClusterInteractsWithFunctionalClusterMapping` that is located in the `ARPackage` “FCInteractions”.

These two elements in the manifest define the `InstanceSpecifier` “/FCInteractions/RadarServiceBrakeEventKeyId” that is used in the Crypto API call from the Communication Management. This `InstanceSpecifier` is resolved to a concrete `CryptoKeySlot` with the information that is available in the Crypto Deployment.

14.1 Com Certificate To Crypto Certificate Mapping

[TPS_MANI_03269]{DRAFT} **Semantics of ComCertificateToCryptoCertificateMapping** ⌈The meta-class ComCertificateToCryptoCertificateMapping provides an anchor for the specification of interaction between the COM FunctionalCluster and the Crypto FunctionalCluster and is used to map a CryptoServiceCertificate defined in COM to a CryptoCertificate defined in the Crypto Stack.⌋ *(RS_MANI_00023)*

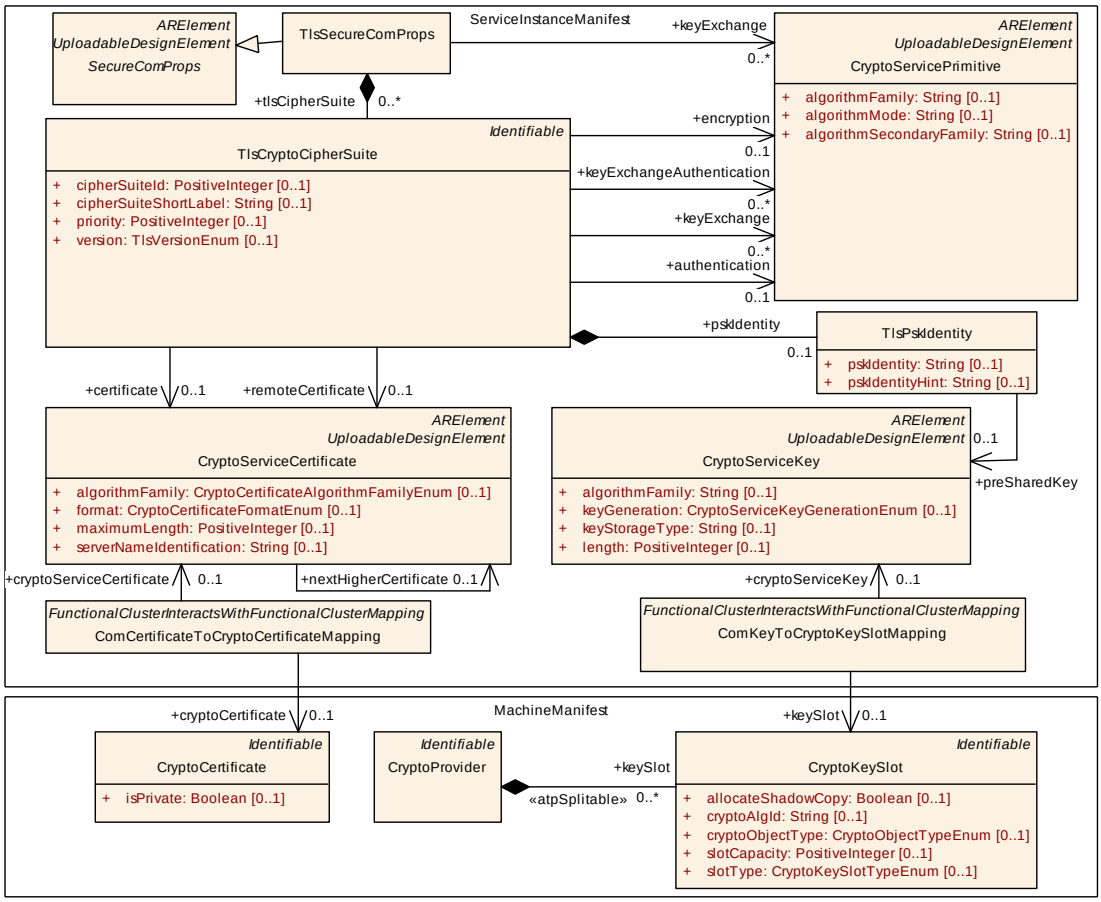


Figure 14.3: Mapping of TLS config elements to crypto objects in the crypto stack

Figure 14.3 shows the ComCertificateToCryptoCertificateMapping used to assign a CryptoServiceCertificate defined in the TLS configuration to a CryptoCertificate in the Crypto Stack.

Class	**ComCertificateToCryptoCertificateMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment
Note	This meta-class maps the CryptoServiceCertificate defined in the COM deployment to the Crypto Certificate defined in the Crypto Stack. **Tags:** atp.recommendedPackage=FCInteractions

▽

△

Class	ComCertificateToCryptoCertificateMapping			
Base	ARElement, ARObject, CollectableElement, FunctionalClusterInteractsWithFunctionalClusterMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadableDeployment Element, UploadablePackageElement			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
crypto Certificate	CryptoCertificate	0..1	ref	Reference to the CryptoCertifcate in the Crypto Stack.
cryptoService Certificate	CryptoService Certificate	0..1	ref	Reference to the cryptoServiceCertificate in the Com deployment

Table 14.2: ComCertificateToCryptoCertificateMapping

14.2 Com Key To Crypto Key Slot Mapping

[TPS_MANI_03270]{DRAFT} **Semantics of `ComKeyToCryptoKeySlotMapping`**
⌈The meta-class `ComKeyToCryptoKeySlotMapping` provides an anchor for the specification of interaction between the COM FunctionalCluster and the Crypto FunctionalCluster and is used to map a `CryptoServiceKey` defined in COM to a `CryptoKeySlot` defined in the Crypto Stack.⌋*(RS_MANI_00023)*

Class	ComKeyToCryptoKeySlotMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class maps the CryptoServiceKey defined in the COM deployment to the CryptoKeySlot defined in the Crypto Stack. **Tags:** atp.recommendedPackage=FCInteractions			
Base	ARElement, ARObject, CollectableElement, FunctionalClusterInteractsWithFunctionalClusterMapping, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, UploadableDeployment Element, UploadablePackageElement			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cryptoService Key	CryptoServiceKey	0..1	ref	Reference to the cryptoServiceKey in the Com deployment
keySlot	CryptoKeySlot	0..1	ref	Reference to the CryptoKeySlot in the Crypto Stack.

Table 14.3: ComKeyToCryptoKeySlotMapping

Figure 14.4 shows the `ComKeyToCryptoKeySlotMapping` used to assign a `CryptoServiceKey` defined in the IPsec configuration to `CryptoKeySlot` in the Crypto Stack.

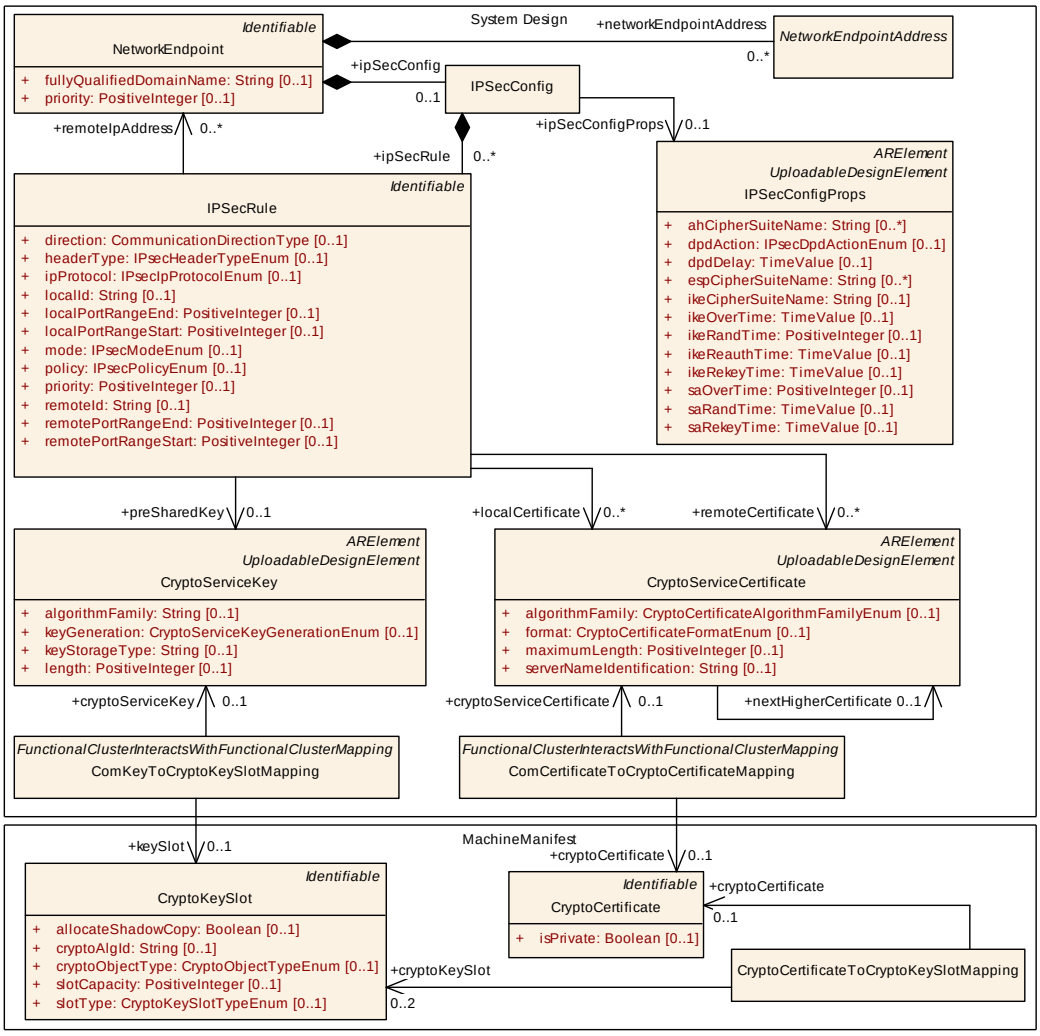


Figure 14.4: Mapping of IPsec config elements to crypto objects in the crypto stack

14.3 Com SecOc To Crypto Key Slot Mapping

[TPS_MANI_03271]{DRAFT} **Semantics of `ComSecOcToCryptoKeySlotMapping`**
⌈The meta-class `ComSecOcToCryptoKeySlotMapping` provides an anchor for the specification of interaction between the COM FunctionalCluster and the Crypto FunctionalCluster and is used to map a `ServiceInterfaceElementSecureComConfig` defined in COM to a `CryptoKeySlot` defined in the Crypto Stack.⌋*(RS_MANI_00023)*

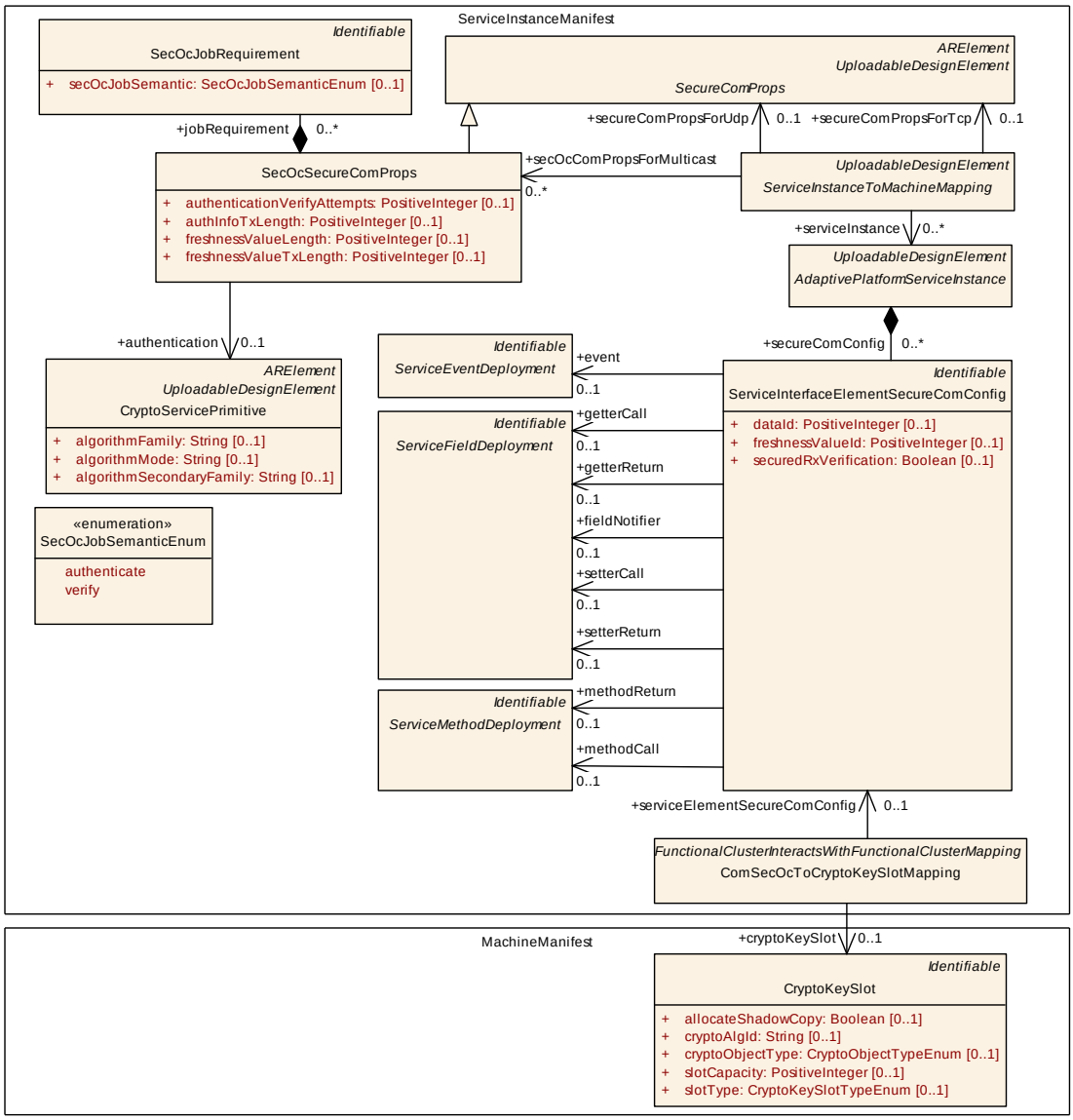


Figure 14.5: Mapping of SecOC config elements to crypto objects in the crypto stack

Class	**ComSecOcToCryptoKeySlotMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class maps the ServiceElementSecureComConfig defined in the COM deployment to the CryptoKeySlot defined in the Crypto Stack. **Tags:** atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cryptoKeySlot	CryptoKeySlot	0..1	ref	Reference to the CryptoKeySlot in the Crypto Stack.

△

Class	ComSecOcToCryptoKeySlotMapping			
serviceElement SecureCom Config	ServiceInterface ElementSecureCom Config	0..1	ref	Reference to the ServiceInterfaceElementSecureCom Config element in the COM config.

Table 14.4: ComSecOcToCryptoKeySlotMapping

Figure 14.5 shows the `ComSecOcToCryptoKeySlotMapping` used to assign a `ServiceInterfaceElementSecureComConfig` defined in the SecOC configuration to `CryptoKeySlot` in the Crypto Stack.

14.4 Persistency Deployment To Crypto Key Slot Mapping

[TPS_MANI_03272]{DRAFT} **Semantics of `PersistencyDeploymentToCryptoKeySlotMapping`** ⌈The meta-class `PersistencyDeploymentToCryptoKeySlotMapping` provides an anchor for the specification of interaction between the Persistency FunctionalCluster and the Crypto FunctionalCluster and is used to map a `PersistencyDeployment` defined in Persistency to a `CryptoKeySlot` defined in the Crypto Stack.⌋*(RS_MANI_00023)*

Class	PersistencyDeploymentToCryptoKeySlotMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to define a mapping between the PersistencyDeployment and a CryptoKeySlot. **Tags:** atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cryptoAlgorithm String	String	0..1	attr	This attribute defines the cryptographic algorithm used for hashing, encryption, decryption, signature/MAC verification, or MAC generation.
cryptoKeySlot	CryptoKeySlot	0..1	ref	This reference represents the mapped CryptoKeySlot.
keySlotUsage	CryptoKeySlotUsage Enum	0..1	attr	This attribute defines the role of the keySlot assignment.
persistency Deployment	PersistencyDeployment	0..1	ref	This reference represents the mapped Persistency Deployment.
verificationHash	String	0..1	attr	This attribute defines the hash of the storage used in case of verification.

Table 14.5: PersistencyDeploymentToCryptoKeySlotMapping

[constr_10232]{DRAFT} **Multiplicity of reference in the role `PersistencyDeploymentToCryptoKeySlotMapping.persistencyDeployment`** ⌈For each `PersistencyDeploymentToCryptoKeySlotMapping`, the reference in the role `persistencyDeployment` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

Enumeration	**CryptoKeySlotUsageEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment
Note	This enum defines the possible roles of the keySlotUsage.
Aggregated by	PersistencyDeploymentElementToCryptoKeySlotMapping.keySlotUsage, PersistencyDeploymentToCryptoKeySlotMapping.keySlotUsage
Literal	***Description***
encryption	Key slot usage for encryption **Tags:** atp.EnumerationLiteralIndex=1
verification	Key slot usage for verification **Tags:** atp.EnumerationLiteralIndex=0

Table 14.6: CryptoKeySlotUsageEnum

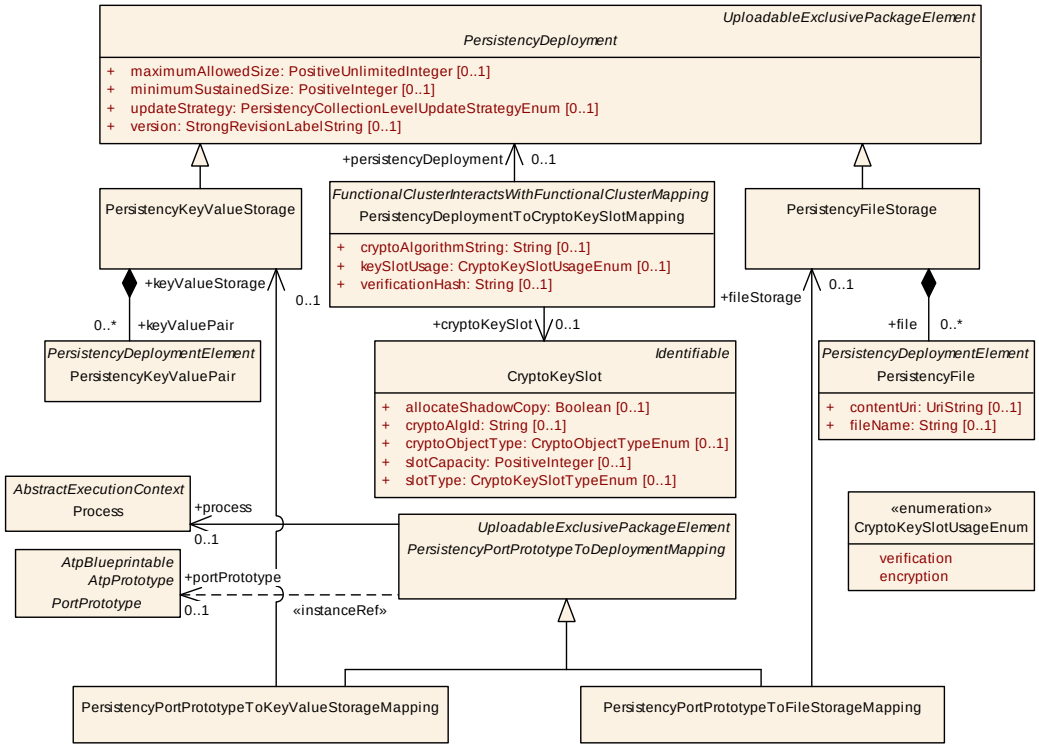


Figure 14.6: Mapping of PersistencyDeployment to crypto objects in the crypto stack

14.5 Persistency Deployment Element To Crypto Key Slot Mapping

[TPS_MANI_03273]{DRAFT} **Semantics of `PersistencyDeploymentElementToCryptoKeySlotMapping`** ⌈The meta-class `PersistencyDeploymentElementToCryptoKeySlotMapping` provides an anchor for the specification of interaction between the Persistency FunctionalCluster and the Crypto FunctionalCluster and is used to map a `PersistencyDeploymentElement` defined in Persistency to a `CryptoKeySlot` defined in the Crypto Stack.⌋*(RS_MANI_00023)*

Class	**PersistencyDeploymentElementToCryptoKeySlotMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to define a mapping between the PersistencyDeploymentElement and a CryptoKeySlot. **Tags:** atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cryptoAlgorithm String	String	0..1	attr	This attribute defines the cryptographic algorithm used for hashing, encryption, decryption, signature/MAC verification, or MAC generation.
cryptoKeySlot	CryptoKeySlot	0..1	ref	This reference represents the mapped CryptoKeySlot.
keySlotUsage	CryptoKeySlotUsage Enum	0..1	attr	This attribute defines the role of the keySlot assignment.
persistency Deployment Element	PersistencyDeployment Element	0..1	ref	This reference represents the mapped Persistency Deployment.
verificationHash	String	0..1	attr	This attribute defines the hash of the storage used in case of verification.

Table 14.7: PersistencyDeploymentElementToCryptoKeySlotMapping

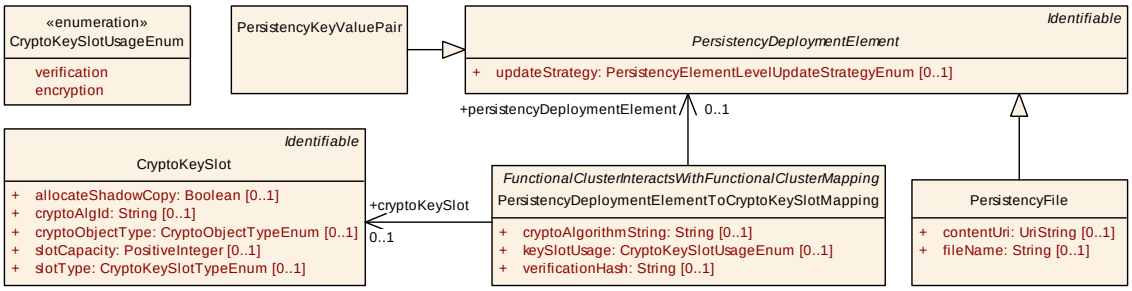


Figure 14.7: Mapping of PersistencyDeploymentElement to crypto objects in the crypto stack

14.6 Persistency Deployment To Dlt Log Sink Mapping

[TPS_MANI_03276]{DRAFT} **Semantics of meta-class `PersistencyDeploymentToDltLogSinkMapping`** ⌈The meta-class `PersistencyDeploymentToDltLogSinkMapping` provides an anchor for the specification of interaction between the Persistency FunctionalCluster and Log & Trace and is used to map a `PersistencyDeployment` defined in Persistency to a `DltLogSink` defined in the `LogAndTraceInstantiation`.⌋*(RS_MANI_00023)*

Class	**PersistencyDeploymentToDltLogSinkMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::LogAndTrace			
Note	This meta-class represents the ability to define a mapping between the PersistencyDeployment and a Dlt LogSink. **Tags:** atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dltContext	DltContext	0..1	ref	Reference to the DltContext to which this mapping belongs.
logSink	DltLogSink	0..1	ref	Reference to the DltLogSink to which the Persistency Deployment is mapped
persistency Deployment	PersistencyDeployment	0..1	ref	This reference represents the mapped Persistency Deployment.

Table 14.8: PersistencyDeploymentToDltLogSinkMapping

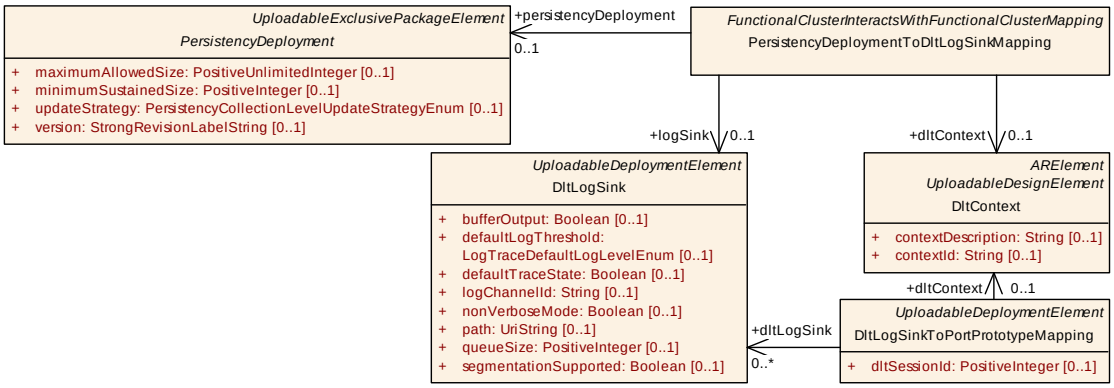


Figure 14.8: Mapping of PersistencyDeployment to DltLogSink

14.7 Ucm To Time Base Resource Mapping

[TPS_MANI_01374]{DRAFT} **Semantics of `UcmToTimeBaseResourceMapping`**
⌈The meta-class `UcmToTimeBaseResourceMapping` provides an anchor for the specification of interaction between the Ucm Functional Cluster and Time Sync Functional Cluster and is used to identify the source of timestamp-information in the UCM.⌋
(RS_MANI_00023)

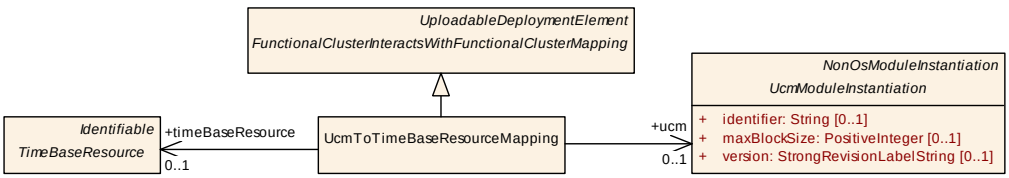


Figure 14.9: Mapping of the Ucm Functional Cluster to the Time Sync Functional Cluster

Class	**UcmToTimeBaseResourceMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Ucm			
Note	This meta-class maps the UCM Module Instantiation to the TimeSync Module Instantiation. **Tags:** atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
timeBase Resource	TimeBaseResource	0..1	ref	This reference identifies the relevant TimeBaseResource.
ucm	UcmModuleInstantiation	0..1	ref	This reference identifies the relevant UcmModule Instantiation.

Table 14.9: UcmToTimeBaseResourceMapping

14.8 Functional Cluster Interacts With Persistency Deployment Mapping

[TPS_MANI_01379]{DRAFT} **Semantics of meta-class `FunctionalClusterInteractsWithPersistencyDeploymentMapping`** ⌈Meta-class `FunctionalClusterInteractsWithPersistencyDeploymentMapping` provides the ability to specify that (and how) a functional cluster that is implemented as a **library component** wants to access persistently stored data (formalized by meta-class `PersistencyDeployment`) on the target platform.⌋*(RS_MANI_00023)*

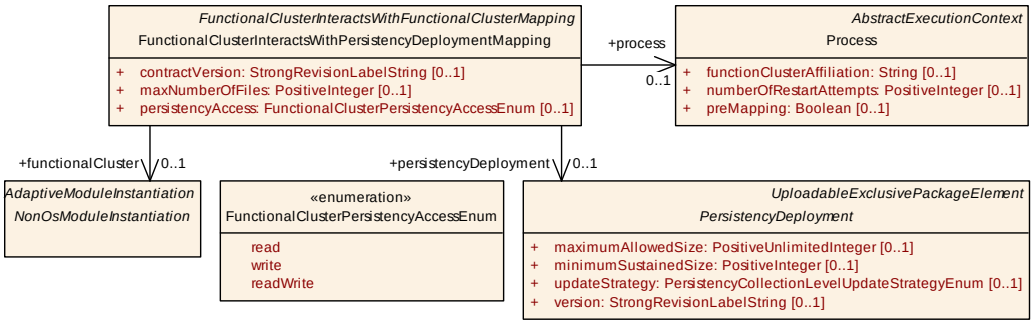


Figure 14.10: Mapping of any non-OS Functional Cluster to the Persistency Functional Cluster

Please note that the functional clusters are typically not dealing with modeled data types and therefore a formalized relation of `FunctionalClusterInteractsWithPersistencyDeploymentMapping` to a subclass of `CppImplementationDataType` is not foreseen.

Class	**FunctionalClusterInteractsWithPersistencyDeploymentMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to define a mapping between any functional cluster modeled as a subclass of NonOsModuleInstantiation and a PersistencyDeployment. **Tags:** atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contractVersion	StrongRevisionLabel String	0..1	attr	This attribute represents the contract version that is used to determine whether the Persistency configuration experienced structural changes and is also used for the check for data type compatibility.
functional Cluster	NonOsModule Instantiation	0..1	ref	This reference identifies the client functional cluster that wants to use persistency.
maxNumberOf Files	PositiveInteger	0..1	attr	This attribute represents the definition of an upper bound for the handling of files at run-time in the context of the enclosing FunctionalClusterInteractsWithPersistency DeploymentMapping.
persistency Access	FunctionalCluster PersistencyAccess Enum	0..1	attr	This attribute represents the definition of the persistency access of all kinds of persisted data at run-time in the context of the enclosing FunctionalClusterInteractsWith PersistencyDeploymentMapping.
persistency Deployment	PersistencyDeployment	0..1	ref	This reference identifies the applicable Persistency Deployment.
process	Process	0..1	ref	"This reference identifies the applicable process.

Table 14.10: FunctionalClusterInteractsWithPersistencyDeploymentMapping

[constr_10484]{DRAFT} **Existence of `FunctionalClusterInteractsWithPersistencyDeploymentMapping.contractVersion`** ⌈In the context of `FunctionalClusterInteractsWithPersistencyDeploymentMapping`, the attribute in the role `contractVersion` shall exist **at the time when the creation of the manifest is finished**.⌋ *()*

Enumeration	**FunctionalClusterPersistencyAccessEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency
Note	This meta-class provides possible values about how functional clusters may use persistency with respect to the direction of access.
Aggregated by	FunctionalClusterInteractsWithPersistencyDeploymentMapping.persistencyAccess
Literal	***Description***
read	Functional Cluster wants to access persistency with a read semantics. **Tags:** atp.EnumerationLiteralIndex=0
readWrite	Functional Cluster wants to access persistency with a read and write semantics. **Tags:** atp.EnumerationLiteralIndex=2
write	Functional Cluster wants to access persistency with a write semantics. **Tags:** atp.EnumerationLiteralIndex=1

Table 14.11: FunctionalClusterPersistencyAccessEnum

14.9 AdaptiveModuleInstantiation to SecurityEventDefinition Mapping

[TPS_MANI_03324]{DRAFT} **Semantics of `FunctionalClusterToSecurityEventDefinitionMapping`** ⌈The meta-class `FunctionalClusterToSecurityEventDefinitionMapping` provides an anchor for the specification of interaction between a Functional Cluster that is modeled as a `AdaptiveModuleInstantiation` and the IDSM (Intrusion Detection System Manager).

`FunctionalClusterToSecurityEventDefinitionMapping` is used to map a security event (formalized by meta-class `SecurityEventDefinition`) and the Functional Cluster that reports this security event.⌋ *(RS_MANI_00023)*

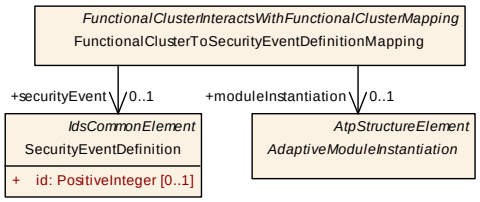


Figure 14.11: Mapping of the interaction between a Functional Cluster and the IdsM

Class	**FunctionalClusterToSecurityEventDefinitionMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::FunctionalClusterInteractsWithFunctionalClusterMapping			
Note	This meta-class represents the ability to define a mapping between the SecurityEventDefinition and the Module Instantiation that reports the security event. **Tags:** atp.recommendedPackage=FunctionalClusterToSecurityEventDefinitionMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
module Instantiation	AdaptiveModule Instantiation	0..1	ref	Reference to the AdaptiveModuleInstantiation that reports the SecurityEvent.
securityEvent	SecurityEventDefinition	0..1	ref	Reference to the SecurityEvent that is reported by the AdaptiveModuleInstantiation. **Tags:** atp.Status=candidate

Table 14.12: FunctionalClusterToSecurityEventDefinitionMapping

[constr_5381]{DRAFT} **Modeling of Security Event reports by FunctionalCluster shall not be done via ProcessToMachineMapping** ⌈If a `Process` is mapped by the `ProcessToMachineMapping` to a `Machine` and this `ProcessToMachineMapping` references a `NonOsModuleInstantiation` in the role `nonOsModuleInstantiation`, then this `Process` shall not reference a `SecurityEventDefinition` in the role `securityEvent` **at the time when the creation of the manifest is finished**.⌋ *()*

14.10 State Management interacts with Network Management and vice versa

The interaction between state management and network management is bi-directional, but the purpose of each direction is different enough that a dedicated modeling of the configuration of each direction makes sense.

14.10.1 State Management interacts with Network Management

State management interacts with network management to request the network management to switch to a specific communication state (see also section 10.14.5.4).

In this interaction, state management acts like a functional cluster rather than an application and therefore the modeling of a subclass of `FunctionalClusterInteractsWithFunctionalClusterMapping` needs to be created: `SmInteractsWithNmMapping`.

[TPS_MANI_01446]{DRAFT} **Semantics of meta-class `SmInteractsWithNmMapping`** ⌈Meta-class `SmInteractsWithNmMapping` supports the interaction between state management and network management by mapping a `StateManagementNmActionItem` (which obviously represents the state management) to an `NmNetworkHandle` (representing the network management).⌋*()*

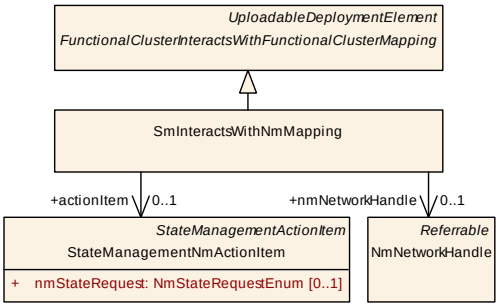


Figure 14.12: Mapping of the interaction between state management and network management

Class	**SmInteractsWithNmMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This mapping represents an interaction from state management to network management. **Tags:** atp.Status=draft atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	SmInteractsWithNmMapping			
actionItem	StateManagementNm ActionItem	0..1	ref	This reference identifies the action item with which the state management wants to interact with network management. **Tags:** atp.Status=draft
nmNetwork Handle	NmNetworkHandle	0..1	ref	This reference identifies the network management handle that is affected by the interaction with the state management. **Tags:** atp.Status=draft

Table 14.13: SmInteractsWithNmMapping

[constr_10513]{DRAFT} **Existence of the reference in the role SmInteractsWithNmMapping.actionItem** ⌈For each SmInteractsWithNmMapping, the reference in the role actionItem shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10514]{DRAFT} **Existence of the reference in the role SmInteractsWithNmMapping.nmNetworkHandle** ⌈For each SmInteractsWithNmMapping, the reference in the role nmNetworkHandle shall exist **at the time when the creation of the manifest is finished**.⌋*()*

14.10.2 Network Management interacts with State Management

Network management interacts with state management to notify the state management that a specific communication state (see also section 10.14.5.4) has been switched to.

This may happen, for example, if the communication bus goes down unexpectedly and state management needs to react to this situation.

In this interaction, state management acts like a functional cluster rather than an application and therefore the modeling of a subclass of FunctionalClusterInteractsWithFunctionalClusterMapping needs to be created: NmInteractsWithSmMapping.

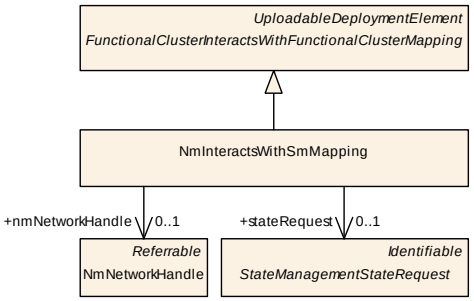


Figure 14.13: Mapping of the interaction between network management and state management

[TPS_MANI_01447]{DRAFT} **Semantics of meta-class NmInteractsWithSmMapping** ⌈Meta-class NmInteractsWithSmMapping supports the interaction between

state management and network management by mapping an NmNetworkHandle (which obviously represents the network management) to a StateManagementStateRequest (representing the state management).⌋()

Class	**NmInteractsWithSmMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::StateManagement			
Note	This mapping represents an interaction from network management to state management. **Tags:** atp.Status=draft atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
nmNetwork Handle	NmNetworkHandle	0..1	ref	This reference identifies the network management handle that wants to interact with state management. **Tags:** atp.Status=draft
stateRequest	StateManagementState Request	0..1	ref	This reference identifies the state management state request that is involved in the interaction with the network management. **Tags:** atp.Status=draft

Table 14.14: NmInteractsWithSmMapping

[constr_10515]{DRAFT} **Existence of the reference in the role NmInteractsWithSmMapping.stateRequest** ⌈For each NmInteractsWithSmMapping, the reference in the role stateRequest shall exist **at the time when the creation of the manifest is finished**.⌋()

[constr_10516]{DRAFT} **Existence of the reference in the role NmInteractsWithSmMapping.nmNetworkHandle** ⌈For each NmInteractsWithSmMapping, the reference in the role nmNetworkHandle shall exist **at the time when the creation of the manifest is finished**.⌋()

15 Software Distribution

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 re-flash the entire ECU. Instead, software packages are uploaded to the ECU where the content is taken care of by responsible platform modules.

The reason why this topic is relevant for the modeling is the fact that an uploadable software package consists not only of software itself but also of manifest content required to support the integration of the uploaded software with the existing platform instance.

As far as the meta-model is concerned, the discussion about manifests and which manifest content needs to go with which other model elements doesn't care about the file granularity. In other words, it would not make sense to formalize the uploadable software package on the basis of references to files that carry model elements.

Instead, the view on the manifest topic from the modeling point of view focuses on model elements that make up manifest content.

Therefore, the modeling of an uploadable software package allows for putting references to all the required model elements that, in their entirety, make up the manifest of the corresponding application software that is also going to end up in the uploadable software package.

From the formal point of view, such an uploadable software package is modeled as a so-called `SoftwareCluster`. This meta-class is the root element that in turn describes all the necessary content of an uploadable software package.

However, the software package obviously isn't created out of thin air. It is the result of a workflow that starts from the formulation of requirements on the content of a `SoftwareCluster`.

These requirements are formalized by means of meta-class `SoftwareClusterDesigns`.

The relation between `SoftwareClusterDesign` and `SoftwareCluster` is depicted in Figure 15.1.

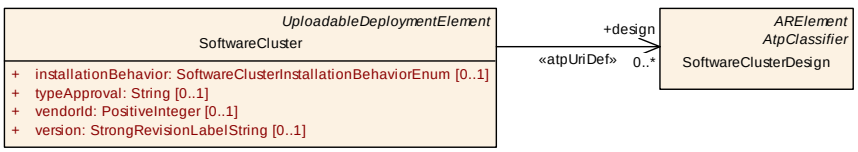


Figure 15.1: Relation of `SoftwareClusterDesign` to `SoftwareCluster`

15.2 Uploadable Package Element

[TPS_MANI_01109]{DRAFT} **Semantics of `UploadablePackageElement`** ⌈In order to keep the complexity of the modeling of `SoftwareCluster` as low as possible abstract meta-class `UploadablePackageElement` has been created.

This allows for the referencing of model elements derived from `UploadablePackageElement` that need to be considered in an uploadable software package from within a `SoftwareCluster` with just the reference `containedPackageElement`.⌋ *(RS_MANI_00035)*

Meta-class `UploadablePackageElement` that has two abstract sub-classes for model elements that are deployment-related (represented by meta-class `UploadableDeploymentElement`) and model elements that are design-related (represented by meta-class `UploadableDesignElement`), but still needed at configuration time on the target system as a dependency of the `UploadableDeploymentElement`s.

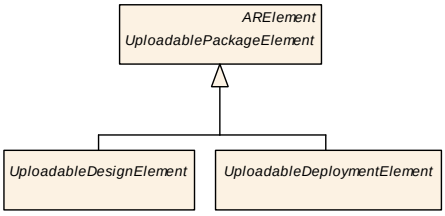


Figure 15.2: Overview of `UploadablePackageElement` and its direct sub-classes

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. This applies for both design-level and deployment-level meta-classes.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*UploadableDeploymentElement*, *UploadableDesignElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 15.1: UploadablePackageElement

[TPS_MANI_01413]{DRAFT} **Semantics of `UploadableDeploymentElement`** ⌈As a refinement of the abstract base class `UploadablePackageElement`, meta-class `UploadableDeploymentElement` represents configuration information that is **related to the deployment phase** and which is expected to be **uploaded to the target** system and **needed for configuration** on the *AUTOSAR adaptive platform* .⌋ *(RS_MANI_00035)*

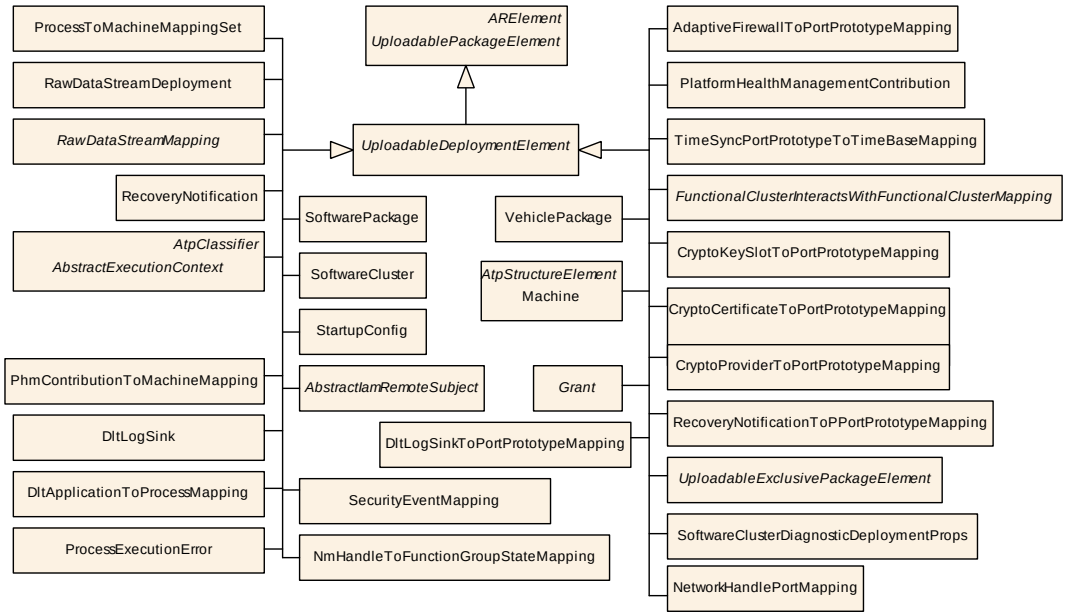


Figure 15.3: Overview of `UploadableDeploymentElement`

Class	***UploadableDeploymentElement*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class acts as an abstract base class for all deployment-level meta-classes that need to be added to an uploadable software package in order to complete the manifest content.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Subclasses	*AbstractExecutionContext*, *AbstractIamRemoteSubject*, AdaptiveFirewallToPortPrototypeMapping, CryptoCertificateToPortPrototypeMapping, CryptoKeySlotToPortPrototypeMapping, CryptoProviderToPortPrototypeMapping, DltApplicationToProcessMapping, DltLogSink, DltLogSinkToPortPrototypeMapping, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Grant*, IdsmContextProviderMapping, IdsmTimestampProviderMapping, Machine, NetworkHandlePortMapping, NmHandleToFunctionGroupStateMapping, PhmContributionToMachineMapping, PlatformHealthManagementContribution, ProcessExecutionError, ProcessToMachineMappingSet, RawDataStreamDeployment, *RawDataStreamMapping*, RecoveryNotification, RecoveryNotificationToPPortPrototypeMapping, SecurityEventMapping, SoftwareCluster, SoftwareClusterDiagnosticDeploymentProps, SoftwarePackage, StartupConfig, TimeSyncPortPrototypeToTimeBaseMapping, *UploadableExclusivePackageElement*, VehiclePackage			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 15.2: UploadableDeploymentElement

[TPS_MANI_01414]{DRAFT} **Semantics of `UploadableDesignElement`** ⌈As a refinement of the abstract base class `UploadablePackageElement`, meta-class `UploadableDesignElement` represents configuration information that is **related to the design phase** and which is nevertheless expected to be **uploaded to the target** system and **needed for configuration** (as a dependency) on the *AUTOSAR adaptive platform.*⌋ *(RS_MANI_00035)*

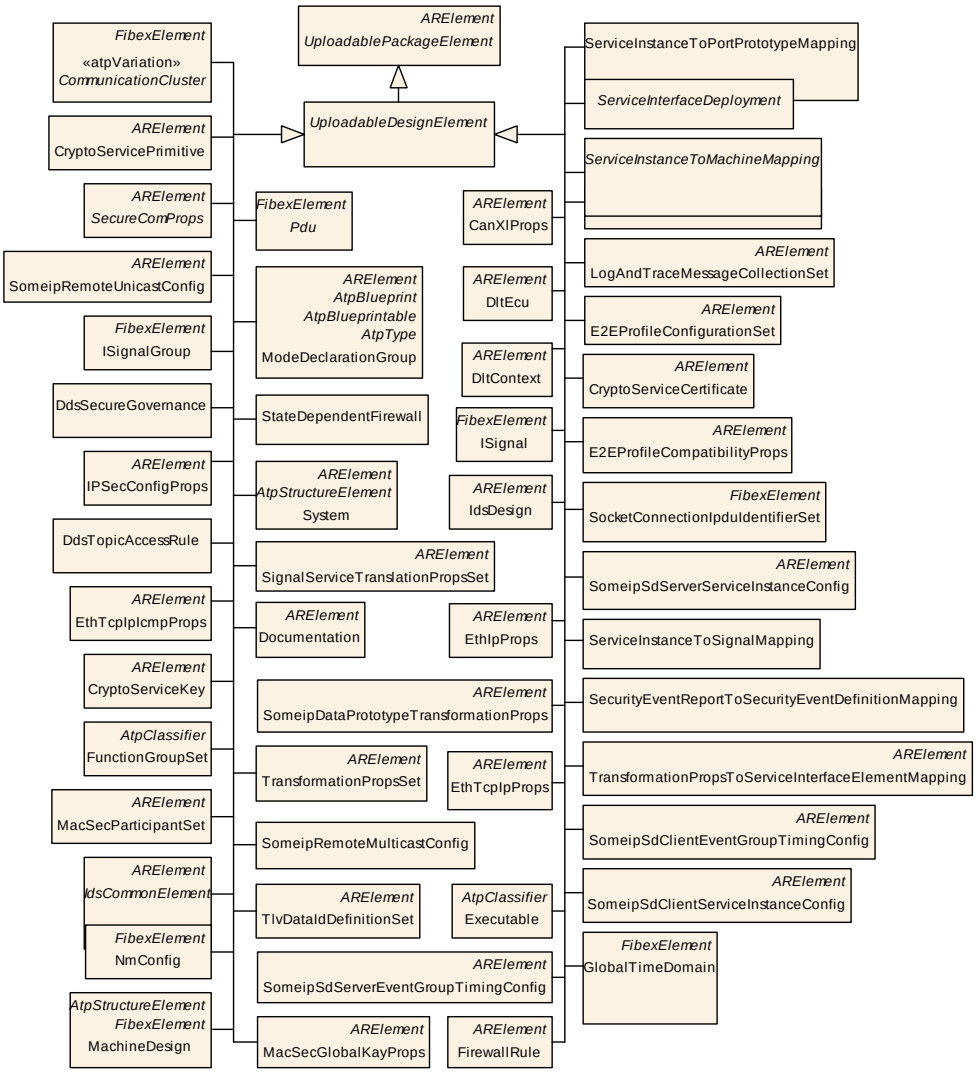


Figure 15.4: Overview of `UploadableDesignElement`

Class	***UploadableDesignElement*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::General
Note	This meta-class acts as an abstract base class for all design-level meta-classes that need to be added to an uploadable software package in order to complete the manifest content.
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*

△

Class	***UploadableDesignElement*** (abstract)			
Subclasses	*AdaptivePlatformServiceInstance*, CanXlProps, *CommunicationCluster*, CryptoServiceCertificate, CryptoServiceKey, CryptoServicePrimitive, DdsSecureGovernance, DdsTopicAccessRule, DltContext, DltEcu, Documentation, E2EProfileCompatibilityProps, E2EProfileConfigurationSet, EthIpProps, EthTcpIpIcmpProps, EthTcpIpProps, Executable, FirewallRule, FunctionGroupSet, GlobalTimeDomain, IPSecConfigProps, ISignal, ISignalGroup, *IdsCommonElement*, IdsDesign, LogAndTraceMessageCollectionSet, MacSecGlobalKayProps, MacSecParticipantSet, MachineDesign, ModeDeclarationGroup, NmConfig, *Pdu*, *SecureComProps*, SecurityEventReportToSecurityEventDefinitionMapping, *ServiceInstanceToMachineMapping*, ServiceInstanceToPortPrototypeMapping, ServiceInstanceToSignalMapping, *ServiceInterfaceDeployment*, SignalServiceTranslationPropsSet, SocketConnectionIpduIdentifierSet, SomeipDataPrototypeTransformationProps, SomeipRemoteMulticastConfig, SomeipRemoteUnicastConfig, SomeipSdClientEventGroupTimingConfig, SomeipSdClientServiceInstanceConfig, SomeipSdServerEventGroupTimingConfig, SomeipSdServerServiceInstanceConfig, StateDependentFirewall, System, TlvDataIdDefinitionSet, TransformationPropsSet, TransformationPropsToServiceInterfaceElementMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 15.3: UploadableDesignElement

Please note that this approach to collecting elements is very similar in semantics to `System.fibexElement` or `DiagnosticContributionSet.element`.

15.3 Software Cluster

15.3.1 Software Cluster General Modeling

[TPS_MANI_01110]{DRAFT} **Semantics of `SoftwareCluster`** ⌈The existence of a `SoftwareCluster` represents an uploadable software package.⌋ *(RS_MANI_00035)*

[TPS_MANI_01213]{DRAFT} **Semantics of meta-class `StrongRevisionLabelString`** ⌈Meta-class `StrongRevisionLabelString` supports the specification of a version number for a `SoftwareCluster` that consists of four components ([constr_1747] applies):

- Major version
- Minor version
- Patch version
- Additional labels for pre-release version and build metadata

⌋ *(RS_MANI_00035)*

[TPS_MANI_01410]{DRAFT} **Semantics of attribute `SoftwareCluster.installationBehavior`** ⌈Attribute `SoftwareCluster.installationBehavior` determines the behavior of a `SoftwareCluster` in terms of whether it will be allowed to remove the `SoftwareCluster` from the target machine.

The decision to declare a SoftwareCluster non-removable is most likely limited to SoftwareClusters with a significant share of platform-level software.⌋(*RS_MANI_-00035*)

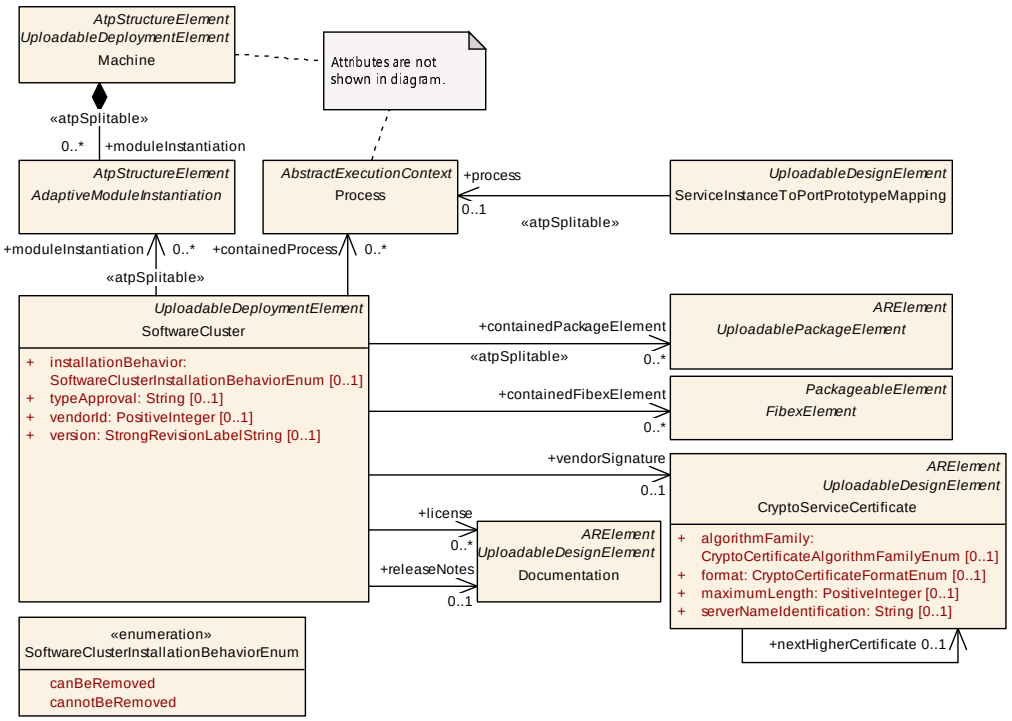


Figure 15.5: Modeling of SoftwareCluster

Class	**SoftwareCluster**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
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.recommendedPackage=SoftwareClusters			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
artifact Checksum	ArtifactChecksum	*	aggr	This aggregation carries the checksums for artifacts contained in the enclosing SoftwareCluster. Please note that the value of these checksums is only applicable at the time of configuration. **Stereotypes:** atpSplittable **Tags:** atp.Splitkey=artifactChecksum.shortName, artifact Checksum.uri
artifactLocator	ArtifactLocator	*	aggr	This aggregation represents the artifact locations that are relevant in the context of the enclosing SoftwareCluster
claimed FunctionGroup	ModeDeclarationGroup Prototype	*	ref	Each SoftwareCluster can reserve the usage of a given functionGroup such that no other SoftwareCluster is allowed to use it

▽

△

Class	**SoftwareCluster**			
conflictsTo	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation handles conflicts. If it yields true then the SoftwareCluster shall not be installed. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=conflictsTo
contained ARElement	ARElement	*	ref	This reference represents the collection of model elements that cannot derive from UploadablePackage Element and that contribute to the completeness of the definition of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=containedARElement
containedFibex Element	FibexElement	*	ref	This allows for referencing FibexElements that need to be considered in the context of a SoftwareCluster.
contained Package Element	UploadablePackage Element	*	ref	This reference identifies model elements that are required to complete the manifest content. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=containedPackageElement
contained Process	Process	*	ref	This reference represent the processes contained in the enclosing SoftwareCluster.
dependsOn	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation can be taken to identify a dependency for the enclosing SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=dependsOn
design	SoftwareClusterDesign	*	ref	This reference represents the identification of all Software ClusterDesigns applicable for the enclosing Software Cluster. **Stereotypes:** atpUriDef
diagnostic Deployment Props	SoftwareCluster DiagnosticDeployment Props	0..1	ref	This reference identifies the applicable SoftwareCluster DiagnosticDeploymentProps that are applicable for the referencing SoftwareCluster.
installation Behavior	SoftwareCluster InstallationBehavior Enum	0..1	attr	This attribute controls the behavior of the SoftwareCluster in terms of installation.
license	Documentation	*	ref	This attribute allows for the inclusion of the full text of a license of the enclosing SoftwareCluster. In many cases open source licenses require the inclusion of the full license text to any software that is released under the respective license.
module Instantiation	AdaptiveModule Instantiation	*	ref	This reference identifies AdaptiveModuleInstantiations that need to be included with the SoftwareCluster in order to establish infrastructure required for the installation of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=moduleInstantiation
releaseNotes	Documentation	0..1	ref	This attribute allows for the explanations of changes since the previous version. The list of changes might require the creation of multiple paragraphs of test.
typeApproval	String	0..1	attr	This attribute carries the homologation information that may be specific for a given country.
vendorId	PositiveInteger	0..1	attr	Vendor ID of this Implementation according to the AUTOSAR vendor list.
vendor Signature	CryptoService Certificate	0..1	ref	This reference identifies the certificate that represents the vendor's signature.
version	StrongRevisionLabel String	0..1	attr	This attribute can be used to describe a version information for the enclosing SoftwareCluster.

Table 15.4: SoftwareCluster

Enumeration	**SoftwareClusterInstallationBehaviorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This enumeration defines possible approaches for the installation behavior of a SoftwareCluster.
Aggregated by	SoftwareCluster.installationBehavior
Literal	***Description***
canBeRemoved	The enclosing SoftwareCluster can be removed from the target Machine or updated with a newer version. **Tags:** atp.EnumerationLiteralIndex=0
cannotBeRemoved	The enclosing SoftwareCluster cannot be removed from the target Machine. It can only be updated with a newer version. **Tags:** atp.EnumerationLiteralIndex=1

Table 15.5: SoftwareClusterInstallationBehaviorEnum

[constr_10410] Value of `SoftwareCluster.installationBehavior` for a `SoftwareCluster` of `category PLATFORM_CORE` ⌈In a `SoftwareCluster` of `category PLATFORM_CORE`, the attribute `installationBehavior` shall exist and its value shall be set to `cannotBeRemoved` **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10233]{DRAFT} **Multiplicity of the reference in the role `SoftwareCluster.vendorSignature`** ⌈For each `SoftwareCluster`, the reference in the role `vendorSignature` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10234]{DRAFT} **Multiplicity of attribute `SoftwareCluster.version`** ⌈For each `SoftwareCluster`, the attribute `version` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

[constr_10235]{DRAFT} **Multiplicity of attribute `SoftwareCluster.vendorId`** ⌈For each `SoftwareCluster`, the attribute `vendorId` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

The modeling of the relation `SoftwareCluster.claimedFunctionGroup` is modeled in Figure 15.6.

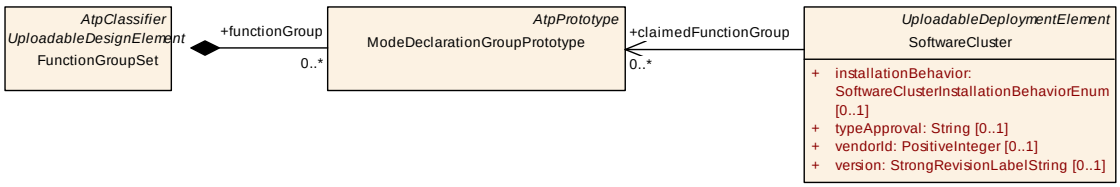


Figure 15.6: Modeling of the relation between `SoftwareCluster` and `Function Group`s

[constr_1747]{DRAFT} **Completeness of the `SoftwareCluster.version`** ⌈The `SoftwareCluster.version` shall contain all the following parts:

- Major version
- Minor version
- Patch version

- Additional labels for pre-release version and build metadata

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

Primitive	**StrongRevisionLabelString**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This primitive represents a revision label which identifies an object under version control. It represents a pattern which requires three integer numbers separated by a dot, representing from left to right Major Version, MinorVersion, PatchVersion and additional labels for pre-release version and build metadata. Legal patterns are for example: 1.0.0-alpha+001 1.0.0+20130313144700 1.0.0-beta+exp.sha.5114f85 **Tags:** xml.xsd.customType=STRONG-REVISION-LABEL-STRING xml.xsd.pattern=(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(-((0|[1-9]\d*|\d*[a-zA-Z-][0-9a-z A-Z-]*)(\.(0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(\+([0-9a-zA-Z-]+(\.[0-9a-zA-Z-]+)*))? xml.xsd.type=string

Table 15.6: StrongRevisionLabelString

Please note that the build number does not necessarily have to be consecutively incremented between two builds. In some cases the build number is created by creating a hash over the build.

In such a case it would not make sense to include the build number in a greater/less comparison while a comparison for equality/inequality may positively make sense. This aspect shall be taken into account when processing the value of an attribute types by a `StrongRevisionLabelString`.

[TPS_MANI_01331]{DRAFT} **Standardized values of attribute `SoftwareCluster.category`** ⌈AUTOSAR standardizes the following values of attribute `SoftwareCluster.category`

- **`PLATFORM_CORE`**: a `SoftwareCluster` of this `category` typically represents any kind of platform software, e.g. bootloader, hypervisor, OS, adaptive platform module. Such a `SoftwareCluster` cannot be removed by a UCM, but updates are possible. Please note that it is not forbidden that such a Software Cluster contains application level software.
- **`PLATFORM`**: a `SoftwareCluster` of this `category` represents the parts of the platform software (e.g. configuration of functional clusters) that could be installed, removed, and updated.
- **`APPLICATION_LAYER`**: a `SoftwareCluster` of this `category` represents a driving-relevant function on application level, e.g. a lane keeping assistant, window lift controller, seat positioning. Such a `SoftwareCluster` can be installed, removed, and updated.

⌋(*RS_MANI_00035*)

Even if the AUTOSAR UCM is not used as the Update Management System on the Machine the `PLATFORM_CORE` `SoftwareCluster` shall be defined for Methodology

reasons in the Manifest file. But in such a case it is allowed to define a single PLATFORM_CORE SoftwareCluster on the Machine that would contain all software that is deployed on this machine (platform software and application software).

Please also note that it is possible to (in addition to the standardized values) define custom values for attribute SoftwareCluster.category.

In this case, however, it is important to use custom values that don't clash with future extensions of the standardized values. A good way to avoid a clash is, for example, to use specific pre- or postfixes that identify a company or project name.

[constr_1788]{DRAFT} **Restriction to SoftwareCluster of category PLATFORM_CORE** ⌈On each Machine, exactly one SoftwareCluster of category PLATFORM_CORE shall be deployed **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_01115]{DRAFT} **Specification of executable software within SoftwareCluster** ⌈One of the most prominent contents of an uploadable software package is the reference to the executable software.

Within the definition of a SoftwareCluster, this reference is implicitly given by means of the reference SoftwareCluster.containedProcess.

The target of SoftwareCluster.containedProcess is a Process that represents an instance of the corresponding executable program (the software image), formalized as Executable⌋*(RS_MANI_00035)*

The prominence of the dedicated reference to Process is amplified by the fact that it would have been technically possible to let Process inherit from UploadablePackageElement and thus include the referenced Process(es) in the bulk of references to other required model elements.

These references are formalized in two different forms. For technical reasons it is not possible to let all model elements that need to be immediately referenced by a SoftwareCluster inherit from UploadablePackageElement.

The main reason is that further model elements need to be referenced by a SoftwareCluster that are also used on the *AUTOSAR classic platform*.

In other words, it would be very questionable to introduce the "useless" concept of an UploadablePackageElement into the scope of the *AUTOSAR classic platform* as a mere (and unwanted) side effect of providing a definition of the SoftwareCluster on the *AUTOSAR classic platform*.

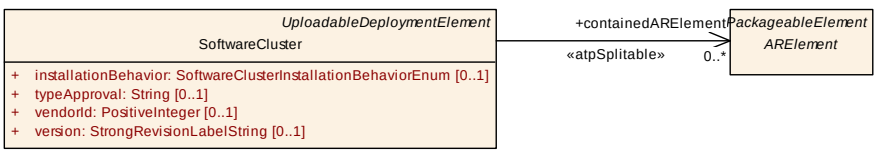


Figure 15.7: SoftwareCluster can reference ARElement

The scope of a single SoftwareCluster in terms of a relation to a Machine is that all software contained in one SoftwareCluster is supposed to be uploaded to one and only one Machine.

The definition of SoftwareCluster shall never include multiple Machines. This is expressed in [constr_1536].

[constr_1536]{DRAFT} **Definition of SoftwareCluster applies for a single Machine** ⌈Within the scope of a SoftwareCluster, each Process referenced in the role containedProcess shall be mapped (e.g. by means of the existence of a ProcessToMachineMapping) to the same Machine **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_01116]{DRAFT} **Reference to model elements included in an uploadable software package** ⌈Beside the ability to explicitly reference a Process in the role containedProcess it is possible to define the following references to required model elements:

- references to meta-classes derived from UploadablePackageElement are formalized by way of SoftwareCluster.containedPackageElement.
- references to meta-classes derived from ARElement are formalized by way of SoftwareCluster.containedARElement.
- references to meta-classes derived from FibexElement are formalized by way of SoftwareCluster.containedFibexElement.

Technically, an UploadablePackageElement is also an ARElement, but it is still mandated to use the dedicated reference specifically for UploadablePackageElement.⌋*(RS_MANI_00035)*

To exemplify the reference to UploadablePackageElement, Figure 15.5 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.

[TPS_MANI_01202]{DRAFT} **Semantics of reference SoftwareCluster.moduleInstantiation** ⌈By means of the reference SoftwareCluster.moduleInstantiation it is possible to express the need for updates of the platform infrastructure along with other resources referenced by the enclosing SoftwareCluster.⌋*(RS_MANI_00035)*

[TPS_MANI_01218]{DRAFT} **Cryptographic signature of SoftwareCluster** ⌈A SoftwareCluster also needs to be signed cryptographically. For this purpose, metaclass CryptoServiceCertificate is referenced in the role vendorSignature.⌋ *(RS_MANI_00035)*

[TPS_MANI_01219]{DRAFT} **License of software in included SoftwareCluster** ⌈It is possible to refer to licenses for software included in a SoftwareCluster by

means of a reference to meta-class Documentation in the role license.⌋(*RS_MANI_00035*)

Class	**Documentation**			
Package	M2::AUTOSARTemplates::GenericStructure::DocumentationOnM1			
Note	This meta-class represents the ability to handle a so called standalone documentation. Standalone means, that such a documentation is not embedded in another ARElement or identifiable object. The standalone documentation is an entity of its own which denotes its context by reference to other objects and instances. **Tags:** atp.recommendedPackage=Documentations			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
context	DocumentationContext	*	aggr	This is the context of the particular documentation.
documentation Content	PredefinedChapter	0..1	aggr	This is the content of the documentation related to the specified contexts. **Tags:** xml.sequenceOffset=200

Table 15.7: Documentation

Please note that Documentation is an ARElement that cannot be owned by a SoftwareCluster. The latter can only refer to it.

This aspect also means that once a given license is formalized by means of a Documentation it is in general possible to refer to this formalization from within different SoftwareClusters.

[TPS_MANI_01220]{DRAFT} **Release notes of software in included SoftwareCluster** ⌈It is possible to refer to release notes for software included in a SoftwareCluster by means of a reference to meta-class Documentation in the role releaseNotes.⌋(*RS_MANI_00035*)

[constr_1566]{DRAFT} **Usage of SoftwareCluster.containedARElement** ⌈The reference SoftwareCluster.containedARElement shall not be used to refer to a SoftwareCluster or a SoftwareClusterDesign **at the time when the creation of the manifest is finished**.⌋()

15.3.2 Relevance of Software Cluster for Diagnostics

15.3.2.1 Diagnostic Props

[TPS_MANI_01349]{DRAFT} **Configuration of diagnostic-related properties of a SoftwareCluster** ⌈The diagnostics-related properties of a SoftwareCluster are configured in the context of meta-class SoftwareClusterDiagnosticDeploymentProps, referenced in the role SoftwareCluster.diagnosticDeploymentProps.⌋(*RS_MANI_00035*)

[constr_10069]{DRAFT} **Existence of SoftwareClusterDiagnosticDeploymentProps.powerDownTime** ⌈The attribute SoftwareClusterDiagnosticDeploymentProps.powerDownTime shall exist **at the time when the creation of the manifest is finished** and have a value between 0 and 254 if the referenced diagnosticExtract that in turn references in the role element a DiagnosticEcuReset where attribute category is set to the value ENABLE_RAPID_POWER_SHUT_DOWN.⌋*()*

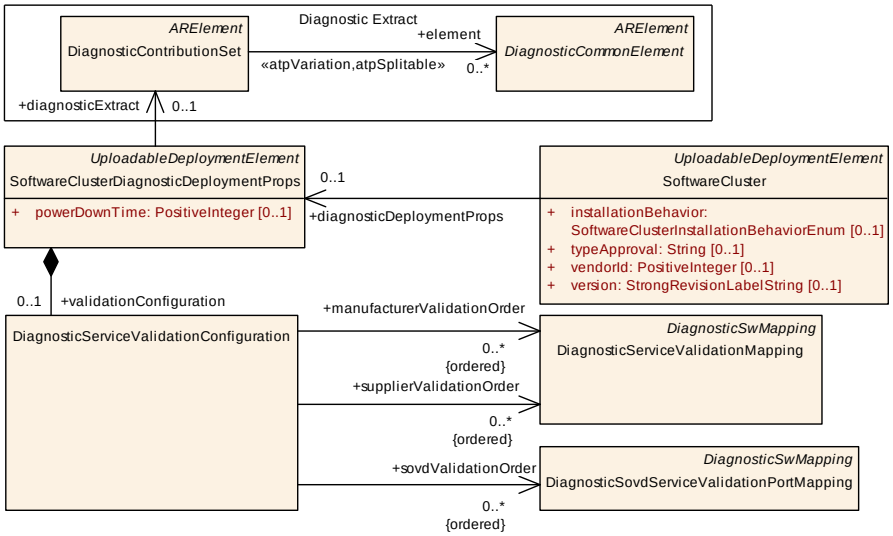


Figure 15.8: Modeling of diagnostic properties of a SoftwareCluster

15.3.2.2 Relation to the Diagnostic Contribution Set

[TPS_MANI_01114]{DRAFT} **Relation of DiagnosticContributionSet to SoftwareCluster** ⌈In AUTOSAR, the formalization of the external behavior of the diagnostic stack is rooted in meta-class DiagnosticContributionSet.

On the *AUTOSAR classic platform* the scope of the “external behavior of the diagnostic stack” is represented by an entire ECU.

This relation changes on the *AUTOSAR adaptive platform* where each uploadable software package is shipped with the definition of the “external behavior of the diagnostic stack” **as far as the software in the scope of respective uploadable software package is concerned**.

To fully support the different approaches of *AUTOSAR classic platform* and *AUTOSAR adaptive platform* it is necessary to provide means for specifying a DiagnosticContributionSet for a given SoftwareCluster.

In particular, this relation is created by means of the reference SoftwareCluster.diagnosticDeploymentProps.diagnosticExtract.⌋*(RS_MANI_00035)*

Please note that the placement of the reference to the DiagnosticContributionSet at the SoftwareClusterDiagnosticDeploymentProps follows the pattern

on the *AUTOSAR adaptive platform* that deployment-related elements shall reference their corresponding design-level elements.

In addition, it is much easier to keep the definition of diagnostic deployment and diagnostic design in sync if the former has a direct reference to the latter.

Please note further that this approach works for all levels of granularity in the definition of diagnostics on the *AUTOSAR adaptive Platform*:

- One `SoftwareCluster` represents a separately diagnosable entity with its own diagnostic address.

  In this case, the `DiagnosticContributionSet` referenced in the role `SoftwareCluster.diagnosticDeploymentProps.diagnosticExtract` is only referenced by this specific `SoftwareCluster`.

- All `SoftwareCluster`s deployed to one Machine share the same `SoftwareClusterDiagnosticDeploymentProps` and therefore also the same `DiagnosticContributionSet`.

Class	**SoftwareClusterDiagnosticDeploymentProps**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class acts as the owner of all deployment-related diagnostic properties of a SoftwareCluster. **Tags:** atp.recommendedPackage=SoftwareClusterDiagnosticProps			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
diagnostic Extract	DiagnosticContribution Set	0..1	ref	This reference identifies the DiagnosticContributionSet that is applicable for the referencing SoftwareCluster.
powerDown Time	PositiveInteger	0..1	attr	This attribute indicates the minimum time of the stand-by sequence the server will remain in the power-down sequence. The unit is seconds.
validation Configuration	DiagnosticService ValidationConfiguration	0..1	aggr	This aggregation represents the ability to define the order of manufacturer and supplier validations in diagnostic management.

Table 15.8: SoftwareClusterDiagnosticDeploymentProps

15.3.2.3 Diagnostic Service Validation

The configuration of the diagnostic management is mostly done on design level by means of the diagnostic extract (see sections 3.3.8 and 4).

Some aspects, like the validation of service request by manufacturer-specific or supplier-specific checks, can only be decided on deployment level and in the context of the enclosing `SoftwareCluster`, see Figure 15.8.

[TPS_MANI_01350]{DRAFT} **Semantics of `DiagnosticServiceValidationConfiguration`** ⌈For the specific purpose of the validation prior to the execution

of diagnostic services, meta-class DiagnosticServiceValidationConfiguration is defined.

The main purpose of DiagnosticServiceValidationConfiguration is to define the order in which manufacturer-specific (by means of the reference in the role manufacturerValidationOrder) and (separately, via the reference in the role supplierValidationOrder) supplier-specific checks are executed.⌋(*RS_MANI_00005*)

Class	**DiagnosticServiceValidationConfiguration**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class has the ability to configure the order of manufacturer/supplier-checks. **Tags:** atp.recommendedPackage=DiagnosticValueConfigurations			
Base	*ARObject*			
Aggregated by	SoftwareClusterDiagnosticDeploymentProps.validationConfiguration			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
manufacturer ValidationOrder (ordered)	DiagnosticService ValidationMapping	*	ref	This reference defines the order in which validations created by manufacturer are executed.
sovdValidation Order (ordered)	DiagnosticSovdService ValidationPortMapping	*	ref	This reference defines the order in which validations of SOVD requests are executed. **Tags:** atp.Status=candidate
supplier ValidationOrder (ordered)	DiagnosticService ValidationMapping	*	ref	This reference defines the order in which validations created by supplier are executed.

Table 15.9: DiagnosticServiceValidationConfiguration

[constr_10065]{DRAFT} **Validity of DiagnosticServiceValidationConfiguration.manufacturerValidationOrder** ⌈Any DiagnosticServiceValidationConfiguration.manufacturerValidationOrder shall only refer to a DiagnosticServiceValidationMapping where attribute category has been set to MANUFACTURER_VALIDATION **at the time when the creation of the manifest is finished**.⌋()

[constr_10066]{DRAFT} **Validity of DiagnosticServiceValidationConfiguration.supplierValidationOrder** ⌈Any DiagnosticServiceValidationConfiguration.supplierValidationOrder shall only refer to a DiagnosticServiceValidationMapping where attribute category has been set to SUPPLIER_VALIDATION **at the time when the creation of the manifest is finished**.⌋()

15.3.3 Software Cluster Dependency

[TPS_MANI_01215]{DRAFT} **Semantics of meta-class SoftwareClusterDependencyFormula** ⌈Meta-class SoftwareClusterDependencyFormula allows for the definition of a formal condition that can be taken to decide about the dependency to or the conflict with a SoftwareCluster.

The modeling of SoftwareClusterDependencyFormula allows for the definition of nested conditions. The attribute operator is applied on the results of the evaluation of the parts.⌋(*RS_MANI_00035*)

Class	**SoftwareClusterDependencyFormula**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define a dependency among SoftwareClusters.			
Base	*ARObject*, *SoftwareClusterDependencyFormulaPart*			
Aggregated by	SoftwareCluster.conflictsTo, SoftwareCluster.dependsOn, SoftwareClusterDependencyFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
category	CategoryString	0..1	attr	This attribute specializes the semantics of the enclosing SoftwareClusterDependencyFormula.
operator	SoftwareCluster DependencyLogical OperatorEnum	0..1	attr	This logical operator can be used to relate the results of different SoftwareClusterDependencyParts.
part (ordered)	SoftwareCluster DependencyFormula Part	*	aggr	This aggregation represents the ordered collection of the parts of the SoftwareClusterDependencyFormula.

Table 15.10: SoftwareClusterDependencyFormula

Enumeration	**SoftwareClusterDependencyLogicalOperatorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This enumeration provides a set of operators to be used in a SoftwareClusterDependencyFormula.
Aggregated by	SoftwareClusterDependencyFormula.operator
Literal	***Description***
logicalAnd	logical and **Tags:** atp.EnumerationLiteralIndex=0
logicalOr	logical or **Tags:** atp.EnumerationLiteralIndex=1

Table 15.11: SoftwareClusterDependencyLogicalOperatorEnum

[TPS_MANI_01216]{DRAFT} **Semantics of meta-class SoftwareClusterDependencyFormulaPart** ⌈Meta-class SoftwareClusterDependencyFormulaPart represents a part of a SoftwareClusterDependencyFormula. The order of the parts of a SoftwareClusterDependencyFormula is significant.⌋(*RS_MANI_00035*)

Class	***SoftwareClusterDependencyFormulaPart*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents an abstract base class for the definition of different formula parts of a SoftwareClusterDependencyFormula.			
Base	*ARObject*			
Subclasses	SoftwareClusterDependencyCompareCondition, SoftwareClusterDependencyFormula			
Aggregated by	SoftwareClusterDependencyFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table 15.12: SoftwareClusterDependencyFormulaPart

At the same time, SoftwareClusterDependencyFormulaPart is the base class of SoftwareClusterDependencyFormula.

This means that the SoftwareClusterDependencyFormula can aggregate all subclasses of SoftwareClusterDependencyFormulaPart, i.e. SoftwareClusterDependencyFormula and SoftwareClusterDependencyCompareCondition.

[TPS_MANI_01164]{DRAFT} **Semantics of SoftwareCluster.dependsOn** ⌈A SoftwareCluster has the ability to express a dependency to other SoftwareClusters in the role dependsOn.

Attribute SoftwareCluster.dependsOn allows for the definition of a **formal** (potentially nested) dependency condition. The dependency shall be applicable only if the condition defined by dependsOn yields true.⌋*(RS_MANI_00035)*

[TPS_MANI_01217]{DRAFT} **Semantics of meta-class SoftwareClusterDependencyCompareCondition** ⌈Meta-class SoftwareClusterDependencyCompareCondition allows for the definition of a formal condition to compare against the version of the referenced softwareCluster using a given compareType.

The ability to specifically decide about whether to consider the build number in the comparison is implemented by means of attribute considerBuildNumber.⌋*(RS_MANI_00035)*

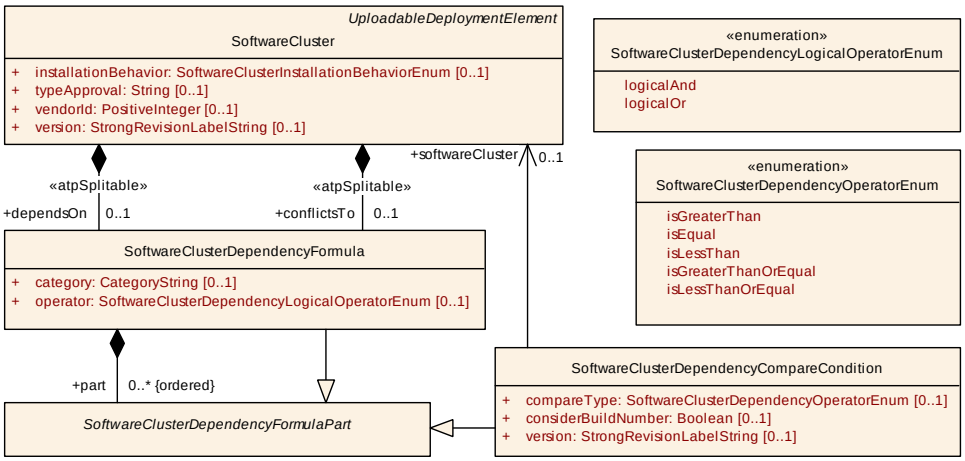


Figure 15.9: Modeling of dependencies in the context of a SoftwareCluster and SoftwareClusterDesign

Class	**SoftwareClusterDependencyCompareCondition**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to specify a concrete dependency condition in the context of a SoftwareClusterDependencyFormula.			
Base	*ARObject*, *SoftwareClusterDependencyFormulaPart*			
Aggregated by	SoftwareClusterDependencyFormula.part			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	SoftwareClusterDependencyCompareCondition			
compareType	SoftwareCluster DependencyOperator Enum	0..1	attr	This attribute identifies the semantics of the compare operator.
considerBuild Number	Boolean	0..1	attr	If this attribute is set to true then the build number shall be taken into account for the comparison. Build numbers don't have to be consecutive but could be created by some kind of hashing algorithm. In such a case it might make sense to include the build number in a test for equality but it is probably not reasonable to apply e.g. a less-than comparison.
softwareCluster	SoftwareCluster	0..1	ref	This reference identifies the SoftwareCluster to which the dependency/conflict applies.
version	StrongRevisionLabel String	0..1	attr	This attribute represents the value of a version against which the comparison shall be executed.

Table 15.13: SoftwareClusterDependencyCompareCondition

[constr_10237]{DRAFT} **Multiplicity of attribute SoftwareClusterDependency-CompareCondition.compareType** ⌈For each SoftwareClusterDependency-CompareCondition, the attribute compareType shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10253]{DRAFT} **Multiplicity of attribute SoftwareClusterDependency-CompareCondition.considerBuildNumber** ⌈For each SoftwareClusterDependencyCompareCondition, the attribute considerBuildNumber shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10254]{DRAFT} **Multiplicity of attribute SoftwareClusterDependencyCompareCondition.version** ⌈For each SoftwareClusterDependency-CompareCondition, the attribute version shall exist **at the time when the creation of the manifest is finished**.⌋*()*

Enumeration	SoftwareClusterDependencyOperatorEnum
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This enumeration provides a choice of operators for comparison within a SoftwareCluster DependencyCompareCondition.
Aggregated by	SoftwareClusterDependencyCompareCondition.compareType
Literal	*Description*
isEqual	equal **Tags:** atp.EnumerationLiteralIndex=1
isGreaterThan	greater than **Tags:** atp.EnumerationLiteralIndex=0
isGreaterThanOr Equal	greater than or equal **Tags:** atp.EnumerationLiteralIndex=3
isLessThan	less than **Tags:** atp.EnumerationLiteralIndex=2
isLessThanOrEqual	less than or equal **Tags:** atp.EnumerationLiteralIndex=4

Table 15.14: SoftwareClusterDependencyOperatorEnum

[TPS_MANI_01214]{DRAFT} **Semantics of `SoftwareCluster.conflictsTo`** ⌈A `SoftwareCluster` has the ability to express a conflict to other `SoftwareCluster`s in the role `conflictsTo`. The semantics is to express that the functionality of the referenced `SoftwareCluster` **inhibits** the installation of the referencing `SoftwareCluster`.

Attribute `SoftwareCluster.conflictsTo` allows for the definition of a **formal** (potentially nested) dependency condition. The dependency shall be applicable only if the condition defined by `conflictsTo` yields `false`.⌋*(RS_MANI_00035)*

An example definition of dependencies among `SoftwareCluster`s can be found in Appendix B.11.1.

15.3.4 References between Software Clusters

There are several strong use cases for the need of referencing into different `SoftwareCluster`s, for example:

- Reference to a `ProvidedApServiceInstance` of `RequiredApServiceInstance` defined in the context of a "host" `SoftwareCluster`.
- Reference to `CommunicationConnector`s defined on `Machine` level from within application `SoftwareCluster`s.
- Reference from a `Process` in one `SoftwareCluster` to an `Executable` in another `SoftwareCluster` with the semantics that the referencing `Process` is just another instance of the `Executable`.

To support such use cases, AUTOSAR provides the definition of dependencies among `SoftwareCluster`s such that a `SoftwareCluster` that contains a reference can define a dependency to another `SoftwareCluster` that contains the referenced object.

[TPS_MANI_01329]{DRAFT} **Reference to model elements in different `SoftwareCluster`s** ⌈If a model element inside a given `SoftwareCluster` defines a reference to another model element and the referenced model element is contained in a different `SoftwareCluster`, then the `SoftwareCluster` that contains the referencing model element shall establish a dependency to the other `SoftwareCluster` by means of an aggregation of `SoftwareClusterDependencyFormula` in the role `dependsOn`.⌋*(RS_MANI_00035)*

[constr_1784]{DRAFT} **Restriction for the reference to `UploadableExclusivePackageElement`** ⌈A reference to an `UploadableExclusivePackageElement` shall not cross the boundary of the enclosing `SoftwareCluster`, i.e. the target `UploadableExclusivePackageElement` of such a reference shall not be located in a different `SoftwareCluster` than the owner of the reference **at the time when the creation of the manifest is finished**.⌋*()*

Class	***UploadableExclusivePackageElement*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class represents an abstract base class for an uploadable package element that is not supposed to be referenced from a different software cluster.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	*PersistencyDeployment*, *PersistencyPortPrototypeToDeploymentMapping*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table 15.15: UploadableExclusivePackageElement

Note that [constr_1784] forbids a reference across `SoftwareCluster`s to an `UploadableExclusivePackageElement`, regardless of whether there is a dependency relation defined or not.

Referencing from one `SoftwareCluster` into another `SoftwareCluster` is only allowed if the referenced `SoftwareCluster` is in the list of dependent `SoftwareCluster`s for the referencing `SoftwareCluster`.

This restriction is formalized in [constr_1785].

[constr_1785]{DRAFT} **Restriction regarding the reference into another `SoftwareCluster`** ⌈A reference from an element in one `SoftwareCluster` to an element located in another `SoftwareCluster` shall only exist if the `SoftwareCluster` that owns the referenced element is referenced by a `SoftwareClusterDependencyCompareCondition` in the context of the mentioned `SoftwareClusterDependencyFormula` in the role `part.softwareCluster` **at the time when the creation of the manifest is finished**.

[constr_1784] applies.⌋*()*

15.3.5 Software Cluster Artifact Checksum

[TPS_MANI_01345]{DRAFT} **Ability to attach checksums to `SoftwareCluster`** ⌈Meta-class `SoftwareCluster` supports the collection of checksums for artifacts like binary executable files, libraries, or persistency files by means of the aggregation of meta-class `ArtifactChecksum` in the role `artifactChecksum`.⌋*(RS_MANI_00035)*

Class	**ArtifactChecksum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This meta-class provides the ability to associate a checksum with a given artifact identified by its URI.
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*
Aggregated by	SoftwareCluster.artifactChecksum

▽

△

Class	**ArtifactChecksum**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checksumValue	String	0..1	attr	This attributes carries the serialized checksum of the corresponding artifact.
uri	UriString	0..1	attr	This attribute represents the URI of the artifact on which the checksum shall be computed. **Stereotypes:** atpIdentityContributor

Table 15.16: ArtifactChecksum

[TPS_MANI_01346]{DRAFT} **No formal definition of checksum algorithm** ⌈The checksum algorithm used for computing the `ArtifactChecksum.checksumValue` is not formally defined.

A description of the algorithm shall be contained in the `CryptoProvider.cryptoProviderDocumentation` of the `CryptoProvider` that is associated with a specific `ArtifactChecksum` via the existence of `ArtifactChecksumToCryptoProviderMapping`.⌋ *(RS_MANI_00035)*

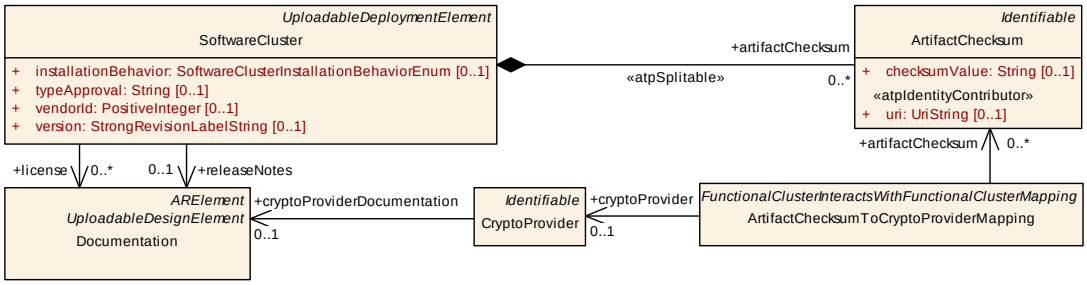


Figure 15.10: Modeling of the artifact checksum of a `SoftwareCluster`

Class	**ArtifactChecksumToCryptoProviderMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class provides the ability to associate a CryptoProvider with a collection of the checksums computed for artifacts. **Tags:** atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
artifact Checksum	ArtifactChecksum	*	ref	This reference identifies the collection of Artifact Checksums associated with the corresponding Crypto Provider.
cryptoProvider	CryptoProvider	0..1	ref	This reference identifies the applicable CryptoProvider.

Table 15.17: ArtifactChecksumToCryptoProviderMapping

15.3.6 Software Cluster Artifact Locator

[TPS_MANI_01381]{DRAFT} **Semantics of meta-class `ArtifactLocator`** ⌈Meta-class `ArtifactLocator` represents a generic approach to configure a location of a referrable model element in the role `representedModelElement` by means of attribute `uri`.⌋*()*

The modeling of meta-class `ArtifactLocator` as a generic concept is depicted in Figure 15.11.

The association between an artifact and its location is a 1:1 relation. It is not allowed to associate a second (or more) location to a model element that is already the target of a reference in the role `ArtifactLocator.representedModelElement`.

[constr_10380]{DRAFT} **Target of `ArtifactLocator.representedModelElement`** ⌈The target of a reference in the role `ArtifactLocator.representedModelElement` shall not be the target of another reference in the role `ArtifactLocator.representedModelElement`.

This rule shall be imposed **`at the time when the creation of the manifest is finished`**.⌋*()*

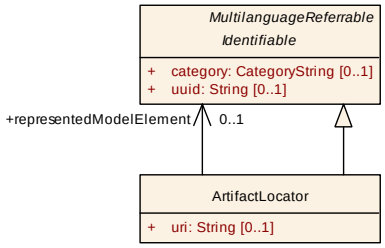


Figure 15.11: Modeling of the artifact location of a model element that belongs to a `SoftwareCluster`

Class	**ArtifactLocator**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class has the ability to define the location of an artifact that is represented by a model element, e.g. Executable.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SoftwareCluster.artifactLocator			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
represented ModelElement	Identifiable	0..1	ref	This reference identifies the model element that is represented by the artifact.
uri	String	0..1	attr	This attribute describes the location of the artifact.

Table 15.18: ArtifactLocator

[constr_10381]{DRAFT} **Existence of attribute `ArtifactLocator.uri`** ⌈For each `ArtifactLocator`, the attribute `uri` shall exist **`at the time when the creation of the manifest is finished`**⌋*()*

[constr_10382]{DRAFT} **Existence of attribute ArtifactLocator.representedModelElement** ⌈For each ArtifactLocator, the attribute representedModelElement shall exist **at the time when the creation of the manifest is finished**⌋()

Meta-class ArtifactLocator is used in the context of the modeling of a SoftwareCluster to facilitate the configuration of model elements that belong to the SoftwareCluster and for which a representation in the software exists that can have a "location" (e.g. a file that contains executable code).

Therefore, one example for the application of ArtifactLocator in the context of the modeling of a SoftwareCluster is the configuration of the location of binary executable code that corresponds to an Executable, which is claimed by a SoftwareCluster via the indirection of SoftwareCluster.containedProcess that in turn references the Executable in the role executable.

It is obviously assumed that the entire code represented by an Executable is locatable in one place described by the attribute uri.

This relation is depicted in Figure 15.12. The ability to refer to an Executable in the role representedModelElement is inherited to Executable by its superclass Identifiable.

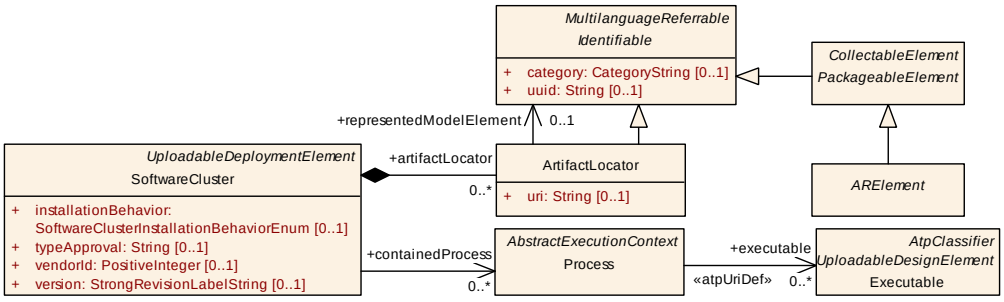


Figure 15.12: Modeling of the location of an Executable that (via a containedProcess) belongs to a SoftwareCluster

[TPS_MANI_01382]{DRAFT} **Location of artifact that contains executable code represented by Executable** ⌈The location of the executable code that is represented by Executable can be identified by means of the aggregation SoftwareCluster.artifactLocator where attribute ArtifactLocator.representedModelElement.dest is set to the value EXECUTABLE.⌋()

15.4 Software Package

The existence of the SoftwareCluster by itself is not sufficient for installation. Actually, the SoftwareCluster gets wrapped into a so-called SoftwarePackage that comes with an own manifest format that is at least partly standardized.

The difference between the semantics of a SoftwareCluster and the semantics of SoftwarePackage is that a SoftwareCluster focuses on the structure of the

software itself while the SoftwarePackage is created to handle the logistics aspect of the software installation.

[TPS_MANI_01221]{DRAFT} **Semantics of meta-class SoftwarePackage** ⌈The purpose of meta-class SoftwarePackage is to cover the "logistics" aspect of the software installation procedure.⌋ *(RS_MANI_00035)*

[constr_1690]{DRAFT} **SoftwareCluster shall only be referenced by a single SoftwarePackage.** ⌈Each SoftwareCluster shall only be referenced by a single SoftwarePackage **at the time when the creation of the manifest is finished**.⌋ *()*

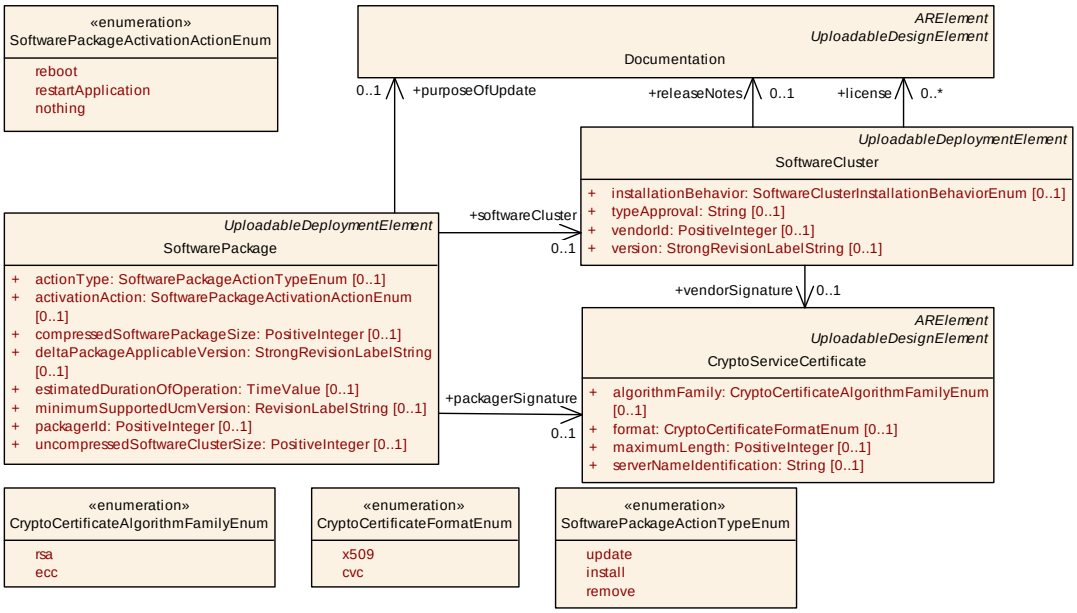


Figure 15.13: Modeling of SoftwarePackage

Class	**SoftwarePackage**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to formalize the content of a software package. **Tags:** atp.recommendedPackage=SoftwarePackages			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
actionType	SoftwarePackageAction TypeEnum	0..1	attr	This attribute defines the action to be taken in the step of processing the enclosing SoftwarePackage.
activationAction	SoftwarePackage ActivationActionEnum	0..1	attr	This attribute governs the action to be taken after the installation of the SoftwareCluster completed.
compressed Software PackageSize	PositiveInteger	0..1	attr	This size represents the size of the compressed Software Package.

△

Class	SoftwarePackage			
deltaPackage ApplicableVersion	StrongRevisionLabelString	0..1	attr	This attribute identifies the version of the included SoftwareCluster for which the enclosing SoftwarePackage can be used as a delta update
estimatedDurationOfOperation	TimeValue	0..1	attr	This attribute provides an estimation about how long the operation of the SoftwarePackage is going to take for its transfer, processing and activation when updated standalone (not within an update campaign)
minimumSupportedUcmVersion	RevisionLabelString	0..1	attr	This attribute identifies the minimum supported version of the UCM for this SoftwarePackage.
packagerId	PositiveInteger	0..1	attr	This attribute identifies Id of the organization that provides the packager generating the SoftwarePackage.
packagerSignature	CryptoServiceCertificate	0..1	ref	This reference identifies the certificate that represents the packager's signature.
purposeOfUpdate	Documentation	0..1	ref	The referenced Documentation is supposed to provide a description of the purpose of the update.
softwareCluster	SoftwareCluster	0..1	ref	This reference identifies the SoftwareCluster that belongs to the SoftwarePackage. The nature of this relation is actually more like an aggregation than a reference. But the relation is still modelled as a reference because two ARElements cannot aggregate each other.
uncompressedSoftwareClusterSize	PositiveInteger	0..1	attr	This attribute gives an indication about the storage that has to be available on the target.

Table 15.19: SoftwarePackage

[constr_10240]{DRAFT} **Multiplicity of attribute `SoftwarePackage.actionType`** ⌈For each `SoftwarePackage`, the attribute `actionType` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10241]{DRAFT} **Multiplicity of attribute `SoftwarePackage.compressedSoftwarePackageSize`** ⌈For each `SoftwarePackage`, the attribute `compressedSoftwarePackageSize` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10242]{DRAFT} **Multiplicity of attribute `SoftwarePackage.minimumSupportedUcmVersion`** ⌈For each `SoftwarePackage`, the attribute `minimumSupportedUcmVersion` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10243]{DRAFT} **Multiplicity of attribute `SoftwarePackage.packagerId`** ⌈For each `SoftwarePackage`, the attribute `packagerId` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10244]{DRAFT} **Multiplicity of reference in the role `SoftwarePackage.packagerSignature`** ⌈For each `SoftwarePackage`, the reference in the role `packagerSignature` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10245]{DRAFT} **Multiplicity of reference in the role SoftwarePackage.softwareCluster** ⌈For each SoftwarePackage, the reference in the role softwareCluster shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10246]{DRAFT} **Multiplicity of attribute SoftwarePackage.uncompressedSoftwareClusterSize** ⌈For each SoftwarePackage, the attribute uncompressedSoftwareClusterSize shall exist **at the time when the creation of the manifest is finished**.⌋*()*

Primitive	**RevisionLabelString**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This primitive represents an internal AUTOSAR revision label which identifies an engineering object. It represents a pattern which • supports three integers representing from left to right MajorVersion, MinorVersion, PatchVersion. • may add an application specific suffix separated by one of ".", "_", ";". Legal patterns are for example: • 4.0.0 • 4.0.0.1234565 • 4.0.0_vendor specific;13 • 4.0.0;12 **Tags:** xml.xsd.customType=REVISION-LABEL-STRING xml.xsd.pattern=[0-9]+\.[0-9]+\.[0-9]+([\._;].*)? xml.xsd.type=string

Table 15.20: RevisionLabelString

In other words, AUTOSAR factually assumes a 1:1 relation between SoftwarePackage and SoftwareCluster. Such a relation would otherwise typically be modeled by means of an aggregation with the multiplicity 1.

However, a SoftwareCluster is derived from base class PackageableElement which is only aggregated by ARPackage. Subclasses of PackageableElement – by convention – shall not be aggregated by any other meta-class.

[TPS_MANI_01222]{DRAFT} **Cryptographic signature of SoftwarePackage** ⌈A SoftwarePackage also needs to be signed cryptographically. For this purpose, meta-class CryptoServiceCertificate is referenced in the role packagerSignature.⌋*(RS_MANI_00035)*

[TPS_MANI_01223]{DRAFT} **Semantics of attribute SoftwarePackage.packagerId** ⌈Attribute SoftwarePackage.packagerId contains the value of the AUTOSAR vendor Id of the organization that created software tool that created the SoftwarePackage.⌋*(RS_MANI_00035)*

For clarification, a UCM can only accept packages that are generated by a packaging tool developed by the same organization that also developed the UCM itself. The vendor of the SoftwareCluster contained in the SoftwarePackage can obviously be different.

[TPS_MANI_01225]{DRAFT} **Actions taken during installation of a SoftwarePackage** ⌈It is necessary to define the concrete activity that shall be taken to handle the SoftwarePackage on the target machine. Possible actions are:

- Do a clean installation of a SoftwareCluster.
- Update a previously installed SoftwareCluster.
- Remove a SoftwareCluster

These options are formalized by means of meta-class SoftwarePackageActionTypeEnum and attribute SoftwarePackage.actionType.⌋*(RS_MANI_00035)*

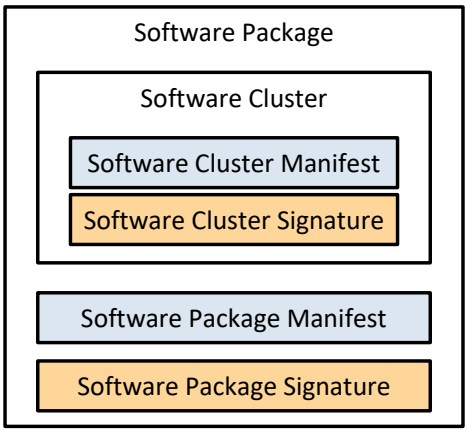


Figure 15.14: Conceptual relation of SoftwarePackage and SoftwareCluster

Enumeration	**SoftwarePackageActionTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This enumeration provides a choice of possible actions for the handling of a software package.
Aggregated by	SoftwarePackage.actionType
Literal	***Description***
install	Do a clean installation of a SoftwareCluster. **Tags:** atp.EnumerationLiteralIndex=1
remove	Remove a SoftwareCluster. **Tags:** atp.EnumerationLiteralIndex=2
update	Update a previously installed SoftwareCluster. **Tags:** atp.EnumerationLiteralIndex=0

Table 15.21: SoftwarePackageActionTypeEnum

[TPS_MANI_01344]{DRAFT} **Actions taken after installation of a SoftwarePackage** ⌈After a SoftwarePackage has been installed on the target machine it is possible to execute one of the following actions:

- Reboot the target platform.
- Restart the installed SoftwareCluster.

- Do nothing.

These options are formalized by means of meta-class SoftwarePackageActivationActionEnum and attribute SoftwarePackage.activationAction.⌋(*RS_MANI_00035*)

Enumeration	**SoftwarePackageActivationActionEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This enumeration provides a choice of possible actions to be executed on installing a Software Package to a target Machine.
Aggregated by	SoftwarePackage.activationAction
Literal	*Description*
nothing	The installation has no immediate consequences in terms of other software on the target. **Tags:** atp.EnumerationLiteralIndex=2
reboot	Reboot the whole Machine. **Tags:** atp.EnumerationLiteralIndex=0
restartApplication	Restart the application software on the target Machine. **Tags:** atp.EnumerationLiteralIndex=1

Table 15.22: SoftwarePackageActivationActionEnum

15.5 Vehicle Package

15.5.1 Overview

The ability to handle SoftwarePackages is the prerequisite for an important further step: the execution of an **update campaign that applies for the whole vehicle**. The basis for the update campaign is the definition of meta-class VehiclePackage.

[TPS_MANI_01290]{DRAFT} **VehiclePackage names affected UCMs** ⌈Meta-class VehiclePackage has the ability to describe the set of UCMs that are affected by the update campaign by means of aggregating meta-class UcmDescription in the role ucm.⌋(*RS_MANI_00035*)

[TPS_MANI_01291]{DRAFT} **Identification of an actual UCM in the context of an update campaign** ⌈It is necessary to unambiguously identify the individual UCMs that are affected in the update campaign. For this purpose, meta-class UcmDescription defines attribute identifier.

By means of the reference to UcmModuleInstantiation in the role ucmModuleInstantiation it is in addition possible to identify the actual UCMs (represented by a UcmModuleInstantiation) that are relevant for the update campaign.

In order to be able to resolve the reference it is necessary to have access to the manifest model of the target Machine.⌋(*RS_MANI_00035*)

[constr_1731]{DRAFT} **Value of `UcmDescription.identifier` in the scope of a `VehiclePackage`** ⌈Within the scope of any given `VehiclePackage`, no two `UcmDescriptions` shall define the same value of attribute `identifier` **at the time when the creation of the manifest is finished**.⌋*()*

[TPS_MANI_01292]{DRAFT} **Definition of fallback-order for UCM master** ⌈The update campaign is executed under the management of one UCM that acts as a "master UCM".

If this UCM goes down for some reason, `VehiclePackage` has the ability to define an **ordered** list of other candidates for becoming the "master UCM" by means of the reference to meta-class `UcmDescription` in the role `ucmMasterFallback`.⌋*(RS_MANI_00035)*

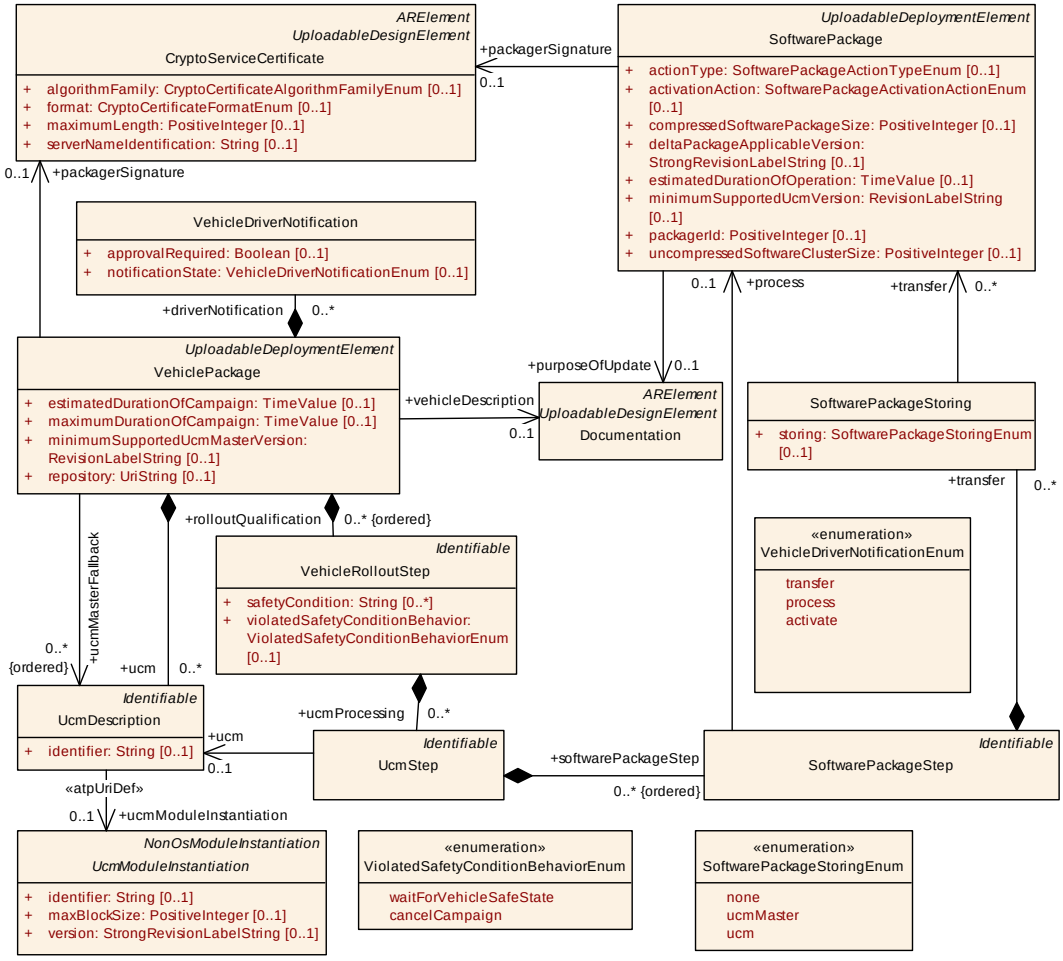


Figure 15.15: Modeling of `VehiclePackage`

[TPS_MANI_01294]{DRAFT} **Update campaign depends on driver's acceptance** ⌈For obvious reasons, it is not possible to arbitrarily trigger the execution of an update campaign at any time. It is the prerogative of the vehicle driver to decide about the amount and consequence of the UCM activities with respect to an update campaign.

For this purpose `VehiclePackage` aggregates meta-class `VehicleDriverNotification` in the role `driverNotification`.⌋*(RS_MANI_00035)*

Class	**VehiclePackage**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define a vehicle package for executing an update campaign. **Tags:** atp.recommendedPackage=VehiclePackages			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
driver Notification	VehicleDriver Notification	*	aggr	This aggregation provides the ability to configure the necessary driver notifications.
estimated DurationOf Campaign	TimeValue	0..1	attr	This attribute provides an estimation about how long the campaign based on the VehiclePackage is going to take.
maximum DurationOf Campaign	TimeValue	0..1	attr	Maximum time allowed for the campaign to be active until UCM Master automatically cancels the campaign.
minimum SupportedUcm MasterVersion	RevisionLabelString	0..1	attr	This attribute identifies the minimum supported version of the UCM Master for this VehiclePackage.
packager Signature	CryptoService Certificate	0..1	ref	This reference identifies the certificate that represents the packager's signature.
repository	UriString	0..1	attr	This attribute identifies the repository where the Vehicle Package is stored.
rollout Qualification (ordered)	VehicleRolloutStep	*	aggr	This represents the rollout qualification.
ucm	UcmDescription	*	aggr	This aggregation represents the UcmDescriptions to be considered in the context of the VehiclePackage.
ucmMaster Fallback (ordered)	UcmDescription	*	ref	This reference lists the fallback order of Ucms that can take over the master role if the master goes down.
vehicle Description	Documentation	0..1	ref	This reference identifies the vehicle description.

Table 15.23: VehiclePackage

[constr_10247]{DRAFT} **Multiplicity of reference in the role `VehiclePackage`. `packagerSignature`** ⌈For each `VehiclePackage`, the reference in the role `packagerSignature` shall exist **`at the time when the creation of the manifest is finished`**.⌋*()*

Class	**UcmDescription**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define an identifier for a given UCM.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	VehiclePackage.ucm			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
identifier	String	0..1	attr	This attribute represents the unique identification of the UcmIdentifier.

▽

△

Class	UcmDescription			
ucmModule Instantiation	UcmModuleInstantiation	0..1	ref	This reference identifies the applicable UcmModule Instantiation. **Stereotypes:** atpUriDef

Table 15.24: UcmDescription

[constr_10248]{DRAFT} **Multiplicity of reference in the role `UcmDescription.identifier`** ⌈For each `UcmDescription`, the reference in the role `identifier` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

Class	VehicleDriverNotification			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class provides the ability to configure a notification of the vehicle driver with respect to the update of vehicle software.			
Base	*ARObject*			
Aggregated by	VehiclePackage.driverNotification			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
approval Required	Boolean	0..1	attr	This attribute controls whether approval is required for the driver notification.
notificationState	VehicleDriver NotificationEnum	0..1	attr	This attribute is used to configure the notification state.

Table 15.25: VehicleDriverNotification

[constr_10249]{DRAFT} **Multiplicity of reference in the role `VehicleDriverNotification.approvalRequired`** ⌈For each `VehicleDriverNotification`, the reference in the role `approvalRequired` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

[constr_10250]{DRAFT} **Multiplicity of reference in the role `VehicleDriverNotification.notificationState`** ⌈For each `VehicleDriverNotification`, the reference in the role `notificationState` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

Enumeration	VehicleDriverNotificationEnum
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This meta-class provides available options for vehicle driver notification.
Aggregated by	VehicleDriverNotification.notificationState
Literal	**Description**
activate	Software package shall be activated. **Tags:** atp.EnumerationLiteralIndex=2
process	Processing of software package shall be executed **Tags:** atp.EnumerationLiteralIndex=1
transfer	Software shall be transferred to the vehicle. **Tags:** atp.EnumerationLiteralIndex=0

Table 15.26: VehicleDriverNotificationEnum

15.5.2 VehicleRolloutStep

[TPS_MANI_01295]{DRAFT} **Semantics of `VehicleRolloutStep`** ⌈The purpose of an update campaign is to roll out the installation or update of `SoftwarePackage`s in the context of given UCMs. Each `VehicleRolloutStep` may apply to several UCMs at the same time.

The activation of the `SoftwarePackage`s processed in the context of the enclosing `VehiclePackage` is triggered as the last element of the `rolloutQualification` is processed.⌋ *(RS_MANI_00035)*

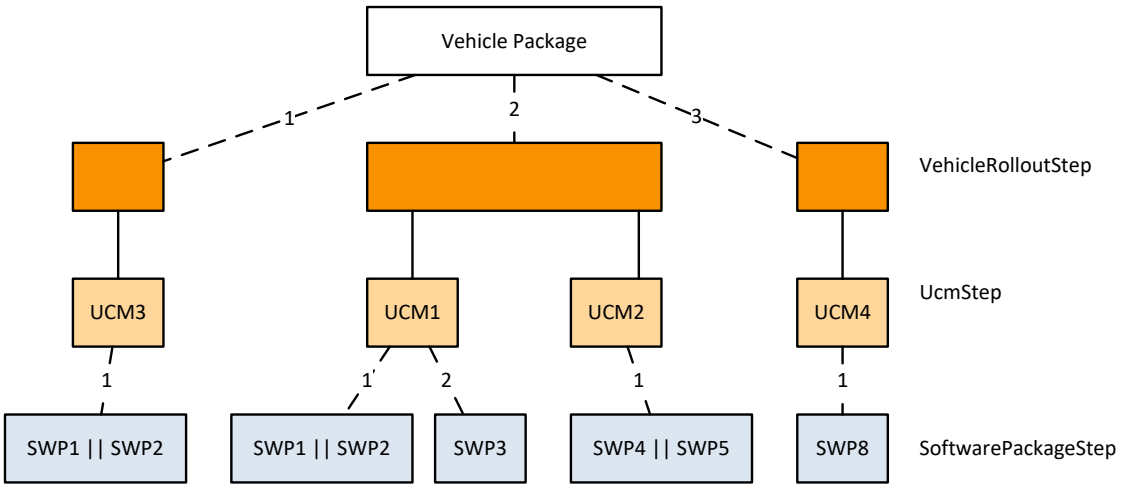


Figure 15.16: Conceptual view on an update campaign

Figure 15.16 takes a conceptual view on the structure of an update campaign and associates the relevant aspects of this view to meta-classes explained in this chapter. Associations that are labeled by a number indicate that an ordering is implied with the respective step.

For example, the execution of the update campaign happens in dedicated steps formalized as `VehicleRolloutStep`, as explained in [TPS_MANI_01295]. Each of the three steps sketched in the picture would be modeled as an individual `VehicleRolloutStep`.

The rollout action is formalized by `UcmStep.softwarePackageStep`. In other words, it is possible to specify a different `softwarePackageStep` for each individual UCM.

The individual `VehicleRolloutStep`s are executed in the order in which they are aggregated at the enclosing `VehiclePackage`. This aspect is in more detail explained by [TPS_MANI_01296].

[TPS_MANI_01296]{DRAFT} **Ordered execution of rollout steps in an update campaign** ⌈The individual `VehicleRolloutStep`s defined in the context of a given `VehiclePackage` are executed in the defined order and therefore the aggregation of `VehicleRolloutStep` at `VehiclePackage` is `ordered`.⌋ *(RS_MANI_00035)*

Class	VehicleRolloutStep			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define a rollout-condition for a vehicle update campaign.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	VehiclePackage.rolloutQualification			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
safetyCondition	String	*	attr	This attribute represents a list of textual safety conditions (e.g.: close the driver window) that need to be fulfilled before the rollout step can proceed and need to be maintained while the campaign's rolloutQualification is executed.
ucmProcessing	UcmStep	*	aggr	This aggregation collects the UcmProcessingSteps that make up the rollout step.
violatedSafety Condition Behavior	ViolatedSafetyCondition BehaviorEnum	0..1	attr	This attribute provides options for the configuration of the reaction to a violated safety condition.

Table 15.27: VehicleRolloutStep

[constr_10429]{DRAFT} **Existence of attribute `VehicleRolloutStep.violatedSafetyConditionBehavior`** ⌈For each `VehicleRolloutStep`, the attribute `violatedSafetyConditionBehavior` shall exist **at the time when the creation of the manifest is finished**.⌋*()*

15.5.3 UcmStep

[TPS_MANI_01297]{DRAFT} **Semantics of meta-class `UcmStep`** ⌈Each `VehicleRolloutStep` consists of a number of `UcmStep`s (aggregated by `VehicleRolloutStep` in the role `ucmProcessing`). Each `UcmStep` refers to a specific UCM (represented by `UcmDescription`) in the role `ucm`.⌋*(RS_MANI_00035)*

[TPS_MANI_01298]{DRAFT} **No ordering of `VehicleRolloutStep.ucmProcessing`** ⌈Each `UcmStep` defined in the context of an enclosing `VehicleRolloutStep` can be handled without the consideration of a dedicated order. Therefore, the aggregation `VehicleRolloutStep.ucmProcessing` is not labeled as `ordered`.⌋*(RS_MANI_00035)*

Class	UcmStep			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define a rollout-condition for a vehicle update campaign.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	VehicleRolloutStep.ucmProcessing			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
software PackageStep (ordered)	SoftwarePackageStep	*	aggr	This aggregation represents the sequence of activities to be carried out in the context of the respective UCM.
ucm	UcmDescription	0..1	ref	This reference identifies the UCM for which the rollout step applies.

Table 15.28: UcmStep

15.5.4 SoftwarePackageStep

[TPS_MANI_01299]{DRAFT} **Aggregation of SoftwarePackageSteps at UcmStep** ⌈Each UcmStep consists of an ordered collection of SoftwarePackageSteps. This means that the order in which SoftwarePackages are handled in the scope of one UcmStep is significant.⌋*(RS_MANI_00035)*

Class	SoftwarePackageStep			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the configuration of an activation step in the context of software package activation.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	UcmStep.softwarePackageStep			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
process	SoftwarePackage	0..1	ref	This reference identifies the SoftwarePackage to be processed in the enclosing SoftwarePackageStep.
transfer	SoftwarePackage Storing	*	aggr	This aggregation clarifies the storing of the Software Package.

Table 15.29: SoftwarePackageStep

[TPS_MANI_01300]{DRAFT} **Semantics of reference SoftwarePackageStep.transfer.transfer** ⌈The reference SoftwarePackageStep.transfer.transfer identifies the SoftwarePackages that are supposed to be transferred in the context of the enclosing SoftwarePackageStep.

It is positively supported that SoftwarePackages are transferred in parallel and therefore the multiplicity of the reference in the role transfer has been set to 0..*.⌋*(RS_MANI_00035)*

[TPS_MANI_01301]{DRAFT} **Semantics of aggregation SoftwarePackageStep.transfer** ⌈By means of the aggregation of SoftwarePackageStoring it is possible to specify for each individual SoftwarePackage to specify whether and where the SoftwarePackage is stored in the vehicle.

This information is specifically provided by attribute SoftwarePackageStoring.storing of type SoftwarePackageStoringEnum.⌋*(RS_MANI_00035)*

Class	SoftwarePackageStoring			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class provides the ability to specify whether and where the referenced SoftwarePackage is stored.			
Base	*ARObject*			
Aggregated by	SoftwarePackageStep.transfer			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
storing	SoftwarePackage StoringEnum	0..1	attr	This attribute clarifies whether and where the referenced SoftwarePackage is stored.
transfer	SoftwarePackage	*	ref	This reference identifies the SoftwarePackage(s) to be transferred in the enclosing SoftwarePackageStep.

Table 15.30: SoftwarePackageStoring

[constr_10256]{DRAFT} **Multiplicity of reference in the role SoftwarePackageStoring.storing** ⌈For each SoftwarePackageStoring, the reference in the role storing shall exist **at the time when the creation of the manifest is finished**.⌋*()*

Enumeration	**SoftwarePackageStoringEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This enumeration controls where software packages are stored.
Aggregated by	SoftwarePackageStoring.storing
Literal	***Description***
none	No storing in vehicle. **Tags:** atp.EnumerationLiteralIndex=0
ucm	Storing in UCM (subordinate). **Tags:** atp.EnumerationLiteralIndex=2
ucmMaster	Storing in Ucm Master. **Tags:** atp.EnumerationLiteralIndex=1

Table 15.31: SoftwarePackageStoringEnum

[TPS_MANI_01302]{DRAFT} **Semantics of reference SoftwarePackageStep.process** ⌈The reference SoftwarePackageStep.process identifies the SoftwarePackage that is supposed to be processed in the context of the enclosing SoftwarePackageStep.

The processing of SoftwarePackages happens strictly one after the other and therefore the reference process can only have the multiplicity 0..1. The strict order of processing is guaranteed by the aggregation of the SoftwarePackageStep at UcmStep.⌋*(RS_MANI_00035)*

[TPS_MANI_01306]{DRAFT} **Simultaneous existence of references in the role SoftwarePackageStep.transfer and SoftwarePackageStep.process** ⌈It is possible that the references SoftwarePackageStep.transfer and SoftwarePackageStep.process simultaneously exist to the identical SoftwarePackage in the context of the same SoftwarePackageStep.

The semantics of such a configuration is that the SoftwarePackage that is referenced by the two roles owned by the same SoftwarePackageStep is "streamed", i.e. transferred and processed in one step represented by the SoftwarePackageStep.⌋ *(RS_MANI_00035)*

Examples for the usage of SoftwarePackageStep can be found in Appendix B.11.2.

16 Interoperability between Classic Platform and Adaptive Platform

This chapter collects restrictions for interoperability between Classic Platform and Adaptive Platform.

16.1 Usage of majorVersion in the SOME/IP network binding

In case of a SOME/IP communication between Classic Platform and Adaptive Platform the usage of SomeipServiceInterfaceDeployment.serviceInterfaceVersion.majorVersion is restricted.

If several ProvidedSomeipServiceInstances are defined with the same SomeipServiceInterfaceDeployment.serviceInterfaceId and different SomeipServiceInterfaceDeployment.serviceInterfaceVersion.majorVersions and these ProvidedSomeipServiceInstances are mapped to the same Socket Address (UDP/TCP Port and IP Address) by SomeipServiceInstanceToMachineMapping then particular restrictions apply in case that the ProvidedSomeipServiceInstances **are consumed on a Classic Platform ECU**.

In such a scenario the same MessageId may be used for ServiceInterface elements like an event in the different ProvidedSomeipServiceInstances that have different majorVersions.

On the Classic Platform in the SoAd module, this will result in different Pdus that have the same headerId in the same SocketConnection. In the AUTOSAR Architecture of the Classic Platform one part of the SOME/IP Header is evaluated in the SOME/IP Transformer (RequestId, Protocol Version, Interface Version, Message Type, Return Code) and the other part in the Socket Adaptor (MessageId, Length).

This means that the Socket Adaptor is not able to evaluate the MajorVersion in the Pdu and can not determine the source of the Pdu.

The following restrictions apply in case that the ServiceInterface contains methods and/or fields with hasGetter or hasSetter set to true: If two or more ProvidedSomeipServiceInstances are defined using the same serviceInterfaceId and different majorVersions and these ProvidedSomeipServiceInstances are mapped to the same Socket Address (same UDP/TCP Port and IP Address) then the destination IP address, the destination port number, and the Level 4 protocol (Udp/Tcp) fields of header of IP packets containing calls that are sent to the ProvidedSomeipServiceInstances are identical.

In such a scenario, the ProvidedSomeipServiceInstances may still use identical methodIds if the following condition applies:

a) At any point in time only one of the ProvidedSomeipServiceInstances is active, and only clients of that ProvidedSomeipServiceInstance send requests to the ProvidedSomeipServiceInstance.

In all other cases, the methodIds of the two ProvidedSomeipServiceInstances should not overlap.

The following restrictions apply for ServiceInterface events and fields with hasNotifier set to true: If two or more ProvidedSomeipServiceInstances are defined using the same serviceInterfaceId and different majorVersions and these ProvidedSomeipServiceInstances are mapped to the same Socket Address (same UDP/TCP Port and IP Address), then the source IP address, the source port number, and the Level 4 protocol (Udp) fields of header of IP packets containing event PDUs that are sent to the clients of the ProvidedSomeipServiceInstances are identical.

In such a scenario, the ProvidedSomeipServiceInstances may use identical eventIds if at least one of the following conditions holds for any pair of the ProvidedSomeipServiceInstances:

a) At any point in time only one of the ProvidedSomeipServiceInstances is active

b) If two or more ProvidedSomeipServiceInstances can send events or Field notifiers at the same time, the ProvidedSomeipServiceInstances may still use identical eventIds if at least one of the following IP header fields of the IP packet containing the event is different for any pair of SomeipEventDeployments identified by the same eventId:

b1) Destination IP address (== IP address of client)

b2) Destination port number (== client port number)

A Reference Material

A.1 Terms and Abbreviations

The main list of terms and abbreviations are defined in [37]. The following table contains the list of terms and abbreviations used in the scope of this document which are not already defined in [37] along with the spelled-out meaning of each of the abbreviations.

Abbreviation	*Meaning*
AES	Advanced Encryption Standard
ATP	AUTOSAR Template Profile
ARXML	AUTOSAR XML
CTM	Counter Mode
DDS	Data Distribution Service
DES	Data Encryption Standard
DHCP	Dynamic Host Configuration Protocol
DM	Diagnostic Manager
DNS	Domain Name System
DTC	Diagnostic Trouble Code
ECB	Electronic Code Book
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
EDDSA	Edwards-Curve Digital Signature Algorithm
FQDN	Fully-Qualified Domain Name
GCM	Galios/Counter Mode
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transport Protocol
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
ICV	MACsec Integrity Check Value
ID	Identifier
IP	Internet Protocol
ISO	International Standardization Organization
JSON	JavaScript Object Notation
KaY	MACsec Key Agreement Entity
LAN	Local Area Network
MAC	Media Access Control
MAC	Message Authentication Code
MACsec	Media Access Control security
MD	Message Digest

▽

△

Abbreviation	*Meaning*
MTU	Maximum Transmission Unit
mDNS	Multicast DNS
NDP	Neighbor Discovery Protocol
NM	Network Management
NV	Non-Volatile
PAE	MACsec Port Access Entity
PHM	Platform Health Management
PKCS	Public Key Cryptography Standards
POSIX	Portable Operating System Interface
PSK	Pre-Shared Key
ROM	Read-Only Memory
RSA	Cryptographic approach according to Rivest, Shamir, and Adleman
SAK	MACsec Secure Association key
SD	Service Discovery
SDG	Special Data Group
SecY	MACsec Security Entity
SHA	Secure Hash Algorithm
SOME/IP	Scalable service-Oriented MiddlewarE over IP
SOVD	Service-Oriented Vehicle Diagnostics
SWC	Software Component
TLS	Transport Layer Security
TLV	Tag Length Value
TTL	Time to Live
UDS	Unified Diagnostic Services
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
VLAN	Virtual Local Area Network
VSA	Variable Size Array

Table A.1: Terms and Abbreviations used in the scope of this Document

A.2 Imposition Times of Constraints

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

The imposition times that are considered applicable in the scope of this document[1] are listed in Table A.2.

Please note that the imposition times are intentionally rendered as technical terms such that it is possible to link back from each constraint to the definition of the affected imposition time in Table A.2.

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

Imposition Time	Description
at the time before the generation of the ara API starts	This imposition time is aimed at the time when a software-component is ready for generating the header files such that the implementation of the software-component can be started.
at the time when the design of the Executable is complete	This imposition time is aimed at the time when an `Executable` is finished, i.e. it shall be used in constraints that target the consistency of the modeling of `Executable`.
at the time when the ProcessDesign is complete	This imposition time is aimed at the time when a `ProcessDesign` is finished, i.e. it shall be used in constraints that target the consistency of the modeling of `ProcessDesign`.
at the time when the GrantDesign is complete	This imposition time is aimed at the time when a `GrantDesign` is finished, i.e. it shall be used in constraints that target the consistency of the modeling of `GrantDesign`.
at the time when the system design is complete	This imposition time denotes the step in the workflow, where the system design is about to be finished.
at the time when the sub-system design is complete	This imposition time denotes the step in the workflow, where the sub-system design is about to be finished.
at the time when the diagnostic design is complete	This imposition time denotes the step in the workflow, where the diagnostic design is about to be finished.
at the time when the machine design is complete	This imposition time denotes the step in the workflow, where the machine design is about to be finished.
at the time when the creation of the manifest is finished	This imposition time denotes the step in the workflow, where the manifest is considered complete such that the installation on a target platform can be started.

Table A.2: Imposition Times considered in the scope of this document

[1] Different imposition times may be defined in the context of other AUTOSAR standard documents

A.3 Requirements Tracing

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

The following table A.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_01001] [TPS_MANI_01010] [TPS_MANI_01370] [TPS_MANI_01383]
[RS_MANI_00002]	Declaration of provided and required services in an application	[TPS_MANI_01039] [TPS_MANI_01040] [TPS_MANI_01053] [TPS_MANI_01057] [TPS_MANI_01190] [TPS_MANI_03210] [TPS_MANI_03211] [TPS_MANI_03212]
[RS_MANI_00003]	Specification of service interfaces	[TPS_MANI_01001] [TPS_MANI_01004] [TPS_MANI_01005] [TPS_MANI_01006] [TPS_MANI_01007] [TPS_MANI_01033] [TPS_MANI_01034] [TPS_MANI_01035] [TPS_MANI_01064] [TPS_MANI_03118] [TPS_MANI_03119] [TPS_MANI_03223] [TPS_MANI_03291] [TPS_MANI_03628]
[RS_MANI_00004]	Support of application design	[TPS_MANI_01010] [TPS_MANI_01228] [TPS_MANI_01229]
[RS_MANI_00005]	Configuration of diagnostic capabilities of an application	[TPS_MANI_01048] [TPS_MANI_01049] [TPS_MANI_01050] [TPS_MANI_01230] [TPS_MANI_01259] [TPS_MANI_01260] [TPS_MANI_01261] [TPS_MANI_01262] [TPS_MANI_01263] [TPS_MANI_01326] [TPS_MANI_01350] [TPS_MANI_01351] [TPS_MANI_01352] [TPS_MANI_01358] [TPS_MANI_01360] [TPS_MANI_01361] [TPS_MANI_01362] [TPS_MANI_01372] [TPS_MANI_01396] [TPS_MANI_01397] [TPS_MANI_01398] [TPS_MANI_01405] [TPS_MANI_01416] [TPS_MANI_01417] [TPS_MANI_01418] [TPS_MANI_01424] [TPS_MANI_01425] [TPS_MANI_01426] [TPS_MANI_01427] [TPS_MANI_01428] [TPS_MANI_01429] [TPS_MANI_01430]
[RS_MANI_00006]	Support of application deployment	[TPS_MANI_01011] [TPS_MANI_01308] [TPS_MANI_01337] [TPS_MANI_03147]
[RS_MANI_00007]	Configuration of application startup behavior	[TPS_MANI_01012] [TPS_MANI_01013] [TPS_MANI_01017] [TPS_MANI_01041] [TPS_MANI_01046] [TPS_MANI_01061] [TPS_MANI_01188] [TPS_MANI_01209] [TPS_MANI_01277] [TPS_MANI_01278] [TPS_MANI_01328] [TPS_MANI_01334] [TPS_MANI_01412] [TPS_MANI_03151]
[RS_MANI_00008]	Service interface deployment to a transport layer mechanism	[TPS_MANI_01136] [TPS_MANI_01137] [TPS_MANI_01210] [TPS_MANI_03036] [TPS_MANI_03037] [TPS_MANI_03038] [TPS_MANI_03039] [TPS_MANI_03070] [TPS_MANI_03071] [TPS_MANI_03072] [TPS_MANI_03073] [TPS_MANI_03074] [TPS_MANI_03101] [TPS_MANI_03103] [TPS_MANI_03104] [TPS_MANI_03105] [TPS_MANI_03106] [TPS_MANI_03107] [TPS_MANI_03108] [TPS_MANI_03116] [TPS_MANI_03117] [TPS_MANI_03217] [TPS_MANI_03278]
[RS_MANI_00009]	Service instance configuration on the network-level	[TPS_MANI_01316] [TPS_MANI_01317] [TPS_MANI_03001] [TPS_MANI_03002] [TPS_MANI_03003] [TPS_MANI_03004] [TPS_MANI_03005] [TPS_MANI_03006] [TPS_MANI_03007] [TPS_MANI_03008] [TPS_MANI_03009] [TPS_MANI_03010] [TPS_MANI_03022] [TPS_MANI_03023] [TPS_MANI_03024] [TPS_MANI_03049] [TPS_MANI_03061] [TPS_MANI_03236] [TPS_MANI_03237] [TPS_MANI_03292] [TPS_MANI_03293] [TPS_MANI_03554] [TPS_MANI_03555] [TPS_MANI_03619]
[RS_MANI_00011]	Instantiation of provided and required services in an application	[TPS_MANI_01275] [TPS_MANI_01276] [TPS_MANI_01282] [TPS_MANI_03000]

▽

△

Requirement	Description	Satisfied by
[RS_MANI_00014]	User defined transport layer mechanisms	[TPS_MANI_01165] [TPS_MANI_03032] [TPS_MANI_03045] [TPS_MANI_03046] [TPS_MANI_03047] [TPS_MANI_03048] [TPS_MANI_03102] [TPS_MANI_03280] [TPS_MANI_03281]
[RS_MANI_00015]	Definition of the nature of a manifest	[TPS_MANI_01000] [TPS_MANI_01019] [TPS_MANI_01020] [TPS_MANI_01021]
[RS_MANI_00016]	Usage of data types specifically on the AUTOSAR Adaptive Platform	[TPS_MANI_01016] [TPS_MANI_01027] [TPS_MANI_01047] [TPS_MANI_01309]
[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_03277] [TPS_MANI_03289]
[RS_MANI_00018]	Network connections of the machine	[TPS_MANI_03052] [TPS_MANI_03053]
[RS_MANI_00019]	Service discovery message exchange configuration	[TPS_MANI_03064]
[RS_MANI_00020]	Hardware resources of the machine	[TPS_MANI_01269] [TPS_MANI_03035] [TPS_MANI_03148]
[RS_MANI_00021]	Description of machine states	[TPS_MANI_03035]
[RS_MANI_00022]	Adaptive Platform configuration	[TPS_MANI_01208] [TPS_MANI_01273] [TPS_MANI_03035]
[RS_MANI_00023]	Adaptive Module configuration	[TPS_MANI_01208] [TPS_MANI_01226] [TPS_MANI_01227] [TPS_MANI_01279] [TPS_MANI_01343] [TPS_MANI_01365] [TPS_MANI_01366] [TPS_MANI_01367] [TPS_MANI_01370] [TPS_MANI_01374] [TPS_MANI_01375] [TPS_MANI_01376] [TPS_MANI_01377] [TPS_MANI_01379] [TPS_MANI_01383] [TPS_MANI_01384] [TPS_MANI_01387] [TPS_MANI_01388] [TPS_MANI_01389] [TPS_MANI_01390] [TPS_MANI_01391] [TPS_MANI_01392] [TPS_MANI_01399] [TPS_MANI_01400] [TPS_MANI_01401] [TPS_MANI_01402] [TPS_MANI_01403] [TPS_MANI_01404] [TPS_MANI_01836] [TPS_MANI_02384] [TPS_MANI_02385] [TPS_MANI_02386] [TPS_MANI_02387] [TPS_MANI_02388] [TPS_MANI_03035] [TPS_MANI_03056] [TPS_MANI_03096] [TPS_MANI_03098] [TPS_MANI_03162] [TPS_MANI_03163] [TPS_MANI_03164] [TPS_MANI_03165] [TPS_MANI_03166] [TPS_MANI_03167] [TPS_MANI_03218] [TPS_MANI_03219] [TPS_MANI_03220] [TPS_MANI_03221] [TPS_MANI_03222] [TPS_MANI_03226] [TPS_MANI_03260] [TPS_MANI_03261] [TPS_MANI_03262] [TPS_MANI_03263] [TPS_MANI_03264] [TPS_MANI_03265] [TPS_MANI_03266] [TPS_MANI_03267] [TPS_MANI_03268] [TPS_MANI_03269] [TPS_MANI_03270] [TPS_MANI_03271] [TPS_MANI_03272] [TPS_MANI_03273] [TPS_MANI_03274] [TPS_MANI_03276] [TPS_MANI_03279] [TPS_MANI_03282] [TPS_MANI_03283] [TPS_MANI_03285] [TPS_MANI_03286] [TPS_MANI_03294] [TPS_MANI_03323] [TPS_MANI_03324] [TPS_MANI_03325] [TPS_MANI_03502] [TPS_MANI_03503] [TPS_MANI_03505] [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_03544] [TPS_MANI_03545] [TPS_MANI_03546] [TPS_MANI_03553] [TPS_MANI_03573] [TPS_MANI_03574] [TPS_MANI_03575] [TPS_MANI_03576] [TPS_MANI_03625] [TPS_MANI_03626] [TPS_MANI_03633] [TPS_MANI_03651] [TPS_MANI_03670] [TPS_MANI_03672] [TPS_MANI_03673]

▽

△

Requirement	Description	Satisfied by
[RS_MANI_00024]	SOME/IP transport layer mechanisms	[TPS_MANI_01136] [TPS_MANI_01137] [TPS_MANI_03002] [TPS_MANI_03003] [TPS_MANI_03004] [TPS_MANI_03005] [TPS_MANI_03006] [TPS_MANI_03007] [TPS_MANI_03008] [TPS_MANI_03009] [TPS_MANI_03010] [TPS_MANI_03011] [TPS_MANI_03012] [TPS_MANI_03013] [TPS_MANI_03014] [TPS_MANI_03015] [TPS_MANI_03016] [TPS_MANI_03017] [TPS_MANI_03018] [TPS_MANI_03020] [TPS_MANI_03021] [TPS_MANI_03022] [TPS_MANI_03023] [TPS_MANI_03024] [TPS_MANI_03025] [TPS_MANI_03026] [TPS_MANI_03027] [TPS_MANI_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_03116] [TPS_MANI_03154] [TPS_MANI_03155] [TPS_MANI_03156] [TPS_MANI_03157] [TPS_MANI_03158] [TPS_MANI_03159] [TPS_MANI_03168] [TPS_MANI_03217] [TPS_MANI_03227] [TPS_MANI_03230] [TPS_MANI_03231] [TPS_MANI_03237] [TPS_MANI_03278] [TPS_MANI_03554] [TPS_MANI_03555]
[RS_MANI_00025]	Definition and configuration of serialization	[TPS_MANI_01210] [TPS_MANI_03101] [TPS_MANI_03102] [TPS_MANI_03103] [TPS_MANI_03104] [TPS_MANI_03105] [TPS_MANI_03106] [TPS_MANI_03107] [TPS_MANI_03108] [TPS_MANI_03109] [TPS_MANI_03117]
[RS_MANI_00026]	Software Component System Design	[TPS_MANI_01191] [TPS_MANI_01192] [TPS_MANI_01198] [TPS_MANI_01274] [TPS_MANI_01863] [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_01073] [TPS_MANI_01078] [TPS_MANI_01079] [TPS_MANI_01080] [TPS_MANI_01081] [TPS_MANI_01135] [TPS_MANI_01138] [TPS_MANI_01139] [TPS_MANI_01140] [TPS_MANI_01142] [TPS_MANI_01144] [TPS_MANI_01146] [TPS_MANI_01147] [TPS_MANI_01148] [TPS_MANI_01149] [TPS_MANI_01150] [TPS_MANI_01155] [TPS_MANI_01156] [TPS_MANI_01157] [TPS_MANI_01159] [TPS_MANI_01160] [TPS_MANI_01179] [TPS_MANI_01180] [TPS_MANI_01182] [TPS_MANI_01187] [TPS_MANI_01194] [TPS_MANI_01196] [TPS_MANI_01197] [TPS_MANI_01204] [TPS_MANI_01205] [TPS_MANI_01206] [TPS_MANI_01207] [TPS_MANI_01313] [TPS_MANI_01314] [TPS_MANI_01315] [TPS_MANI_01319] [TPS_MANI_01320] [TPS_MANI_01321] [TPS_MANI_01322] [TPS_MANI_01323] [TPS_MANI_01371] [TPS_MANI_01378] [TPS_MANI_01380]
[RS_MANI_00028]	Configuration of Safety protection	[TPS_MANI_01324] [TPS_MANI_01325] [TPS_MANI_01327] [TPS_MANI_03127] [TPS_MANI_03128] [TPS_MANI_03129] [TPS_MANI_03130] [TPS_MANI_03131] [TPS_MANI_03132] [TPS_MANI_03228] [TPS_MANI_03229] [TPS_MANI_03252]
[RS_MANI_00029]	Mapping description between Signal-based communication and Service-Oriented communication	[TPS_MANI_03124] [TPS_MANI_03627] [TPS_MANI_03629] [TPS_MANI_03635] [TPS_MANI_03665]
[RS_MANI_00030]	Definition of optional elements in composite data structures	[TPS_MANI_01097] [TPS_MANI_01184] [TPS_MANI_01185] [TPS_MANI_01186] [TPS_MANI_01270] [TPS_MANI_01333]
[RS_MANI_00031]	Interaction with Crypto Software	[TPS_MANI_03253] [TPS_MANI_03254] [TPS_MANI_03255] [TPS_MANI_03256] [TPS_MANI_03257] [TPS_MANI_03258] [TPS_MANI_03259]

▽

△

Requirement	Description	Satisfied by
[RS_MANI_00032]	Support for platform health management	[TPS_MANI_01280] [TPS_MANI_03500] [TPS_MANI_03502] [TPS_MANI_03503] [TPS_MANI_03505] [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_03534] [TPS_MANI_03544] [TPS_MANI_03545] [TPS_MANI_03546] [TPS_MANI_03553] [TPS_MANI_03573] [TPS_MANI_03574] [TPS_MANI_03575] [TPS_MANI_03576] [TPS_MANI_03624] [TPS_MANI_03625] [TPS_MANI_03626] [TPS_MANI_03630] [TPS_MANI_03631] [TPS_MANI_03633] [TPS_MANI_03651] [TPS_MANI_03670] [TPS_MANI_03672] [TPS_MANI_03673]
[RS_MANI_00034]	Specification of intents	[TPS_MANI_01106] [TPS_MANI_01107] [TPS_MANI_01108] [TPS_MANI_03209]
[RS_MANI_00035]	Definition of an uploadable software package	[TPS_MANI_01109] [TPS_MANI_01110] [TPS_MANI_01112] [TPS_MANI_01114] [TPS_MANI_01115] [TPS_MANI_01116] [TPS_MANI_01117] [TPS_MANI_01118] [TPS_MANI_01119] [TPS_MANI_01164] [TPS_MANI_01189] [TPS_MANI_01202] [TPS_MANI_01211] [TPS_MANI_01213] [TPS_MANI_01214] [TPS_MANI_01215] [TPS_MANI_01216] [TPS_MANI_01217] [TPS_MANI_01218] [TPS_MANI_01219] [TPS_MANI_01220] [TPS_MANI_01221] [TPS_MANI_01222] [TPS_MANI_01223] [TPS_MANI_01225] [TPS_MANI_01290] [TPS_MANI_01291] [TPS_MANI_01292] [TPS_MANI_01294] [TPS_MANI_01295] [TPS_MANI_01296] [TPS_MANI_01297] [TPS_MANI_01298] [TPS_MANI_01299] [TPS_MANI_01300] [TPS_MANI_01301] [TPS_MANI_01302] [TPS_MANI_01306] [TPS_MANI_01329] [TPS_MANI_01331] [TPS_MANI_01344] [TPS_MANI_01345] [TPS_MANI_01346] [TPS_MANI_01349] [TPS_MANI_01369] [TPS_MANI_01410] [TPS_MANI_01413] [TPS_MANI_01414] [TPS_MANI_01434]
[RS_MANI_00036]	Configuration of security protection	[TPS_MANI_03133] [TPS_MANI_03134] [TPS_MANI_03137] [TPS_MANI_03138] [TPS_MANI_03139] [TPS_MANI_03140] [TPS_MANI_03199] [TPS_MANI_03200] [TPS_MANI_03203] [TPS_MANI_03204] [TPS_MANI_03205] [TPS_MANI_03206] [TPS_MANI_03207] [TPS_MANI_03208] [TPS_MANI_03213] [TPS_MANI_03214] [TPS_MANI_03215] [TPS_MANI_03216] [TPS_MANI_03232] [TPS_MANI_03233] [TPS_MANI_03234] [TPS_MANI_03240] [TPS_MANI_03241] [TPS_MANI_03242] [TPS_MANI_03317] [TPS_MANI_03318] [TPS_MANI_03319] [TPS_MANI_03320] [TPS_MANI_03661] [TPS_MANI_03664]
[RS_MANI_00037]	Configuration of logging and tracing	[TPS_MANI_03284]
[RS_MANI_00038]	DDS transport layer mechanisms	[TPS_MANI_03525] [TPS_MANI_03526] [TPS_MANI_03527] [TPS_MANI_03528] [TPS_MANI_03529] [TPS_MANI_03530] [TPS_MANI_03531] [TPS_MANI_03532] [TPS_MANI_03533] [TPS_MANI_03556] [TPS_MANI_03557] [TPS_MANI_03558] [TPS_MANI_03561] [TPS_MANI_03562] [TPS_MANI_03567] [TPS_MANI_03568] [TPS_MANI_03622] [TPS_MANI_03650] [TPS_MANI_03662]

▽

△

Requirement	Description	Satisfied by
[RS_MANI_00039]	Usage of implementation specific data types	[TPS_MANI_01166] [TPS_MANI_01167] [TPS_MANI_01168] [TPS_MANI_01169] [TPS_MANI_01171] [TPS_MANI_01172] [TPS_MANI_01173] [TPS_MANI_01174] [TPS_MANI_01175] [TPS_MANI_01176] [TPS_MANI_01177] [TPS_MANI_01201] [TPS_MANI_01212] [TPS_MANI_01393] [TPS_MANI_01411] [TPS_MANI_03169] [TPS_MANI_03170] [TPS_MANI_03171] [TPS_MANI_03172] [TPS_MANI_03173] [TPS_MANI_03174] [TPS_MANI_03175] [TPS_MANI_03176] [TPS_MANI_03177] [TPS_MANI_03178] [TPS_MANI_03179] [TPS_MANI_03180] [TPS_MANI_03181] [TPS_MANI_03183] [TPS_MANI_03184] [TPS_MANI_03185] [TPS_MANI_03186] [TPS_MANI_03187] [TPS_MANI_03188] [TPS_MANI_03189] [TPS_MANI_03190] [TPS_MANI_03191] [TPS_MANI_03192] [TPS_MANI_03193] [TPS_MANI_03196] [TPS_MANI_03197] [TPS_MANI_03198] [TPS_MANI_03201] [TPS_MANI_03202]
[RS_MANI_00040]	Support for access to synchronized time	[TPS_MANI_03535] [TPS_MANI_03536] [TPS_MANI_03537] [TPS_MANI_03539] [TPS_MANI_03541] [TPS_MANI_03542] [TPS_MANI_03543] [TPS_MANI_03547] [TPS_MANI_03548] [TPS_MANI_03549] [TPS_MANI_03551] [TPS_MANI_03632] [TPS_MANI_03671]
[RS_MANI_00041]	Configuration of function groups	[TPS_MANI_03145] [TPS_MANI_03152] [TPS_MANI_03194] [TPS_MANI_03195]
[RS_MANI_00060]	Support of Identity and Access Management	[TPS_MANI_01231] [TPS_MANI_01232] [TPS_MANI_01234] [TPS_MANI_01235] [TPS_MANI_01236] [TPS_MANI_01237] [TPS_MANI_01238] [TPS_MANI_01239] [TPS_MANI_01240] [TPS_MANI_01284] [TPS_MANI_01307] [TPS_MANI_03238] [TPS_MANI_03239] [TPS_MANI_03240] [TPS_MANI_03241] [TPS_MANI_03242] [TPS_MANI_03244] [TPS_MANI_03245] [TPS_MANI_03246] [TPS_MANI_03247] [TPS_MANI_03248] [TPS_MANI_03249] [TPS_MANI_03250] [TPS_MANI_03251] [TPS_MANI_03290] [TPS_MANI_03322]
[RS_MANI_00061]	Support of Diagnostic Interfaces	[TPS_MANI_01048] [TPS_MANI_01049] [TPS_MANI_01050] [TPS_MANI_01242] [TPS_MANI_01243] [TPS_MANI_01244] [TPS_MANI_01245] [TPS_MANI_01246] [TPS_MANI_01247] [TPS_MANI_01248] [TPS_MANI_01249] [TPS_MANI_01250] [TPS_MANI_01251] [TPS_MANI_01252] [TPS_MANI_01253] [TPS_MANI_01254] [TPS_MANI_01255] [TPS_MANI_01259] [TPS_MANI_01260] [TPS_MANI_01261] [TPS_MANI_01262] [TPS_MANI_01263] [TPS_MANI_01265] [TPS_MANI_01326] [TPS_MANI_01332] [TPS_MANI_01347] [TPS_MANI_01348] [TPS_MANI_01351] [TPS_MANI_01352] [TPS_MANI_01353] [TPS_MANI_01359] [TPS_MANI_01360] [TPS_MANI_01361] [TPS_MANI_01362] [TPS_MANI_01363] [TPS_MANI_01373] [TPS_MANI_01394] [TPS_MANI_01395] [TPS_MANI_01416] [TPS_MANI_01417] [TPS_MANI_01418] [TPS_MANI_01420] [TPS_MANI_01421] [TPS_MANI_01422] [TPS_MANI_01423] [TPS_MANI_01424] [TPS_MANI_01425] [TPS_MANI_01426] [TPS_MANI_01427] [TPS_MANI_01428] [TPS_MANI_01429] [TPS_MANI_01430]
[RS_MANI_00062]	Support for Partial Networking	[TPS_MANI_03224] [TPS_MANI_03225]

▽

△

Requirement	Description	Satisfied by
[RS_MANI_00063]	The Manifest specification shall support the translation between signal-based and service-oriented communication	[TPS_MANI_03287] [TPS_MANI_03577] [TPS_MANI_03578] [TPS_MANI_03579] [TPS_MANI_03580] [TPS_MANI_03581] [TPS_MANI_03582] [TPS_MANI_03583] [TPS_MANI_03585] [TPS_MANI_03589] [TPS_MANI_03590] [TPS_MANI_03591] [TPS_MANI_03592] [TPS_MANI_03593] [TPS_MANI_03594] [TPS_MANI_03595] [TPS_MANI_03597] [TPS_MANI_03598] [TPS_MANI_03599] [TPS_MANI_03600] [TPS_MANI_03601] [TPS_MANI_03602] [TPS_MANI_03603] [TPS_MANI_03604] [TPS_MANI_03605] [TPS_MANI_03606] [TPS_MANI_03607] [TPS_MANI_03608] [TPS_MANI_03609] [TPS_MANI_03610] [TPS_MANI_03611] [TPS_MANI_03612] [TPS_MANI_03614] [TPS_MANI_03615] [TPS_MANI_03620] [TPS_MANI_03621] [TPS_MANI_03636] [TPS_MANI_03637] [TPS_MANI_03638] [TPS_MANI_03639] [TPS_MANI_03640] [TPS_MANI_03641] [TPS_MANI_03642] [TPS_MANI_03643] [TPS_MANI_03644] [TPS_MANI_03645] [TPS_MANI_03646] [TPS_MANI_03647] [TPS_MANI_03648] [TPS_MANI_03649] [TPS_MANI_03652] [TPS_MANI_03653] [TPS_MANI_03654] [TPS_MANI_03655] [TPS_MANI_03656] [TPS_MANI_03657] [TPS_MANI_03658] [TPS_MANI_03659] [TPS_MANI_03660] [TPS_MANI_03663] [TPS_MANI_03666] [TPS_MANI_03667]
[RS_MANI_00064]	Service contract version for a service interface	[TPS_MANI_03616]
[RS_MANI_00065]	Service contract versioning for all Transport Deployment Protocols	[TPS_MANI_03617]
[RS_MANI_00066]	Service Versioning Blocklist	[TPS_MANI_03618]
[RS_MANI_00067]	Raw data stream deployment	[TPS_MANI_01285] [TPS_MANI_01287] [TPS_MANI_01307] [TPS_MANI_01354] [TPS_MANI_01355] [TPS_MANI_01356] [TPS_MANI_01357]
[RS_MANI_00068]	Support for security event reporting interface definitions	[TPS_MANI_01338] [TPS_MANI_01339] [TPS_MANI_01340]
[RS_MANI_00069]	Support for deployment definition of the Intrusion Detection System Manager	[TPS_MANI_01341] [TPS_MANI_01342] [TPS_MANI_01440] [TPS_MANI_01441]
[RS_MANI_00070]	Support for SOVD	[TPS_MANI_01394] [TPS_MANI_01395] [TPS_MANI_01396] [TPS_MANI_01397] [TPS_MANI_01398] [TPS_MANI_01399] [TPS_MANI_01400] [TPS_MANI_01401] [TPS_MANI_01402] [TPS_MANI_01403] [TPS_MANI_01404] [TPS_MANI_01405] [TPS_MANI_01420] [TPS_MANI_01421] [TPS_MANI_01422] [TPS_MANI_01423]

Table A.3: RequirementsTracing

B 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.

B.1 Data Type

B.1.1 Definition of Map Data Type

Figure B.1 contains a graphical representation of an example model for ApplicationAssocMapDataType.

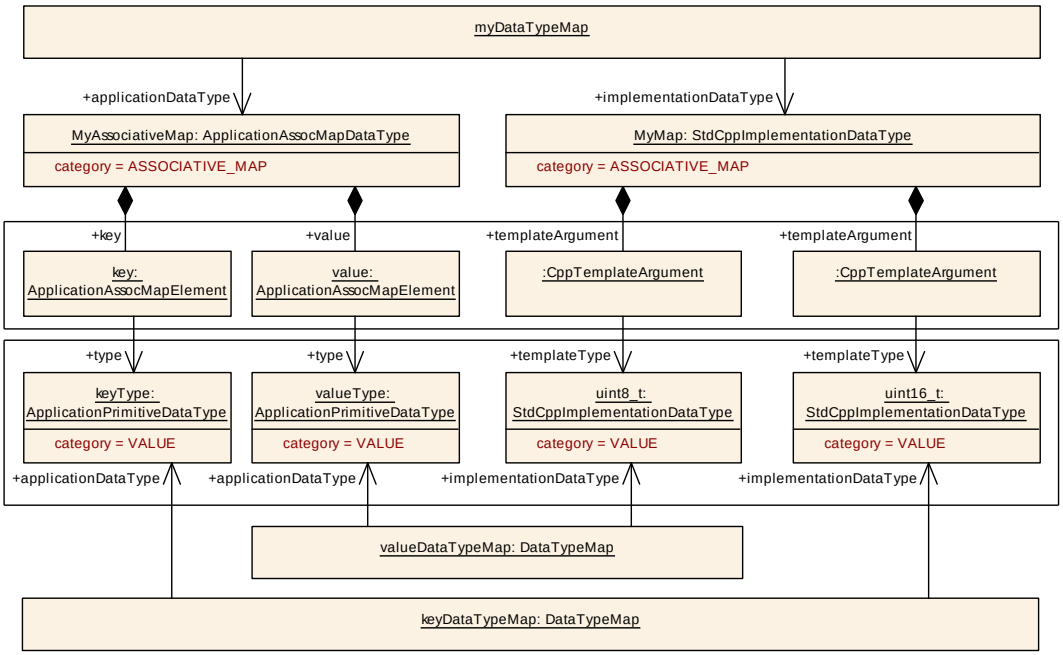


Figure B.1: Example of the model of an associative map

Listing B.1 provides the corresponding ARXML serialization of the example model of an ApplicationAssocMapDataType depicted in Figure B.1.

```
<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>
```

Listing B.1: Example for the definition of an ApplicationAssocMapDataType

B.1.2 Definition of String Data Type

The example depicted in Figure B.2 contains the definition of both an ApplicationDataType as well as the definition of the corresponding CppImplementationDataType.

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

Another aspect of the example in Figure B.2 is that it defines the intended encoding of the modeled data type in the scope of the ApplicationPrimitiveDataType.

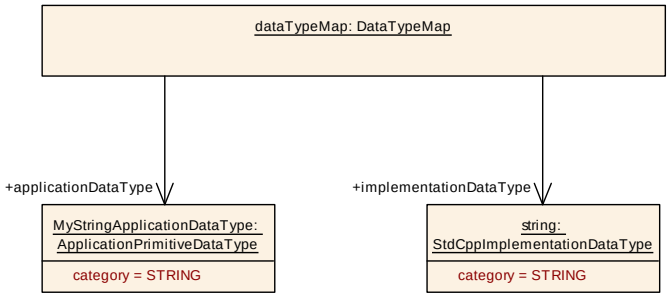


Figure B.2: Example of the model of a string with UTF-8 encoding

B.1.3 Definition of String Data Type

Figure B.3 shows an example of an one-dimensional array of uint16 elements with arraySize = 5.

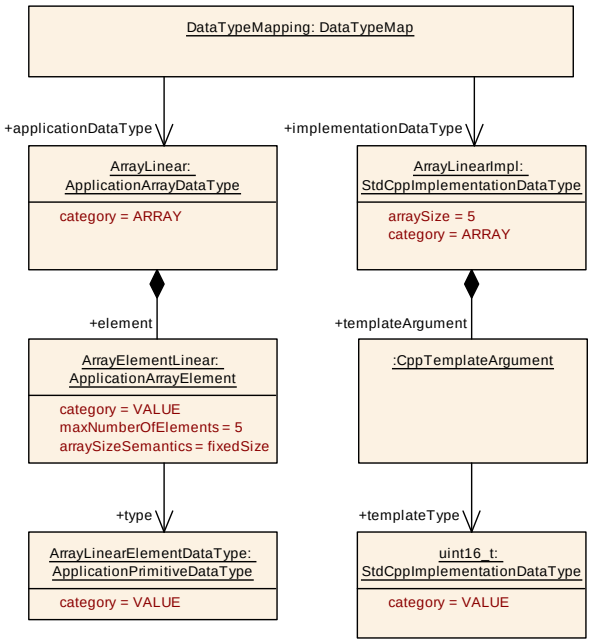


Figure B.3: Example of the model of a one-dimensional array

Figure B.4 shows an example of a multidimensional array where a `CppImplementationDataType` of `category ARRAY` with `arraySize` = 5 has a `templateArgument` that points to the inner `CppImplementationDataType` of `category ARRAY` in the role `templateType`.

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

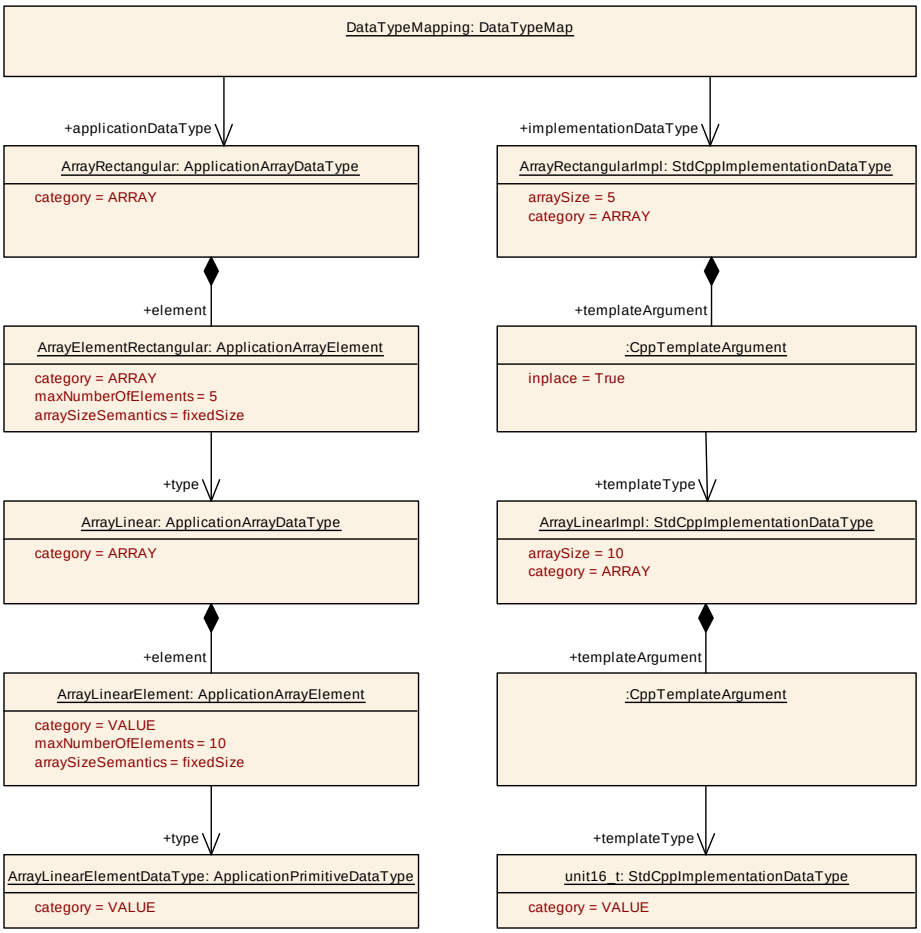


Figure B.4: Example of the model of a multidimensional array

B.1.4 Definition of Vector Data Type

Figure B.5 shows an example of an one-dimensional vector of uint16 elements.

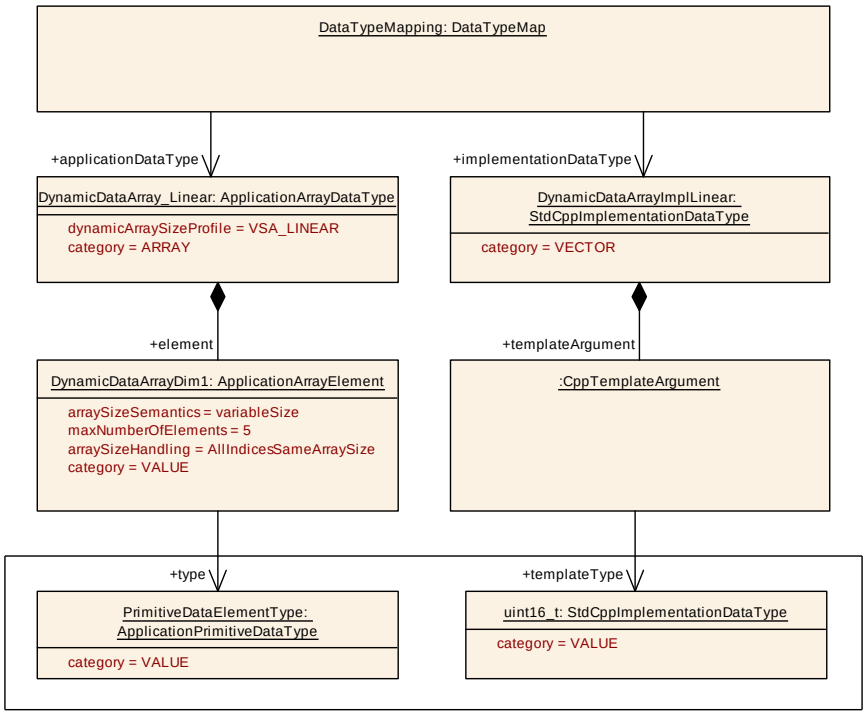


Figure B.5: Example of the model of a one-dimensional vector

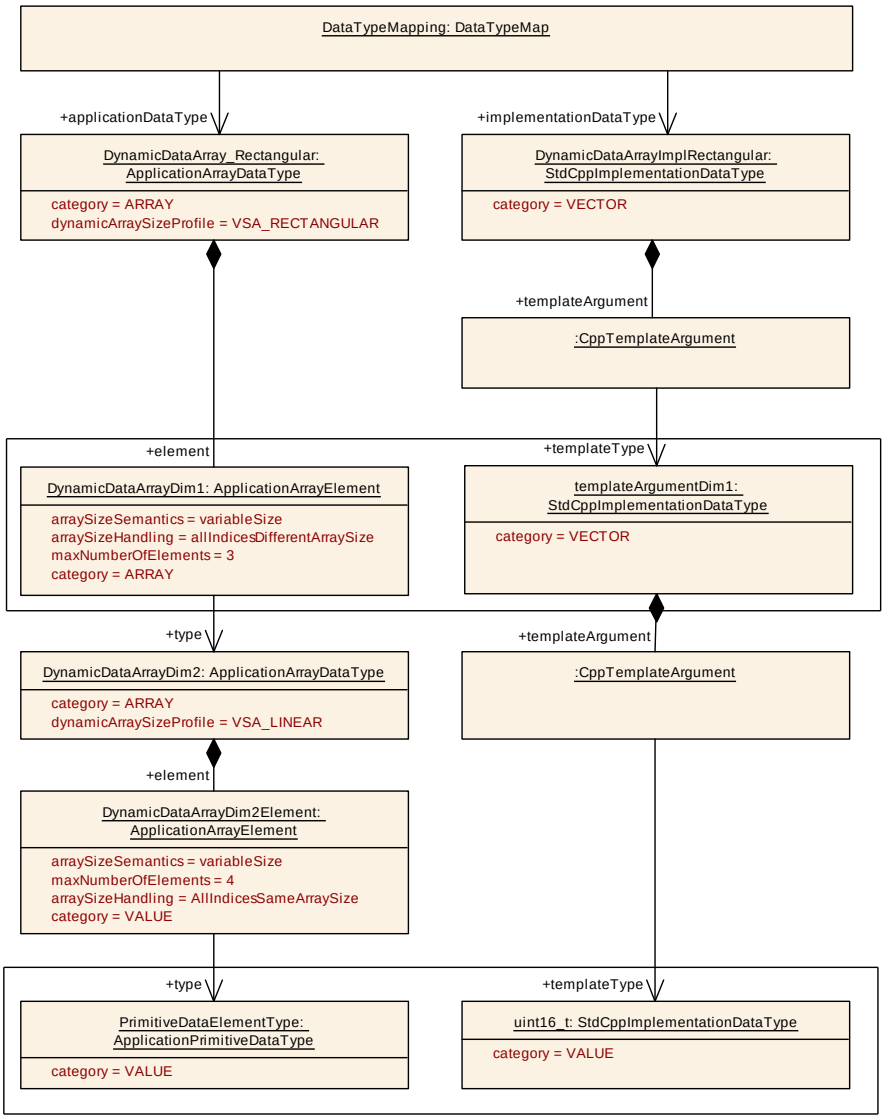


Figure B.6: Example of the model of a multidimensional vector

Figure B.6 shows an example of a multidimensional vector where a `CppImplementationDataType` of `category VECTOR` has a `templateArgument` that points to the inner `CppImplementationDataType` of `category VECTOR` in the role `templateType`. The inner `CppImplementationDataType` has a `templateArgument` that finally points with the `templateType` reference to a primitive type.

B.1.5 Definition of Struct Data Type

The example depicted in Figure B.7 shows the definition of a Structure, called `MyStruct`, that has two members. The `typeReference` of the `subElements` with the `shortName` `ArrayElement` is classified with `inplace` = **`true`**.

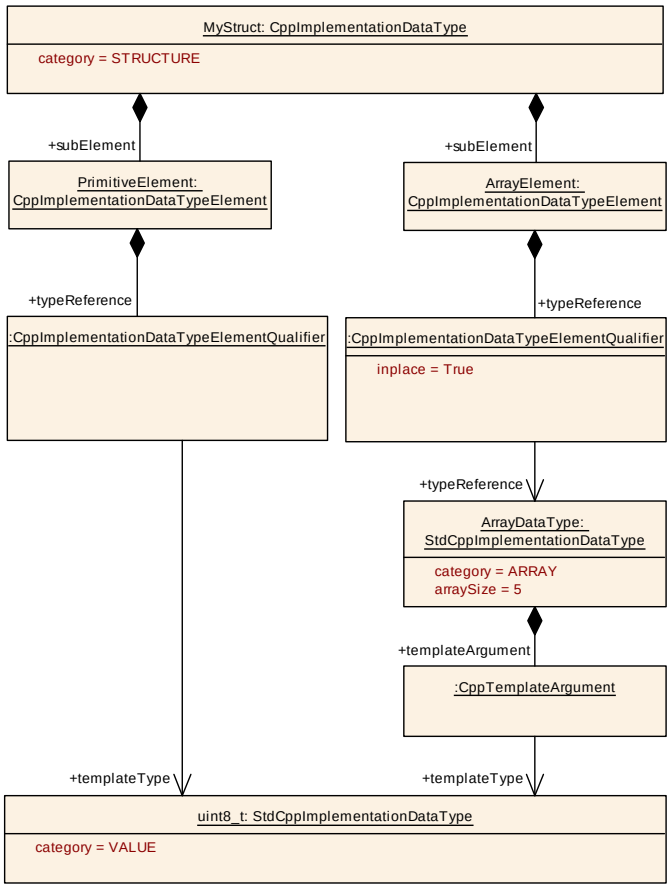


Figure B.7: Example of the model of a Struct

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

B.1.6 Definition of Enumeration Data Type

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

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

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

Listing B.2: example for enumeration

B.1.7 Definition of Map Data Type

The example depicted in Figure B.8 shows the definition of a ASSOCIATIVE_MAP that has two CppTemplateArguments, one for the key and one for the value.

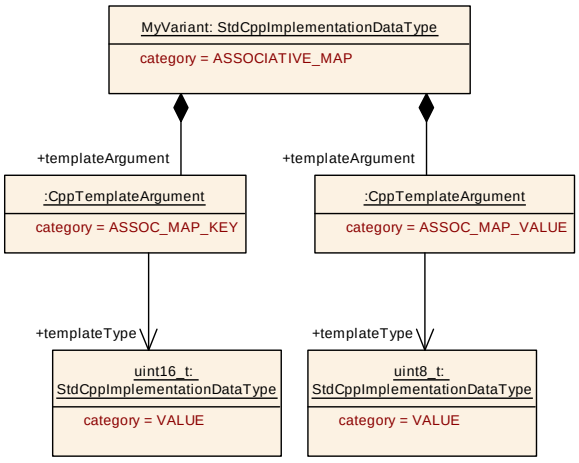


Figure B.8: Example of the model of an ASSOCIATIVE_MAP

B.1.8 Definition of Variant Data Type

The example depicted in Figure B.9 shows the definition of a VARIANT that has two CppTemplateArguments. Each one represents one alternative type. Please note

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

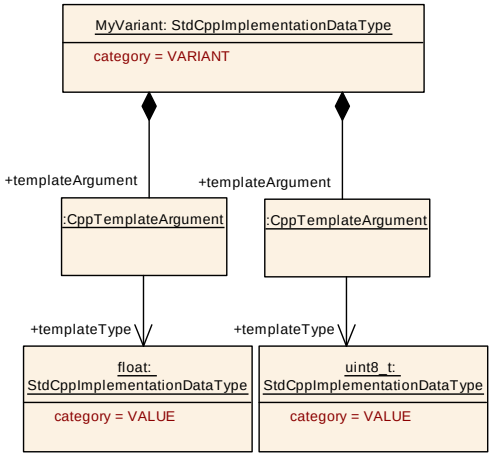


Figure B.9: Example of the model of an VARIANT

B.1.9 Scenarios to define a Vector

This section contains a non-comprehensive list of possible scenarios for the definition of a CppImplementationDataType of category VECTOR.

Please note that the general information contained in this chapter does not exclusively apply on to the vector data type. The latter has been picked as an arbitrary example for the visualization of the effect of configuration settings on the language binding.

Consequently, there is no further discussion of this topic with respect to a different kind of container data type.

Please note that for these example scenarios the namespace of a CustomCppImplementationDataType is assumed to be set to x::y and the shortName is assumed to be set to CustVec.

The shortName of a StdCppImplementationDataType is assumed to be set to MyVec.

If a custom Allocator is used in a scenario the value of Allocator.shortName shall be assumed to have the value CustAlloc.

Scenario	array size	custom allocator	custom type	Resulting C++ Code
I	No	No	No	`using MyVec = ara::core::Vector<std::uint8_t>`
II	Yes	No	No	`using MyVec = ara::core::Vector<std::uint8_t> //generator warning`
III	Yes	Yes	No	`using MyVec = ara::core::Vector<std::uint8_t, CustAlloc<std::uint8_t, MaxSize>>`
IV	No	Yes	No	`using MyVec = ara::core::Vector<std::uint8_t,CustAlloc<std::uint8_t>>`
V	Yes	Yes	Yes	`x::y::CustVec<ara::core::uint8_t, CustAlloc<std::uint8_t, MaxSize>>`
VI	Yes	No	Yes	`x::y::CustVec<std::uint8_t>`
VII	No	Yes	Yes	`x::y::CustVec<std::uint8_t, CustAlloc<std::uint8_t>>`
VIII	No	No	Yes	`x::y::CustVec<std::uint8_t>`

Table B.1: Example definitions of a `CppImplementationDataType` of `category VECTOR`

B.2 Service-oriented Communication

B.2.1 Radar and Camera ServiceInterface example

The example in figure B.10 shows a *Radar* `ServiceInterface` with a *BrakeEvent* and two `method`s: *Calibrate* and *Adjust*. The *Camera* `ServiceInterface` shown in figure B.11 has two events: *LaneEvent* and *SpeedLimitEvent* and one *Calibrate* `method`.

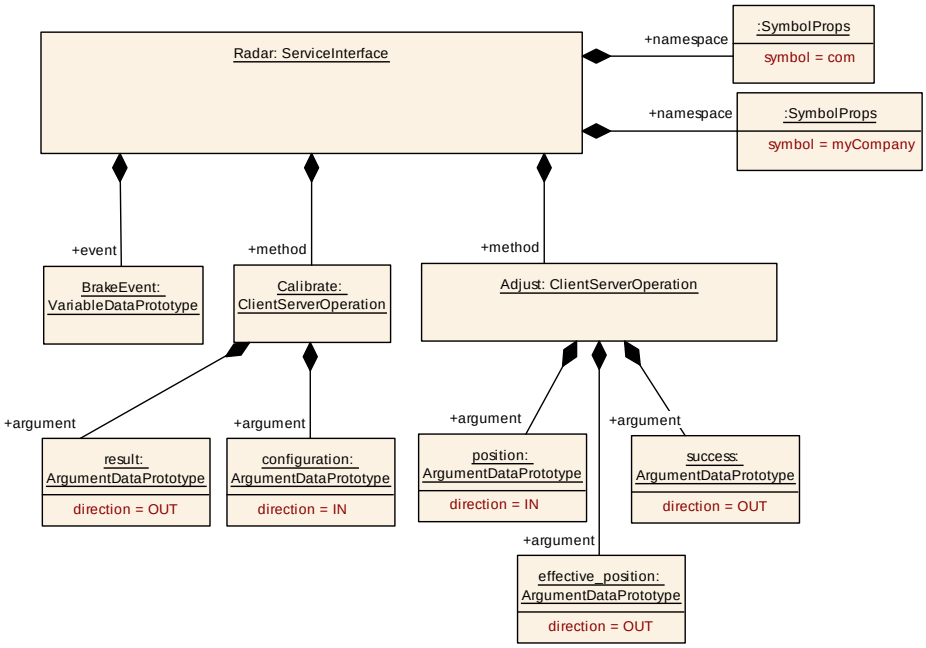


Figure B.10: Radar Service Interface

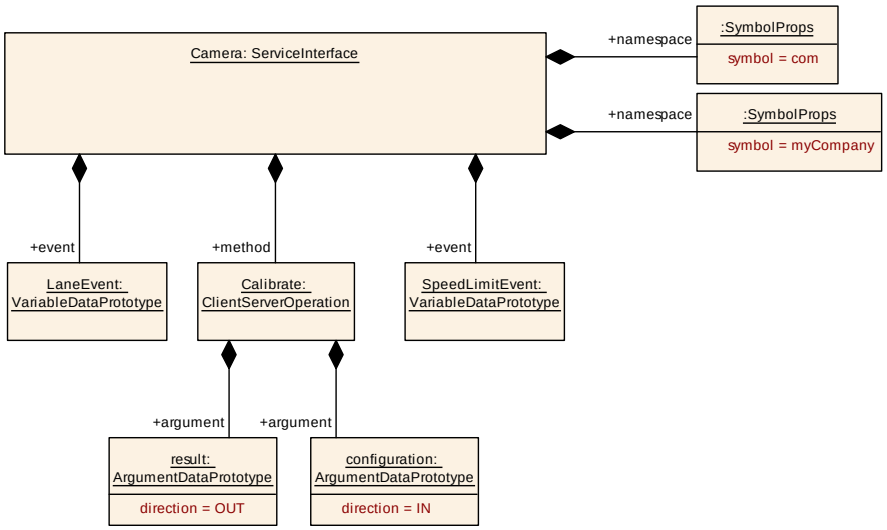


Figure B.11: Camera Service Interface

Both `ServiceInterface`s *Radar* and *Camera* are mapped to a combined *RadarAndCamera* `ServiceInterface` with an Service Interface Element Mapping since both `ServiceInterface`s have a `method` with the same name: *Calibrate*.

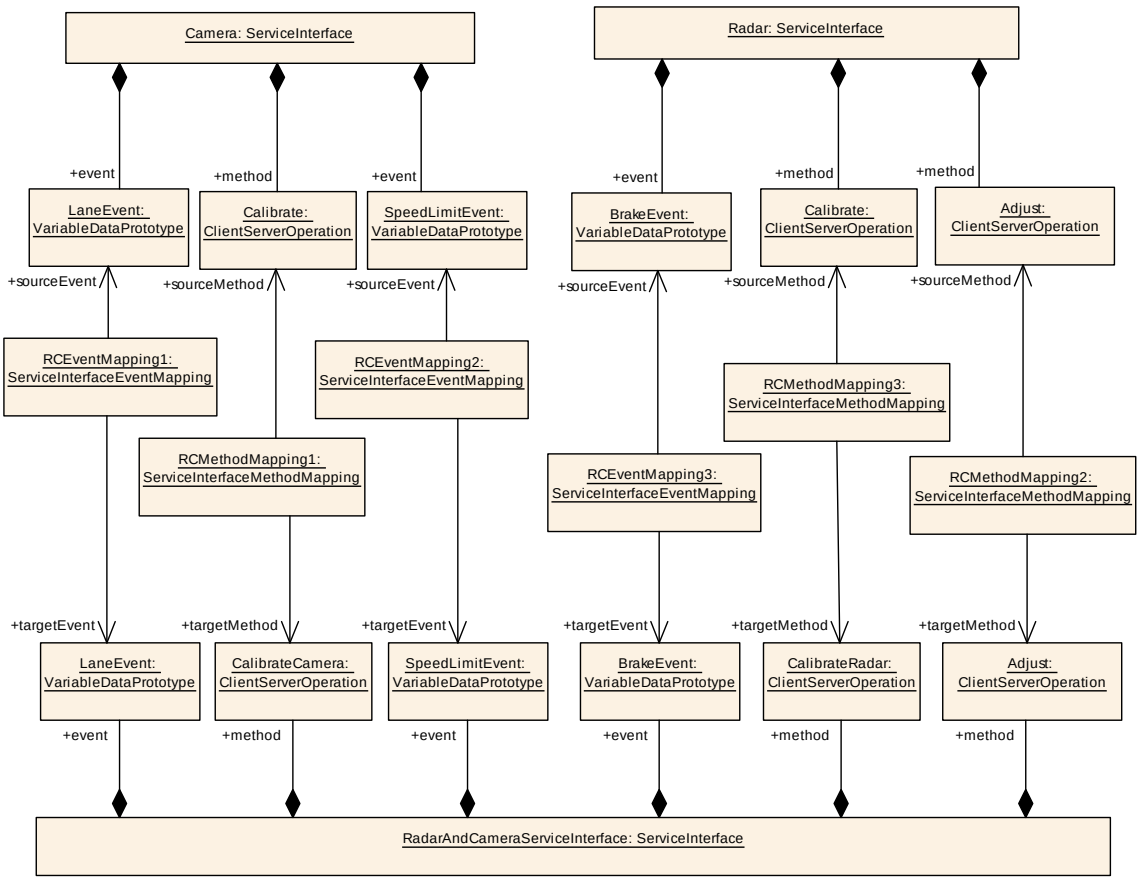


Figure B.12: Service Interface Element Mapping example

The combined ServiceInterface is offered over the network as a SOME/IP Service. Figure B.13 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.

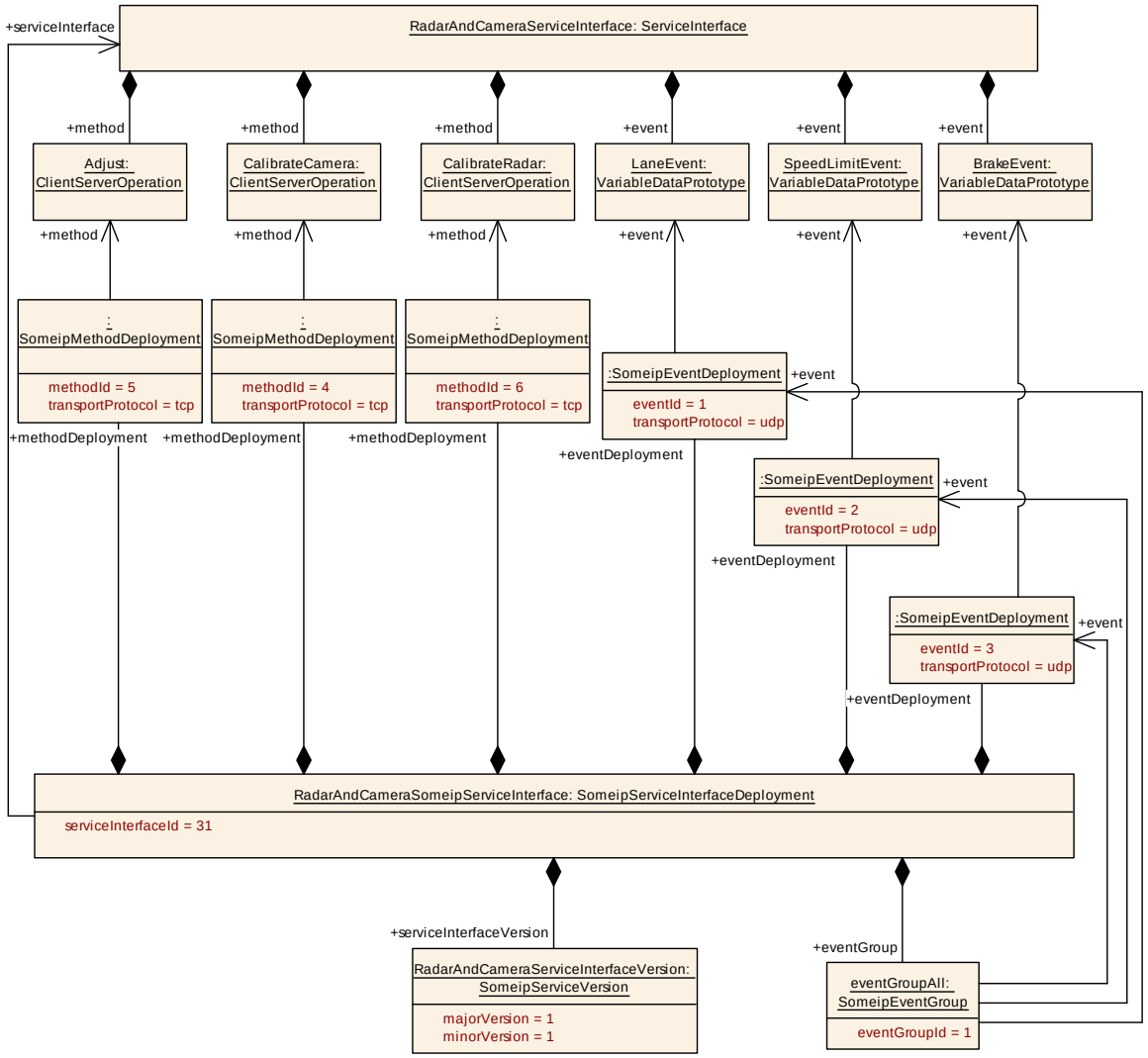


Figure B.13: SOME/IP Deployment

Figure B.14 shows a modeled ProvidedSomeipServiceInstance that is mapped to a Machine.

The displayed configuration in figure B.14 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), PortNumber (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 B.15.

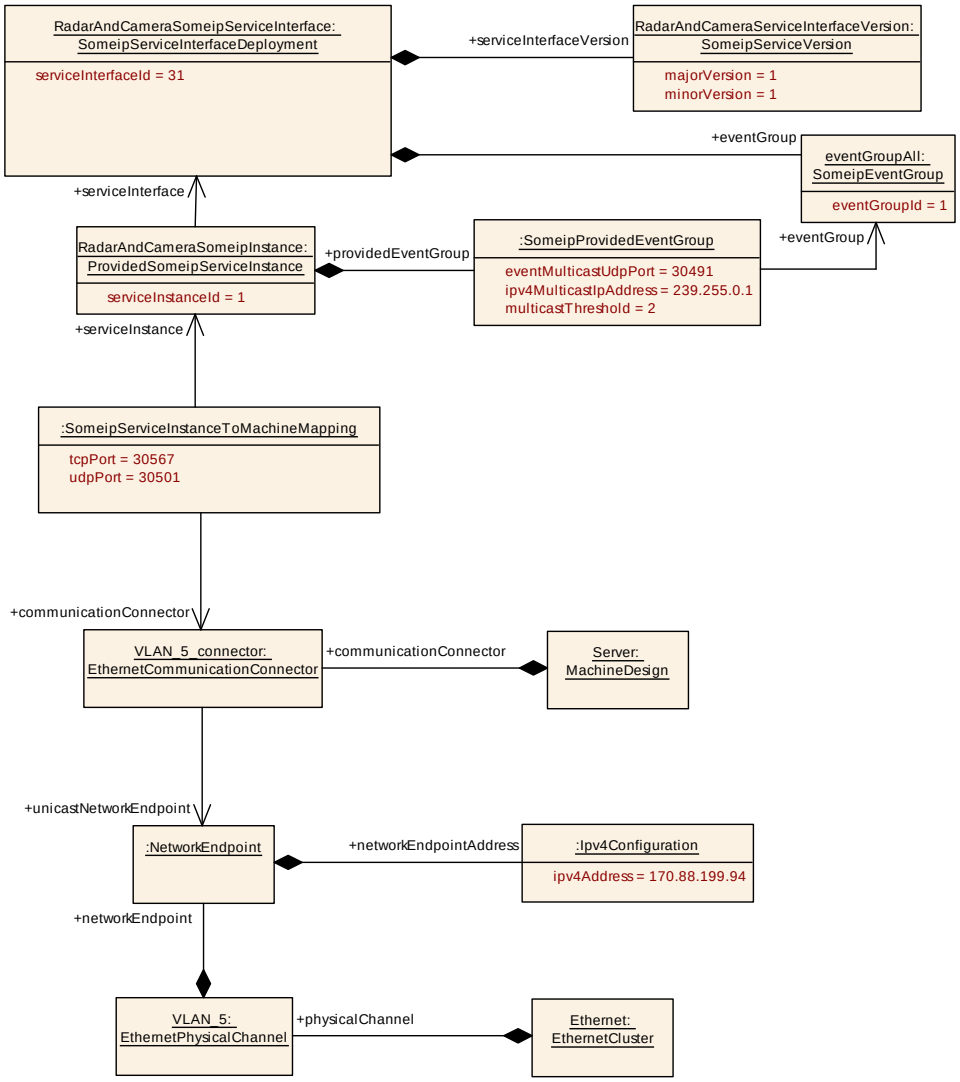


Figure B.14: SOME/IP Provided Service Instance

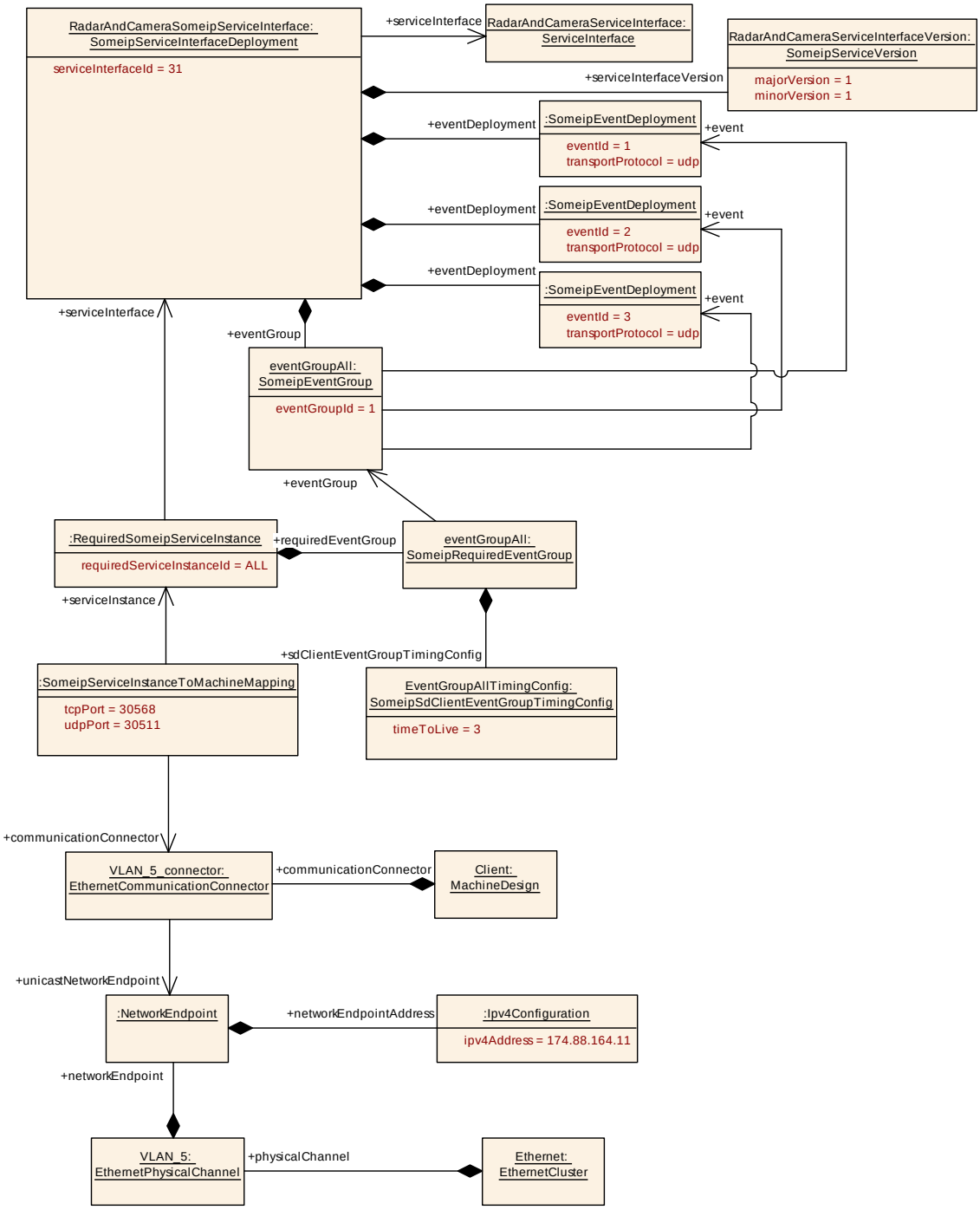


Figure B.15: SOME/IP Required Service Instance

The displayed configuration in figure B.15 leads to a SOME/IP Find Service Message with the following content:

- ServiceId => `serviceInterfaceId` = 31
- InstanceId => `RequiredSomeipServiceInstance.requiredServiceInstanceId` = ALL
- MajorVersion => `majorVersion` = 1

- MinorVersion => `minorVersion` = 1

The displayed configuration in figure B.14 also leads to a SOME/IP SubscribeEventGroup Message content that is sent from the Service Requester to the Service Provider:

- ServiceId => taken from the OfferMessage
- InstanceId => taken from the OfferMessage
- MajorVersion => taken from the OfferMessage
- MinorVersion => taken from the OfferMessage
- Eventgroup ID => `RequiredSomeipServiceInstance.requiredEventGroup.eventGroup.eventGroupId` = 1
- TTL => `RequiredSomeipServiceInstance.requiredEventGroup.sdClientEventGroupTimingConfig.timeToLive` = 3
- IPv4 Endpoint Option with IPv4 Address (170.88.164.11), Protocol (UDP), PortNumber (30511)

B.2.2 Service Interface Mapping

B.2.2.1 Service Interface Element Mapping

Figure B.16 provides an example of how a mapping on element basis looks like. Note that, in this example, both `ServiceInterface1` and `ServiceInterface2` aggregate a `field` with the `shortName` `field1`.

This configuration 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" `ServiceInterface`s quite nicely:

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

The formal modeling of the individual mappings is described in section 3.3.3.

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

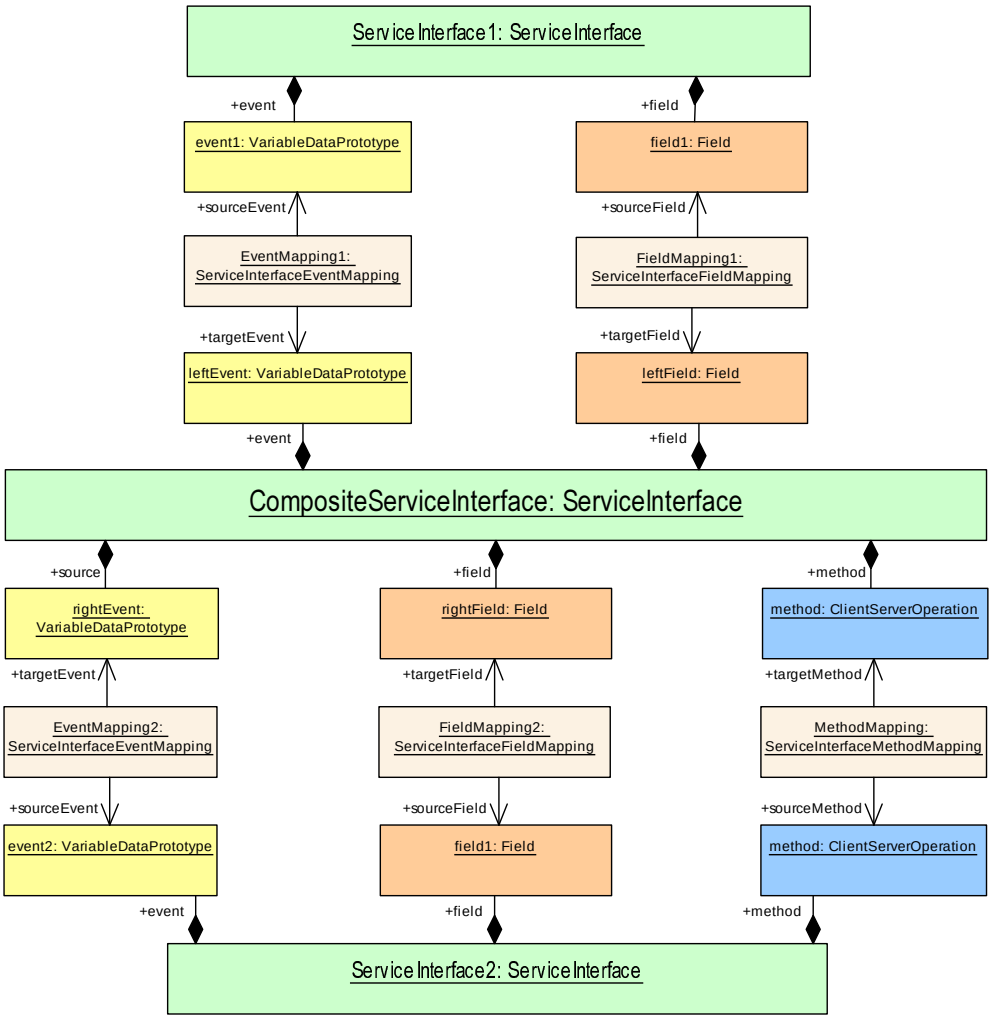


Figure B.16: Example for a mapping of elements of `ServiceInterface`

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

B.2.2.2 Service Interface Event Mapping

It is 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 B.17.

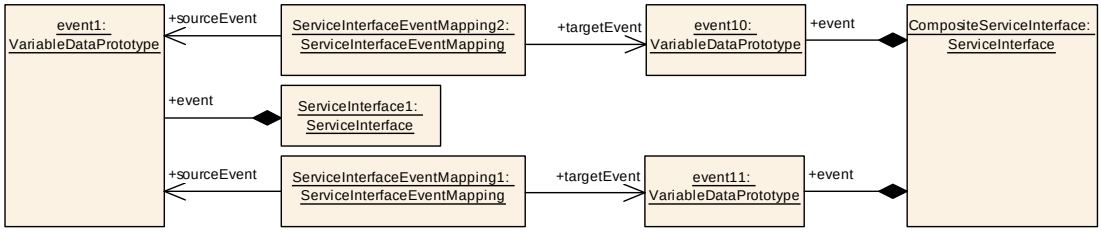


Figure B.17: Example for the application of a `ServiceInterfaceEventMapping`

B.2.2.3 Service Interface Field Mapping

It is 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 B.18.

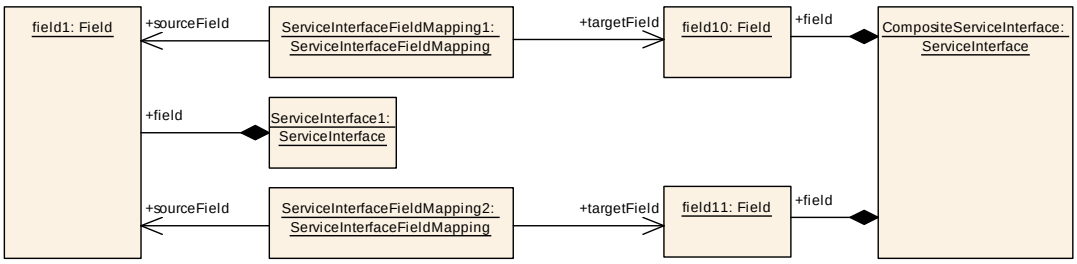


Figure B.18: Example for the application of a `ServiceInterfaceFieldMapping`

B.2.2.4 Service Interface Method Mapping

It is 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 B.19.

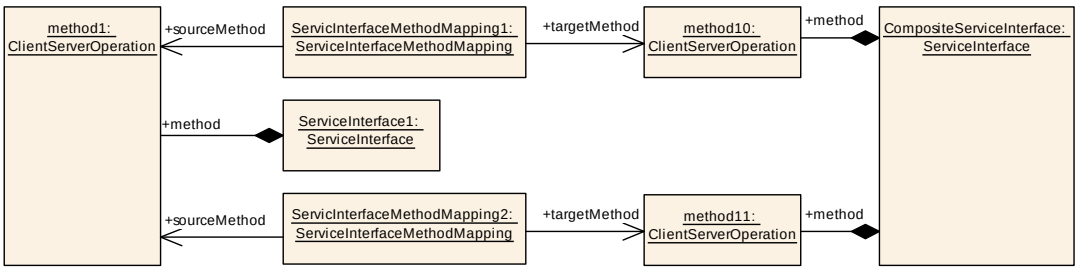


Figure B.19: Example for the application of a `ServiceInterfaceMethodMapping`

B.2.3 Service Instance Deployment

B.2.3.1 Service Instance Deployment by Service Interface Mapping

The example in Figure B.21 sketches the modeling of a `ProvidedSomeipService-Instance` in the presence of a `ServiceInterfaceMapping`, that references two `ServiceInterfaces` in the role `sourceServiceInterface`.

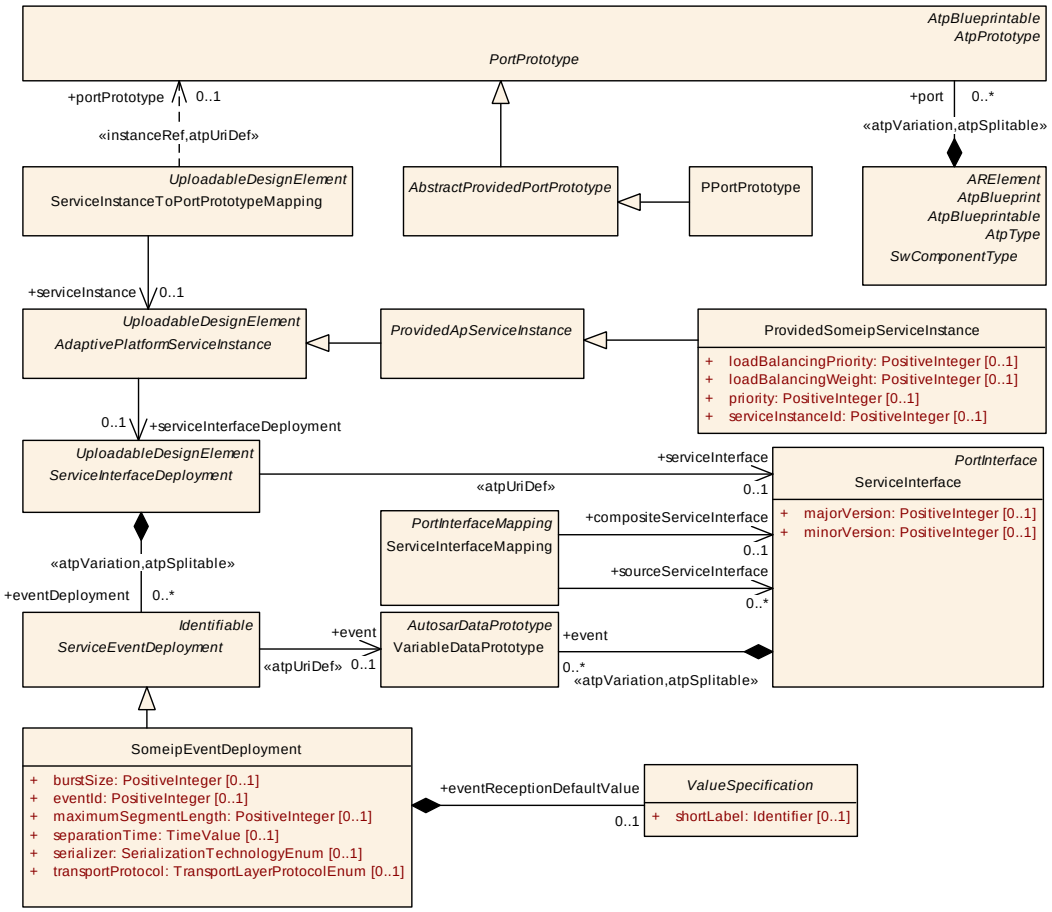


Figure B.20: Meta-model excerpt relevant for the example

For support, Figure B.20 contains an excerpt from the meta-model that contains the relevant meta-classes that have been instantiated to create the example sketched in Figure B.21.

Note further that the example depicted in Figure B.21 is not limited to the explanation of the actual `ServiceInterfaceMapping`.

As the main use case for this is the usage of `ServiceInterface`s for the definition of an “outside” communication binding the example also contains the modeling of such a binding, in this case to SOME/IP.

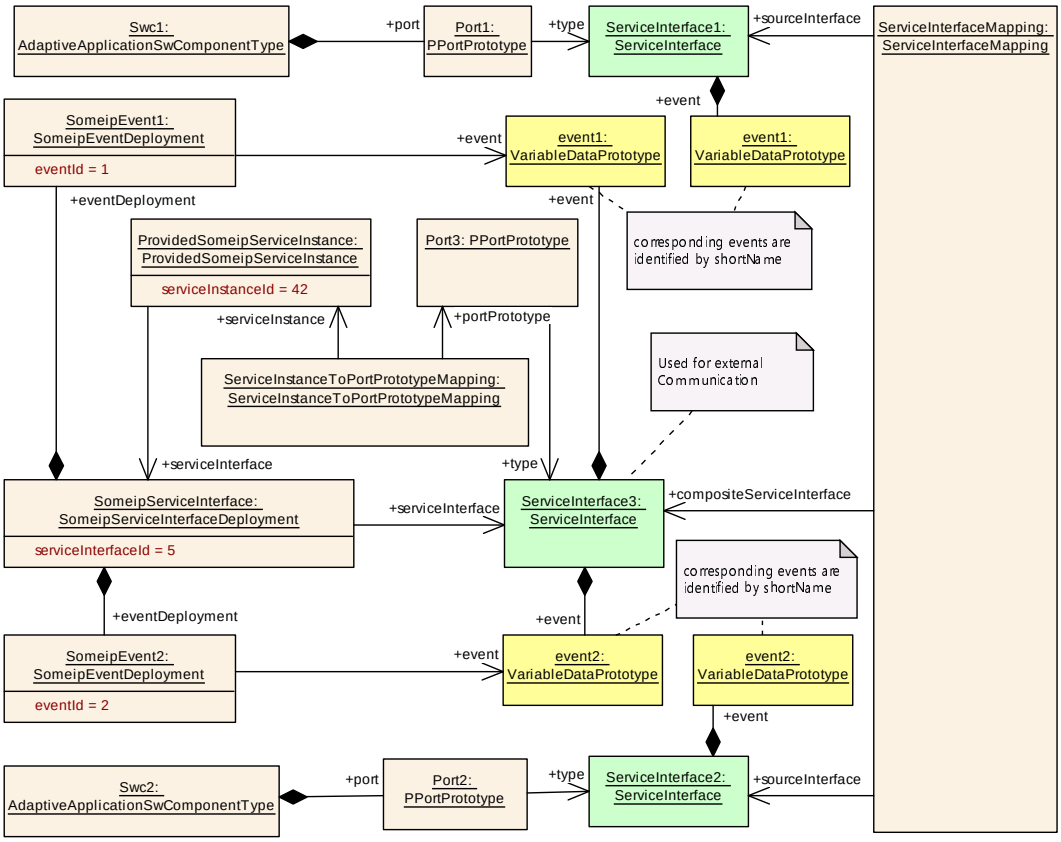


Figure B.21: Example for the deployments of a service in the presence of a `ServiceInterfaceMapping`

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.

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]).

B.2.3.2 Service Instance Deployment by Service Interface Element Mapping

The example in Figure B.23 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 B.21.

In contrast to the example sketched in Figure B.21, the example depicted in Figure B.23 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 B.22.

Note further that the example depicted in Figure B.22 is not limited to the explanation of the actual `ServiceInterfaceElementMapping`.

As the main use case for this is the usage of `ServiceInterface`s for the definition of an “outside” communication binding the example also contains the modeling of such a binding, in this case to SOME/IP.

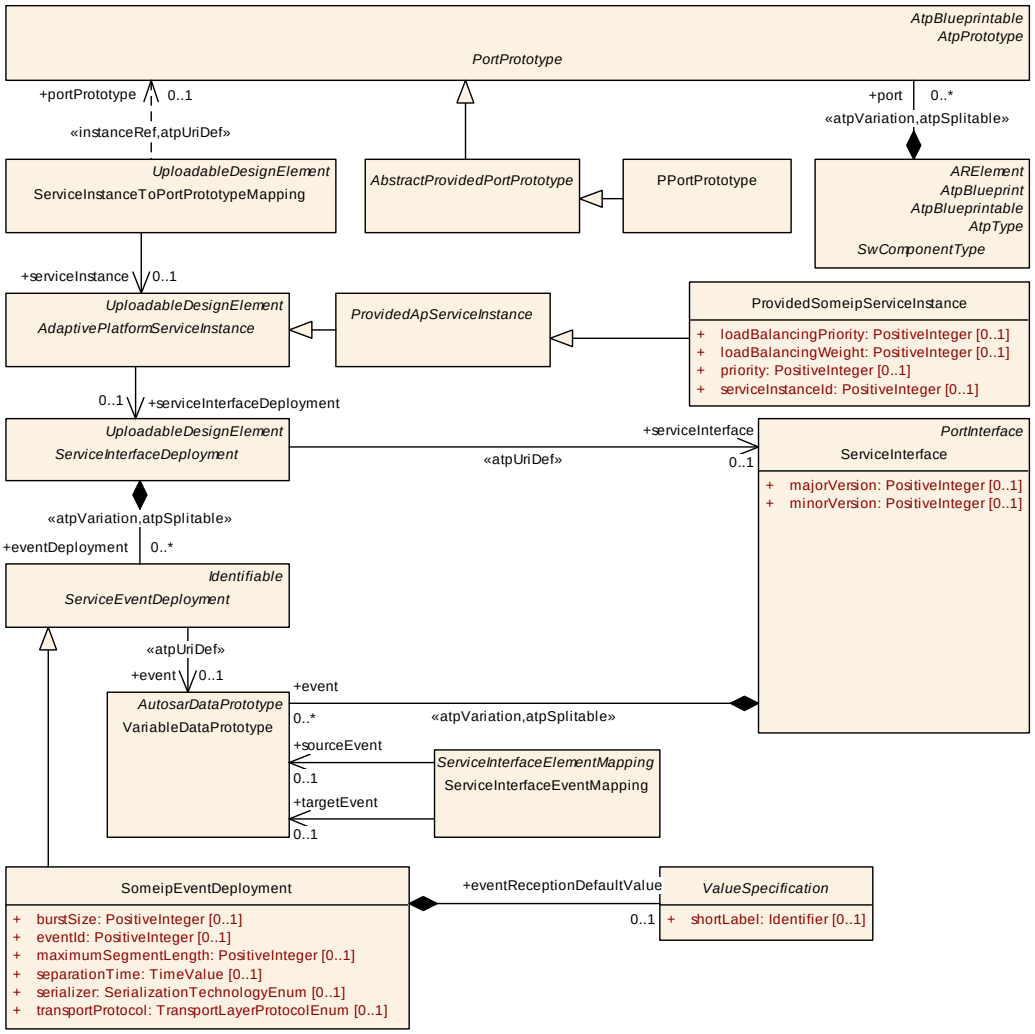


Figure B.22: Excerpt of the relevant meta-classes for the `ServiceInterfaceEventMapping` example

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.

By mapping individual elements of ServiceInterfaces, it is possible to map element with different shortNames to each other. In this example, the event with the shortName event1 is mapped to another event with the shortName eventLeft.

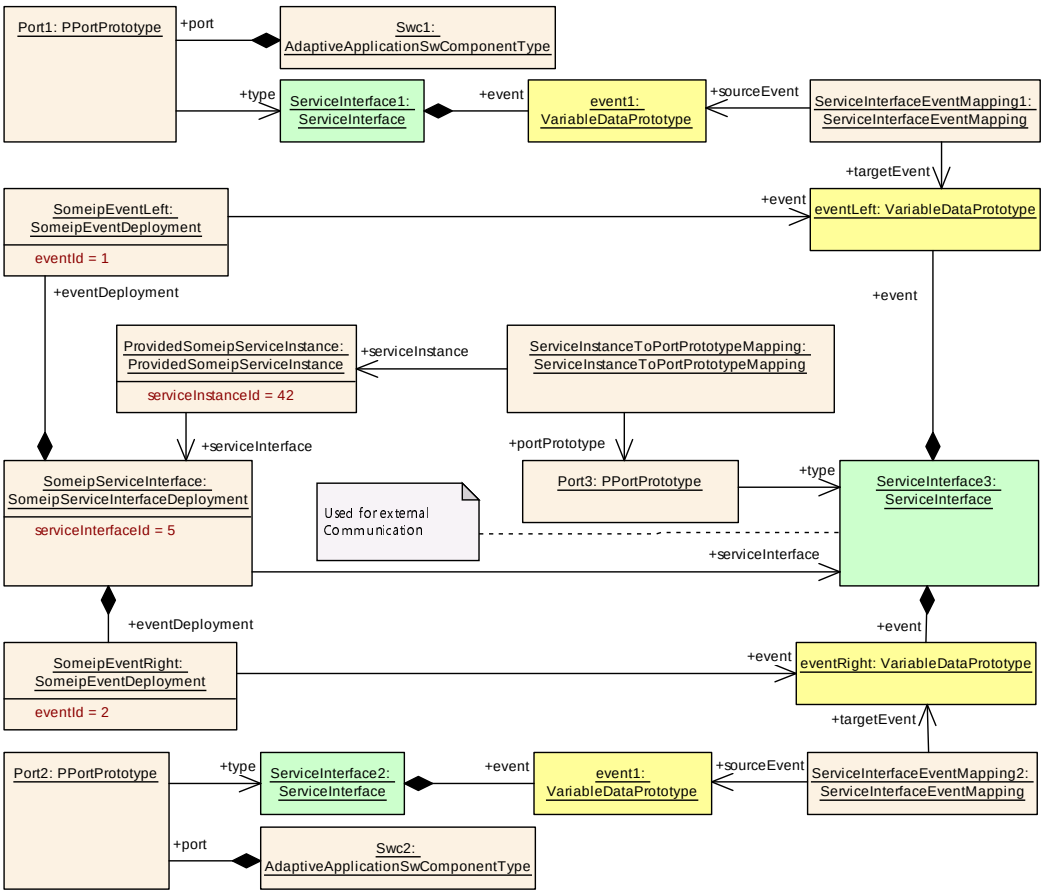


Figure B.23: Example for the deployment of a service in the presence of a ServiceInterfaceEventMapping

In Figure B.23, 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 B.23, Figure B.24 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.

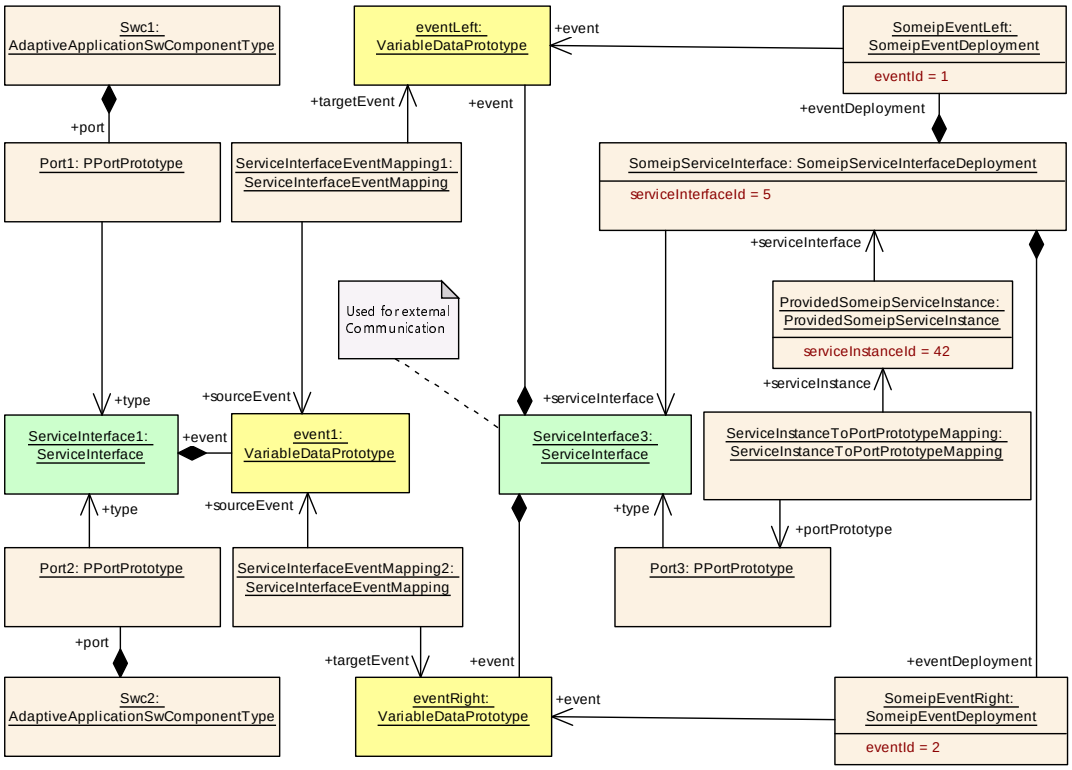


Figure B.24: Example for the deployment of a service in the presence of a `ServiceInterfaceEventMapping` to the same source `ServiceInterface`

Again, this scenario **requires** the existence of appropriately configured `ServiceInterfaceElementMapping`s.

B.2.4 Service Instance Mapping

B.2.4.1 SOME/IP 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 B.25 shows the binding of `PortPrototype`s 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 `shortName`s of the two affected `PortPrototype`s are different. In other words, the `shortName`s of the `PortPrototype`s are not used as a way to identify the opposite end of the service binding.

Instead, the existence of a `ServiceInstanceToPortPrototypeMapping` that maps a `PortPrototype` to a `ProvidedSomeipServiceInstance` or `RequiredSomeipServiceInstance` with the **identical value** of attribute `serviceInstanceId` creates the actual binding between the “find” and the “offer” end.

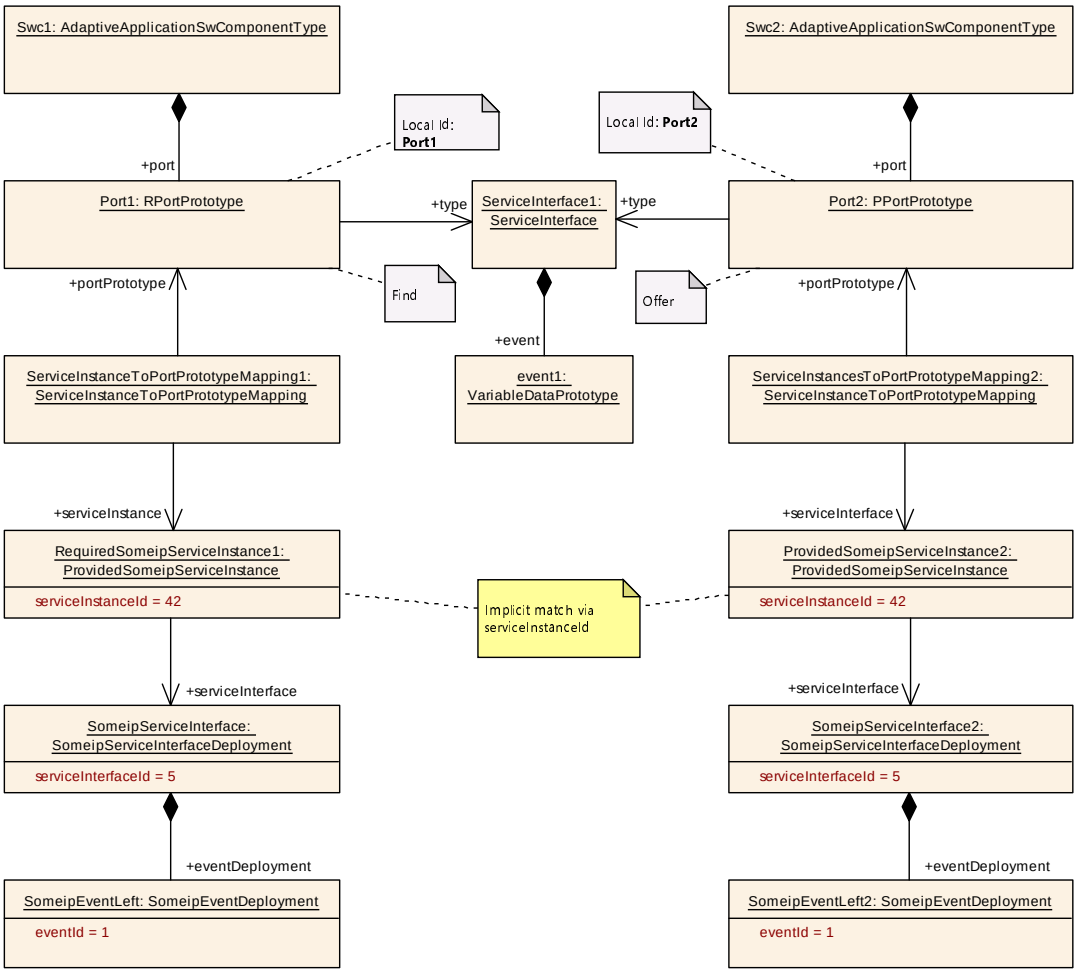


Figure B.25: Port-based binding of a service instance to the application using SOME/IP

B.2.4.2 User-Defined Service Instance Mapping

The next example (depicted in Figure B.26) shows a binding of `PortPrototype`s 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 B.25, the example described in Figure B.26 uses a simple identification based on a numerical value. Again, this is an arbitrary scenario created just for the sake of explanation.

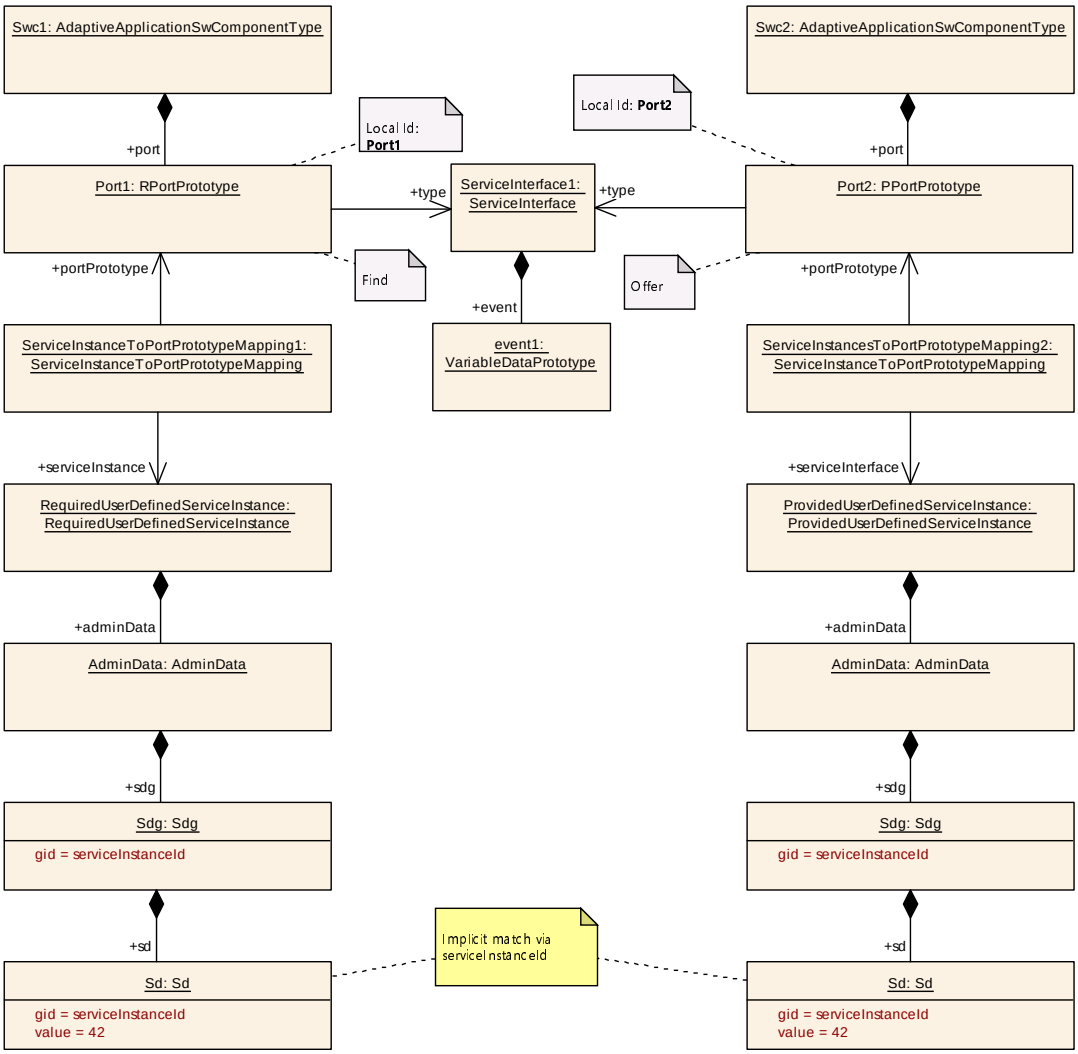


Figure B.26: Port-based binding of a service instance to the application using a user-defined binding

B.2.5 Definition of VLAN Priority

The following table shows an example for the usage of the VLAN priority (please find the background information in section 11.1.1).

Port (Default Priority)	NetworkEndpoint (e.g. IP Address)	ApApplicationEndpoint (e.g. Tcp Port)
CouplingPort1: Prio. 0	NEP1.1: Prio. —	AEP 1.1.1: Prio. —
		AEP 1.1.2: Prio. 1
	NEP1.2: Prio. 0	AEP 1.2.1: Prio. —
		AEP 1.2.2: Prio. 1
CouplingPort2: Prio. 0	NEP2.1: Prio. 1	AEP 2.1.1: Prio. 2
		AEP 2.1.2: Prio. 3

▽

△

	NEP2.2: Prio. 2	AEP 2.2.1: Prio. 1
		AEP 2.2.2: Prio. 0

Table B.2: VLAN Priority Example

Two CouplingPorts are defined and both have a reference to VLAN (EthernetPhysicalChannel) that defines two NetworkEndpoints. The CouplingPort defines the default priority. This default priority can be overwritten on the NetworkEndpoint level, and again on the ApApplicationEndpoint level.

For NetworkEndpoint NEP1.1 no priority is defined. But since on ApApplicationEndpoint AEP 1.1.2 a priority is defined, all messages on CouplingPort1 have the Priority 0 ("best effort"), except for messages that are going over AEP 1.1.2. These messages have the priority 1 (higher priority).

On CouplingPort2, the priority is overwritten on several levels. Please note that ApApplicationEndpoints AEP 2.2.1 and AEP 2.2.2 are reducing the priority that is defined on the NetworkEndpoint NEP2.2.

B.2.6 Static Configuration of remote Peers

B.2.6.1 Provided Service Instance with remote Peers

The following example in Figure B.27 shows the setup of a Service Provider with two statically defined remote peers. The ServiceInstanceToMachineMapping maps two ProvidedSomeipServiceInstances to ApApplicationEndpoints: Camera and Radar_1.

Two remote peers are referenced by the same ServiceInstanceToMachineMapping in the role remoteUnicastConfig: RemotePeer1 and RemotePeer2.

RemotePeer1 is subscribed to EG2 and EG3 of ServiceInstance Radar_1. RemotePeer2 is subscribed to EG1 and EG2 of ServiceInstance Radar_1.

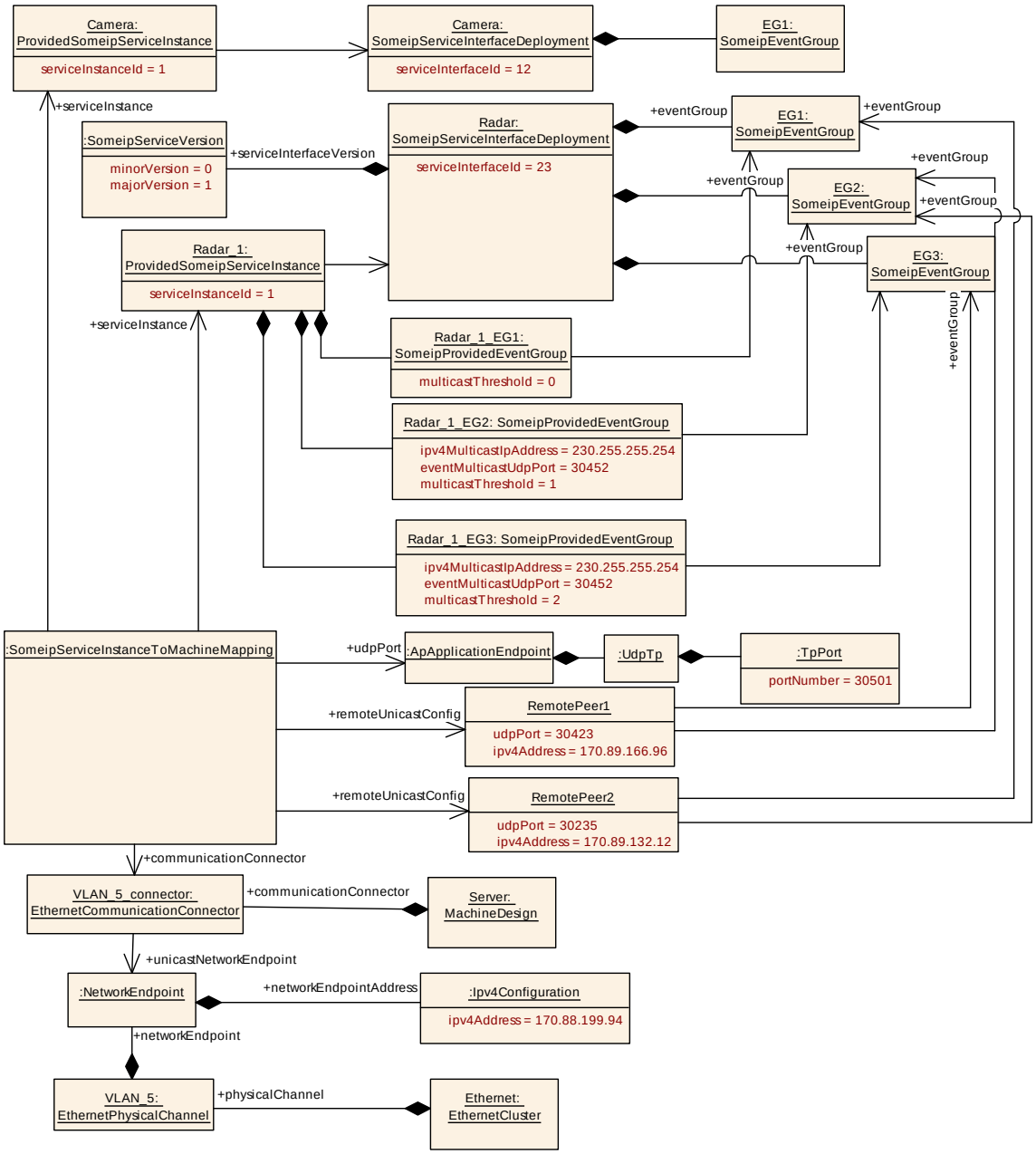


Figure B.27: Example of a Service Provider with static configured remote peers

B.2.6.2 Required Service Instance with remote Peers

The following example in Figure B.28 shows the setup of a Service Consumer with a statically defined remote peer. The `ServiceInstanceToMachineMapping` maps the Radar_CSI1 `RequiredSomeipServiceInstance` to an `ApApplicationEndpoint`.

The remote peer is referenced by the same `ServiceInstanceToMachineMapping` in the role `remoteUnicastConfig`.

The service consumer may receive event notifications of EG2 and EG3 over the multicast address and therefore the `ServiceInstanceToMachineMapping` references the `SomeipRemoteMulticastConfig`.

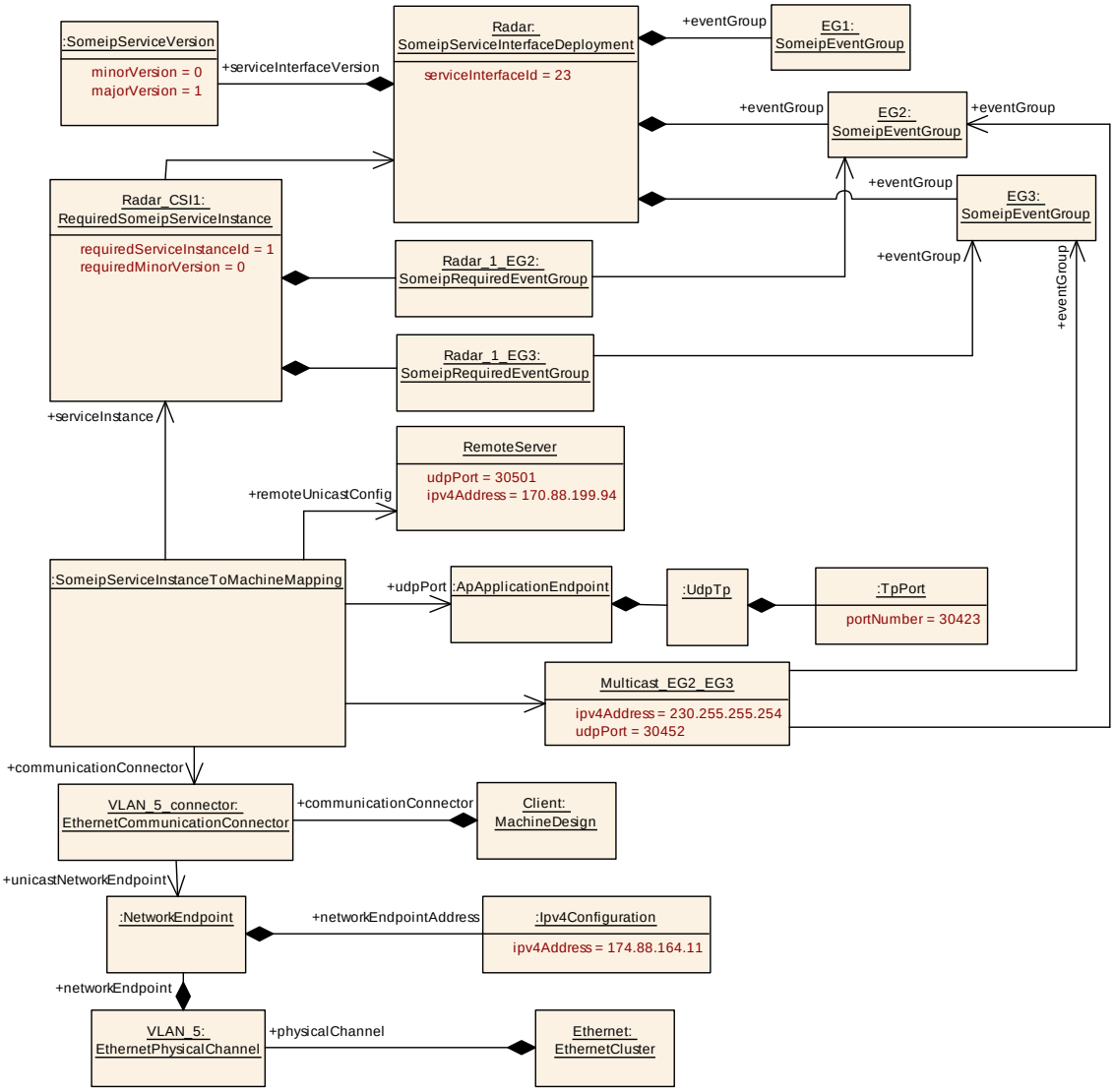


Figure B.28: Example of a Service Consumer with static configured remote peer

B.3 Execution Management

B.3.1 Definition of Startup Configuration

As already mentioned, the mode-dependent startup configuration is directly aggregated by the definition of a `Process`:

```
<PROCESS>
  <SHORT-NAME>AA1</SHORT-NAME>
  <STATE-DEPENDENT-STARTUP-CONFIGS>
    <STATE-DEPENDENT-STARTUP-CONFIG>
```

```
        <EXECUTION-DEPENDENCYS>
          <EXECUTION-DEPENDENCY>
            <PROCESS-STATE-IREF>
              <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
                 DECLARATION-GROUP-PROTOTYPE">/Processes/MWC/
                 ProcessStateMachine</CONTEXT-MODE-DECLARATION-GROUP-
                 PROTOTYPE-REF>
              <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
                 ModeDeclarationGroups/ProcessStateMachine/Running</TARGET-
                 MODE-DECLARATION-REF>
            </PROCESS-STATE-IREF>
          </EXECUTION-DEPENDENCY>
          <EXECUTION-DEPENDENCY>
            <PROCESS-STATE-IREF>
              <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
                 DECLARATION-GROUP-PROTOTYPE">/Processes/MSM/
                 ProcessStateMachine</CONTEXT-MODE-DECLARATION-GROUP-
                 PROTOTYPE-REF>
              <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
                 ModeDeclarationGroups/ProcessStateMachine/Running</TARGET-
                 MODE-DECLARATION-REF>
            </PROCESS-STATE-IREF>
          </EXECUTION-DEPENDENCY>
        </EXECUTION-DEPENDENCYS>
        <FUNCTION-GROUP-STATE-IREFS>
          <FUNCTION-GROUP-STATE-IREF>
            <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
               DECLARATION-GROUP-PROTOTYPE">/FunctionGroupSets/ExampleFGS/
               ExampleFG</CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF>
            <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
               ModeDeclarationGroups/ExampleFG/Running</TARGET-MODE-
               DECLARATION-REF>
          </FUNCTION-GROUP-STATE-IREF>
        </FUNCTION-GROUP-STATE-IREFS>
        <RESOURCE-GROUP-REF DEST="RESOURCE-GROUP">/Machines/ExampleMachine/
           Linux/resourceGroup2</RESOURCE-GROUP-REF>
        <STARTUP-CONFIG-REF DEST="STARTUP-CONFIG">/StartupConfigs/AA1_Startup
           </STARTUP-CONFIG-REF>
      </STATE-DEPENDENT-STARTUP-CONFIG>
    </STATE-DEPENDENT-STARTUP-CONFIGS>
</PROCESS>
```

Listing B.3: Example for the definition of the `StateDependentStartupConfig` owned by a `Process`

In this example, launch dependencies exist on two other `Process`es. Both `Process`es MWC and MSM need to be in the ProcessState “Running” before AA1 is started.

The reference `StateDependentStartupConfig.functionGroupState` refers to a `ModeDeclaration` with the `shortName` “Running” within the `Function Group` “ExampleFG”.

In other words, the referenced `StartupConfig` that is defined in Listing B.4 is valid if the `Function Group` “ExampleFG” is in state “Running”.

```
<STARTUP-CONFIG>
```

```
  <SHORT-NAME>AA1_Startup</SHORT-NAME>
  <PROCESS-ARGUMENTS>
    <PROCESS-ARGUMENT>
      <ARGUMENT>-a</ARGUMENT>
    </PROCESS-ARGUMENT>
    <PROCESS-ARGUMENT>
      <ARGUMENT>-b</ARGUMENT>
    </PROCESS-ARGUMENT>
    <PROCESS-ARGUMENT>
      <ARGUMENT>-d</ARGUMENT>
    </PROCESS-ARGUMENT>
    <PROCESS-ARGUMENT>
      <ARGUMENT>XYZ</ARGUMENT>
    </PROCESS-ARGUMENT>
  </PROCESS-ARGUMENTS>
  <SCHEDULING-POLICY>SCHEDULING-POLICY-FIFO</SCHEDULING-POLICY>
  <SCHEDULING-PRIORITY>20</SCHEDULING-PRIORITY>
</STARTUP-CONFIG>
```

Listing B.4: Example for a `StartupConfig`

The `StateDependentStartupConfig` of the `Process` is assigned to the `ResourceGroup` named “ResourceGroup2” that is defined in the Machine Manifest.

The corresponding definition of a `Machine` contains a `OsModuleInstantiation` that in turn owns the two `ResourceGroup`s named “ResourceGroup1” and “ResourceGroup2”. This aspect can be found in Listing B.5.

```
<MACHINE>
  <SHORT-NAME>ExampleMachine</SHORT-NAME>
  <MODULE-INSTANTIATIONS>
    <OS-MODULE-INSTANTIATION>
      <SHORT-NAME>Linux</SHORT-NAME>
      <RESOURCE-GROUPS>
        <RESOURCE-GROUP>
          <SHORT-NAME>resourceGroup1</SHORT-NAME>
          <CPU-USAGE>60</CPU-USAGE>
          <MEM-USAGE>1000000</MEM-USAGE>
        </RESOURCE-GROUP>
        <RESOURCE-GROUP>
          <SHORT-NAME>resourceGroup2</SHORT-NAME>
          <CPU-USAGE>70</CPU-USAGE>
          <MEM-USAGE>2000000</MEM-USAGE>
        </RESOURCE-GROUP>
      </RESOURCE-GROUPS>
    </OS-MODULE-INSTANTIATION>
  </MODULE-INSTANTIATIONS>
</MACHINE>
```

Listing B.5: Example for the definition of a `Machine`

The example definition of a `FunctionGroupSet` is sketched in Listing B.6.

```
<FUNCTION-GROUP-SET>
  <SHORT-NAME>ExampleFGS</SHORT-NAME>
  <FUNCTION-GROUPS>
    <MODE-DECLARATION-GROUP-PROTOTYPE>
```

```
      <SHORT-NAME>ExampleFG</SHORT-NAME>
      <TYPE-TREF DEST="MODE-DECLARATION-GROUP">/ModeDeclarationGroups/
          ExampleFG</TYPE-TREF>
    </MODE-DECLARATION-GROUP-PROTOTYPE>
  </FUNCTION-GROUPS>
</FUNCTION-GROUP-SET>
```

Listing B.6: Example for the definition of a FunctionGroupSet

The definition of the ModeDeclarationGroup that represents the Function Group is contained in Listing B.7.

```
<MODE-DECLARATION-GROUP>
  <SHORT-NAME>ExampleFG</SHORT-NAME>
  <INITIAL-MODE-REF DEST="MODE-DECLARATION">/ModeDeclarationGroups/
      ExampleFG/Off</INITIAL-MODE-REF>
  <MODE-DECLARATIONS>
    <MODE-DECLARATION>
      <SHORT-NAME>Off</SHORT-NAME>
    </MODE-DECLARATION>
    <MODE-DECLARATION>
      <SHORT-NAME>Running</SHORT-NAME>
    </MODE-DECLARATION>
    <MODE-DECLARATION>
      <SHORT-NAME>Fallback</SHORT-NAME>
    </MODE-DECLARATION>
    <MODE-DECLARATION>
      <SHORT-NAME>Diag</SHORT-NAME>
    </MODE-DECLARATION>
  </MODE-DECLARATIONS>
</MODE-DECLARATION-GROUP>
```

Listing B.7: Example for the definition of ModeDeclarationGroups

The definition of the ModeDeclarationGroup that represents the process state machine is contained in Listing B.8.

```
<MODE-DECLARATION-GROUP>
  <SHORT-NAME>ProcessStateMachine</SHORT-NAME>
  <INITIAL-MODE-REF DEST="MODE-DECLARATION">/ModeDeclarationGroups/
      ProcessStateMachine/Terminated</INITIAL-MODE-REF>
  <MODE-DECLARATIONS>
    <MODE-DECLARATION>
      <SHORT-NAME>Running</SHORT-NAME>
    </MODE-DECLARATION>
    <MODE-DECLARATION>
      <SHORT-NAME>Terminated</SHORT-NAME>
    </MODE-DECLARATION>
  </MODE-DECLARATIONS>
</MODE-DECLARATION-GROUP>
```

Listing B.8: Example for the definition of ModeDeclarationGroups

B.4 Persistency

B.4.1 Definition of Persistent Data

This chapter contains examples for the modeling of persistent data and file storage starting form the design aspect down to the definition of the persistent storage and the mapping between design and deployment.

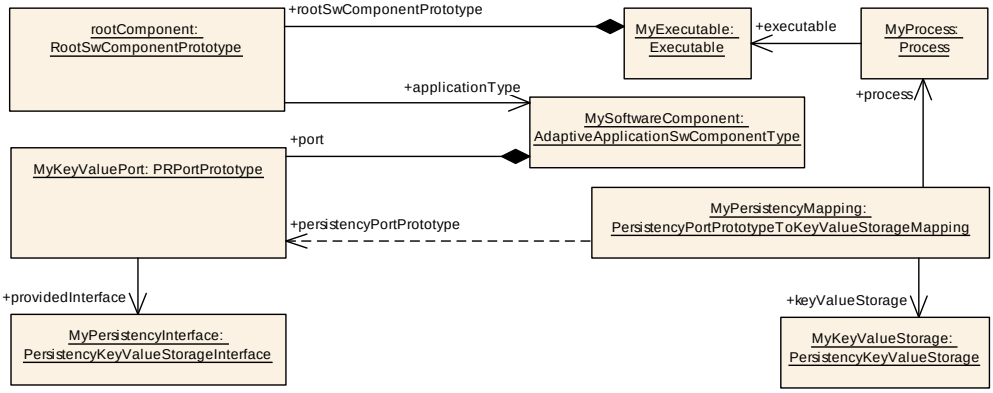


Figure B.29: Simple example modeling of persistent data (design + deployment)

The setup presented in Figure B.29 represents a case with reduced modeling of persistent data.

It is possible to extend the modeling to a deeper level of detail and also formally describe the individual data that is subject to persistency on both design and deployment level, see Figure B.30.

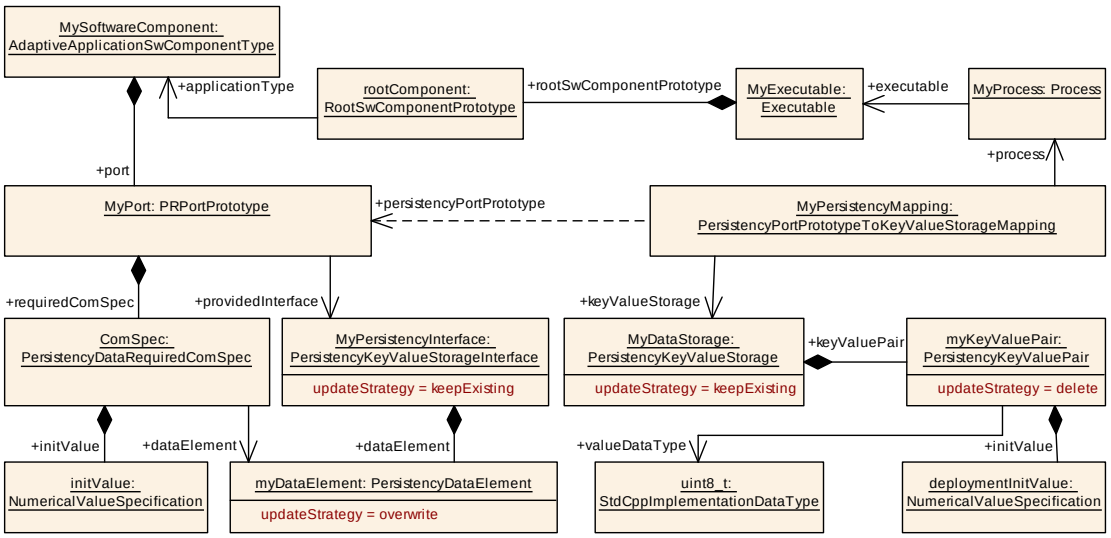


Figure B.30: Advanced example modeling of persistent data (design + deployment)

B.4.2 Definition of Persistent File

The setup presented in Figure B.31 represents a case with reduced modeling of persistent files.

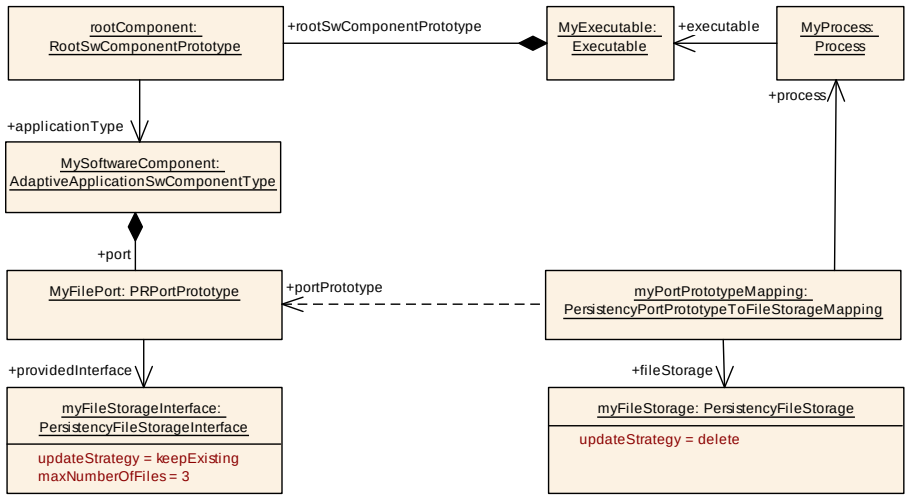


Figure B.31: Simple example modeling of persistent file (design + deployment)

It is possible to extend the modeling to a deeper level of detail and also formally describe the individual file that is subject to persistency on both design and deployment level, see Figure B.32.

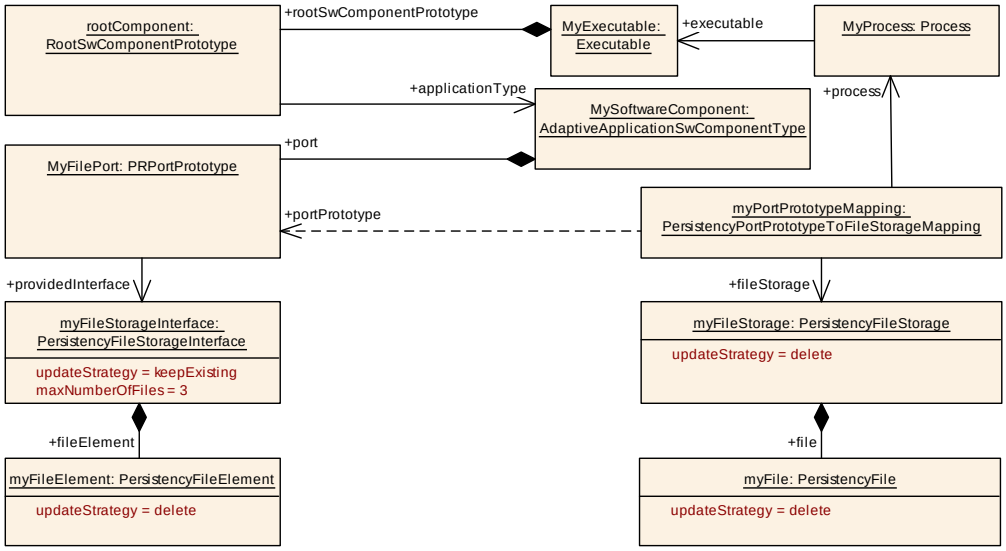


Figure B.32: Advanced example modeling of persistent file (design + deployment)

B.5 Interaction with the PHM

This chapter contains examples for the modeling of platform health management. The example is structured into Application design and platform health management configuration.

B.5.1 Relation between Instance Specifier and Instance Ref

Figure B.33 exemplifies the relation between an `InstanceSpecifier` and an instanceRef. If the `InstanceSpecifier` as well as the model structure is known then the instanceRef can be constructed on the base of the information contained in the `InstanceSpecifier` and vice versa.

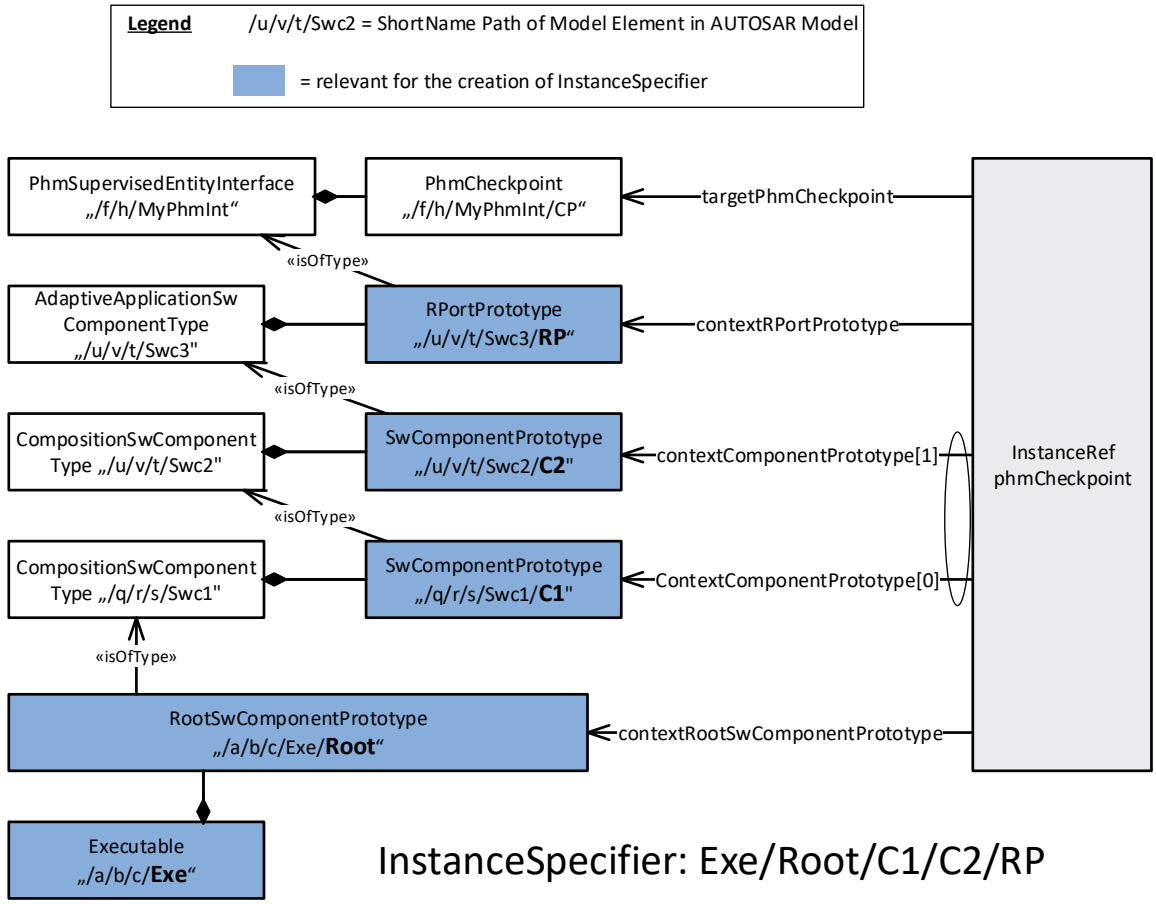


Figure B.33: Example for the creation of an `InstanceSpecifier` of a *SupervisedEntity*

B.5.2 Phm Application Design example

The simple example provided in figure B.34 shows the definition of a `PhmHealthChannelInterface` and a `PhmSupervisedEntityInterface`. This example will also be used in the subsequent section to define the platform health management configuration.

The `PhmHealthChannelInterface` *HealthChannel_A* defines two status attributes:

- *Good*
- *Bad*

The `PhmSupervisedEntityInterface` *SupervisedEntity_B* defines two checkpoints:

- *CP1*
- *CP2*

The `AdaptiveApplicationSwComponentType` *AdaptiveApplication* defines two `RPortPrototype`s

- *Hc_A* typed by *HealthChannel_A*
- *Se_B* typed by *SupervisedEntity_B*

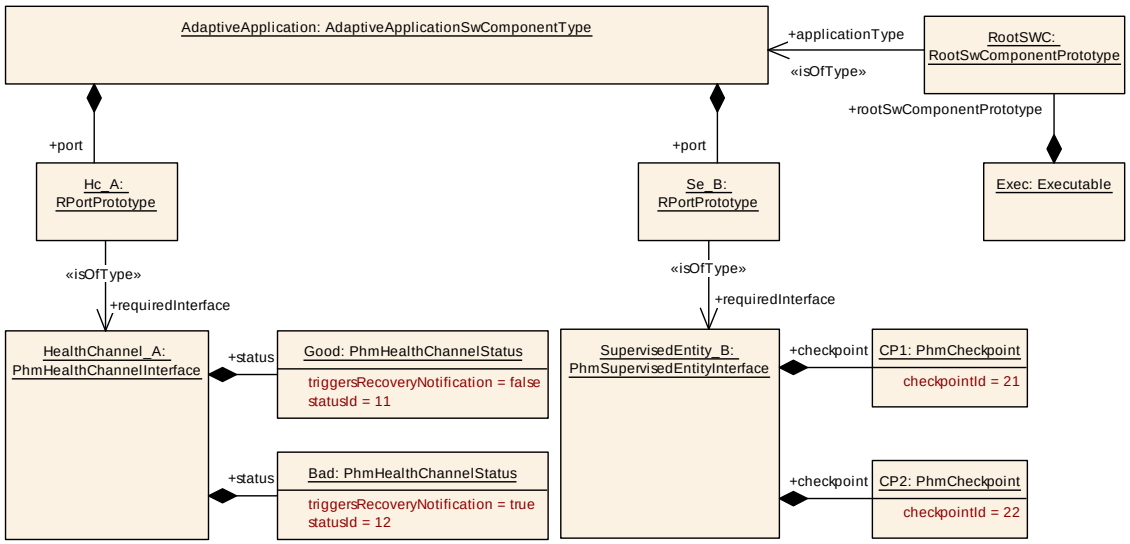


Figure B.34: Example modeling of Health Channel and Supervised Entity

B.5.3 Phm configuration example

When defining the configuration contribution for Phm it is required to first create representatives of the application design model artifacts (health channel status and supervised entity checkpoints) in the Phm configuration context. This is shown in figure B.35.

In this example the *PHM* `PlatformHealthManagementContribution` defines placeholder elements which refer to the respective application design model artifacts:

Example health channel:

- *Hc_Status_Good* refers to the *Good* status of *HealthChannel_A*
- *Hc_Status_Bad* refers to the *Bad* status of *HealthChannel_A*

Example supervision checkpoint:

- *Se_B_Cp1* refers to the *CP1* checkpoint of *SupervisedEntity_B*
- *Se_B_Cp2* refers to the *CP2* checkpoint of *SupervisedEntity_B*

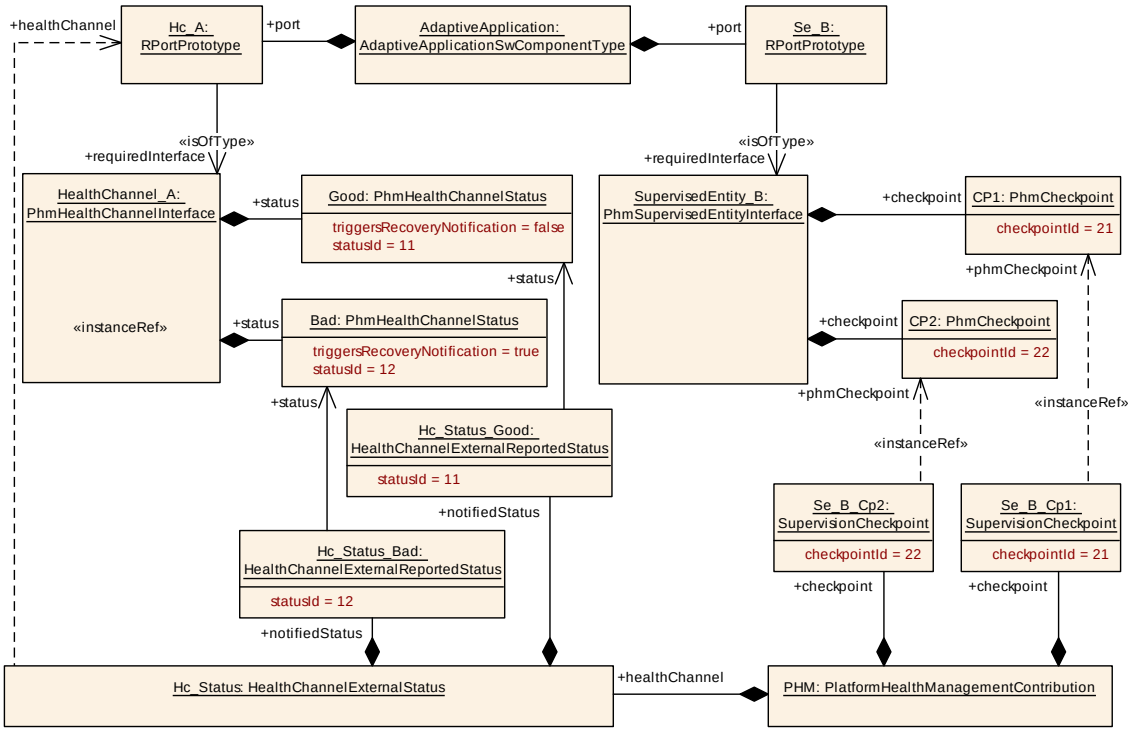


Figure B.35: Example modeling of Phm placeholder definition

Note how the numbers from the `PhmCheckpoint.checkpointId` have been replicated in `SupervisionCheckpoint.checkpointId` and `PhmHealthChannelStatus.statusId` have been replicated in `HealthChannelExternalReportedStatus.statusId`.

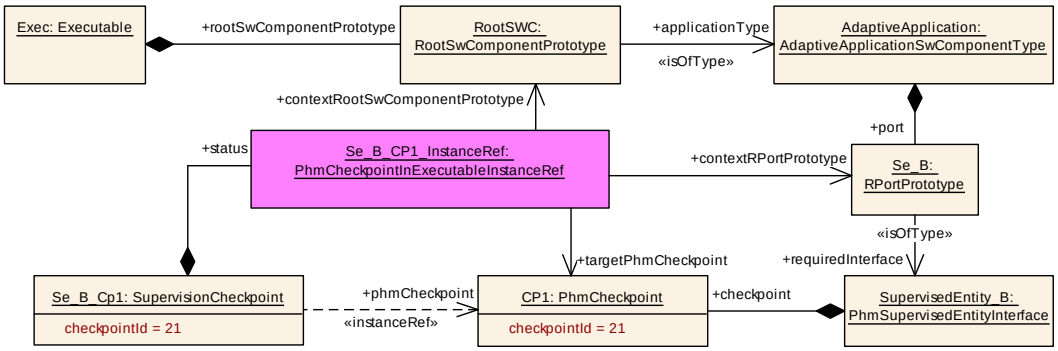


Figure B.36: Example modeling of Phm instance reference

Note that these instance references have a composite nature, which is shown in example figure B.36. Here it is shown that in order to instance reference from the `SupervisionCheckpoint` *Se_B_Cp1* to the `PhmCheckpoint` *CP1* there is the structured reference required consisting of

- `contextRootSwComponentPrototype`
- `contextRPortPrototype`
- `targetPhmCheckpoint`

B.6 State Management

B.6.1 Consideration of User Perspective

The modeling of state management is described in sections 3.3.4 and 10.14.

Obviously, the definition of the meta-classes to serialize the state management configuration does not have to directly reflect the mental model which some creators of a specific state management instance have in mind.

This means that the design approach in the user domain might be entirely different from the definition of the respective meta-classes. Such a situation is manageable as long as the mental model applied on the side of the users can be mapped to the meta-classes and their relationships.

To be more concrete, some of the users who define state management configurations, **prefer to think in terms of tables that combine an input trigger or error value with a target state**, see Table B.3 as an example.

This chapter exemplifies such an approach for how a table-based configuration of state changes could look like and how the table columns map to the meta-model.

B.6.2 Scenario for Trigger

For the definition of the response to triggers, some people define a table, which maps the *trigger value* to the assumed *current state* and the *next state*.

In other words, if the *trigger value* is found in one row and the value of the current state in the state machine is equal to the assumed *current state* in the applicable table row, then the state machine shall be switched to the *next state* listed in the table row.

Please note that the concrete table entries for the states might consist of numerals or literals, depending on the personal preference of the user.

Of course, the meta-model allows to create a connection between the table values either by means of the numerical `ModeDeclaration.value` or the textual `ModeDeclaration.shortName`.

An example for such a table is listed in Table B.3.

Trigger Value	Current State	Next State
1001	0	1
1000	1	0
1000	2	0
1000	3	0
1000	4	0

Table B.3: Example table for associating trigger with state changes using the `value`s

It is important that it is possible to map this mental model of state management configuration to the existing meta-model.

Alternatively, it would be possible to create the table using the `shortName` of the respective `ModeDeclaration`s and thereby create the exact same semantics, see Table B.4.

Trigger Value	Current State	Next State
1001	StateOff	StateFull_primary
1000	StateFull_primary	StateOff
1000	State_Fallback	StateOff
1000	State3	StateOff
1000	State4	StateOff

Table B.4: Example table for associating trigger with state changes using the `shortName`

Thus, the correspondence of table columns to meta-model elements is documented in Table B.5.

Table Column of Table B.3	Meta-Model Element, see Figure 10.67
Trigger Value	StateManagementCompareCondition.compareValue
Current State	StateManagementTriggerCompare-Rule.assumedCurrentState
Next State	StateManagementRequestRule.nextState

Table B.5: Correspondence of table columns and meta-model elements for triggers

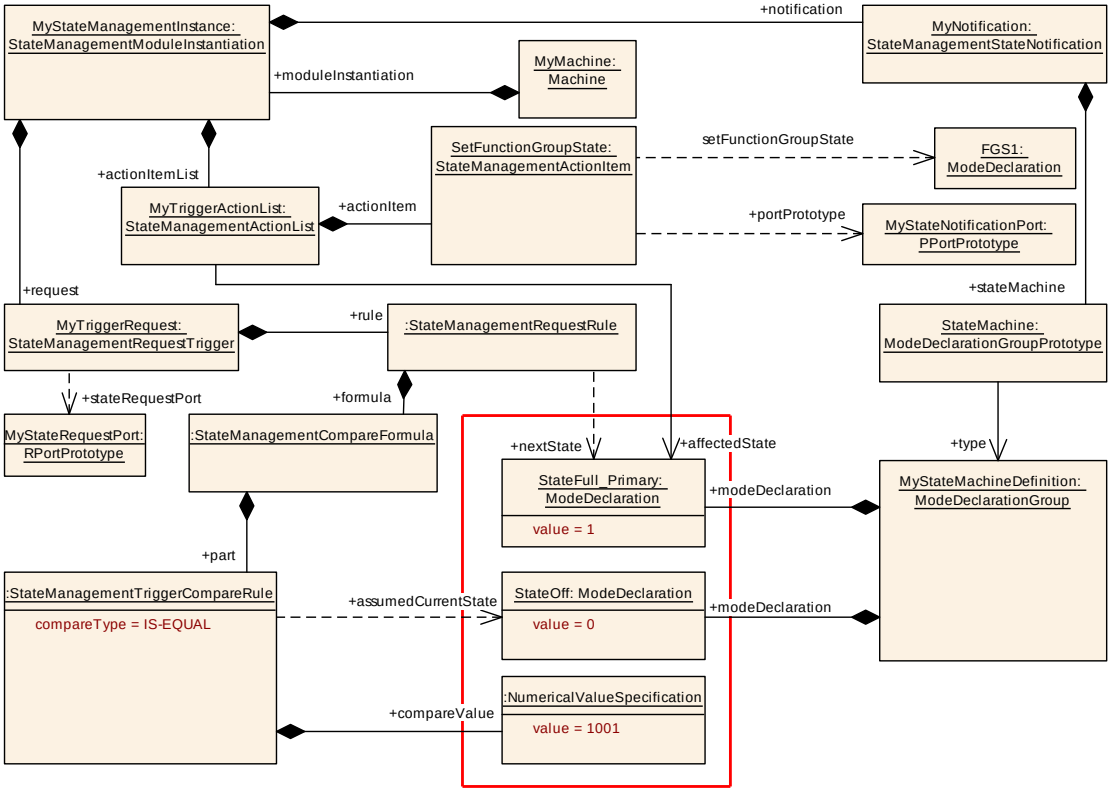


Figure B.37: Example state management modeling for a trigger request

In particular, the Figure B.37 and, with more detail, Listing B.9 depict the modeling of the relevant part for the first row of Table B.3.

The relevant parts of the table row are located inside the red box in Figure B.37.

The concrete usage of StateManagementCompareCondition.compareValue can be found in the listing as part of the STATE-MANAGEMENT-TRIGGER-COMPARE-RULE, specifically by COMPARE-TYPE and COMPARE-VALUE.

The StateManagementTriggerCompareRule.assumedCurrentState is represented by the ASSUMED-CURRENT-STATE-IREF.

And finally, the StateManagementRequestRule.nextState can be found at the NEXT-STATE-IREF inside the STATE-MANAGEMENT-RQUEST-RULE.

```
<STATE-MANAGEMENT-REQUEST-TRIGGER>
  <SHORT-NAME>MyTriggerRequest</SHORT-NAME>
  <STATE-REQUEST-PORT-IREF>
    <CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF DEST="ROOT-SW-COMPONENT-
        PROTOTYPE">/Executables/MyStateManager/MyRootComponent</CONTEXT-ROOT
        -SW-COMPONENT-PROTOTYPE-REF>
    <TARGET-R-PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">/SwComponentTypes/
        StateManagerComp/MyStateRequestPort</TARGET-R-PORT-PROTOTYPE-REF>
  </STATE-REQUEST-PORT-IREF>
  <RULES>
    <STATE-MANAGEMENT-REQUEST-RULE>
      <FORMULA>
        <PARTS>
          <STATE-MANAGEMENT-TRIGGER-COMPARE-RULE>
            <COMPARE-TYPE>IS-EQUAL</COMPARE-TYPE>
            <COMPARE-VALUE>
              <NUMERICAL-VALUE-SPECIFICATION>
                <VALUE>1001</VALUE>
              </NUMERICAL-VALUE-SPECIFICATION>
            </COMPARE-VALUE>
            <ASSUMED-CURRENT-STATE-IREF>
              <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
                  DECLARATION-GROUP-PROTOTYPE">/Machines/MyMachine/
                  MyStateManagementInstance/MyNotification/StateMachine</
                  CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF>
              <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
                  ModeDeclarationGroups/MyStateMachineDefinition/StateOff</
                  TARGET-MODE-DECLARATION-REF>
            </ASSUMED-CURRENT-STATE-IREF>
          </STATE-MANAGEMENT-TRIGGER-COMPARE-RULE>
        </PARTS>
      </FORMULA>
      <NEXT-STATE-IREF>
        <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
            DECLARATION-GROUP-PROTOTYPE">/Machines/MyMachine/
            MyStateManagementInstance/MyNotification/StateMachine</CONTEXT-
            MODE-DECLARATION-GROUP-PROTOTYPE-REF>
        <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
            ModeDeclarationGroups/MyStateMachineDefinition/StateFull_primary
            </TARGET-MODE-DECLARATION-REF>
      </NEXT-STATE-IREF>
    </STATE-MANAGEMENT-REQUEST-RULE>
  </RULES>
</STATE-MANAGEMENT-REQUEST-TRIGGER>
```

Listing B.9: Example for the definition of a StateManagementRequestTrigger

The ModeDeclarationGroup that types the internal state machine is depicted in Listing B.10. The definition of values that represent the states in Table B.3 are visible in the listing, see VALUE.

```
<MODE-DECLARATION-GROUP>
  <SHORT-NAME>MyStateMachineDefinition</SHORT-NAME>
  <MODE-DECLARATIONS>
    <MODE-DECLARATION>
      <SHORT-NAME>StateOff</SHORT-NAME>
```

```
      <VALUE>0</VALUE>
    </MODE-DECLARATION>
    <MODE-DECLARATION>
      <SHORT-NAME>StateFull_primary</SHORT-NAME>
      <VALUE>1</VALUE>
    </MODE-DECLARATION>
    <MODE-DECLARATION>
      <SHORT-NAME>State_Fallback</SHORT-NAME>
      <VALUE>2</VALUE>
    </MODE-DECLARATION>
  </MODE-DECLARATIONS>
</MODE-DECLARATION-GROUP>
```

Listing B.10: Example for the definition of a `ModeDeclarationGroup` to define the states of an internal state machine

The modeling of the applicable `StateManagementActionList` is sketched in Listing B.11.

Please note that the modeling of the `PortPrototype`s and the internal structure of the `Executable` is not explicitly documented in this example because there are other example demonstrating this aspect already.

```
<STATE-MANAGEMENT-ACTION-LIST>
  <SHORT-NAME>MyTriggerActionList</SHORT-NAME>
  <ACTION-ITEMS>
    <STATE-MANAGEMENT-SET-FUNCTION-GROUP-STATE-ACTION-ITEM>
      <SHORT-NAME>SetFunctionGroupState</SHORT-NAME>
      <PORT-PROTOTYPE-IREF>
        <CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF DEST="ROOT-SW-COMPONENT-
            PROTOTYPE">/Executables/MyStateManager/MyRootComponent</CONTEXT-
            ROOT-SW-COMPONENT-PROTOTYPE-REF>
        <TARGET-P-PORT-PROTOTYPE-REF DEST="P-PORT-PROTOTYPE">/
            SwComponentTypes/StateManagerComp/MyStateNotificationPort</
            TARGET-P-PORT-PROTOTYPE-REF>
      </PORT-PROTOTYPE-IREF>
      <SET-FUNCTION-GROUP-STATE-IREF>
        <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
            DECLARATION-GROUP-PROTOTYPE">/FunctionGroups/MyFumctionGroups/
            FG1</CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF>
        <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
            ModeDeclarationGroups/MyFunctionGroupStates/FGS1</TARGET-MODE-
            DECLARATION-REF>
      </SET-FUNCTION-GROUP-STATE-IREF>
    </STATE-MANAGEMENT-SET-FUNCTION-GROUP-STATE-ACTION-ITEM>
  </ACTION-ITEMS>
</STATE-MANAGEMENT-ACTION-LIST>
```

Listing B.11: Example for the definition of a `StateManagementActionList` for the trigger request

The modeling of the `FunctionGroupSet` mentioned in the example is sketched in Listing B.12.

```
<FUNCTION-GROUP-SET>
  <SHORT-NAME>MyFumctionGroups</SHORT-NAME>
```

```
  <FUNCTION-GROUPS>
    <MODE-DECLARATION-GROUP-PROTOTYPE>
      <SHORT-NAME>FG1</SHORT-NAME>
      <TYPE-TREF DEST="MODE-DECLARATION-GROUP">/ModeDeclarationGroups/
          MyFunctionGroupStates</TYPE-TREF>
    </MODE-DECLARATION-GROUP-PROTOTYPE>
  </FUNCTION-GROUPS>
</FUNCTION-GROUP-SET>
```

Listing B.12: Example for the definition of a FunctionGroupSet for the discussed example

The modeling of the ModeDeclarationGroup that defines the Function Group States named “FGS1” and “FGS2” that are used in the example is sketched in Listing B.13.

```
<MODE-DECLARATION-GROUP>
  <SHORT-NAME>MyFunctionGroupStates</SHORT-NAME>
  <MODE-DECLARATIONS>
    <MODE-DECLARATION>
      <SHORT-NAME>FGS1</SHORT-NAME>
    </MODE-DECLARATION>
    <MODE-DECLARATION>
      <SHORT-NAME>FGS2</SHORT-NAME>
    </MODE-DECLARATION>
  </MODE-DECLARATIONS>
</MODE-DECLARATION-GROUP>
```

Listing B.13: Example for the definition of a ModeDeclarationGroup for defining the Function Group States used in the discussed example

Finally, the modeling of the StateManagementStateNotification is sketched in in Listing B.14.

Please note that the definition of the STATE-MACHINE in Listing B.14 relates to the definition of internal states (modeled as a MODE-DECLARATION-GROUP), as sketched in Listing B.10.

```
<STATE-MANAGEMENT-STATE-NOTIFICATION>
  <SHORT-NAME>MyNotification</SHORT-NAME>
  <NOTIFICATION-PORT-IREF>
    <CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF DEST="ROOT-SW-COMPONENT-
        PROTOTYPE">/Executables/MyStateManager/MyRootComponent</CONTEXT-ROOT
        -SW-COMPONENT-PROTOTYPE-REF>
    <TARGET-P-PORT-PROTOTYPE-REF DEST="P-PORT-PROTOTYPE">/SwComponentTypes/
        StateManagerComp/MyStateNotificationPort</TARGET-P-PORT-PROTOTYPE-
        REF>
  </NOTIFICATION-PORT-IREF>
  <STATE-MACHINE>
    <SHORT-NAME>StateMachine</SHORT-NAME>
    <TYPE-TREF DEST="MODE-DECLARATION-GROUP">/ModeDeclarationGroups/
        MyStateMachineDefinition</TYPE-TREF>
  </STATE-MACHINE>
</STATE-MANAGEMENT-STATE-NOTIFICATION>
```

Listing B.14: Example for the definition of a StateManagementStateNotification for communicating state changes within the discussed example

B.6.3 Scenario for Error

For the definition of the response to errors, some people define a table, which maps the *trigger value* and the *next state* that is to be switched to if the particular error value is received.

In other words, if the value of *error code* is found in one row, then the state machine shall be switched to the *next state* listed in the table row, **independently** of the values of the current state of the state machine.

An example for such a table is listed in Table B.6.

Error Code	Next State
11	2
12	3
111	2
23	4
24	0
244	5

Table B.6: Example table for associating error with state changes

The correspondence of table columns to meta-model elements is documented in Table B.7.

Table Column of Table B.6	Meta-Model Element, see Figure 10.68
ErrorCode	`StateManagementCompareCondition.compareValue`
Next State	`StateManagementRequestRule.nextState`

Table B.7: Correspondence of table columns and meta-model elements for errors

In particular, the Figure B.38 and, with more detail, Listing B.15 depict the modeling of the relevant part for the first row of Table B.6.

The relevant parts of the table row are located inside the red box in Figure B.38.

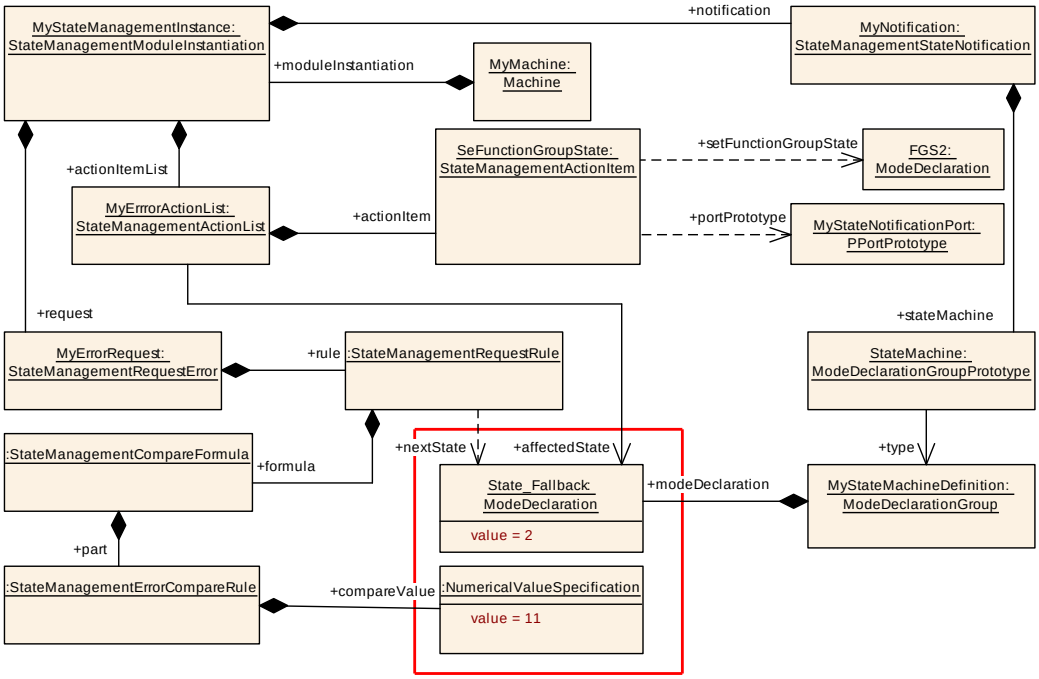


Figure B.38: Example state management modeling for an error request

This example features partly the same model elements as the previous trigger example.

But in this specific case, there is a different `StateManagementActionList` named “MyErrorActionList” that refers to different `modeDeclaration` named “State_Fallback” (that is represented by the numerical value 2) in the role `affectedState`.

Obviously, a `StateManagementRequestError` named “MyErrorRequest” exists that aggregates its own `rule`. The `StateManagementErrorCompareRule.compareValue` has the `value` 11.

This fits to the first row of Table B.6.

```
<STATE-MANAGEMENT-REQUEST-ERROR>
  <SHORT-NAME>MyErrorRequest</SHORT-NAME>
  <STATE-REQUEST-PORT-IREF>
    <CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF DEST="ROOT-SW-COMPONENT-
        PROTOTYPE">/Executables/MyStateManager/MyRootComponent</CONTEXT-ROOT
        -SW-COMPONENT-PROTOTYPE-REF>
    <TARGET-R-PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">/SwComponentTypes/
        StateManagerComp/MyErrorRequestPort</TARGET-R-PORT-PROTOTYPE-REF>
  </STATE-REQUEST-PORT-IREF>
  <RULES>
    <STATE-MANAGEMENT-REQUEST-RULE>
      <FORMULA>
        <PARTS>
          <STATE-MANAGEMENT-ERROR-COMPARE-RULE>
            <COMPARE-TYPE>IS-EQUAL</COMPARE-TYPE>
            <COMPARE-VALUE>
              <NUMERICAL-VALUE-SPECIFICATION>
```

```
                <VALUE>11</VALUE>
              </NUMERICAL-VALUE-SPECIFICATION>
            </COMPARE-VALUE>
          </STATE-MANAGEMENT-ERROR-COMPARE-RULE>
        </PARTS>
      </FORMULA>
      <NEXT-STATE-IREF>
        <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
            DECLARATION-GROUP-PROTOTYPE">/Machines/MyMachine/
            MyStateManagementInstance/MyNotification/StateMachine</CONTEXT-
            MODE-DECLARATION-GROUP-PROTOTYPE-REF>
        <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
            ModeDeclarationGroups/MyStateMachineDefinition/State_Fallback</
            TARGET-MODE-DECLARATION-REF>
      </NEXT-STATE-IREF>
    </STATE-MANAGEMENT-REQUEST-RULE>
  </RULES>
</STATE-MANAGEMENT-REQUEST-ERROR>
```

Listing B.15: Example for the definition of a `StateManagementRequestError`

The modeling of the applicable `StateManagementActionList` is sketched in Listing B.16.

```
<STATE-MANAGEMENT-ACTION-LIST>
  <SHORT-NAME>MyErrorActionList</SHORT-NAME>
  <ACTION-ITEMS>
    <STATE-MANAGEMENT-SET-FUNCTION-GROUP-STATE-ACTION-ITEM>
      <SHORT-NAME>SetFunctionGroupState</SHORT-NAME>
      <PORT-PROTOTYPE-IREF>
        <CONTEXT-ROOT-SW-COMPONENT-PROTOTYPE-REF DEST="ROOT-SW-COMPONENT-
            PROTOTYPE">/Executables/MyStateManager/MyRootComponent</CONTEXT-
            ROOT-SW-COMPONENT-PROTOTYPE-REF>
        <TARGET-P-PORT-PROTOTYPE-REF DEST="P-PORT-PROTOTYPE">/
            SwComponentTypes/StateManagerComp/MyStateNotificationPort</
            TARGET-P-PORT-PROTOTYPE-REF>
      </PORT-PROTOTYPE-IREF>
      <SET-FUNCTION-GROUP-STATE-IREF>
        <CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF DEST="MODE-
            DECLARATION-GROUP-PROTOTYPE">/FunctionGroups/MyFumctionGroups/
            FG1</CONTEXT-MODE-DECLARATION-GROUP-PROTOTYPE-REF>
        <TARGET-MODE-DECLARATION-REF DEST="MODE-DECLARATION">/
            ModeDeclarationGroups/MyFunctionGroupStates/FGS2</TARGET-MODE-
            DECLARATION-REF>
      </SET-FUNCTION-GROUP-STATE-IREF>
    </STATE-MANAGEMENT-SET-FUNCTION-GROUP-STATE-ACTION-ITEM>
  </ACTION-ITEMS>
</STATE-MANAGEMENT-ACTION-LIST>
```

Listing B.16: Example for the definition of a `StateManagementActionList` for the error request

B.7 Crypto

B.7.1 Relation between a Crypto Certificate and a Crypto Key Slot

The following figure B.39 shows an example how the `SwcServiceDependency` is used to create a relation between a Crypto Certificate and a Crypto KeySlot.

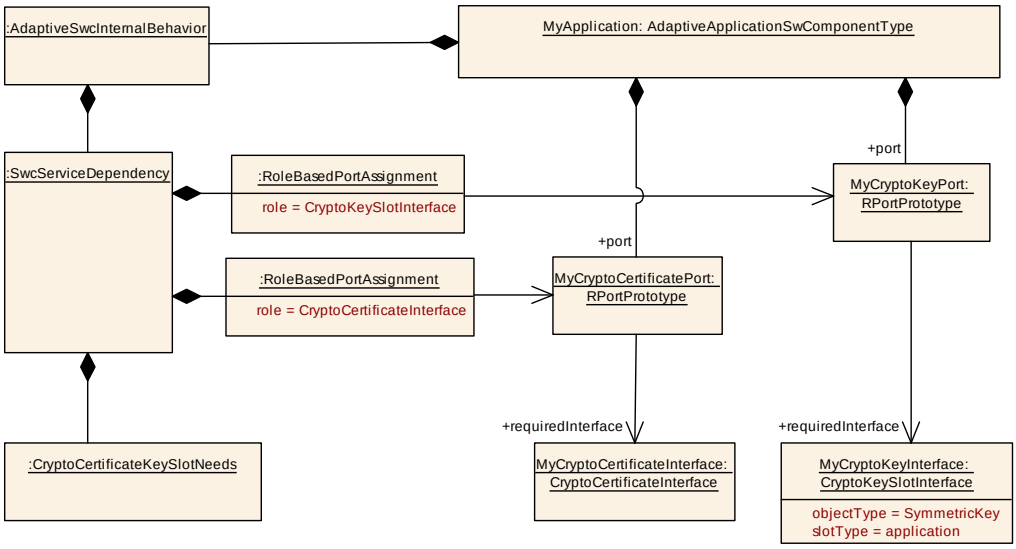


Figure B.39: Example that shows a link between a Port typed by `CryptoKeySlotInterface` and a Port typed by `CryptoCertificateInterface`

B.8 Machine Manifest

B.8.1 Distributed Definition of Environment Variables

The aggregation `Machine.environmentVariable` has been defined with the stereotype ≪`atpSplitable`≫. The consequence of this modeling is that it is possible to contribute to the definition of environment variables (see section 9.3) from **different sources**.

As an example, assume two partial models (sketched in Listing B.17 and B.18) which both add a folder to the search path of a machine running on an adaptive platform ECU.

```
<MACHINE>
  <SHORT-NAME>Machine</SHORT-NAME>
  <ENVIRONMENT-VARIABLES>
    <TAG-WITH-OPTIONAL-VALUE>
      <KEY>PATH</KEY>
      <SEQUENCE-OFFSET>10</SEQUENCE-OFFSET>
      <VALUE>/usr/application-x/bin</VALUE>
    </TAG-WITH-OPTIONAL-VALUE>
  </ENVIRONMENT-VARIABLES>
```

```
</MACHINE>
```

Listing B.17: Example for the definition of environmentVariable (file 1)

```
<MACHINE>
  <SHORT-NAME>Machine</SHORT-NAME>
  <ENVIRONMENT-VARIABLES>
    <TAG-WITH-OPTIONAL-VALUE>
      <KEY>PATH</KEY>
      <SEQUENCE-OFFSET>20</SEQUENCE-OFFSET>
      <VALUE>/usr/application-y/bin</VALUE>
    </TAG-WITH-OPTIONAL-VALUE>
  </ENVIRONMENT-VARIABLES>
</MACHINE>
```

Listing B.18: Example for the definition of environmentVariable (file 2)

Merging the splitable elements from the partial models results in the following content. Please note that the merged model exists only internally in the AUTOSAR tool. The Listing B.19 is therefore shown **only for illustration**.

```
<MACHINE>
  <SHORT-NAME>Machine</SHORT-NAME>
  <ENVIRONMENT-VARIABLES>
    <TAG-WITH-OPTIONAL-VALUE>
      <KEY>PATH</KEY>
      <SEQUENCE-OFFSET>10</SEQUENCE-OFFSET>
      <VALUE>/usr/application-x/bin</VALUE>
    </TAG-WITH-OPTIONAL-VALUE>
    <TAG-WITH-OPTIONAL-VALUE>
      <KEY>PATH</KEY>
      <SEQUENCE-OFFSET>20</SEQUENCE-OFFSET>
      <VALUE>/usr/application-y/bin</VALUE>
    </TAG-WITH-OPTIONAL-VALUE>
  </ENVIRONMENT-VARIABLES>
</MACHINE>
```

Listing B.19: Example for the definition of environmentVariable (merged)

The generator for the target machine configuration may use this information to create the following environment variable:

```
PATH=/usr/application-x/bin;/usr/application-y/bin;
```

B.9 Ethernet Raw Data Stream Configuration

B.9.1 Preamble

The examples discussed in this section of the document are based on the modeling of two `EthernetCommunicationConnector`s, one for the client (see Listing B.20) and one for the server (see Listing B.21)

```
<ETHERNET-COMMUNICATION-CONNECTOR>
```

```
  <SHORT-NAME>ClientConn</SHORT-NAME>
  <AP-APPLICATION-ENDPOINTS>
    <AP-APPLICATION-ENDPOINT>
      <SHORT-NAME>ClientUnicastAEP</SHORT-NAME>
      <TP-CONFIGURATION>
        <UDP-TP>
          <UDP-TP-PORT>
            <PORT-NUMBER>3333</PORT-NUMBER>
          </UDP-TP-PORT>
        </UDP-TP>
      </TP-CONFIGURATION>
    </AP-APPLICATION-ENDPOINT>
    <AP-APPLICATION-ENDPOINT>
      <SHORT-NAME>ClientMulticastAEP</SHORT-NAME>
      <TP-CONFIGURATION>
        <UDP-TP>
          <UDP-TP-PORT>
            <PORT-NUMBER>7654</PORT-NUMBER>
          </UDP-TP-PORT>
        </UDP-TP>
      </TP-CONFIGURATION>
    </AP-APPLICATION-ENDPOINT>
  </AP-APPLICATION-ENDPOINTS>
  <UNICAST-NETWORK-ENDPOINT-REFS>
    <UNICAST-NETWORK-ENDPOINT-REF DEST="NETWORK-ENDPOINT">/
       CommunicationClusters/Ether/VLAN1/ClientEP</UNICAST-NETWORK-ENDPOINT
       -REF>
  </UNICAST-NETWORK-ENDPOINT-REFS>
</ETHERNET-COMMUNICATION-CONNECTOR>
```

Listing B.20: Definition of `EthernetCommunicationConnector` for the client

```
<ETHERNET-COMMUNICATION-CONNECTOR>
  <SHORT-NAME>ServerConn</SHORT-NAME>
  <AP-APPLICATION-ENDPOINTS>
    <AP-APPLICATION-ENDPOINT>
      <SHORT-NAME>ServerUnicastAEP</SHORT-NAME>
      <TP-CONFIGURATION>
        <UDP-TP>
          <UDP-TP-PORT>
            <PORT-NUMBER>7777</PORT-NUMBER>
          </UDP-TP-PORT>
        </UDP-TP>
      </TP-CONFIGURATION>
    </AP-APPLICATION-ENDPOINT>
    <AP-APPLICATION-ENDPOINT>
      <SHORT-NAME>ServerMulticastAEP</SHORT-NAME>
      <TP-CONFIGURATION>
        <UDP-TP>
          <UDP-TP-PORT>
            <PORT-NUMBER>7654</PORT-NUMBER>
          </UDP-TP-PORT>
        </UDP-TP>
      </TP-CONFIGURATION>
    </AP-APPLICATION-ENDPOINT>
  </AP-APPLICATION-ENDPOINTS>
```

```
  <UNICAST-NETWORK-ENDPOINT-REFS>
    <UNICAST-NETWORK-ENDPOINT-REF DEST="NETWORK-ENDPOINT">/
        CommunicationClusters/Ether/VLAN1/ServerEP</UNICAST-NETWORK-ENDPOINT
        -REF>
  </UNICAST-NETWORK-ENDPOINT-REFS>
</ETHERNET-COMMUNICATION-CONNECTOR>
```

Listing B.21: Definition of `EthernetCommunicationConnector` for the server

On top of that, the definition of a `EthernetPhysicalChannel` is also considered, see Listing B.22.

```
<ETHERNET-PHYSICAL-CHANNEL>
  <SHORT-NAME>VLAN1</SHORT-NAME>
  <NETWORK-ENDPOINTS>
    <NETWORK-ENDPOINT>
      <SHORT-NAME>ClientEP</SHORT-NAME>
      <NETWORK-ENDPOINT-ADDRESSES>
        <IPV-4-CONFIGURATION>
          <IPV-4-ADDRESS>192.168.2.240</IPV-4-ADDRESS>
        </IPV-4-CONFIGURATION>
      </NETWORK-ENDPOINT-ADDRESSES>
    </NETWORK-ENDPOINT>
    <NETWORK-ENDPOINT>
      <SHORT-NAME>ServerEP</SHORT-NAME>
      <NETWORK-ENDPOINT-ADDRESSES>
        <IPV-4-CONFIGURATION>
          <IPV-4-ADDRESS>192.168.2.223</IPV-4-ADDRESS>
        </IPV-4-CONFIGURATION>
      </NETWORK-ENDPOINT-ADDRESSES>
    </NETWORK-ENDPOINT>
  </NETWORK-ENDPOINTS>
</ETHERNET-PHYSICAL-CHANNEL>
```

Listing B.22: Definition of `EthernetPhysicalChannel` for connection between client and server

B.9.2 Fully formalized Model Example

A scenario where both the raw data stream client and raw data stream server are fully formalized inside an AUTOSAR model is depicted in Figure B.40.

In the case of a fully formalized model it may be possible to derive the client configuration (that also consists of the credentials of the remote server) from the server's model if the server model is accessible.

The AUTOSAR meta-model foresees the specification of the server credentials in the client's configuration by means of the aggregation `EthernetRawDataStreamClientMapping.remoteServerConfig`.

If multicast communication from server to client is foreseen, then the aggregation in the role `EthernetRawDataStreamClientMapping.remoteServerConfig.multicastCredentials` also needs to exist.

For the sake of simplicity, this example has been created to communicate the general idea of the configuration approach rather than to demonstrate a true-to-the-meta-model representation.

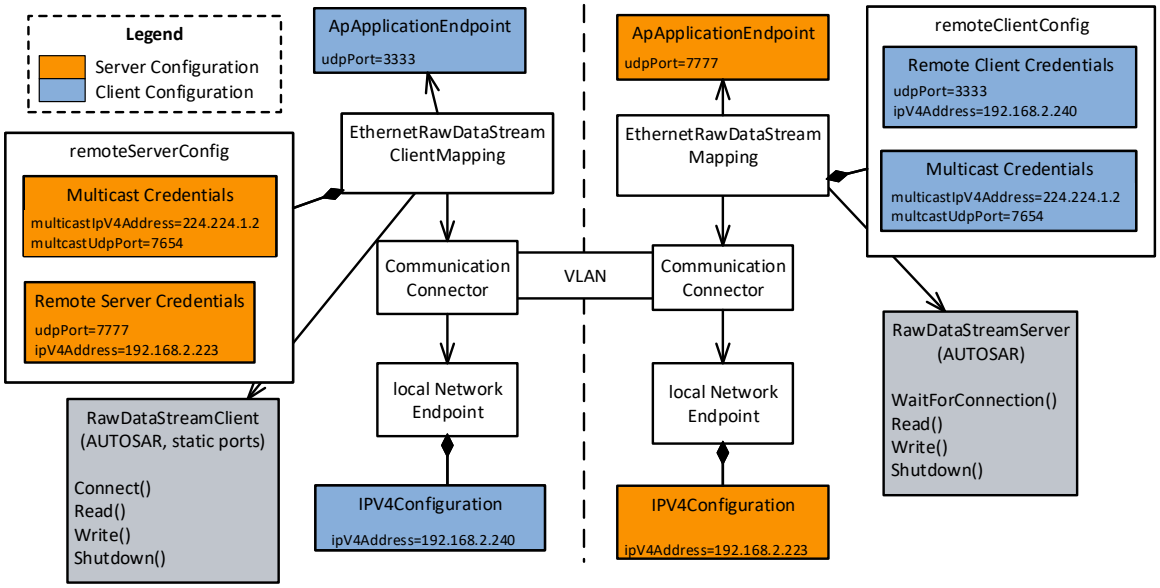


Figure B.40: Scenario where both the raw data stream client and raw data stream server are fully formalized inside an AUTOSAR model

Please note that this example utilizes UDP communication only. If the unicast communication were based on TCP, the server's configuration would (as mentioned in section 12.3.2) not need the client's TCP credentials, i.e. `EthernetRawDataStreamServerMapping.remoteClientConfig` (the "Remote Client Credentials" in Figure B.40) would not exist.

In a fully formalized model, the configuration of the multicast credentials shall be identical for all affected communication ends. The simplified (the focus is on the Ethernet configuration, based on the modeling described in section B.9.1) modeling of the client mapping is sketched in Listing B.23.

```
<ETHERNET-RAW-DATA-STREAM-CLIENT-MAPPING>
  <SHORT-NAME>ClientMap</SHORT-NAME>
  <LOCAL-ENDPOINT-CONFIG>
    <LOCAL-COMM-CONNECTOR-REF DEST="ETHERNET-COMMUNICATION-CONNECTOR">/
        MachineDesigns/Mach/ClientConn</LOCAL-COMM-CONNECTOR-REF>
    <LOCAL-UDP-PORT-REF DEST="AP-APPLICATION-ENDPOINT">/MachineDesigns/Mach
        /ClientConn/ClientUnicastAEP</LOCAL-UDP-PORT-REF>
  </LOCAL-ENDPOINT-CONFIG>
  <REMOTE-SERVER-CONFIG>
    <MULTICAST-CREDENTIALS>
      <IP-V-4-ADDRESS>224.224.1.2</IP-V-4-ADDRESS>
      <UDP-PORT>7654</UDP-PORT>
    </MULTICAST-CREDENTIALS>
    <UNICAST-CREDENTIALS>
      <IP-V-4-ADDRESS>192.168.2.223</IP-V-4-ADDRESS>
      <UDP-PORT>7777</UDP-PORT>
```

```
    </UNICAST-CREDENTIALS>
  </REMOTE-SERVER-CONFIG>
</ETHERNET-RAW-DATA-STREAM-CLIENT-MAPPING>
```

Listing B.23: Modeling of the client mapping

The simplified (the focus is on the Ethernet configuration, based on the modeling described in section B.9.1) modeling of the server mapping is sketched in Listing B.24.

```
<ETHERNET-RAW-DATA-STREAM-SERVER-MAPPING>
  <SHORT-NAME>ServerMap</SHORT-NAME>
  <LOCAL-ENDPOINT-CONFIG>
    <LOCAL-COMM-CONNECTOR-REF DEST="ETHERNET-COMMUNICATION-CONNECTOR">/
       MachineDesigns/Mach/ServerConn</LOCAL-COMM-CONNECTOR-REF>
    <LOCAL-UDP-PORT-REF DEST="AP-APPLICATION-ENDPOINT">/MachineDesigns/Mach
       /ServerConn/ServerUnicastAEP</LOCAL-UDP-PORT-REF>
  </LOCAL-ENDPOINT-CONFIG>
  <REMOTE-CLIENT-CONFIG>
    <UNICAST-UDP-CREDENTIALS>
      <IP-V-4-ADDRESS>192.168.2.240</IP-V-4-ADDRESS>
      <UDP-PORT>3333</UDP-PORT>
    </UNICAST-UDP-CREDENTIALS>
  </REMOTE-CLIENT-CONFIG>
</ETHERNET-RAW-DATA-STREAM-SERVER-MAPPING>
```

Listing B.24: Modeling of the server mapping

B.9.3 Only Client is formalized

The case that **only** the client is formalized (and the server does not appear in the form of an `EthernetRawDataStreamServerMapping`) can be represented by a subset of the modeling described in section B.9.2 as well as the model sketched in Listing B.23.

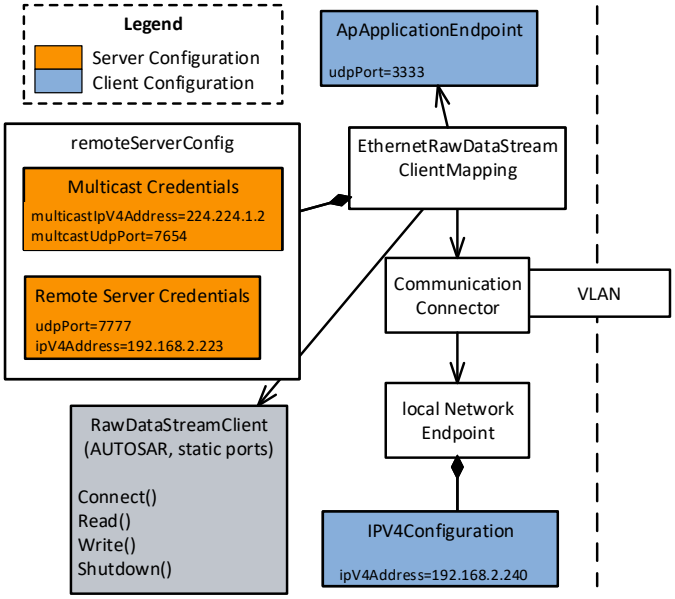


Figure B.41: Scenario where only the raw data stream client is fully formalized inside an AUTOSAR model

B.9.4 Only Server is formalized

The case that **only** the server is formalized (and the client does not appear in the form of an `EthernetRawDataStreamClientMapping`) can be represented by a subset of the modeling described in section B.9.2 as well as the model sketched in Listing B.24.

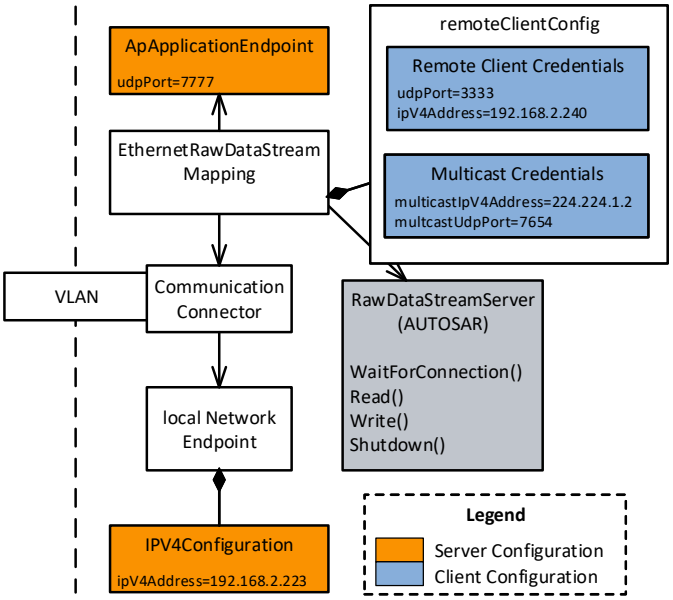


Figure B.42: Scenario where only the raw data stream server is fully formalized inside an AUTOSAR model

B.10 Signal/Service Translation

B.10.1 Signal/Service Mapping

B.10.1.1 Map Event to Signal

In the example sketched in Figure B.43 the *TestEvent* in the *TestServiceInterface* is of type *struct1* that consists of a primitive element *a* and struct *b*. The struct *b* consists of the primitive elements *x*, *y* and *z*.

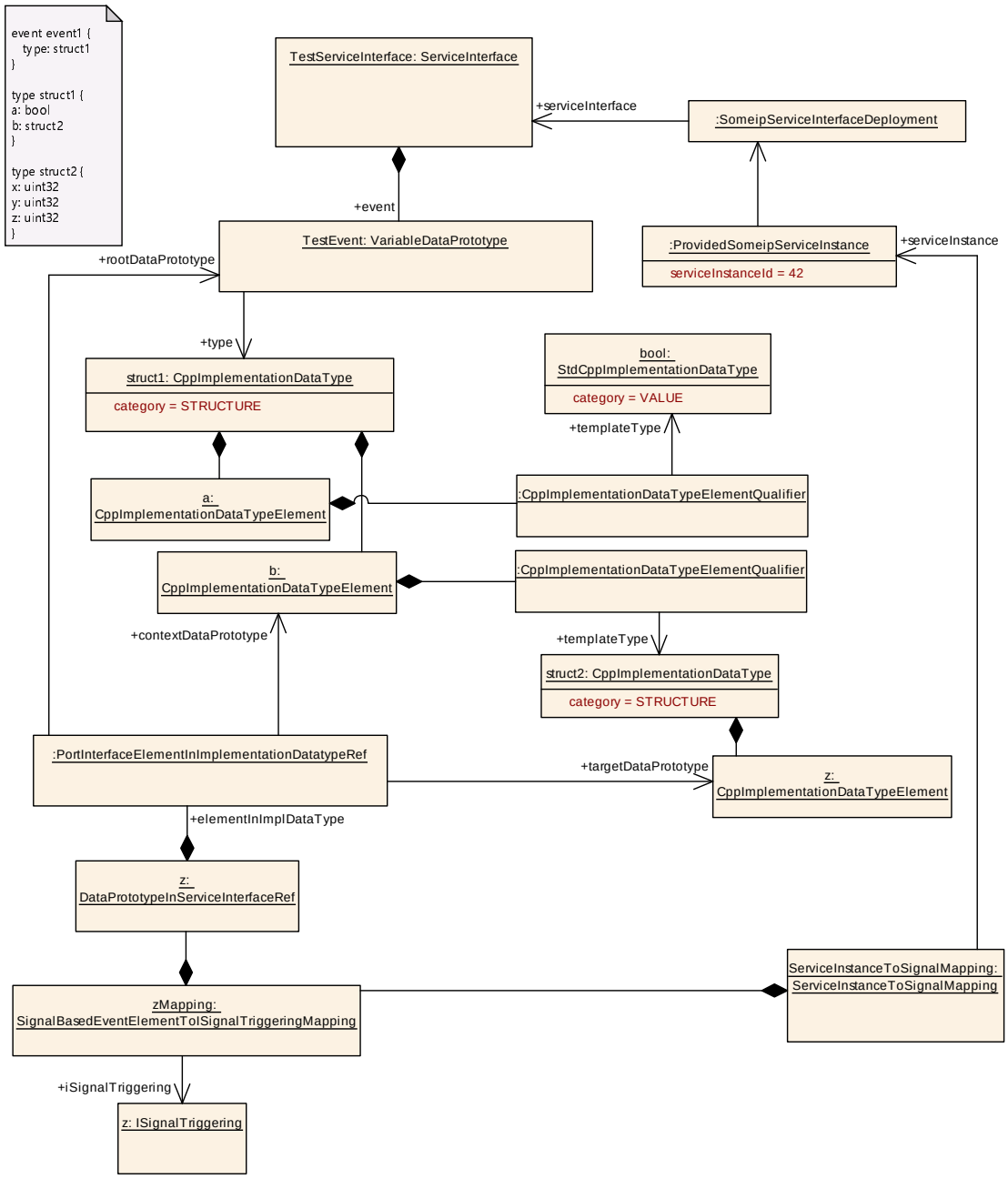


Figure B.43: Example for a mapping of event content to a Signal

Several `SignalBasedEventElementToISignalTriggeringMapping`s are used to map the primitive `DataPrototype`s *a*, *x*, *y* and *z* to `ISignalTriggering`s of `ISignal`s located in the `ISignalGroup` representing the *TestEvent*.

The example shows the mapping of *z* to the `ISignalTriggering`. Here, the `elementInImplDatatype` role of `DataPrototypeInServiceInterfaceRef` is used to refer to the `targetDataPrototype` since it refers to the internal structure of an `AutosarDataPrototype` which is typed by a `CppImplementationDataType`. The context of this reference is defined by the *TestEvent* and struct *b*.

B.10.1.2 Map Field to Signal

In the example sketched in Figure B.44 the *testField* in the *testServiceInterface* is of type *struct1* that consists of the primitive elements *a* and *b*. The *testField* defines a notifier and a setter method.

One `SignalBasedFieldToISignalTriggeringMapping` maps the *TestField* to `ISignalTriggering`s for the Setter-Call and Setter-Return.

Here the `dataPrototype` role of `DataPrototypeInServiceInterfaceRef` is used to refer to the `targetDataPrototype` (`field`) according to the rules defined in [TPS_MANI_01136] and [TPS_MANI_01137].

Additional `SignalBasedFieldToISignalTriggeringMapping`s are necessary to map the field notifier to the corresponding `ISignalTriggering`s.

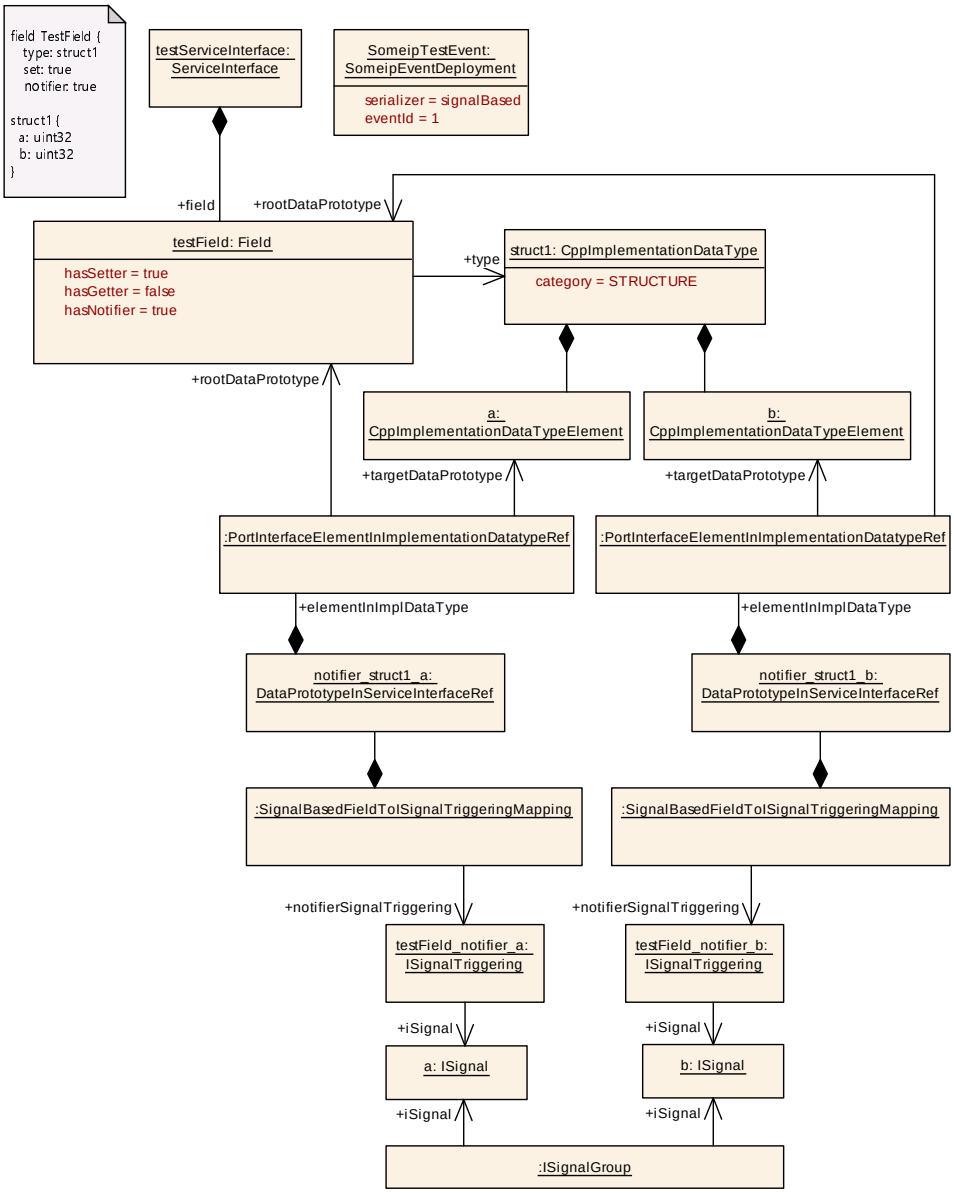


Figure B.44: Example for a mapping of a field to Signals

The primitive DataPrototypes *a* and *b* are mapped by additional SignalBasedFieldToISignalTriggeringMappings to ISignalTriggerings of ISignals located in the ISignalGroup.

Here, the elementInImplDatatype role of DataPrototypeInServiceInterfaceRef is used to refer to the targetDataPrototype since it refers to the internal structure of an AutosarDataPrototype which is typed by a CppImplementationDataType. The context of this reference is defined by the *testField*.

B.10.2 Signal/Service Translation Behavior

B.10.2.1 Optional Structure Element in the context of Signal/Service Translation

Figure B.45 illustrates the usage of update-bits in the source ISignalIPdu as the cause for the definition of isOptional members of the event.

The left part of figure B.45 shows that the *signal group x1*, *signal y*, and *signal z* have update-bits defined and thus the resulting event definition for the ServiceInterface needs to respect this. In this example the event *xyz* is defined as an ApplicationRecordDataType containing 3 ApplicationRecordElements where each element has the attribute isOptional set to true.

The right part of figure B.45 takes one specific ISignalIPdu instance with only the *updateBit_y* set to 1, thus only the *signal y* shall be considered as updated. Therefore the resulting event notification contains an event *xyz* where only the member *y* is defined. The other two members of *xyz* are not set in this example.

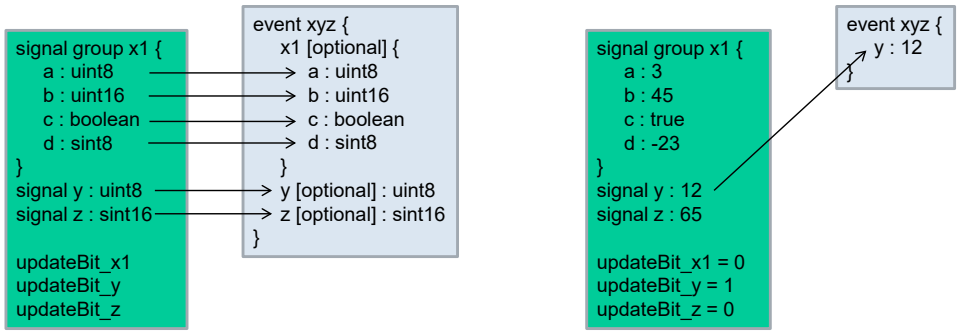


Figure B.45: Example of update-bits for signal-service-translation

B.10.2.2 Optional Structure Element in the context of Service/Signal Translation

Figure B.46 illustrates the usage of isOptional elements in the event / field definition as the cause for the definition of update-bits in the target ISignalIPdu.

The left part of figure B.46 shows that the *event klm* has two optional members (*k* and *l*) and one mandatory member (*m*). These are mapped into an `ISignalIPdu` with two update-bits (*updateBit_k* and *updateBit_l*).

The right part of figure B.46 takes one specific *event klm* instance with values defined for *l* (which is optional but present) and *m* (which is not optional and therefore has to be present). There is no value defined for the optional member *k*. The resulting `ISignalIPdu` takes the values for *l* and *m* and sets the update-bits accordingly (*updateBit_k=0* and *updateBit_l=1*). The values for the not-set optional members stay unchanged.

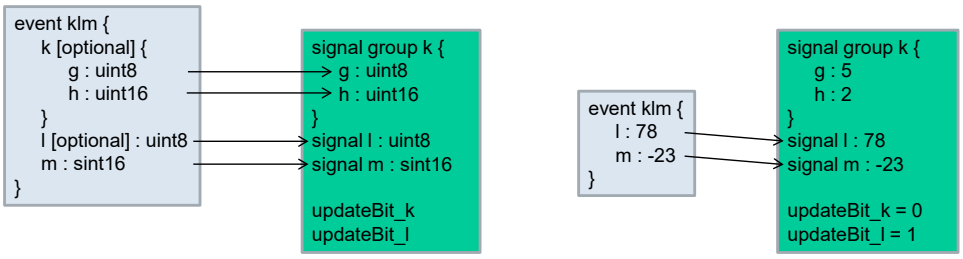


Figure B.46: Example of update-bits for `service-signal-translation`

B.10.2.3 Translation from different Sources

B.10.2.3.1 Signal/Service Translation from different Sources

The example in figure B.47 shows a setup where several sources of information are coming from `signal-service-translation`s. As the source information is accessed using ara::com APIs and the reception semantics is implementing a queue for each input source, the application component is responsible to fetch the data from the various sources and to decide which input data shall be put to the output event *klm*. As the output event *klm* does not have any optional members defined, it is required to provide all three members *k*, *l*, and *m* with valid values for each instance of *klm*.

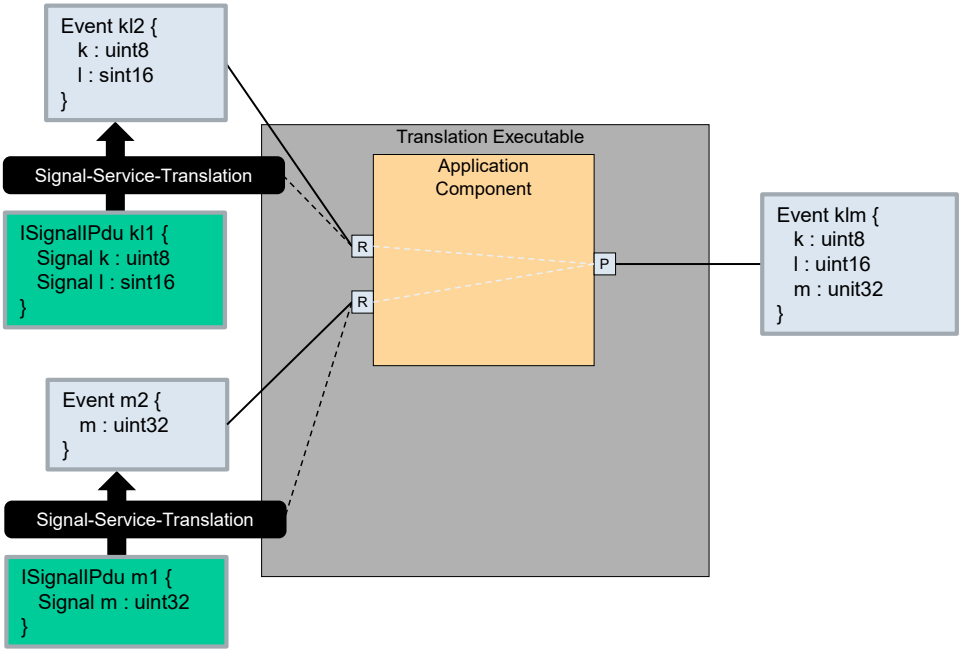


Figure B.47: Example `signal-service-translation` and forwarding from several sources

B.10.2.3.2 ServiceSignalTranslation from different Sources

Figure B.48 shows an example for several sources coming from service-oriented communication. The task of the application component is to fetch the input information for the source events *x1*, *y*, and *z*. The output event *xyz1* does not have any optional members, also the output `ISignalIPdu` *xyz2* does not use any update bits, thus it is required for the application component to provide the full set of information for *a*, *b*, *y*, and *z* when the output event *xyz1* is produced.

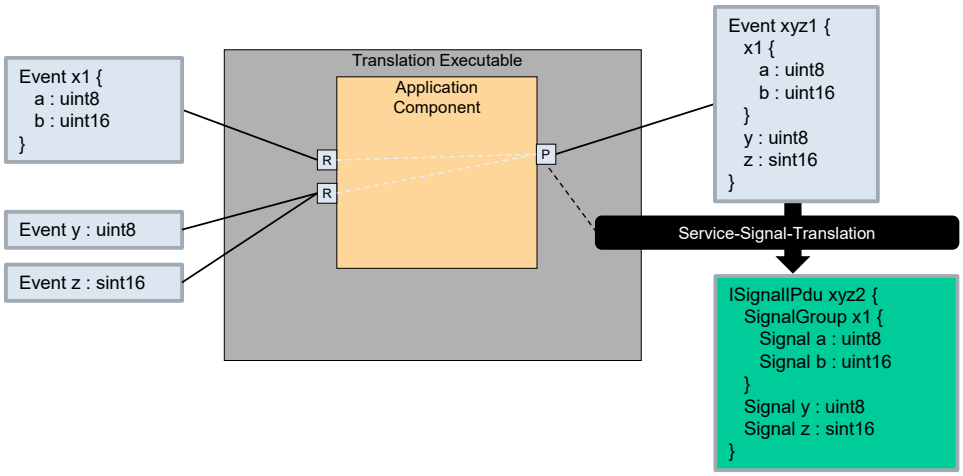


Figure B.48: Example `service-signal-translation` and forwarding from several sources

B.10.2.4 Translation to different Targets

An example of such a setup is illustrated in figure B.49: The ISignalIPdu *xyz* consists of an end-to-end protected ISignalGroup *x1* and two not protected ISignals *y* and *z*.

It is essential for the application component to be able to check the end-to-end status for the event *x1* which holds the end-to-end protected ISignalGroup payload. The event *x1* covers just the end-to-end protected part of the ISignalIPdu and is handled according to the definitions in section 13.9.1.

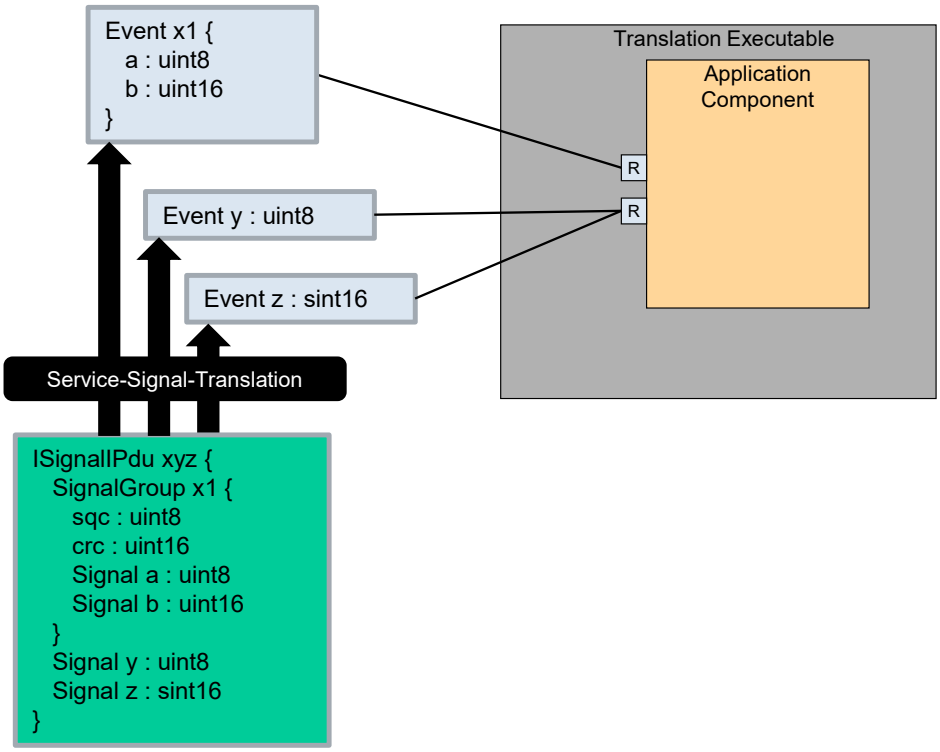


Figure B.49: Example signal-service-translation to several targets

The single events *y* and *z* are accessible individually and there is not end-to-end protection involved in their translation.

B.11 Software Distribution

B.11.1 Dependencies among Software Clusters

The definition of dependencies among SoftwareClusters is exemplified by the sketch in Figure B.50 (where the orange ellipsis represent SoftwareClusterDependencyFormula and the blue ellipsis represent SoftwareClusterDependencyCompareCondition) and a corresponding ARXML formalization, see Listing B.25:

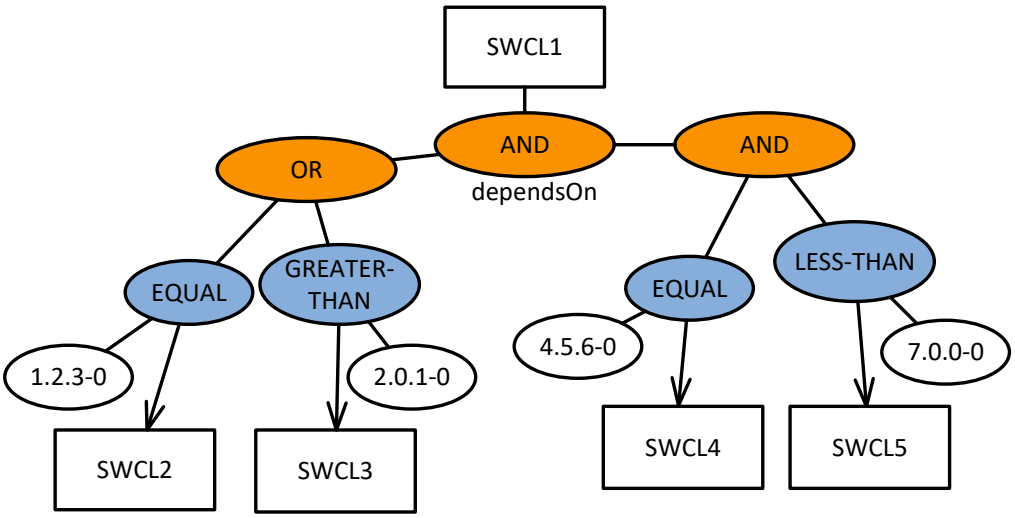


Figure B.50: Example how dependencies among `SoftwareCluster`s can be defined

```
<SOFTWARE-CLUSTER>
  <SHORT-NAME>SWCL1</SHORT-NAME>
  <DEPENDS-ON>
    <OPERATOR>LOGICAL-AND</OPERATOR>
    <PARTS>
      <SOFTWARE-CLUSTER-DEPENDENCY-FORMULA>
        <PARTS>
          <SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
            <COMPARE-TYPE>IS-EQUAL</COMPARE-TYPE>
            <SOFTWARE-CLUSTER-REF DEST="SOFTWARE-CLUSTER">/SoftwareClusters
                /SWCL2</SOFTWARE-CLUSTER-REF>
            <VERSION>1.2.3-0</VERSION>
          </SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
          <SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
            <COMPARE-TYPE>IS-GREATER-THAN</COMPARE-TYPE>
            <SOFTWARE-CLUSTER-REF DEST="SOFTWARE-CLUSTER">/SoftwareClusters
                /SWCL3</SOFTWARE-CLUSTER-REF>
            <VERSION>2.0.1-0</VERSION>
          </SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
        </PARTS>
      </SOFTWARE-CLUSTER-DEPENDENCY-FORMULA>
      <SOFTWARE-CLUSTER-DEPENDENCY-FORMULA>
        <OPERATOR>LOGICAL-AND</OPERATOR>
        <PARTS>
          <SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
            <COMPARE-TYPE>IS-EQUAL</COMPARE-TYPE>
            <SOFTWARE-CLUSTER-REF DEST="SOFTWARE-CLUSTER">/SoftwareClusters
                /SWCL4</SOFTWARE-CLUSTER-REF>
            <VERSION>4.5.6-0</VERSION>
          </SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
          <SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
            <COMPARE-TYPE>IS-LESS-THAN</COMPARE-TYPE>
            <SOFTWARE-CLUSTER-REF DEST="SOFTWARE-CLUSTER">/SoftwareClusters
                /SWCL5</SOFTWARE-CLUSTER-REF>
            <VERSION>7.0.0-0</VERSION>
```

```
          </SOFTWARE-CLUSTER-DEPENDENCY-COMPARE-CONDITION>
        </PARTS>
      </SOFTWARE-CLUSTER-DEPENDENCY-FORMULA>
    </PARTS>
  </DEPENDS-ON>
</SOFTWARE-CLUSTER>
```

Listing B.25: Example for the definition of a dependency between `SoftwareCluster`s

B.11.2 Usage of SoftwarePackageStep

The semantics of the references

- `transfer`
- `process`

shall be explained along a set of examples.

B.11.2.1 Usage of transfer and process

The first example (as depicted in Figure B.51) assumes a scenario where three `SoftwarePackage`s shall be activated and the three `SoftwarePackage`s are transferred in parallel in the context of one `SoftwarePackageStep`.

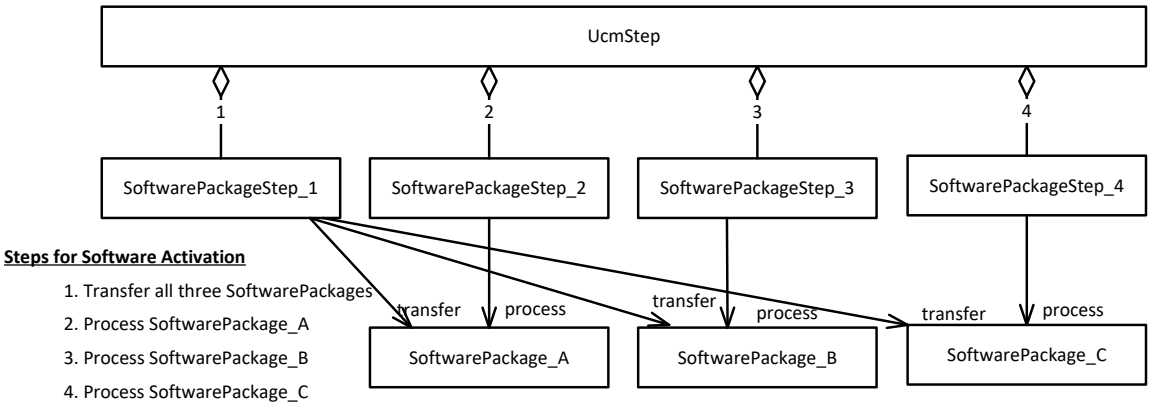


Figure B.51: Example 1 of the configuration of `SoftwarePackageStep`

After that, the three `SoftwarePackage`s are processed one by one in the context of three further `SoftwarePackageStep`s.

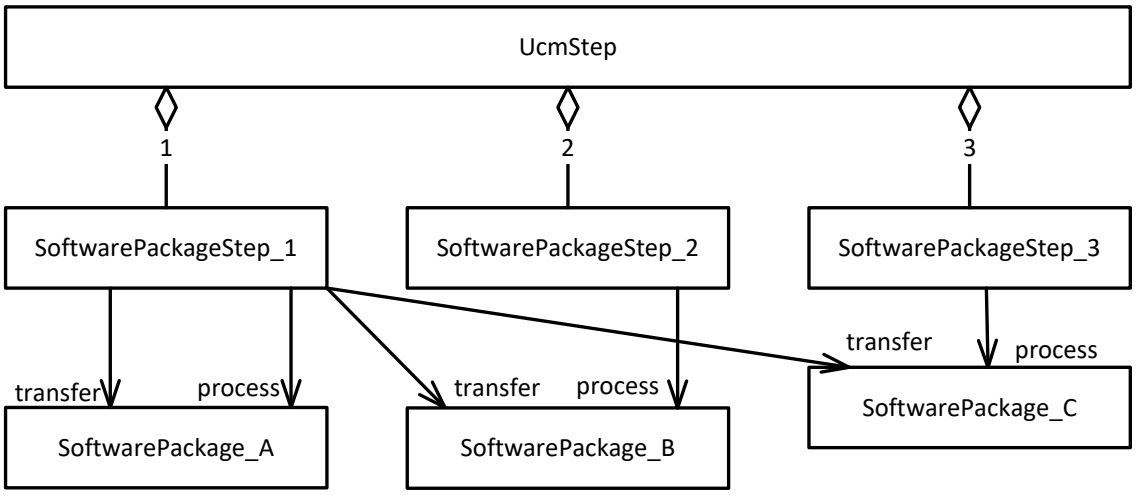


Figure B.52: Example 2 of the configuration of `SoftwarePackageStep`

The ordering of the `SoftwarePackageStep`s in the context of the enclosing `UcmStep` is depicted explicitly by assigning numerical values to the aggregation as well as naming the respective `SoftwarePackageStep` using the same numerical values.

The second example introduces a scenario where the three `SoftwarePackage`s are transferred in the first `SoftwarePackageStep`. One of the `SoftwarePackage`s is additionally referenced in the role `process` and this indicates that the respective `SoftwarePackage` is streamed.

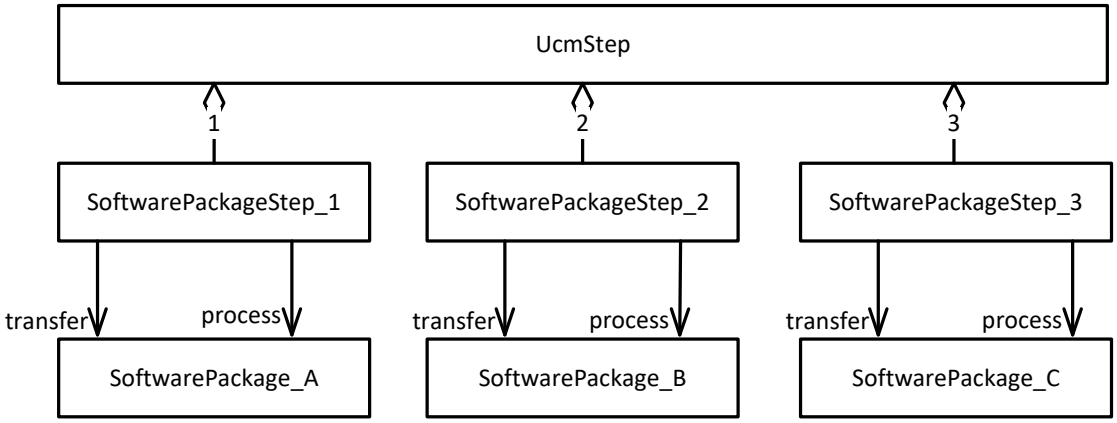
Because the first `SoftwarePackage` is streamed, only three[1] `SoftwarePackageStep`s are needed to model the example.

The rest of the `SoftwarePackage`s are processed in a dedicated `SoftwarePackageStep`. The example is depicted in Figure B.52.

The third example, depicted in Figure B.53, sketches a scenario where three `SoftwarePackage`s are streamed one after the other.

Consequently, this scenario requires the existence of three `SoftwarePackageStep`s that each reference the same `SoftwarePackage` in the roles `transfer` and `process`.

[1] As opposed to four `SoftwarePackageStep`s required to model the first example.



Steps for Software Activation

1. Transfer and process (i.e. stream) SoftwarePackage_A
2. Transfer and process (i.e. stream) SoftwarePackage_B
3. Transfer and process (i.e. stream) SoftwarePackage_C

Figure B.53: Example 3 of the configuration of `SoftwarePackageStep`

C Custom Model Extension

C.1 Overview

The AUTOSAR meta-model is obviously limited to cover the standardized part of AUTOSAR modeling. This means that the modeling stops in some cases although it is clear that the modeling capability provided by the meta-model is not sufficient to create a viable configuration.

One example for this aspect is the meta-class `ProvidedUserDefinedServiceInstance`. AUTOSAR intentionally does nothing but provide this meta-class as a basis for custom additions that configure custom transport for services.

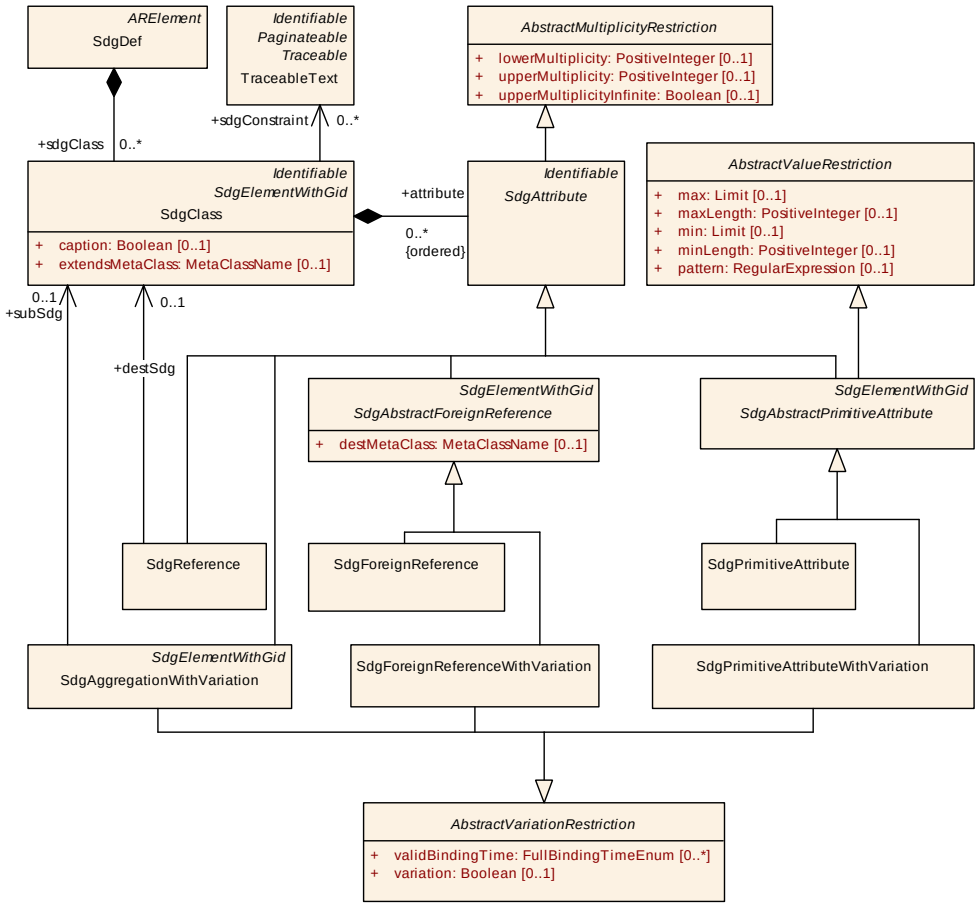


Figure C.1: Modeling of the custom model extensions

The necessary details that are required in order to actually work with a user-defined service instance are not mentioned because there are too many possibilities of how such a `ProvidedUserDefinedServiceInstance` could be modeled that AUTOSAR is simply unable to cover them all.

The typical solution for such a problem is to rely on the definition of *special data groups*, formalized as `Sdg`. However, the direct usage of an `Sdg` with in a project comes with the risk that the `Sdg` is used slightly different in certain parts of the project.

Class	**Sdg**			
Package	M2::MSR::AsamHdo::SpecialData			
Note	Sdg (SpecialDataGroup) is a generic model which can be used to keep arbitrary information which is not explicitly modeled in the meta-model. Sdg can have various contents as defined by sdgContentsType. Special Data should only be used moderately since all elements should be defined in the meta-model. Thereby SDG should be considered as a temporary solution when no explicit model is available. If an sdg Caption is available, it is possible to establish a reference to the sdg structure.			
Base	*ARObject*			
Aggregated by	AdminData.sdg, BuildActionEnvironment.sdg, BuildActionInvocator.sdg, BuildActionIoElement.sdg, File InfoComment.sdg, RptHook.sdg, SdgContents.sdg, VariationPoint.sdg			
Attribute	***Type***	***Mult.***	***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. **Stereotypes:** atpIdentityContributor **Tags:** xml.sequenceOffset=20
sdgContents Type	SdgContents	0..1	aggr	This is the content of the Sdg. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

Table C.1: Sdg

It would therefore be good if there were a way to describe in terms of the AUTOSAR meta-model how a `Sdg` is supposed to be used.

Fortunately, this is possible by means of a corner of the meta-model that had been created for exactly this purpose: `SdgClass` aggregates abstract class `SdgAttribute` that in turn inherits to a bunch of different sub-classes.

Class	**SdgClass**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	An SdgClass specifies the name and structure of the SDG that may be used to store proprietary data in an AUTOSAR model. The SdgClass is similar to an UML stereotype.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgElementWithGid*			
Aggregated by	SdgDef.sdgClass			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
attribute (ordered)	SdgAttribute	*	aggr	Defintion of the structure of the Sdg **Tags:** xml.sequenceOffset=30

▽

△

Class	**SdgClass**			
caption	Boolean	0..1	attr	Specifies if a caption is required. Note: only Sdgs that have a caption can be referenced **Tags:** xml.sequenceOffset=20
extendsMeta Class	MetaClassName	0..1	attr	The AUTOSAR Meta-Class that may be extended by this SdgClass. **Tags:** xml.sequenceOffset=10
sdgConstraint	TraceableText	*	ref	Semantic constraints that restrict the structure of the special data group. **Tags:** xml.sequenceOffset=40

Table C.2: SdgClass

Class	***SdgAttribute*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	Describes the attributes of an Sdg.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*SdgAbstractForeignReference*, *SdgAbstractPrimitiveAttribute*, SdgAggregationWithVariation, Sdg Reference			
Aggregated by	SdgClass.attribute			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table C.3: SdgAttribute

C.2 Custom Attribute Definition

C.2.1 Custom Primitive Attribute Definition

In other words, `SdgClass` and `SdgAttribute` mimic the pattern found in the meta-model itself: meta-classes have attributes of different kinds.

With this mechanism it is possible to extend meta-class `ProvidedUserDefined-ServiceInstance` in order to e.g. add the ability to describe an instance Id. This chapter contains a comprehensive description of how the extension mechanism can be used to implement the instance Id.

The definition starts with an `SdgDef` that aggregates an `SdgClass` with the shortName `ProvidedUserDefinedServiceInstance`. Attribute extendsMetaClass names meta-class `ProvidedUserDefinedServiceInstance` as the subject to extension.

Class	**SdgDef**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	A SdgDef groups several SdgClasses which belong to the same extension. The concept of an SdgDef is similiar to an UML Profile. **Tags:** atp.recommendedPackage=SdgDefs			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sdgClass	SdgClass	*	aggr	The owned sdgClasses which define the structure of the Sdgs **Tags:** xml.namePlural=SDG-CLASSES

Table C.4: SdgDef

The extension itself is modeled by an `SdgPrimitiveAttribute` named instanceId that is mandatory for the implementation of the user-defined service and thus has lower and upper multiplicity set to 1. The supported value interval ranges from 0..4294967295.

Class	**SdgPrimitiveAttribute**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	Describes primitive special data attributes without variation. This class accepts a special data "sd" attribute.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *AbstractValueRestriction*, *Identifiable*, *Multilanguage Referrable*, *Referrable*, *SdgAbstractPrimitiveAttribute*, *SdgAttribute*, *SdgElementWithGid*			
Aggregated by	SdgClass.attribute			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table C.5: SdgPrimitiveAttribute

```
<SDG-DEF>
  <SHORT-NAME>InstanceExtensions</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>ProvidedUserDefinedServiceInstance</SHORT-NAME>
      <GID>acme:instanceExtensions</GID>
      <EXTENDS-META-CLASS>ProvidedUserDefinedServiceInstance</EXTENDS-META-
          CLASS>
      <ATTRIBUTES>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>instanceId</SHORT-NAME>
          <CATEGORY>INTEGER</CATEGORY>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:instanceId</GID>
          <MAX>4294967295</MAX>
          <MIN>0</MIN>
        </SDG-PRIMITIVE-ATTRIBUTE>
        </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
```

```
</SDG-DEF>
```

Listing C.1: Example for the definition of a custom service instance id via `SdgClass`

Please note the definition of `gid` with value "acme:instanceExtensions" on the level of the `SdgPrimitiveAttribute` and the `gid` with value "acme:instanceId" on the level of the `SdgAttribute`.

The usage of the extension is summarized below. Note the usage of the `gid` that reflects the definition in the `SdgClass` and `SdgPrimitiveAttribute`.

Please note further that the definition of the `SdgPrimitiveAttribute` defines the “data type” of the

```
<PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
  <SHORT-NAME>UDSI</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="acme:instanceExtensions">
        <SD GID="acme:instanceId">42</SD>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
```

Listing C.2: Example for the specification of the value of a custom service instance id

C.2.2 Custom Complex Attribute Definition

Other extensions, e.g. using `SdgAggregationWithVariation` can be used to implement the aggregation of a complex attribute (that in turn may own primitive attributes or references).

In this case it makes sense to put the role of the aggregation into the value of `SdgAggregationWithVariation.gid`.

Note that the `SdgAggregationWithVariation` doesn’t aggregate further elements but refers to an `SdgClass` that in turn contains the attributes and references.

Class	**SdgAggregationWithVariation**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	Describes that the Sdg may contain another Sdg. The gid of the nested Sdg is defined by subSdg. Represents ’sdg’.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *AbstractVariationRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgAttribute*, *SdgElementWithGid*			
Aggregated by	SdgClass.attribute			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
subSdg	SdgClass	0..1	ref	Supported sub Sdg Class

Table C.6: SdgAggregationWithVariation

```
<SDG-DEF>
  <SHORT-NAME>DepExt</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>UserDefinedServiceInterfaceDeployment</SHORT-NAME>
      <GID>acme:deploymentExtensions</GID>
      <EXTENDS-META-CLASS>UserDefinedServiceInterfaceDeployment</EXTENDS-
          META-CLASS>
      <ATTRIBUTES>
        <SDG-AGGREGATION-WITH-VARIATION>
          <SHORT-NAME>Version</SHORT-NAME>
          <GID>acme:version</GID>
          <VARIATION>false</VARIATION>
          <SUB-SDG-REF DEST="SDG-CLASS">/CustomME/DepExt/VersionInformation
              </SUB-SDG-REF>
        </SDG-AGGREGATION-WITH-VARIATION>
      </ATTRIBUTES>
    </SDG-CLASS>
    <SDG-CLASS>
      <SHORT-NAME>VersionInformation</SHORT-NAME>
      <GID>acme:deploymentExtensions</GID>
      <ATTRIBUTES>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>MajorVersion</SHORT-NAME>
          <CATEGORY>INTEGER</CATEGORY>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:majorVersion</GID>
          <MAX>4294967295</MAX>
          <MIN>0</MIN>
        </SDG-PRIMITIVE-ATTRIBUTE>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>MinorVersion</SHORT-NAME>
          <CATEGORY>INTEGER</CATEGORY>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:minorVersion</GID>
          <MAX>4294967295</MAX>
          <MIN>0</MIN>
        </SDG-PRIMITIVE-ATTRIBUTE>
      </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing C.3: Example for the definition of a complex version information `SdgClass`

The modeling of the complex aggregation is sketched in Listing C.3. an `SdgClass` with the `gid` set to “acme:deploymentExtensions” defines an attribute that is an `SdgAggregationWithVariation` with the `shortName` “Version“ that has attribute `gid` set to “acme:version”.

This means that the `SdgAggregationWithVariation` with the `shortName` “Version“ is aggregated in the role “version” (derived from the value of the `gid`) at the `SdgClass`.

The SdgAggregationWithVariation in turn references another SdgClass with the shortName set to “VersionInformation”. This SdgClass contains two SdgPrimitiveAttributes for carrying the optional major version and the minor version. The value range of both version information is from 0 to 4294967295.

The definition side of this example is sketched in Listing C.3, the respective value side is provided in Listing C.4.

```
<USER-DEFINED-SERVICE-INTERFACE-DEPLOYMENT>
  <SHORT-NAME>UDSID</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="acme:deploymentExtensions">
        <SDG GID="acme:version">
          <SD GID="acme:majorVersion">1</SD>
          <SD GID="acme:minorVersion">0</SD>
        </SDG>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</USER-DEFINED-SERVICE-INTERFACE-DEPLOYMENT>
```

Listing C.4: Example for the specification of the value of a custom version information

C.3 Custom Foreign Reference Definition

Another aspect of custom modeling is the creation of references to meta-classes derived from Referrable. For this purpose, the meta-class SdgForeignReference resp. SdgForeignReferenceWithVariation.

Class	**SdgForeignReference**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	A reference without variation support that can point to any referrable object in an AUTOSAR Model. This class accepts the special data "Sdx" reference.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgAbstractForeignReference*, *SdgAttribute*, *SdgElementWithGid*			
Aggregated by	SdgClass.attribute			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table C.7: SdgForeignReference

Class	**SdgForeignReferenceWithVariation**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef
Note	A reference with variation support that can point to any referrable object in an AUTOSAR Model. This class accepts the special data "Sdxf" reference.
Base	*ARObject*, *AbstractMultiplicityRestriction*, *AbstractVariationRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgAbstractForeignReference*, *SdgAttribute*, *SdgElementWithGid*

▽

△

Class	SdgForeignReferenceWithVariation			
Aggregated by	SdgClass.attribute			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table C.8: SdgForeignReferenceWithVariation

The particle “foreign” in the name of these classes represents a hint that the reference’s target is situated outside the custom modeling “bubble” depicted in Figure C.1[1].

The creation of a reference to another meta-class obviously implies the specification of the role in which the reference shall be used.

In the case of the `SdgForeignReference` the role of the reference may be defined in the attribute `gid`, contributed by the inheritance from meta-class `SdgElementWithGid`.

Class	SdgElementWithGid (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	A special data group element with gid is an abstract element that shall have a name (gid, "Generic Identifier").			
Base	*ARObject*			
Subclasses	*SdgAbstractForeignReference*, *SdgAbstractPrimitiveAttribute*, SdgAggregationWithVariation, SdgClass			
Attribute	Type	Mult.	Kind	Note
gid	NameToken	0..1	attr	Specifies the name that identifies the element.

Table C.9: SdgElementWithGid

In true AUTOSAR fashion, a reference should always announce the intended meta-class to which it refers to. In the case of the `SdgForeignReference`, this information can be explicitly provided by means of the attribute `destMetaClass`, inherited from `SdgAbstractForeignReference`.

Class	SdgAbstractForeignReference (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	An abstract reference that can point to any referrable object in an AUTOSAR Model.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgAttribute*, *SdgElementWithGid*			
Subclasses	SdgForeignReference, SdgForeignReferenceWithVariation			
Aggregated by	SdgClass.attribute			
Attribute	Type	Mult.	Kind	Note
destMetaClass	MetaClassName	0..1	attr	specifies the destination meta-class of the reference.

Table C.10: SdgAbstractForeignReference

The example created for the explanation of `SdgForeignReference` assumes that a `ProvidedUserDefinedServiceInstance` wants to re-use an existing configuration for SOME/IP SD.

[1] Of course, this naming is also a hat tip to the meta-class `EcucForeignReferenceDef`

For this purpose, a custom extension of the meta-class ProvidedUserDefinedServiceInstance with the ability to refer to at most one SomeipSdServerServiceInstanceConfig is created.

In particular, the extension consists of the definition of aSdgForeignReference with the gid set to the value “acme:sdServerTimeConfig” as the representation of the role of the reference is created.

The nature of the SdgForeignReference defined in this example is determined by means of the value of attribute destMetaClass, in this case “SomeipSdServerServiceInstanceConfig”.

The custom definition of the SdgForeignReference is sketched in Listing C.5.

```
<SDG-DEF>
  <SHORT-NAME>InstanceExtensions</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>ProvidedUserDefinedServiceInstance</SHORT-NAME>
      <GID>acme:instanceExtensions</GID>
      <EXTENDS-META-CLASS>ProvidedUserDefinedServiceInstance</EXTENDS-META-
          CLASS>
      <ATTRIBUTES>
        <SDG-FOREIGN-REFERENCE>
          <SHORT-NAME>SdServerTimeConfig</SHORT-NAME>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:sdServerTimeConfig</GID>
          <DEST-META-CLASS>SomeipSdServerServiceInstanceConfig</DEST-META-
              CLASS>
        </SDG-FOREIGN-REFERENCE>
      </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing C.5: Example for the specification of a custom foreign reference

The value side of the example in Listing C.5 can be found in Listing C.6. Note that the formalization of the reference to the respective SomeipSdServerServiceInstanceConfig is implemented by means of the reference SdgContents.sdx, wrapped into an Sdg where the attribute gid is set to the role of the reference, in this case “acme:sdServerTimeConfig”.

Class	<<atpMixed>> **SdgContents**			
Package	M2::MSR::AsamHdo::SpecialData			
Note	This meta-class represents the possible contents of a special data group. It can be an arbitrary mix of references, of primitive special data and nested special data groups.			
Base	*ARObject*			
Aggregated by	Sdg.sdgContentsType			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	<<atpMixed>> **SdgContents**			
sd	Sd	0..1	aggr	This is one particular special data element. **Tags:** xml.sequenceOffset=40
sdf	Sdf	0..1	aggr	This is one particular special data element. **Tags:** xml.sequenceOffset=60
sdg	Sdg	0..1	aggr	This aggregation allows to express nested special data groups. By this, any structure can be represented in SpeicalData. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=sdg.sdgCaption.shortName, sdg.variation Point.shortLabel vh.latestBindingTime=postBuild xml.sequenceOffset=50
sdx	Referrable	0..1	ref	Reference to any identifiable element. This allows to use Sdg even to establish arbitrary relationships.
sdxf	Referrable	0..1	ref	Additional reference with variant support. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=sdxf.referrable, sdxf.variationPoint.short Label vh.latestBindingTime=postBuild

Table C.11: SdgContents

```
<PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
  <SHORT-NAME>UDSI</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="acme:instanceExtensions">
        <SDG GID="acme:sdServerTimeConfig">
          <SDX-REF DEST="SOMEIP-SD-SERVER-SERVICE-INSTANCE-CONFIG">/SD/
              MyConf</SDX-REF>
        </SDG>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
```

Listing C.6: Example for the specification of the value of a custom foreign reference

C.4 Custom Subclass Configuration

Using the mechanism of custom model extensions it is also possible to mimic the creation of custom “subclasses”.

A possible candidate for the creation of a “subclass” outside the explicitly formalized meta-model could be the `ProvidedUserDefinedServiceInstance`. The intention could, for example, be to provide a configuration for an IPC-specific “subclass”.

A simple example of how an extension that defines a “subclass” might look like is sketched in Listing C.7.

The specific approach in this case is to define an SdgClass to extend ProvidedUserDefinedServiceInstance and this extension refers - via the aggregation of an SdgAggregationWithVariation - to another SdgClass (in this case with the shortName “IpcProvidedServiceInstanceAttributes”) where the attributes of the custom “subclass” are defined.

On the value side, the definition of the attribute values of the “subclass” are defined in a quite straight-forward manner (see Listing C.8).

Please note that the value side in this example does not reflect the definition side in terms of the usage of values of attribute gid 100%. The existence of the SdgClass with shortName “IpcProvidedServiceInstanceAttributes” is not represented on the value side.

But, on the other hand, the intended semantics of defining an instance identifier in the context of the custom model of a “ProvidedIpcServiceInstance” can be conveyed perfectly without it.

```
<SDG-DEF>
  <SHORT-NAME>SubclassConfiguration</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>ProvidedUserDefinedServiceInstance</SHORT-NAME>
      <GID>acme:providedIpcServiceInstance</GID>
      <EXTENDS-META-CLASS>ProvidedUserDefinedServiceInstance</EXTENDS-META-
          CLASS>
      <ATTRIBUTES>
        <SDG-AGGREGATION-WITH-VARIATION>
          <SHORT-NAME>IpcSubclass</SHORT-NAME>
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:attributes</GID>
          <VARIATION>false</VARIATION>
          <SUB-SDG-REF DEST="SDG-CLASS">/CustomME/SubclassConfiguration/
              IpcProvidedServiceInstanceAttributes</SUB-SDG-REF>
        </SDG-AGGREGATION-WITH-VARIATION>
      </ATTRIBUTES>
    </SDG-CLASS>
    <SDG-CLASS>
      <SHORT-NAME>IpcProvidedServiceInstanceAttributes</SHORT-NAME>
      <GID>acme:ipcProvidedServiceInstanceAttributes</GID>
      <ATTRIBUTES>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>instanceId</SHORT-NAME>
          <CATEGORY>INTEGER</CATEGORY>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <GID>acme:instanceId</GID>
          <MAX>65535</MAX>
          <MIN>0</MIN>
        </SDG-PRIMITIVE-ATTRIBUTE>
      </ATTRIBUTES>
    </SDG-CLASS>
```

```
  </SDG-CLASSES>
```

Listing C.7: Example for the specification of a custom “subclass”

And since the point of the whole approach is the creation of a custom modeling anyway, the only relevant condition for the validity of such modeling is that the affected AUTOSAR tools know how to properly parse and interpret the resulting model.

```
<PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
  <SHORT-NAME>ipcProvidedSI</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="acme:providedIpcServiceInstance">
        <SDG GID="acme:attributes">
          <SD GID="acme:instanceId">3485</SD>
        </SDG>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
```

Listing C.8: Example for the specification of a service instance in an IPC “subclass”

C.5 Custom Constraints

Another aspect that can be solved by means of model customizations is the definition of model constraints that go beyond the potentially existing constraints formulated in AUTOSAR.

In the example sketched in Listing C.9 a constraint is formulated for attribute `Executable.minimumTimerGranularity`.

```
<SDG-DEF>
  <SHORT-NAME>ModelConstraint</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>Executable</SHORT-NAME>
      <GID>acme:executableExtensions</GID>
      <EXTENDS-META-CLASS>Executable</EXTENDS-META-CLASS>
      <ATTRIBUTES>
        <SDG-PRIMITIVE-ATTRIBUTE>
          <SHORT-NAME>minimumTimerGranularity</SHORT-NAME>
          <CATEGORY>FLOAT</CATEGORY>
          <LOWER-MULTIPLICITY>1</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <MAX>0.5</MAX>
          <MIN>0.001</MIN>
        </SDG-PRIMITIVE-ATTRIBUTE>
      </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing C.9: Example for the specification of a custom model constraint

On the one hand, the example expresses the expectation that the attribute Executable.minimumTimerGranularity shall exist and the second aspect of the constraint is that it shall have a value that's between 0.001 and 0.5.

C.6 Definition of Reference from SdgClass to SdgClass

Another case that could be relevant for a custom model extension is the creation of a reference between two Sdg elements on the value side.

Semi-formal support for this scenario can be defined by means of the definition of two SdgClass elements where one defines a reference to the other.

A caveat applies. Of course, two SdgClass elements could reference each other by means of their shortName-paths (because they are derived from Identifiable).

But that's not the point, the intended reference on the value side shall exist from one Sdg (that corresponds to one of the SdgClass elements on the definition side) to another Sdg (which corresponds to the other SdgClass element on the definition side).

Sdg itself is not derived from Referrable and therefore does not have a shortName that could be used for reference building purposes.

In order to support the creation of a reference from one Sdg to another a mechanism was created using a reference from an Sdg to an SdgCaption in the role sdx.

In other words, an Sdg may aggregate an SdgCaption in the role sdgCaption. And if it does, it becomes (by extension) a valid target of a reference to this SdgCaption.

Class	**SdgCaption**			
Package	M2::MSR::AsamHdo::SpecialData			
Note	This meta-class represents the caption of a special data group. This allows to have some parts of special data as identifiable.			
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	Sdg.sdgCaption			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the special data 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 special data in question.

Table C.12: SdgCaption

Therefore, the SdgClass that represents the Sdg on the target side of the reference shall define attribute sdgCaption and set it to true.

Class	**SdgReference**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::SpecialDataDef			
Note	Describes an attribute of a SdgClass which is used on the definition side to model a reference from one Sdg to another Sdg on the value side.			
Base	*ARObject*, *AbstractMultiplicityRestriction*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SdgAttribute*			
Aggregated by	SdgClass.attribute			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
destSdg	SdgClass	0..1	ref	Refers to a SdgClass which is used on the definition side to model the destination type of the referenced Sdg. On the value side the reference is realized by means of the originating Sdg defining an sdgx attribute which refers to the sdgCaption of the referenced Sdg.

Table C.13: SdgReference

The `SdgClass` on the source side of the reference shall define an `attribute` that is actually an `SdgReference` to the `SdgClass` that represents the target `Sdg` on the definition side by means of the reference in the role `destSdg`.

As a first step, the following model fragment defines a custom model extension of an IPC channel[2] (note that attribute `caption` is set to `true`):

```
<SDG-DEF>
  <SHORT-NAME>IpcChannels</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>IpcChannel</SHORT-NAME>
      <GID>acme:ipcChannels</GID>
      <CAPTION>true</CAPTION>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing C.10: Example for the specification of a custom model extension that represents an IPC channel

This IPC channel (named “MyIpcChannel”) shall be referenced from the definition of an `SdgClass` that is supposed to extend the `ProvidedUserDefinedServiceInstance`:

```
<SDG-DEF>
  <SHORT-NAME>InstanceExtensions</SHORT-NAME>
  <SDG-CLASSES>
    <SDG-CLASS>
      <SHORT-NAME>ProvidedUserDefinedServiceInstance</SHORT-NAME>
      <GID>acme:instanceExtensions</GID>
      <EXTENDS-META-CLASS>ProvidedUserDefinedServiceInstance</EXTENDS-META-
          CLASS>
      <CAPTION>false</CAPTION>
      <ATTRIBUTES>
        <SDG-REFERENCE>
          <SHORT-NAME>IpcChannelRef</SHORT-NAME>
```

[2]Please note that this example represents an incomplete model that does not care about details of the actual configuration of the hypothetical IPC Channel and is entirely focused on the referencing topic.

```
          <LOWER-MULTIPLICITY>0</LOWER-MULTIPLICITY>
          <UPPER-MULTIPLICITY>1</UPPER-MULTIPLICITY>
          <DEST-SDG-REF DEST="SDG-CLASS">/SdgClasses/IpcChannels/IpcChannel
             </DEST-SDG-REF>
        </SDG-REFERENCE>
      </ATTRIBUTES>
    </SDG-CLASS>
  </SDG-CLASSES>
</SDG-DEF>
```

Listing C.11: Example for the specification of a custom model extension for a `ProvidedUserDefinedServiceInstance` with a reference to an IPC channel

The model on the value side where the `ProvidedUserDefinedServiceInstance` references the IPC channel is sketched by the following model fragment:

```
<AR-PACKAGE>
  <SHORT-NAME>ServiceInstances</SHORT-NAME>
  <ELEMENTS>
    <PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
      <SHORT-NAME>MyService</SHORT-NAME>
      <ADMIN-DATA>
        <SDGS>
          <SDG GID="acme:ipcChannelRef">
            <SDX-REF DEST="SDG-CAPTION">/IpcChannels/IpcChannel1</SDX-REF>
          </SDG>
        </SDGS>
      </ADMIN-DATA>
    </PROVIDED-USER-DEFINED-SERVICE-INSTANCE>
  </ELEMENTS>
</AR-PACKAGE>
<AR-PACKAGE>
  <SHORT-NAME>IpcChannels</SHORT-NAME>
  <ADMIN-DATA>
    <SDGS>
      <SDG GID="acme:ipcChannel1">
        <SDG-CAPTION>
          <SHORT-NAME>IpcChannel1</SHORT-NAME>
        </SDG-CAPTION>
      </SDG>
    </SDGS>
  </ADMIN-DATA>
</AR-PACKAGE>
```

Listing C.12: Example for the specification of reference between `Sdg`

D 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.

D.1 Reference to a DataPrototype in a PortInterface

D.1.1 Reference to the inside of an ApplicationDataType

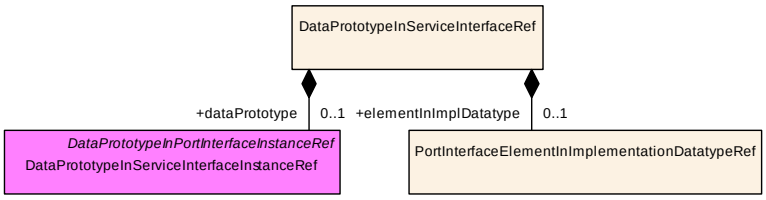


Figure D.1: Modeling of DataPrototypeInServiceInterfaceRef

Class	**DataPrototypeInServiceInterfaceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInPortInterfaceInstanceRef			
Note	This meta-class represents the ability to refer to an AUTOSAR DataPrototype in the context of a Service Interface.			
Base	*ARObject*			
Aggregated by	SignalBasedEventElementToISignalTriggeringMapping.dataPrototypeInServiceInterfaceRef, SignalBasedFieldToISignalTriggeringMapping.dataPrototypeInServiceInterfaceRef, SomeipDataPrototypeTransformationProps.dataPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataPrototype	DataPrototype	0..1	iref	This element represents the ability to: • refer to a DataPrototype in the context of a Service Interface. • refer to the internal structure of a DataPrototype in which is typed by an ApplicationDatatype the context of a ServiceInterface. **InstanceRef implemented by:** DataPrototypeInServiceInterfaceInstanceRef
elementInImplDatatype	PortInterfaceElementInImplementationDatatypeRef	0..1	aggr	This element represents the ability to refer to the internal structure of an AutosarDataPrototype which is typed by an ImplementationDatatype in the context of a Service Interface.

Table D.1: DataPrototypeInServiceInterfaceRef

Please note that the modeling of the reference to a DataPrototype in the context of a PortInterface can only be executed as the abstract template for concrete specializations because the abstract meta-class PortInterface does not aggregate a DataPrototype directly.

The abstract modeling of meta-class DataPrototypeInPortInterfaceInstanceRef is depicted in Figure D.2.

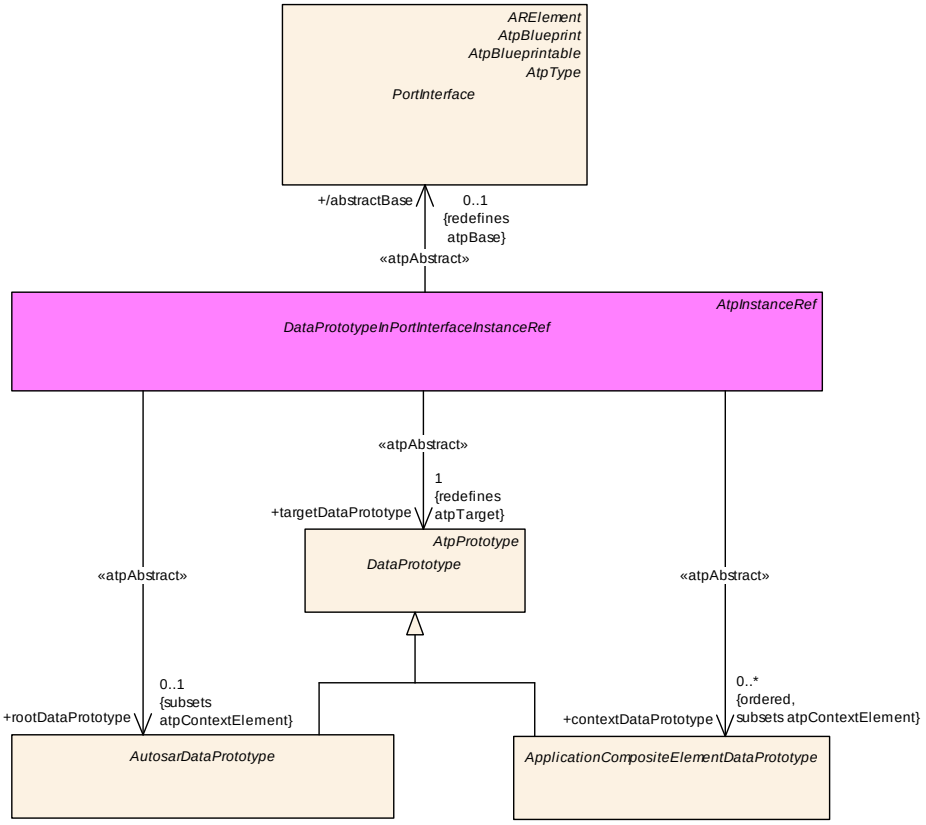


Figure D.2: Modeling of DataPrototypeInPortInterfaceInstanceRef

Class	***DataPrototypeInPortInterfaceInstanceRef*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer::InstanceRef			
Note	This meta-class represents the ability to: • refer to a DataPrototype in the context of a PortInterface. • refer to the internal structure of a DataPrototype which is typed by an ApplicationDatatype in the context of a PortInterface.			
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	DataPrototypeInServiceInterfaceInstanceRef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
abstractBase	PortInterface	0..1	ref	**Stereotypes:** atpAbstract
contextData Prototype (ordered)	ApplicationComposite ElementDataPrototype	*	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=20
rootData Prototype	AutosarDataPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=10
targetData Prototype	DataPrototype	1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=30

Table D.2: DataPrototypeInPortInterfaceInstanceRef

The concrete specialization for the aggregation of a DataPrototype in the concrete ServiceInterface is depicted in Figure D.3.

The meta-class DataPrototypeInServiceInterfaceInstanceRef inherits from DataPrototypeInPortInterfaceInstanceRef.

The individual references modeled in the context of DataPrototypeInServiceInterfaceInstanceRef specialize the abstract structure defined in the context of DataPrototypeInPortInterfaceInstanceRef

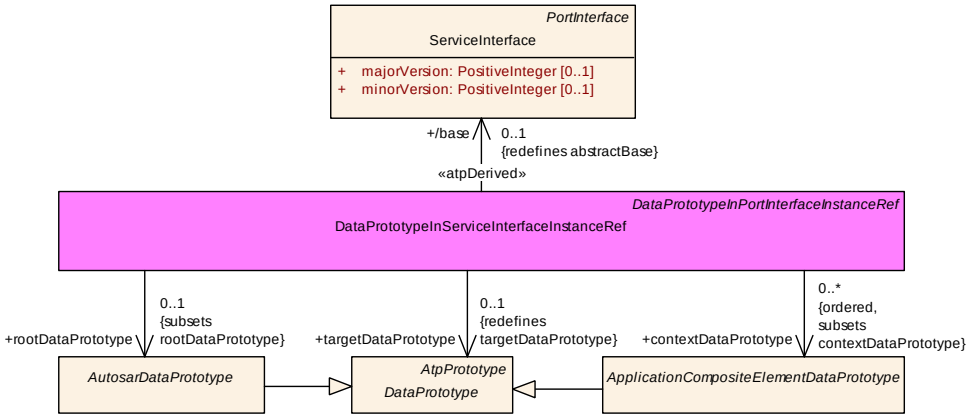


Figure D.3: Modeling of DataPrototypeInServiceInterfaceInstanceRef

Class	**DataPrototypeInServiceInterfaceInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInPortInterfaceInstanceRef			
Note				
Base	*ARObject*, *AtpInstanceRef*, *DataPrototypeInPortInterfaceInstanceRef*			
Aggregated by	DataPrototypeInPortInterfaceRef.dataPrototypeInServiceInterface, DataPrototypeInServiceInterfaceRef.dataPrototype, SignalBasedFireAndForgetMethodToISignalTriggeringMapping.dataPrototypeInMethodArgumentInstanceRef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	ServiceInterface	0..1	ref	**Stereotypes:** atpDerived
contextData Prototype (ordered)	ApplicationComposite ElementDataPrototype	*	ref	**Tags:** xml.sequenceOffset=20
rootData Prototype	AutosarDataPrototype	0..1	ref	**Tags:** xml.sequenceOffset=10
targetData Prototype	DataPrototype	0..1	ref	**Tags:** xml.sequenceOffset=30

Table D.3: DataPrototypeInServiceInterfaceInstanceRef

D.1.2 Reference to the inside of a CppImplementationDataType

Please note that the modeling of instanceRef-like references into the internals of a CppImplementationDataType differs from the way how internals of an ImplementationDataType could be referenced.

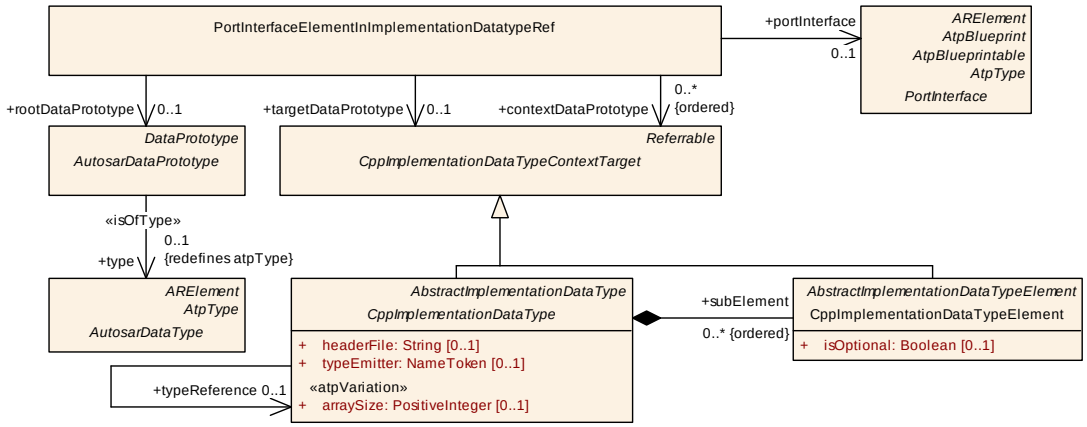


Figure D.4: Modeling of `PortInterfaceElementInImplementationDatatypeRef`

In particular, references to context elements can be directed to `ImplementationDataTypeElement` because both arrays and structures are modeled by means of `ImplementationDataTypeElement`.

This approach has changed with the advent of `CppImplementationDataType` and therefore the same approach is not possible for `CppImplementationDataTypeElement`.

In the case of `CppImplementationDataType`, both `CppImplementationDataTypeElement` and `CppImplementationDataType` can become the target of a context reference. And since the context reference is supposed to be ordered it is simply not possible to straight up model two context references, one for `CppImplementationDataType` and one for `CppImplementationDataTypeElement`.

Instead, it is necessary to introduce an abstract base class named `CppImplementationDataTypeContextTarget` for both `CppImplementationDataType` and `CppImplementationDataTypeElement` and then **direct context references at the abstract base class**.

Class	**PortInterfaceElementInImplementationDatatypeRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General			
Note	This meta-class represents the ability to refer to the internal structure of an AutosarDataPrototype which is typed by an implementationDatatype in the context of a PortInterface. In other words, this meta-class shall not be used to model a reference to the **AutosarDataPrototype as a target itself**, **even** if the AutosarDataPrototype is typed by an ImplementationDataType **and even** if that ImplementationDataType represents a composite data type.			
Base	*ARObject*			
Aggregated by	DataPrototypeInServiceInterfaceRef.elementInImplDatatype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contextData Prototype (ordered)	CppImplementation DataTypeContextTarget	*	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.
portInterface	PortInterface	0..1	ref	This is the PortInterface that contains the rootData Prototype.

△

Class	**PortInterfaceElementInImplementationDatatypeRef**			
rootData Prototype	AutosarDataPrototype	0..1	ref	This rootDataPrototype defines the AutosarDataPrototype in which the target can be found.
targetData Prototype	CppImplementation DataTypeContextTarget	0..1	ref	This is the target reference to a subElement that is defined inside of the rootDataPrototype.

Table D.4: PortInterfaceElementInImplementationDatatypeRef

Class	***CppImplementationDataTypeContextTarget*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class has the ability to serve as the context in instanceRef-like modeling for Cpp ImplementationDataType and CppImplementationDataTypeElement			
Base	*ARObject*, *Referrable*			
Subclasses	*CppImplementationDataType*, CppImplementationDataTypeElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table D.5: CppImplementationDataTypeContextTarget

D.2 Reference to a AutosarDataPrototype in an Executable

The creation of the meta-model for creating a reference to an `AutosarDataPrototype` in the context of an `Executable` is executed in a two-step approach where first an abstract structure of the reference is created.

The abstract structure is the basis for the refinement with respect to specific roles of `AutosarDataPrototype`s.

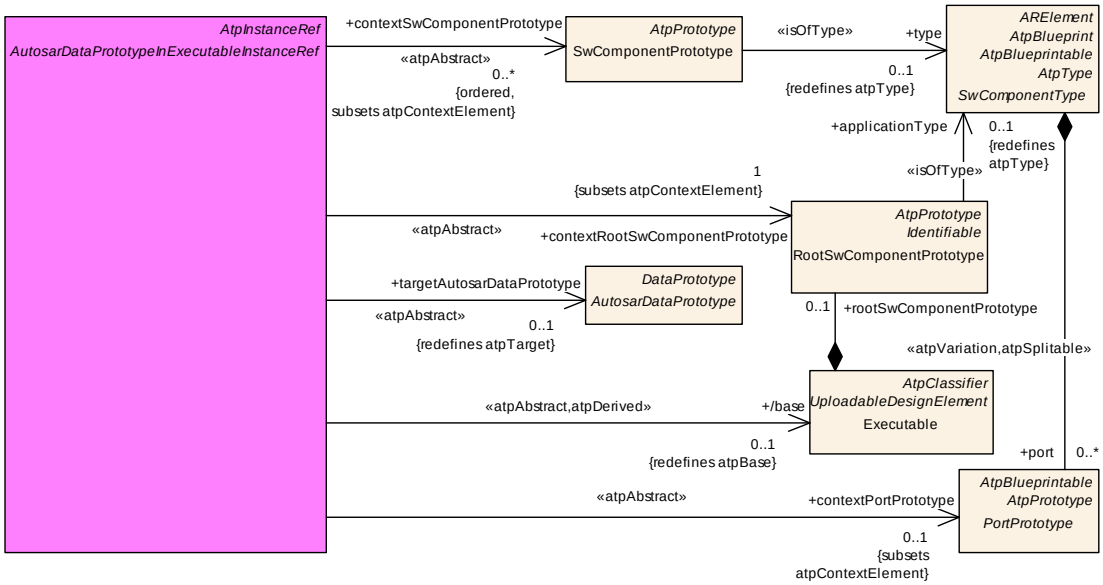


Figure D.5: Modeling of abstract `AutosarDataPrototypeInExecutableInstanceRef`

Class	***AutosarDataPrototypeInExecutableInstanceRef*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note				
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	EventInExecutableInstanceRef, FieldInExecutableInstanceRef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpAbstract; atpDerived
contextPort Prototype	PortPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=10
contextSw Component Prototype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=20
targetAutosar DataPrototype	AutosarDataPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=40

Table D.6: AutosarDataPrototypeInExecutableInstanceRef

Two specializations of AutosarDataPrototypeInExecutableInstanceRef exist:

- EventInExecutableInstanceRef
- FieldInExecutableInstanceRef

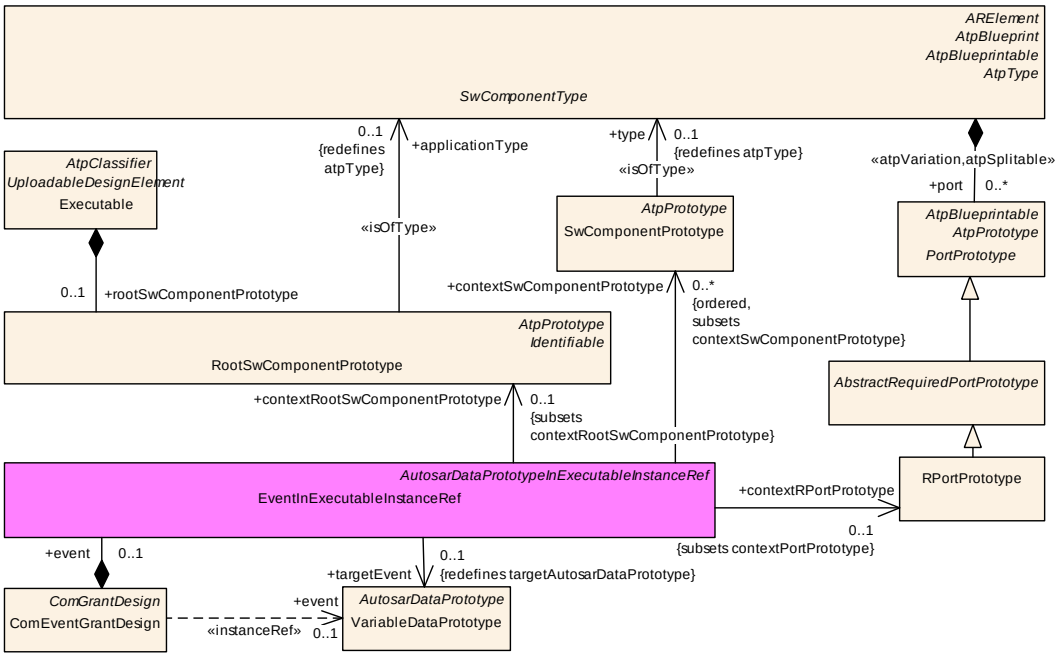


Figure D.6: Modeling of concrete EventInExecutableInstanceRef derived from AutosarDataPrototypeInExecutableInstanceRef

Class	**EventInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note				
Base	*ARObject*, *AtpInstanceRef*, *AutosarDataPrototypeInExecutableInstanceRef*			
Aggregated by	ComEventGrantDesign.event			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Tags:** xml.sequenceOffset=10
contextRPort Prototype	RPortPrototype	0..1	ref	**Tags:** xml.sequenceOffset=30
contextSw Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** xml.sequenceOffset=20
targetEvent	VariableDataPrototype	0..1	ref	**Tags:** xml.sequenceOffset=40

Table D.7: EventInExecutableInstanceRef

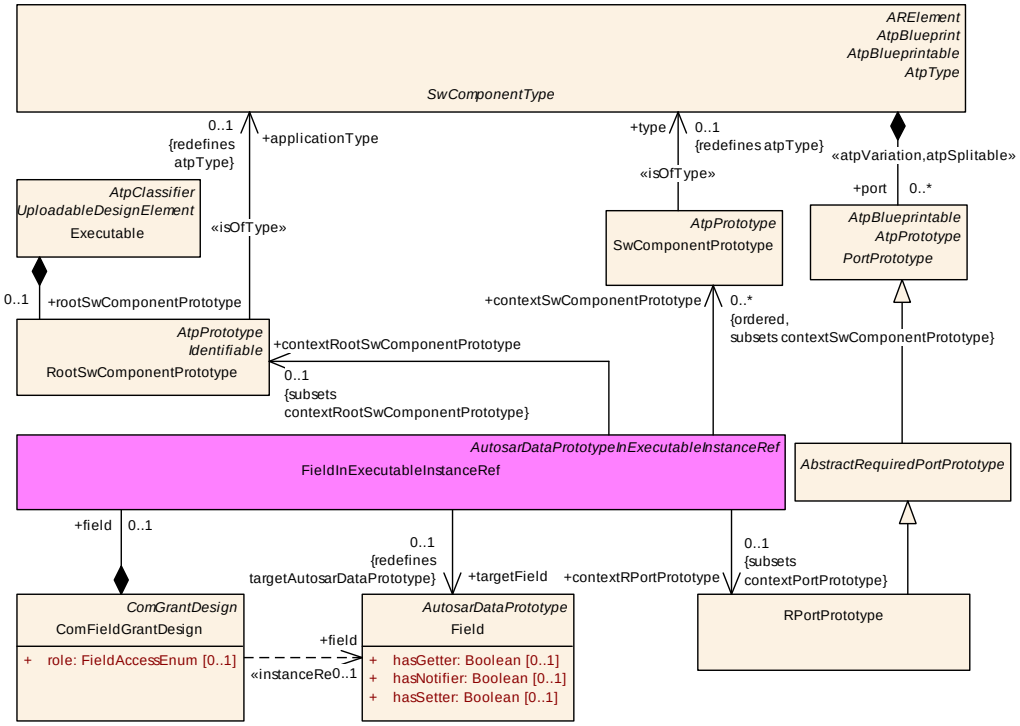


Figure D.7: Modeling of concrete `FieldInExecutableInstanceRef` derived from `AutosarDataPrototypeInExecutableInstanceRef`

Class	**FieldInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note				
Base	*ARObject*, *AtpInstanceRef*, *AutosarDataPrototypeInExecutableInstanceRef*			
Aggregated by	ComFieldGrantDesign.field			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

△

Class	**FieldInExecutableInstanceRef**			
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Tags:** xml.sequenceOffset=10
contextRPort Prototype	RPortPrototype	0..1	ref	**Tags:** xml.sequenceOffset=30
contextSw Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** xml.sequenceOffset=20
targetField	Field	0..1	ref	**Tags:** xml.sequenceOffset=40

Table D.8: FieldInExecutableInstanceRef

D.3 Reference to a PortPrototype in an Executable

The creation of the meta-model for creating a reference to a `PortPrototype` in the context of an `Executable` is executed in a two-step approach where first an abstract structure of the reference is created.

The abstract structure is the basis for the refinement with respect to specific roles of `PortPrototype`s.

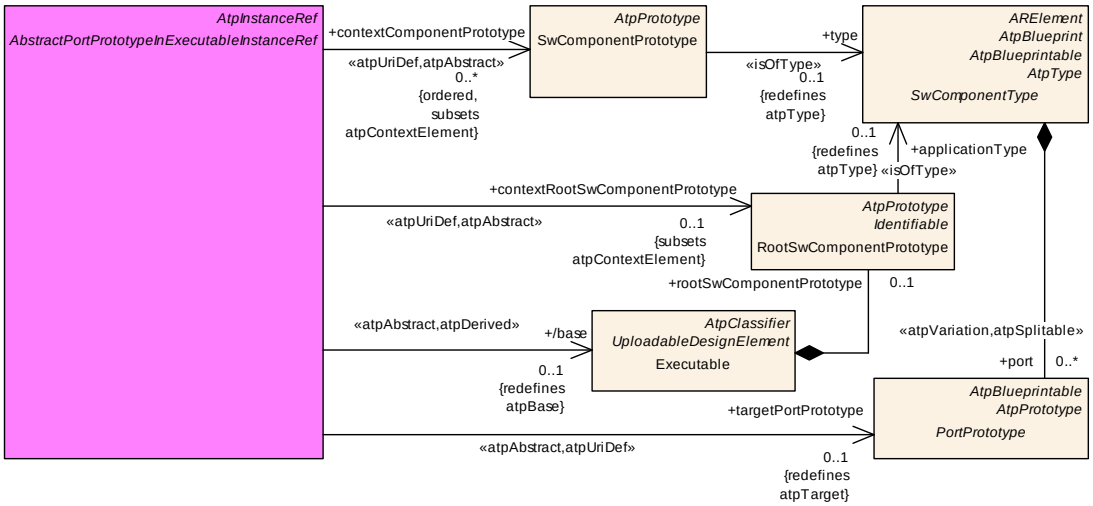


Figure D.8: Modeling of abstract `AbstractPortPrototypeInExecutableInstanceRef`

Class	***AbstractPortPrototypeInExecutableInstanceRef*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef
Note	
Base	*ARObject*, *AtpInstanceRef*
Subclasses	PPortPrototypeInExecutableInstanceRef, PortPrototypeInExecutableInstanceRef, RPortPrototypeIn ExecutableInstanceRef

▽

△

Class	***AbstractPortPrototypeInExecutableInstanceRef*** (abstract)			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpAbstract; atpDerived
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpAbstract; atpUriDef **Tags:** xml.sequenceOffset=20
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Stereotypes:** atpAbstract; atpUriDef **Tags:** xml.sequenceOffset=10
targetPort Prototype	PortPrototype	0..1	ref	**Stereotypes:** atpAbstract; atpUriDef **Tags:** xml.sequenceOffset=30

Table D.9: AbstractPortPrototypeInExecutableInstanceRef

Three specializations of AbstractPortPrototypeInExecutableInstanceRef exist:

- PPortPrototypeInExecutableInstanceRef
- RPortPrototypeInExecutableInstanceRef
- PortPrototypeInExecutableInstanceRef

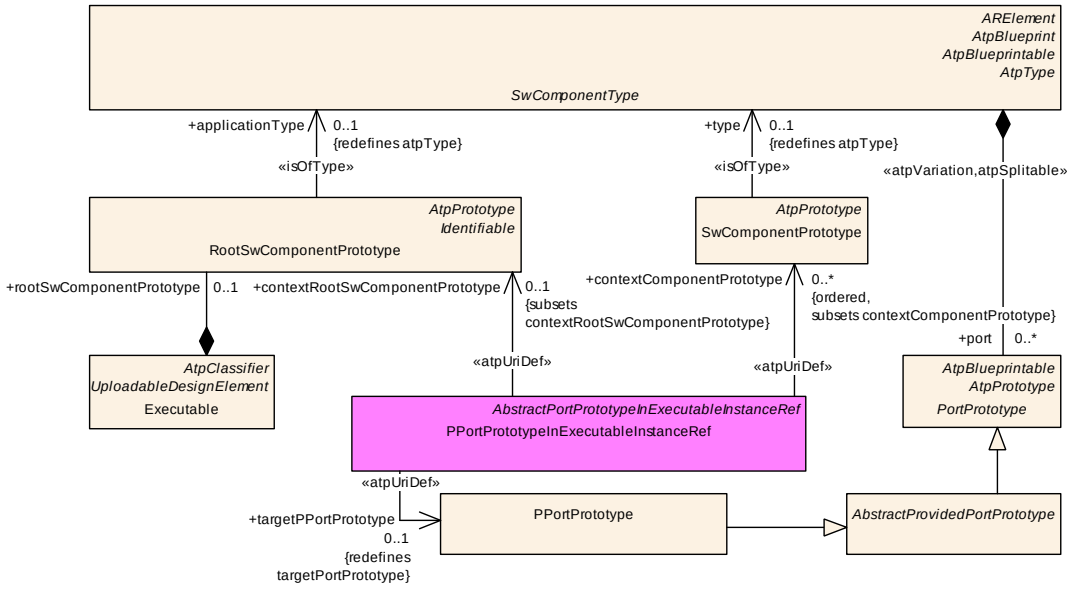


Figure D.9: Modeling of concrete PPortPrototypeInExecutableInstanceRef derived from AbstractPortPrototypeInExecutableInstanceRef

Class	**PPortPrototypeInExecutableInstanceRef**
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef
Note	
Base	*ARObject*, *AbstractPortPrototypeInExecutableInstanceRef*, *AtpInstanceRef*

▽

△

Class	PPortPrototypeInExecutableInstanceRef			
Aggregated by	ComOfferServiceGrantDesign.providedServicePort, CompositionPPortToExecutablePPortMapping.executableProvidedPort, DiagnosticAuthenticationPortMapping.pPortPrototypeInExecutable, DiagnosticDataPortMapping.pPortPrototypeInExecutable, DiagnosticEnvDataElementCondition.pPortPrototype, DiagnosticSecurityLevelPortMapping.pPortPrototypeInExecutable, DiagnosticServiceGenericMapping.pPortPrototypeInExecutable, DiagnosticServiceValidationMapping.pPortPrototypeInExecutable, DiagnosticSovdAuthorizationPortMapping.pPortPrototypeInExecutable, DiagnosticSovdBulkDataPortMapping.pPortPrototypeInExecutable, DiagnosticSovdConfigurationPortMapping.pPortPrototypeInExecutable, DiagnosticSovdProximityChallengePortMapping.pPortPrototypeInExecutable, DiagnosticSovdServiceValidationPortMapping.pPortPrototypeInExecutable, DiagnosticSovdUpdatePortMapping.pPortPrototypeInExecutable, DltLogSinkToPortPrototypeMapping.pPortPrototype, IdsmContextProviderMapping.pPortPrototypeInExecutable, IdsmTimestampProviderMapping.pPortPrototypeInExecutable, NetworkHandlePortMapping.pPortPrototypeInExecutable, RecoveryNotificationToPPortPrototypeMapping.recoveryAction, StateManagementSetFunctionGroupStateActionItem.portPrototype, StateManagementStateNotification.notificationPort, TimeSyncPortPrototypeToTimeBaseMapping.timeSyncPPortPrototype			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=20
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=10
targetPPort Prototype	PPortPrototype	0..1	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=30

Table D.10: PPortPrototypeInExecutableInstanceRef

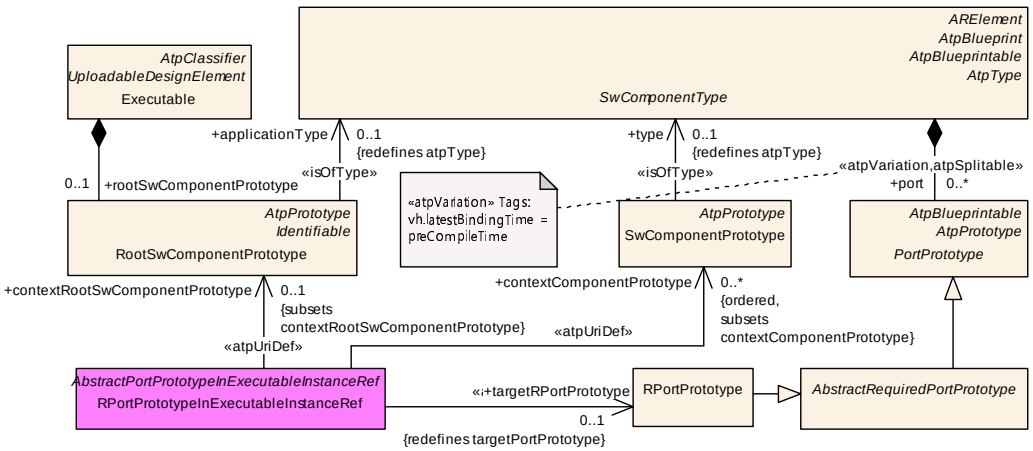


Figure D.10: Modeling of concrete `RPortPrototypeInExecutableInstanceRef` derived from `AbstractPortPrototypeInExecutableInstanceRef`

Class	RPortPrototypeInExecutableInstanceRef
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef
Note	
Base	*ARObject*, *AbstractPortPrototypeInExecutableInstanceRef*, *AtpInstanceRef*

▽

△

Class	**RPortPrototypeInExecutableInstanceRef**			
Aggregated by	CompositionRPortToExecutableRPortMapping.executableRequiredPort, CryptoCertificateToPort PrototypeMapping.portPrototype, CryptoKeySlotToPortPrototypeMapping.portPrototype, CryptoProvider ToPortPrototypeMapping.portPrototype, DiagnosticClearConditionPortMapping.rPortPrototypeIn Executable, DiagnosticEnableConditionPortMapping.rPortPrototypeInExecutable, DiagnosticEventPort Mapping.rPortPrototypeInExecutable, DiagnosticExternalAuthenticationPortMapping.rPortPrototypeIn Executable, DiagnosticIndicatorPortMapping.rPortPrototypeInExecutable, DiagnosticMemoryDestination PortMapping.rPortPrototypeInExecutable, DiagnosticMonitorPortMapping.rPortPrototypeInExecutable, DiagnosticMultipleConditionPortMapping.rPortPrototypeInExecutable, DiagnosticMultipleEventPort Mapping.rPortPrototypeInExecutable, DiagnosticMultipleMonitorPortMapping.rPortPrototypeIn Executable, DiagnosticOperationCyclePortMapping.rPortPrototypeInExecutable, DltLogSinkToPort PrototypeMapping.rPortPrototype, HealthChannelExternalStatus.healthChannel, NoSupervision.target PhmSupervisedEntity, *RawDataStreamMapping*.portPrototype, SecurityEventMapping.reportingPort Prototype, *StateManagementStateRequest*.stateRequestPort, TimeSyncPortPrototypeToTimeBase Mapping.timeSyncRPortPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=20
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=10
targetRPort Prototype	RPortPrototype	0..1	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=30

Table D.11: RPortPrototypeInExecutableInstanceRef

Class	**PortPrototypeInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note				
Base	*ARObject*, *AbstractPortPrototypeInExecutableInstanceRef*, *AtpInstanceRef*			
Aggregated by	*PersistencyPortPrototypeToDeploymentMapping*.portPrototype, ServiceInstanceToPortPrototype Mapping.portPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpDerived **Tags:** xml.sequenceOffset=10
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Tags:** xml.sequenceOffset=20
targetPort Prototype	PortPrototype	0..1	ref	**Tags:** xml.sequenceOffset=40

Table D.12: PortPrototypeInExecutableInstanceRef

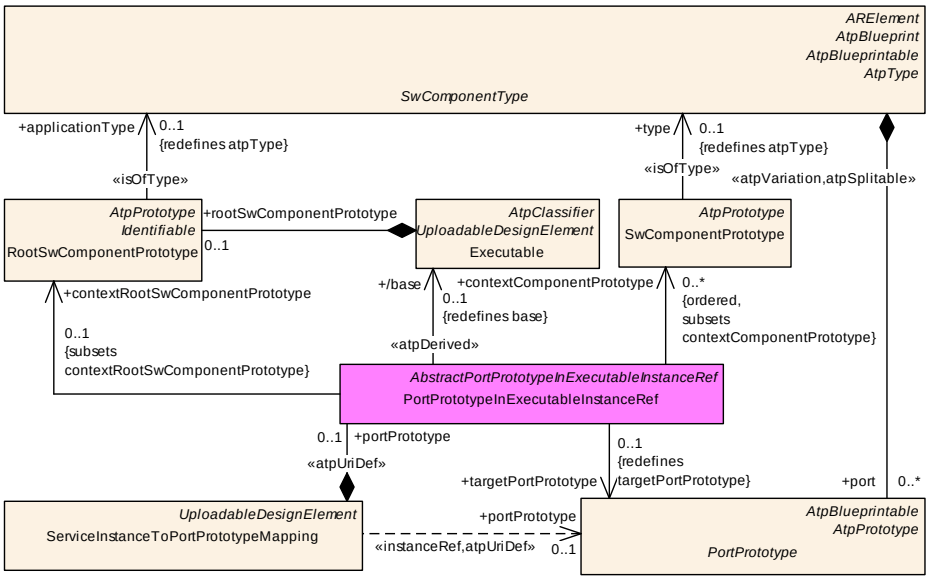


Figure D.11: Modeling of `PortPrototypeInExecutableInstanceRef`

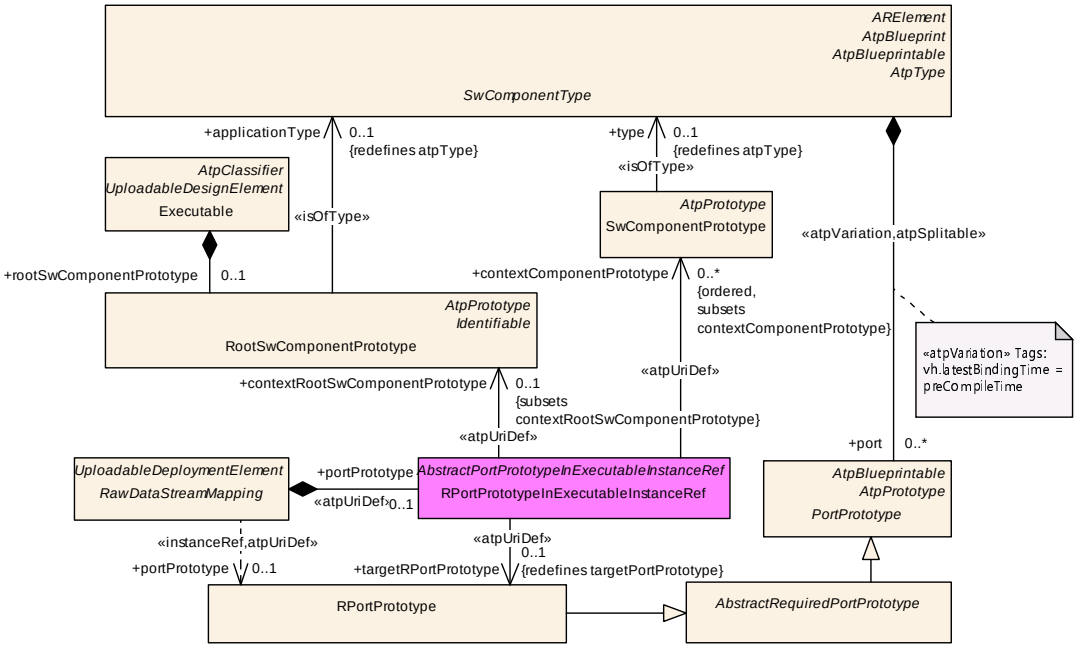


Figure D.12: Modeling of raw data streaming by means of an `RPortPrototype`

D.4 Modeling of a Method in an Executable

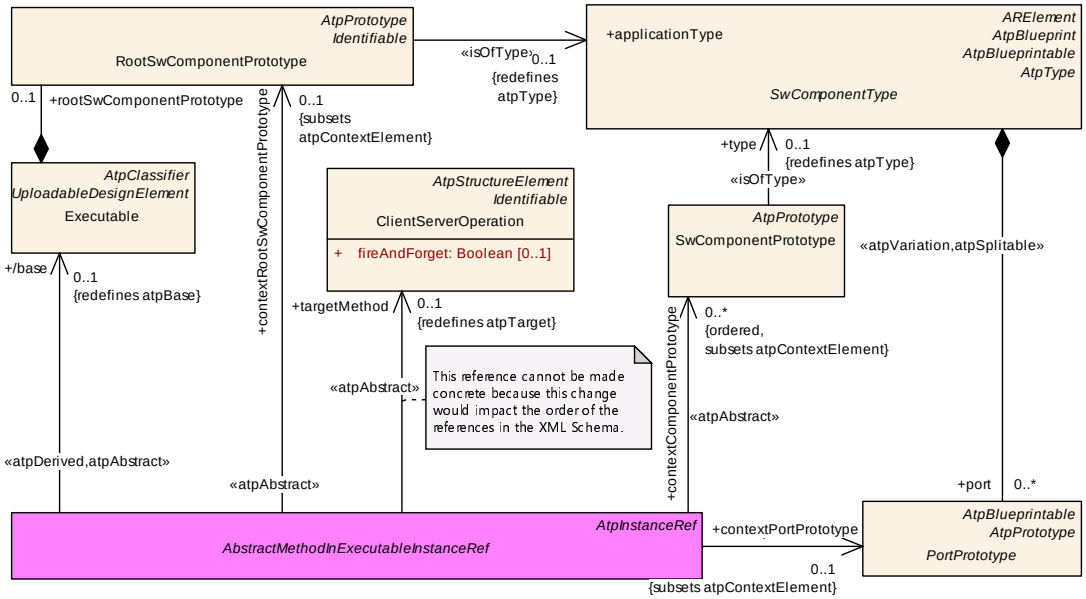


Figure D.13: Modeling of `AbstractMethodInExecutableInstanceRef`

Class	***AbstractMethodInExecutableInstanceRef*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef::MethodIn Executable			
Note				
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	RequiredMethodInExecutableInstanceRef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpAbstract; atpDerived
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=20
contextPort Prototype	PortPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=10
targetMethod	ClientServerOperation	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=40

Table D.13: AbstractMethodInExecutableInstanceRef

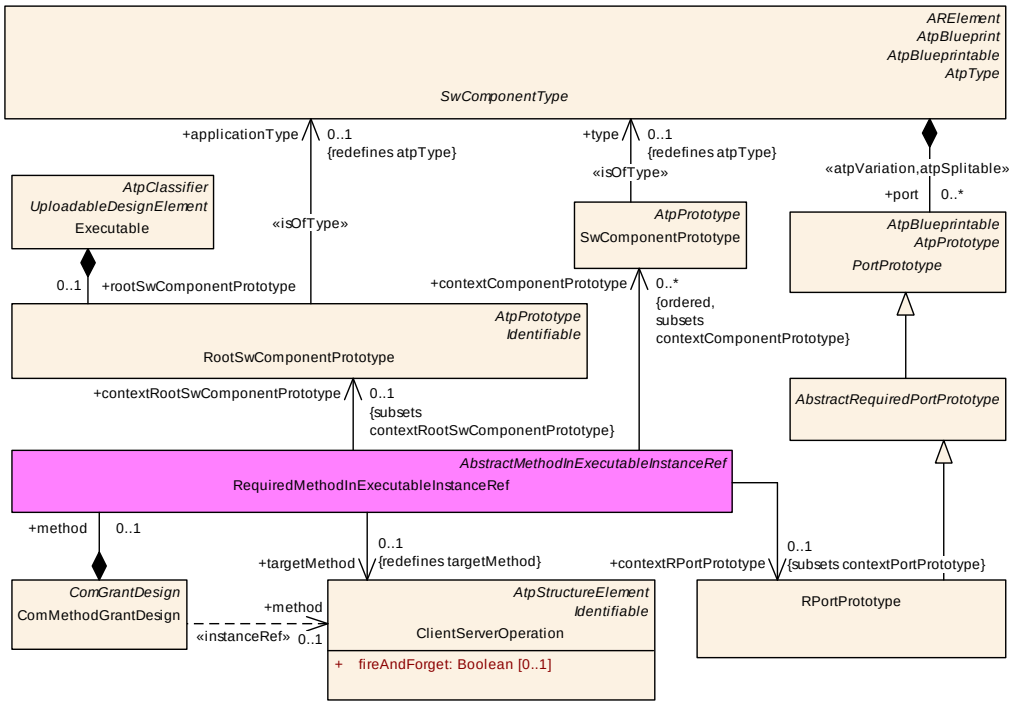


Figure D.14: Modeling of `RequiredMethodInExecutableInstanceRef`

Class	**RequiredMethodInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef::MethodInExecutable			
Note				
Base	*ARObject*, *AbstractMethodInExecutableInstanceRef*, *AtpInstanceRef*			
Aggregated by	ComMethodGrantDesign.method			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** xml.sequenceOffset=20
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Tags:** xml.sequenceOffset=10
contextRPort Prototype	RPortPrototype	0..1	ref	**Tags:** xml.sequenceOffset=30
targetMethod	ClientServerOperation	0..1	ref	**Tags:** xml.sequenceOffset=40

Table D.14: RequiredMethodInExecutableInstanceRef

D.5 Modeling of Diagnostic-related InstanceRefs

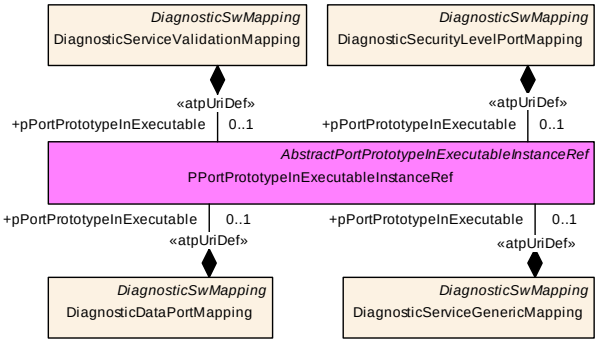


Figure D.15: Modeling of various `DiagnosticSwMapping`s based on `PPortPrototype-InExecutableInstanceRef`

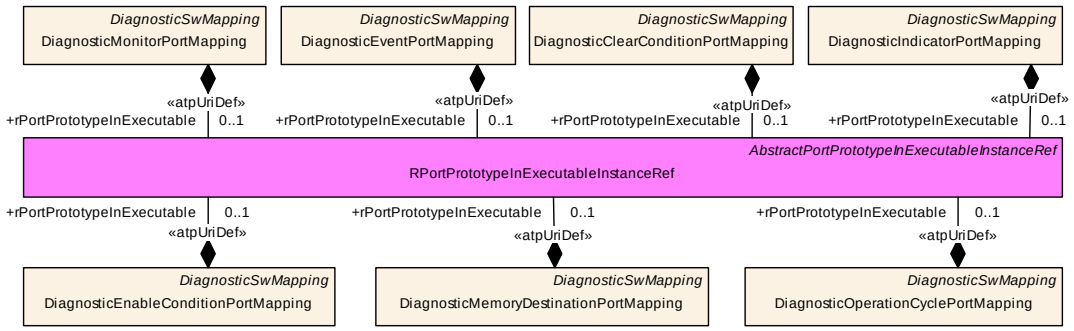


Figure D.16: Modeling of various `DiagnosticSwMapping`s based on `RPortPrototype-InExecutableInstanceRef`

D.6 Modeling of PHM-related InstanceRefs

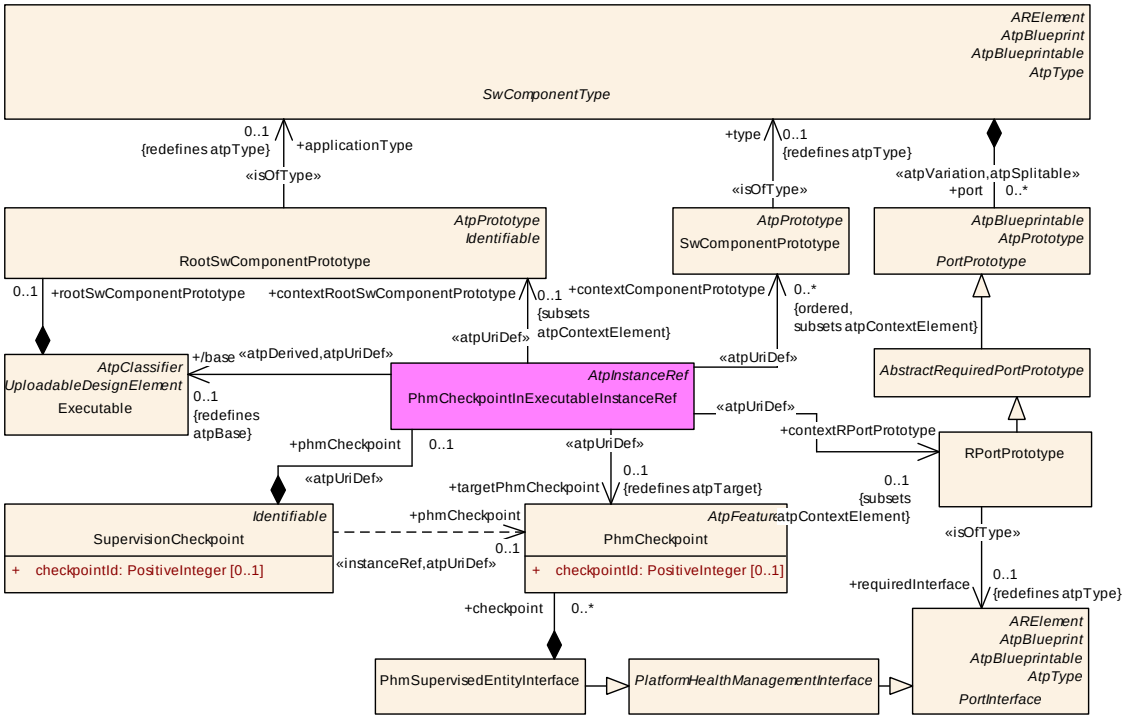


Figure D.17: Modeling of `PhmCheckpointInExecutableInstanceRef`

Class	**PhmCheckpointInExecutableInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealth Management::InstanceRefs			
Note				
Base	*ARObject*, *AtpInstanceRef*			
Aggregated by	SupervisionCheckpoint.phmCheckpoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Executable	0..1	ref	**Stereotypes:** atpDerived; atpUriDef **Tags:** xml.sequenceOffset=10
context Component Prototype (ordered)	SwComponent Prototype	*	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=30
contextRootSw Component Prototype	RootSwComponent Prototype	0..1	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=20
contextRPort Prototype	RPortPrototype	0..1	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=40
targetPhm Checkpoint	PhmCheckpoint	0..1	ref	**Stereotypes:** atpUriDef **Tags:** xml.sequenceOffset=50

Table D.15: PhmCheckpointInExecutableInstanceRef

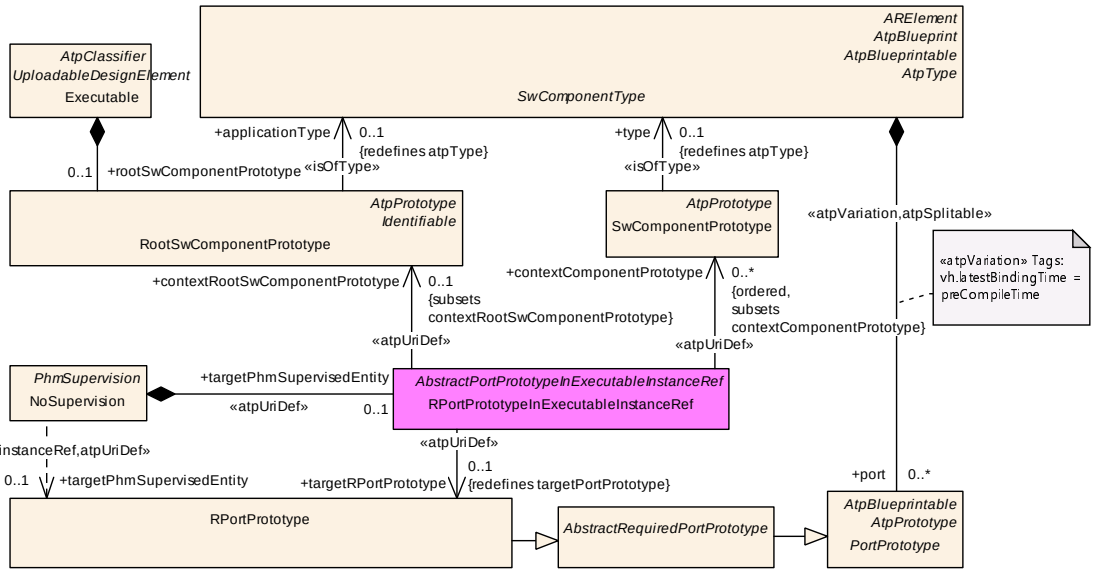


Figure D.18: Modeling of `NoSupervision` `RPortPrototypeInExecutableInstanceRef`

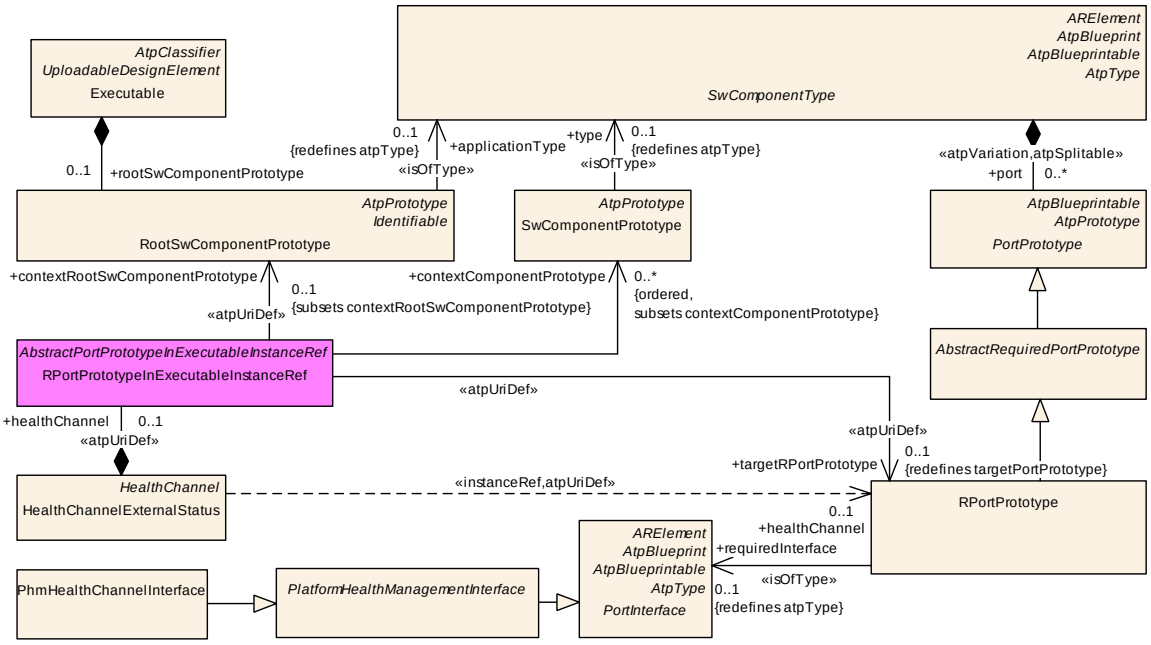


Figure D.19: Modeling of `HealthChannelExternalStatus`

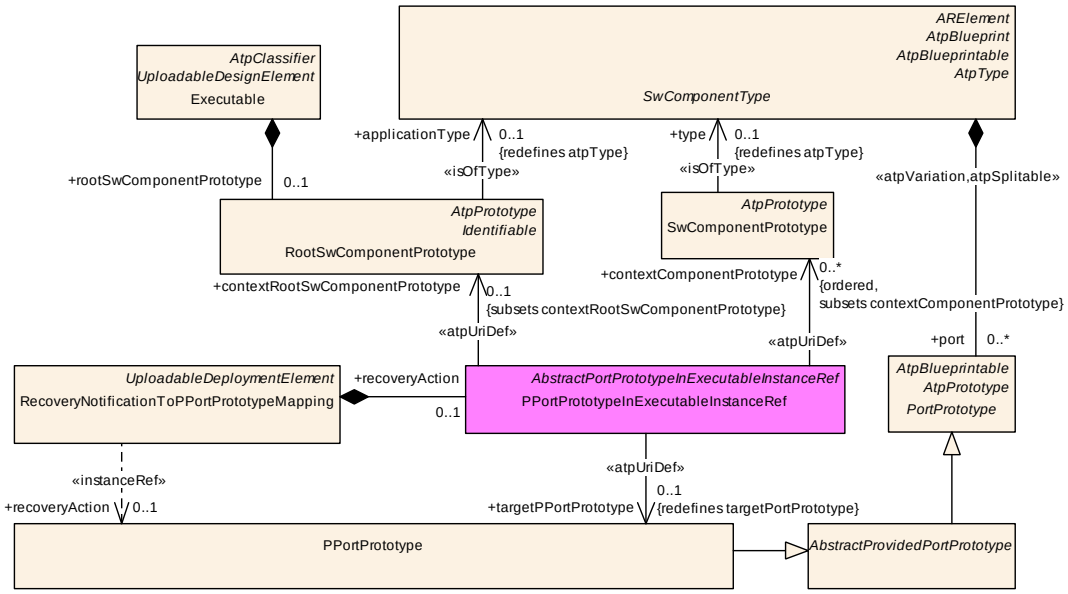


Figure D.20: Modeling of `RecoveryNotificationToPPortPrototypeMapping`

D.7 Modeling of Time-related InstanceRefs

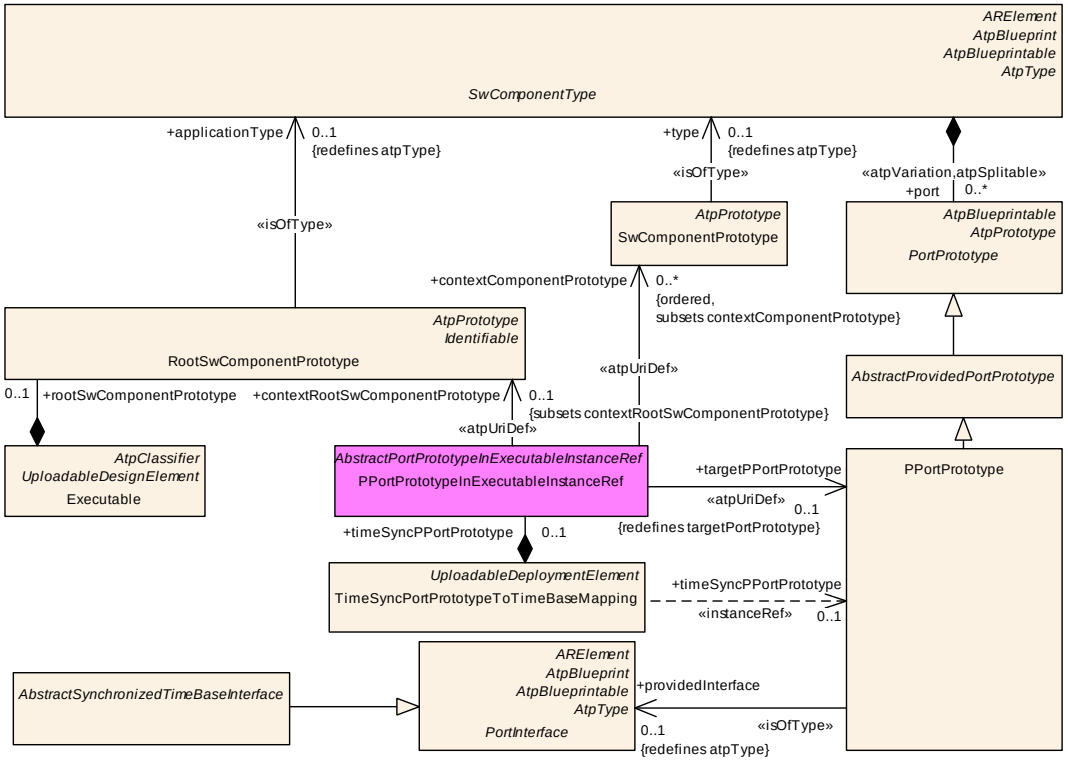


Figure D.21: Modeling of `TimeSyncPortPrototypeToTimeBaseMapping.timeSyncPPortPrototype`

D.8 Modeling of Persistency-related InstanceRefs

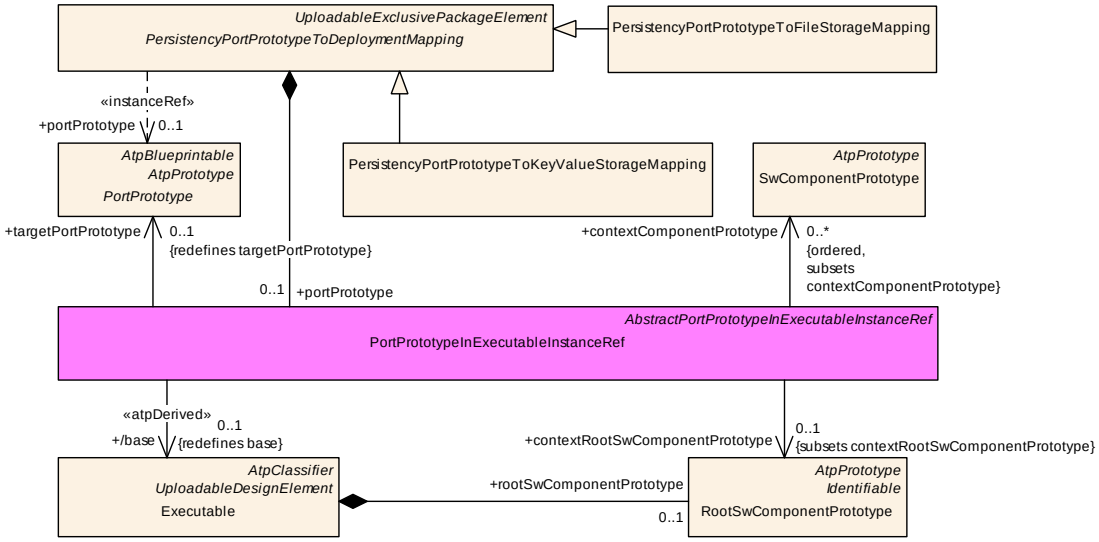


Figure D.22: Modeling of `PersistencyPortPrototypeToDeploymentMapping`

D.9 Modeling of SoftwareClusterDesign-related InstanceRefs

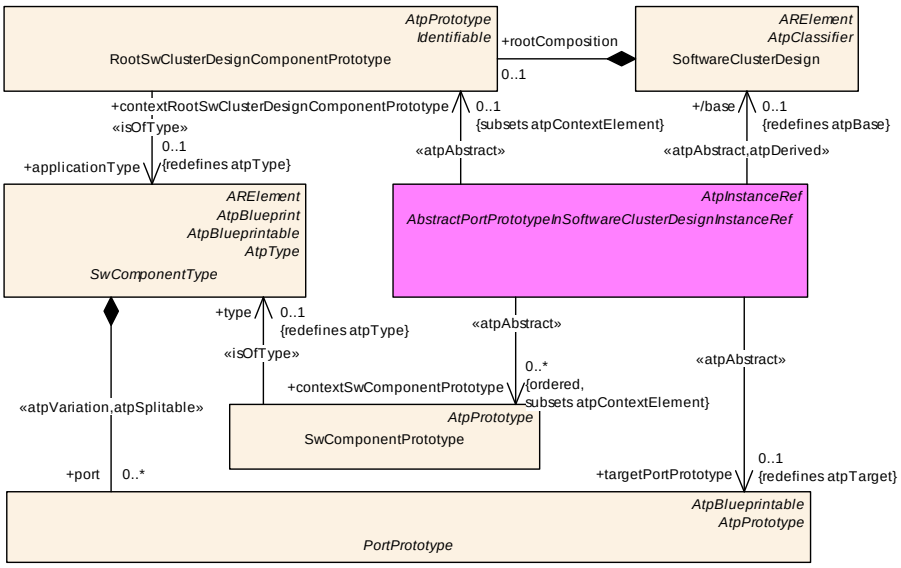


Figure D.23: Modeling of `AbstractPortPrototypeInSoftwareClusterDesignInstanceRef`

Class	***AbstractPortPrototypeInSoftwareClusterDesignInstanceRef*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInSoftwareClusterDesignInstanceRef			
Note				
Base	*ARObject*, *AtpInstanceRef*			
Subclasses	PPortPrototypeInSoftwareClusterDesignInstanceRef, RPortPrototypeInSoftwareClusterDesignInstanceRef			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	SoftwareClusterDesign	0..1	ref	**Stereotypes:** atpAbstract; atpDerived
contextRootSwClusterDesignComponentPrototype	RootSwClusterDesignComponentPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=10
contextSwComponentPrototype (ordered)	SwComponentPrototype	*	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=20
targetPortPrototype	PortPrototype	0..1	ref	**Stereotypes:** atpAbstract **Tags:** xml.sequenceOffset=30

Table D.16: AbstractPortPrototypeInSoftwareClusterDesignInstanceRef

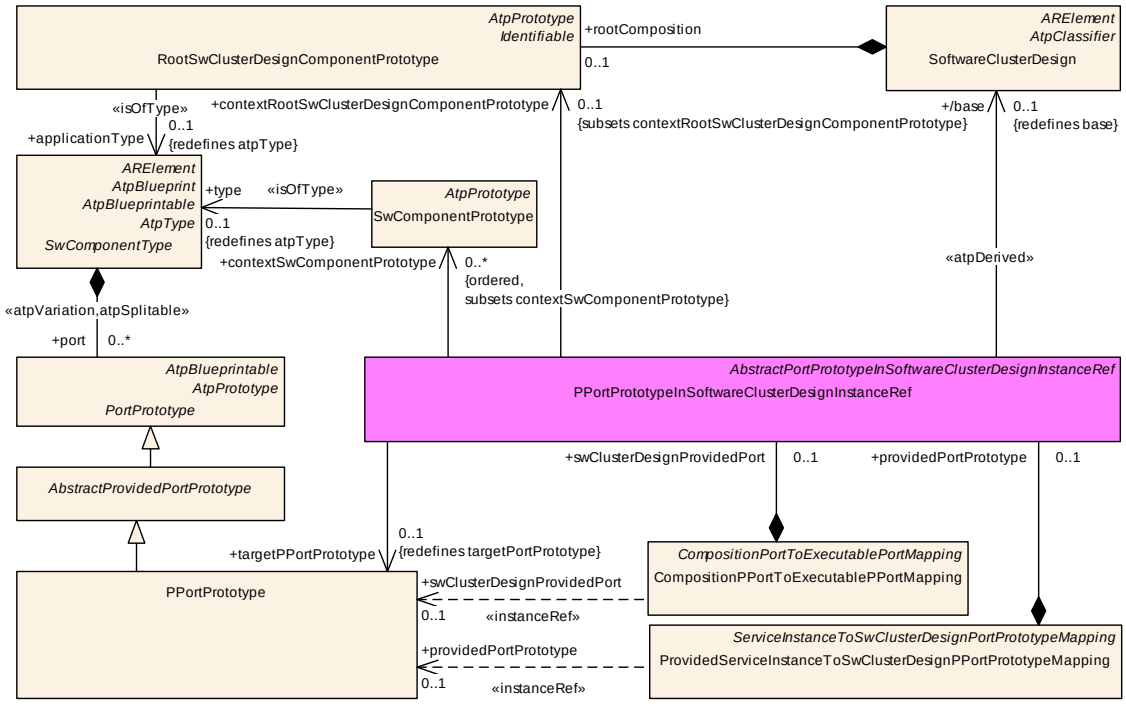


Figure D.24: Modeling of `PPortPrototypeInSoftwareClusterDesignInstanceRef`

Class	**PPortPrototypeInSoftwareClusterDesignInstanceRef**
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInSoftwareClusterDesignInstanceRef
Note	
Base	*ARObject*, *AbstractPortPrototypeInSoftwareClusterDesignInstanceRef*, *AtpInstanceRef*
Aggregated by	CompositionPPortToExecutablePPortMapping.swClusterDesignProvidedPort, ProvidedServiceInstanceToSwClusterDesignPPortPrototypeMapping.providedPortPrototype

△

Class	**PPortPrototypeInSoftwareClusterDesignInstanceRef**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	SoftwareClusterDesign	0..1	ref	**Stereotypes:** atpDerived
contextRootSw ClusterDesign Component Prototype	RootSwClusterDesign ComponentPrototype	0..1	ref	**Tags:** xml.sequenceOffset=10
contextSw Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** xml.sequenceOffset=20
targetPPort Prototype	PPortPrototype	0..1	ref	**Tags:** xml.sequenceOffset=30

Table D.17: PPortPrototypeInSoftwareClusterDesignInstanceRef

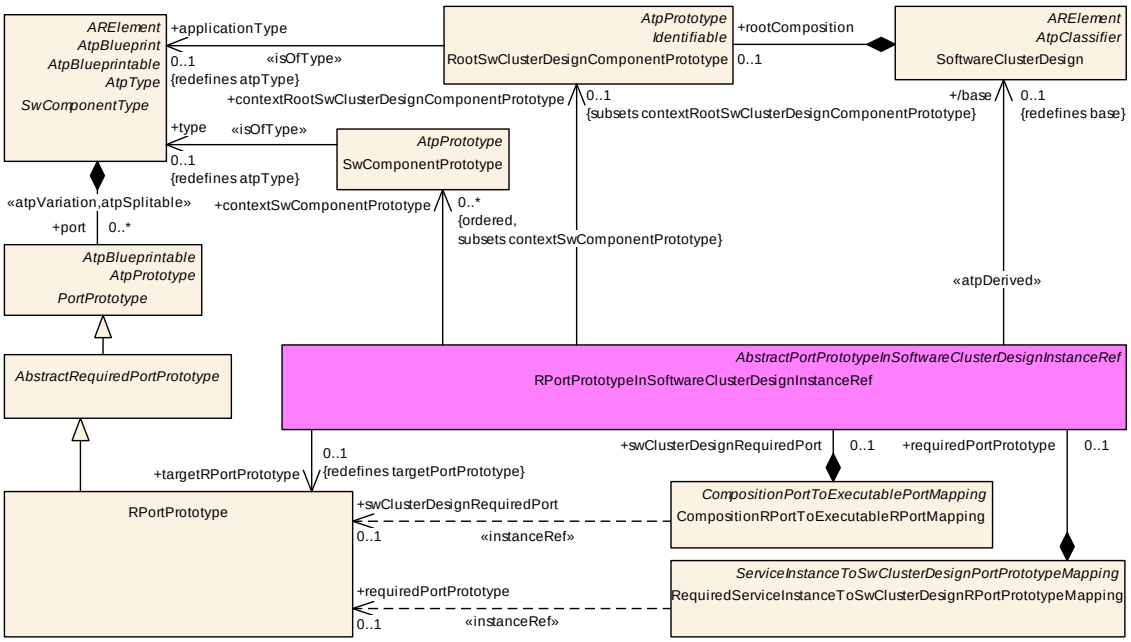


Figure D.25: Modeling of `RPortPrototypeInSoftwareClusterDesignInstanceRef`

Class	**RPortPrototypeInSoftwareClusterDesignInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInSoftwareClusterDesignInstanceRef			
Note				
Base	*ARObject*, *AbstractPortPrototypeInSoftwareClusterDesignInstanceRef*, *AtpInstanceRef*			
Aggregated by	CompositionRPortToExecutableRPortMapping.swClusterDesignRequiredPort, RequiredServiceInstance ToSwClusterDesignRPortPrototypeMapping.requiredPortPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	SoftwareClusterDesign	0..1	ref	**Stereotypes:** atpDerived
contextRootSw ClusterDesign Component Prototype	RootSwClusterDesign ComponentPrototype	0..1	ref	**Tags:** xml.sequenceOffset=10

▽

△

Class	**RPortPrototypeInSoftwareClusterDesignInstanceRef**			
contextSw Component Prototype (ordered)	SwComponent Prototype	*	ref	**Tags:** xml.sequenceOffset=20
targetRPort Prototype	RPortPrototype	0..1	ref	**Tags:** xml.sequenceOffset=30

Table D.18: RPortPrototypeInSoftwareClusterDesignInstanceRef

D.10 Modeling of State-Management-related InstanceRefs

This section illustrates the concrete modeling of the instance references used in the modeling of the state management deployment in this document.

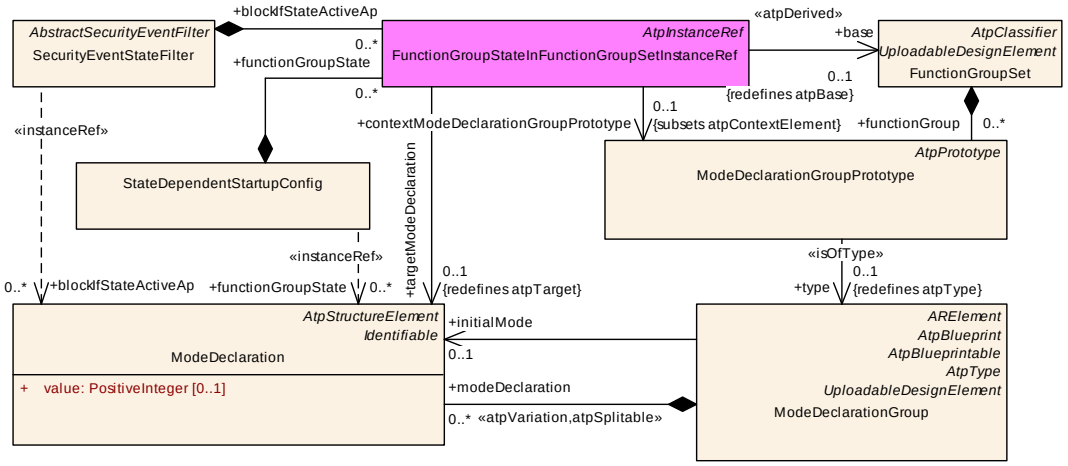


Figure D.26: Modeling of `FunctionGroupStateInFunctionGroupSetInstanceRef`

Class	**FunctionGroupStateInFunctionGroupSetInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest::InstanceRefs			
Note				
Base	*ARObject*, *AtpInstanceRef*			
Aggregated by	FunctionGroupPhmStateReference.functionGroupState, NmHandleToFunctionGroupStateMapping.functionGroupState, SecurityEventStateFilter.blockIfStateActiveAp, StateDependentStartupConfig.functionGroupState, StateManagementSetFunctionGroupStateActionItem.setFunctionGroupState			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	FunctionGroupSet	0..1	ref	**Stereotypes:** atpDerived
contextMode Declaration GroupPrototype	ModeDeclarationGroup Prototype	0..1	ref	**Tags:** xml.sequenceOffset=10
targetMode Declaration	ModeDeclaration	0..1	ref	**Tags:** xml.sequenceOffset=20

Table D.19: FunctionGroupStateInFunctionGroupSetInstanceRef

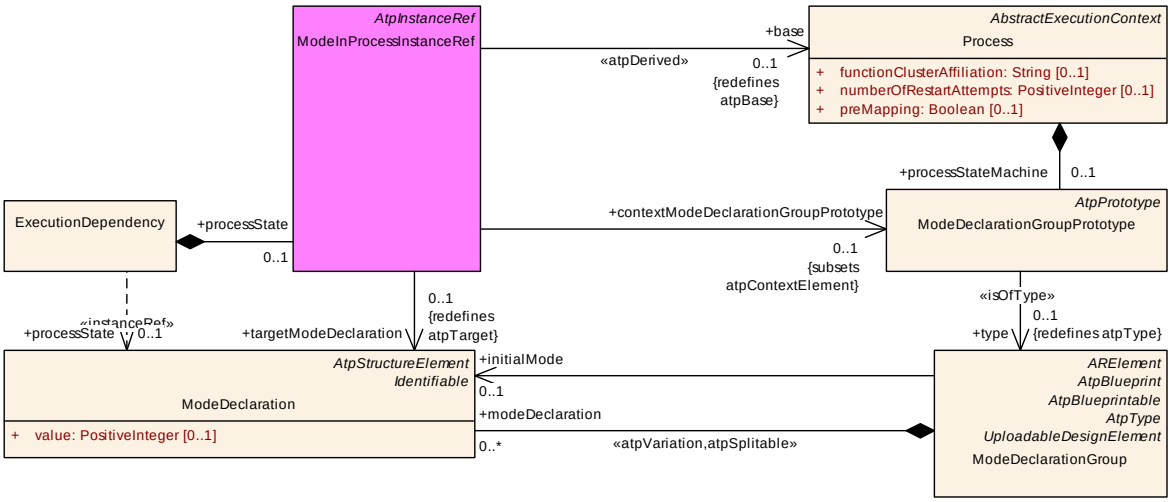


Figure D.27: Modeling of ModeInProcessInstanceRef

Class	**ModeInProcessInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest::InstanceRefs			
Note				
Base	*ARObject*, *AtpInstanceRef*			
Aggregated by	ExecutionDependency.processState			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Process	0..1	ref	**Stereotypes:** atpDerived **Tags:** xml.sequenceOffset=10
contextMode Declaration GroupPrototype	ModeDeclarationGroup Prototype	0..1	ref	**Tags:** xml.sequenceOffset=20
targetMode Declaration	ModeDeclaration	0..1	ref	**Tags:** xml.sequenceOffset=30

Table D.20: ModeInProcessInstanceRef

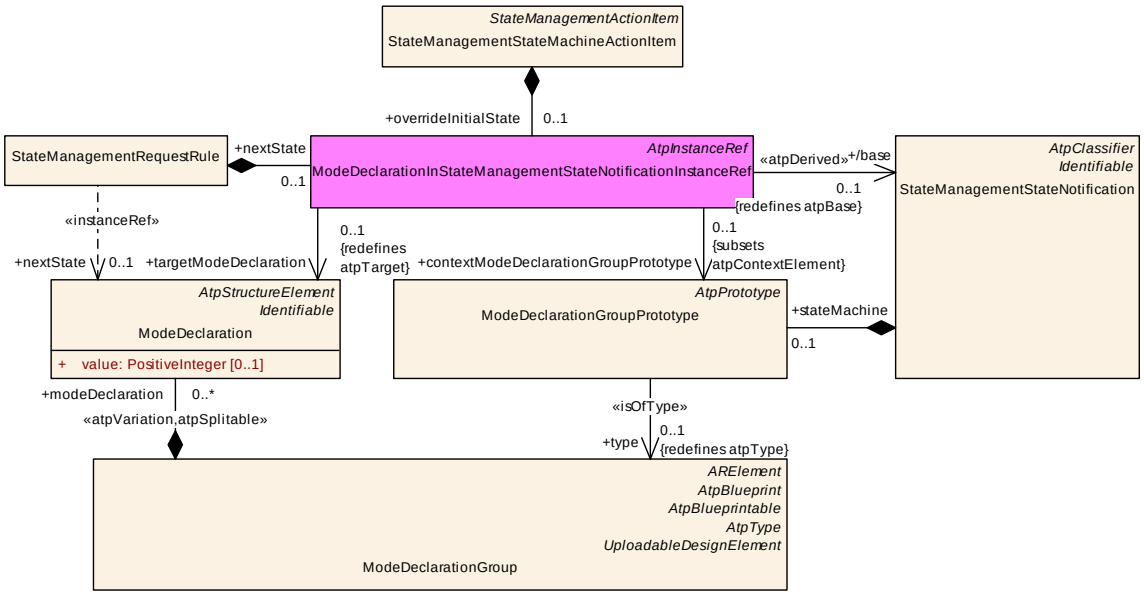


Figure D.28: Modeling of ModeDeclarationInStateManagementStateNotificationInstanceRef

Class	**ModeDeclarationInStateManagementStateNotificationInstanceRef**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::General::SomethingInExecutableInstanceRef			
Note	**Tags:** atp.Status=draft			
Base	*ARObject*, *AtpInstanceRef*			
Aggregated by	StateManagementActionList.affectedState, StateManagementRequestRule.nextState, StateManagement StateMachineActionItem.overrideInitialState, StateManagementTriggerCompareRule.assumedCurrent State			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	StateManagementState Notification	0..1	ref	**Stereotypes:** atpDerived **Tags:** atp.Status=draft
contextMode Declaration GroupPrototype	ModeDeclarationGroup Prototype	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=10
targetMode Declaration	ModeDeclaration	0..1	ref	**Tags:** atp.Status=draft xml.sequenceOffset=20

Table D.21: ModeDeclarationInStateManagementStateNotificationInstanceRef

E Mentioned Class Tables

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

Class	***ARElement*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
Note	An element that can be defined stand-alone, i.e. without being part of another element (except for packages of course).			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	AclObjectSet, AclOperation, AclPermission, AclRole, AliasNameSet, Allocator, ApApplicationError, ApApplicationErrorDomain, ApApplicationErrorSet, ApplicabilityInfoSet, *AutosarDataType*, *BaseType*, BlueprintMappingSet, BuildActionManifest, CalibrationParameterValueSet, CanXlProps, ClientIdDefinitionSet, Collection, *CompositionPortToExecutablePortMapping*, CompuMethod, ConsistencyNeedsBlueprintSet, ConstantSpecification, ConstantSpecificationMappingSet, CryptoEllipticCurveProps, CryptoServiceCertificate, CryptoServiceKey, CryptoServicePrimitive, CryptoServiceQueue, CryptoSignatureScheme, DataConstr, DataExchangePoint, DataTransformationSet, DataTypeMappingSet, DdsCpConfig, *DiagnosticCommonElement*, DiagnosticConnection, DiagnosticContributionSet, DltContext, DltEcu, Documentation, E2EProfileCompatibilityProps, E2EProfileConfigurationSet, EndToEndProtectionSet, EthIpProps, EthTcpIpIcmpProps, EthTcpIpProps, EvaluatedVariantSet, FMFeature, FMFeatureMap, FMFeatureModel, FMFeatureSelectionSet, FirewallRule, GeneralPurposeConnection, *GrantDesign*, HwCategory, HwElement, HwType, *IEEE1722TpConnection*, IPSecConfigProps, *IdsCommonElement*, IdsDesign, ImpositionTimeDefinitionGroup, InterfaceMapping, InterpolationRoutineMappingSet, KeywordSet, LTMessageCollectionToPortPrototypeMapping, LifeCycleInfoSet, LifeCycleStateDefinitionGroup, LogAndTraceMessageCollectionSet, MacSecGlobalKayProps, MacSecParticipantSet, McFunction, McGroup, ModeDeclarationGroup, ModeDeclarationMappingSet, PhysicalDimension, PhysicalDimensionMappingSet, *PlatformModuleEndpointConfiguration*, *PortInterface*, PortInterfaceMappingSet, PortInterfaceToDataTypeMapping, PortPrototypeBlueprint, PostBuildVariantCriterion, PostBuildVariantCriterionValueSet, PredefinedVariant, ProcessDesign, ProcessDesignToMachineDesignMapping, RapidPrototypingScenario, SdgDef, *SecureComProps*, *ServiceInstanceToSwClusterDesignPortPrototypeMapping*, *ServiceInterfaceElementMapping*, ServiceInterfacePedigree, SignalServiceTranslationPropsSet, SoftwareClusterDesign, SomeipDataPrototypeTransformationProps, SomeipRemoteUnicastConfig, SomeipSdClientEventGroupTimingConfig, SomeipSdClientServiceInstanceConfig, SomeipSdServerEventGroupTimingConfig, SomeipSdServerServiceInstanceConfig, SwAddrMethod, SwAxisType, *SwComponentType*, SwRecordLayout, SwSystemconst, SwSystemconstantValueSet, System, SystemSignal, SystemSignalGroup, *TimingExtension*, TlsConnectionGroup, TlvDataIdDefinitionSet, TransformationPropsSet, TransformationPropsToServiceInterfaceElementMapping, Unit, UnitGroup, *UploadablePackageElement*, ViewMapSet			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.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*			
Aggregated by	ARPackage.arPackage, AUTOSAR.arPackage			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

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

Table E.2: ARPackage

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***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data of an Autosar file. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=adminData xml.sequenceOffset=10
arPackage	ARPackage	*	aggr	This is the top level package in an AUTOSAR model. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=arPackage.shortName, arPackage.variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
fileInfo Comment	FileInfoComment	0..1	aggr	This represents a possibility to provide a structured comment in an AUTOSAR file. **Stereotypes:** atpStructuredComment **Tags:** xml.roleElement=true xml.sequenceOffset=-10 xml.typeElement=false

▽

△

Class	**AUTOSAR**			
introduction	DocumentationBlock	0..1	aggr	This represents an introduction on the Autosar file. It is intended for example to represent disclaimers and legal notes. **Tags:** xml.sequenceOffset=20

Table E.3: AUTOSAR

Class	***AbstractProvidedPortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This abstract class provides the ability to become a provided PortPrototype.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Subclasses	PPortPrototype, PRPortPrototype			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
providedComSpec	PPortComSpec	*	aggr	Provided communication attributes per interface element (data element or operation). **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=providedComSpec

Table E.4: AbstractProvidedPortPrototype

Class	***AbstractRequiredPortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This abstract class provides the ability to become a required PortPrototype.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Subclasses	PRPortPrototype, RPortPrototype			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
requiredComSpec	RPortComSpec	*	aggr	Required communication attributes, one for each interface element. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=requiredComSpec

Table E.5: AbstractRequiredPortPrototype

Class	***AbstractServiceInstance*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
Note	Provided and Consumed Ethernet Service Instances that are available at the ApplicationEndpoint.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ConsumedServiceInstance, *DdsCpServiceInstance*, ProvidedServiceInstance			
Aggregated by	ServiceInstanceCollectionSet.serviceInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	***AbstractServiceInstance*** (abstract)			
capability Record	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=capabilityRecord, capabilityRecord.variation Point.shortLabel vh.latestBindingTime=postBuild
majorVersion	PositiveInteger	0..1	attr	Major Version of the ServiceInterface. Value can be set to a number that represents the Major Version of the service.
method Activation RoutingGroup	PduActivationRouting Group	0..1	aggr	The ServiceDiscovery module is able to activate and deactivate the PDU routing for ClientServerOperations (SOME/IP methods). **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=methodActivationRoutingGroup.shortName, methodActivationRoutingGroup.variationPoint.shortLabel vh.latestBindingTime=postBuild
routingGroup	SoAdRoutingGroup	*	ref	The ServiceDiscovery module is able to activate and deactivate the PDU routing from and to TCP/IP-sockets. **Tags:** atp.Status=obsolete

Table E.6: AbstractServiceInstance

Class	**AdminData**			
Package	M2::MSR::AsamHdo::AdminData			
Note	AdminData represents the ability to express administrative information and custom extensions 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 the following 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 Beside that a custom extension of model-data is possible by • Special data			
Base	*ARObject*			
Aggregated by	AUTOSAR.adminData, *Describable*.adminData, *Identifiable*.adminData			
Attribute	***Type***	***Mult.***	***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

▽

△

Class	**AdminData**			
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. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=sdg.sdgCaption.shortName xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=60 xml.typeElement=false xml.typeWrapperElement=false
usedLanguages	MultiLanguagePlainText	0..1	aggr	This property specifies the languages which are provided in the document. Therefore it should only be specified in the top level admin data. For each language provided in the document there is one entry in MultilanguagePlain Text. The content of each entry can be used for illustration of the language. The used language itself depends on the language attribute in the entry. **Tags:** xml.sequenceOffset=30

Table E.7: AdminData

Class	**ApplicationArrayDataType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	An application data type which is an array, each element is of the same application data type. **Tags:** atp.recommendedPackage=ApplicationDataTypes			
Base	*ARElement*, *ARObject*, *ApplicationCompositeDataType*, *ApplicationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicArraySizeProfile	String	0..1	attr	Specifies the profile which the array will follow if it is a variable size array.
element	ApplicationArrayElement	0..1	aggr	This association implements the concept of an array element. That is, in some cases it is necessary to be able to identify single array elements, e.g. as input values for an interpolation routine.

Table E.8: ApplicationArrayDataType

Class	**ApplicationArrayElement**
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes
Note	Describes the properties of the elements of an application array data type.
Base	*ARObject*, *ApplicationCompositeElementDataPrototype*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	**ApplicationArrayElement**			
Aggregated by	ApplicationArrayDataType.element, *AtpClassifier*.atpFeature			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arraySize Handling	ArraySizeHandling Enum	0..1	attr	The way how the size of the array is handled.
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls how the information about the array size shall be interpreted.
indexDataType	ApplicationPrimitive DataType	0..1	ref	This reference can be taken to assign a CompuMethod of category TEXTTABLE to the array. The texttable entries associate a textual value to an index number such that the element with that index number is represented by a symbolic name.
maxNumberOf Elements	PositiveInteger	0..1	attr	The maximum number of elements that the array can contain. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table E.9: ApplicationArrayElement

Class	***ApplicationCompositeDataType*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	Abstract base class for all application data types composed of other data types.			
Base	*ARElement*, *ARObject*, *ApplicationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	ApplicationArrayDataType, ApplicationAssocMapDataType, ApplicationRecordDataType			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.10: ApplicationCompositeDataType

Class	**ApplicationError**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	This is a user-defined error that is associated with an element of an AUTOSAR interface. It is specific for the particular functionality or service provided by the AUTOSAR software component.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ClientServerInterface.possibleError			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
errorCode	Integer	0..1	attr	The RTE generator is forced to assign this value to the corresponding error symbol. Note that for error codes certain ranges are predefined (see RTE specification).

Table E.11: ApplicationError

Class	**ApplicationSwComponentType**
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components
Note	The ApplicationSwComponentType is used to represent the application software. **Tags:** atp.recommendedPackage=SwComponentTypes

▽

△

Class	ApplicationSwComponentType			
Base	ARElement, ARObject, AtomicSwComponentType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType			
Aggregated by	ARPackage.element			
Attribute	Type	Mult.	Kind	Note
–	–	–	–	–

Table E.12: ApplicationSwComponentType

Class	ApplicationValueSpecification			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This meta-class represents values for DataPrototypes typed by ApplicationDataTypes (this includes in particular compound primitives). For further details refer to ASAM CDF 2.0. This meta-class corresponds to some extent with SW-INSTANCE in ASAM CDF 2.0.			
Base	ARObject, CompositeRuleBasedValueArgument, ValueSpecification			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValueSpecification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, CompositeRuleBasedValueSpecification.compoundPrimitiveArgument, ConstantSpecification.valueSpec, CryptoServiceKey.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElementCondition.compareValue, FieldSenderComSpec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, SomeipEventDeployment.eventReceptionDefaultValue, StateManagementCompareCondition.compareValue, SwDataDefProps.invalidValue, VariableDataPrototype.initValue			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
category	Identifier	0..1	attr	Specifies to which category of ApplicationDataType this ApplicationValueSpecification can be applied (e.g. as an initial value), thus imposing constraints on the structure and semantics of the contained values, see [constr_1006] and [constr_2051].
swAxisCont (ordered)	SwAxisCont	*	aggr	This represents the axis values of a Compound Primitive Data Type (curve or map). The first swAxisCont describes the x-axis, the second swAxisCont describes the y-axis, the third swAxisCont describes the z-axis. In addition to this, the axis can be denoted in swAxisIndex.
swValueCont	SwValueCont	0..1	aggr	This represents the values of a Compound Primitive Data Type.

Table E.13: ApplicationValueSpecification

Class	ArgumentDataPrototype
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface
Note	An argument of an operation, much like a data element, but also carries direction information and is owned by a particular ClientServerOperation.
Base	ARObject, AtpFeature, AtpPrototype, AutosarDataPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable
Aggregated by	AtpClassifier.atpFeature, ClientServerOperation.argument

▽

△

Class	**ArgumentDataPrototype**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
direction	ArgumentDirection Enum	0..1	attr	This attribute specifies the direction of the argument prototype.
serverArgument ImplPolicy	ServerArgumentImpl PolicyEnum	0..1	attr	This defines how the argument type of the servers RunnableEntity is implemented. If the attribute is not defined this has the same semantics as if the attribute is set to the value useArgumentType for primitive arguments and structures.

Table E.14: ArgumentDataPrototype

Enumeration	**ArgumentDirectionEnum**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	Use cases: • Arguments in ClientServerOperation can have different directions that need to be formally indicated because they have an impact on how the function signature looks like eventually. • Arguments in BswModuleEntry already determine a function signature, but the direction is used to specify the semantics, especially of pointer arguments.
Aggregated by	ArgumentDataPrototype.direction, SwServiceArg.direction
Literal	***Description***
in	The argument value is passed to the callee. **Tags:** atp.EnumerationLiteralIndex=0
inout	The argument value is passed to the callee but also passed back from the callee to the caller. **Tags:** atp.EnumerationLiteralIndex=1
out	The argument value is passed from the callee to the caller. **Tags:** atp.EnumerationLiteralIndex=2

Table E.15: ArgumentDirectionEnum

Class	**ArrayValueSpecification**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Specifies the values for an array.			
Base	*ARObject*, *CompositeValueSpecification*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValueSpecification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, CompositeRuleBasedValueSpecification.argument, ConstantSpecification.valueSpec, CryptoServiceKey.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElementCondition.compareValue, FieldSenderComSpec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, SomeipEventDeployment.eventReceptionDefaultValue, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableDataPrototype.initValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	ArrayValueSpecification			
element (ordered)	ValueSpecification	*	aggr	The value for a single array element. All Value Specifications aggregated by ArrayValueSpecification shall have the same structure. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=element, element.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
intendedPartial Initialization Count	PositiveInteger	0..1	attr	This attribute shall only have a meaning for dynamic arrays and shall be taken as a sanity check: the number filled in the attribute shall be identical to the number of ArrayValueSpecification.element. If the attribute does not exist it means that no partial initialization is intended.

Table E.16: ArrayValueSpecification

Class	AssemblySwConnector			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	AssemblySwConnectors are exclusively used to connect SwComponentPrototypes in the context of a CompositionSwComponentType.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SwConnector*			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.connector			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
provider	AbstractProvidedPort Prototype	0..1	iref	Instance of providing port. **InstanceRef implemented by:** PPortInComposition InstanceRef
requester	AbstractRequiredPort Prototype	0..1	iref	Instance of requiring port. **InstanceRef implemented by:** RPortInComposition InstanceRef

Table E.17: AssemblySwConnector

Class	*AutosarDataPrototype* (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Base class for prototypical roles of an AutosarDataType.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ArgumentDataPrototype, Field, ParameterDataPrototype, PersistencyDataElement, VariableData Prototype			
Aggregated by	*AtpClassifier*.atpFeature			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
type	AutosarDataType	0..1	tref	This represents the corresponding data type. **Stereotypes:** isOfType

Table E.18: AutosarDataPrototype

Class	***AutosarDataType*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	Abstract base class for user defined AUTOSAR data types for software.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractImplementationDataType*, *ApplicationDataType*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
swDataDefProps	SwDataDefProps	0..1	aggr	The properties of this AutosarDataType. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=swDataDefProps

Table E.19: AutosarDataType

Class	***BaseType*** (abstract)			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This abstract meta-class represents the ability to specify a platform dependent base type.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	SwBaseType			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**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 E.20: BaseType

Class	**BaseTypeDirectDefinition**			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This BaseType is defined directly (as opposite to a derived BaseType)			
Base	*ARObject*, *BaseTypeDefinition*			
Aggregated by	*BaseType*.baseTypeDefinition			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
baseTypeEncoding	BaseTypeEncodingString	0..1	attr	This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence. **Tags:** xml.sequenceOffset=90
baseTypeSize	PositiveInteger	0..1	attr	Describes the length of the data type specified in the container in bits. **Tags:** xml.sequenceOffset=70
byteOrder	ByteOrderEnum	0..1	attr	This attribute specifies the byte order of the base type. **Tags:** xml.sequenceOffset=110

▽

△

Class	**BaseTypeDirectDefinition**			
memAlignment	PositiveInteger	0..1	attr	This attribute describes the alignment of the memory object in bits. E.g. "8" specifies, that the object in question is aligned to a byte while "32" specifies that it is aligned four byte. If the value is set to "0" the meaning shall be interpreted as "unspecified". **Tags:** xml.sequenceOffset=100
native Declaration	NativeDeclarationString	0..1	attr	This attribute describes the declaration of such a base type in the native programming language, primarily in the Programming language C. This can then be used by a code generator to include the necessary declarations into a header file. For example BaseType with shortName: "MyUnsignedInt" native Declaration: "unsigned short" Results in typedef unsigned short MyUnsignedInt; If the attribute is not defined the referring Implementation DataTypes will not be generated as a typedef by RTE. If a nativeDeclaration type is given it shall fulfill the characteristic given by basetypeEncoding and baseType Size. This is required to ensure the consistent handling and interpretation by software components, RTE, COM and MCM systems. **Tags:** xml.sequenceOffset=120

Table E.21: BaseTypeDirectDefinition

Class	**CanControllerXlConfiguration**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Can::CanTopology			
Note	This meta-class represents the CAN XL-specific controller attributes.			
Base	*ARObject*			
Aggregated by	*AbstractCanCommunicationControllerAttributes*.canControllerXlAttributes, CanXlProps.canXlConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
errorSignaling Enabled	Boolean	0..1	attr	Specifies if error signaling shall be enabled. This is not possible when the transceiver is switched to PWM mode (trcvPwmModeEnabled set to TRUE). TRUE: Error signaling shall be enabled. FALSE: Error signaling shall be disabled.
propSeg	PositiveInteger	0..1	attr	Specifies propagation delay in time quantas.
pwmL	PositiveInteger	0..1	attr	Specifies the PWM long phase length.
pwmO	PositiveInteger	0..1	attr	Specifies the PWM time offset.
pwmS	PositiveInteger	0..1	attr	Specifies the PWM short phase length.
sspOffset	PositiveInteger	0..1	attr	Specifies the Transmitter Delay Compensation Offset in minimum time quanta. Transmitter Delay Compensation Offset is used to adjust the position of the Secondary Sample Point (SSP), relative to the beginning of the received bit. If this parameter is configured, the Transmitter Delay Compensation is done by measurement of the CAN controller. If not specified Transmitter Delay Compensation is disabled.

▽

△

Class	**CanControllerXlConfiguration**			
syncJumpWidth	PositiveInteger	0..1	attr	Specifies the synchronization jump width for the controller in time quantas.
timeSeg1	PositiveInteger	0..1	attr	Specifies phase segment 1 in time quantas.
timeSeg2	PositiveInteger	0..1	attr	Specifies phase segment 2 in time quantas.
trcvPwmMode Enabled	Boolean	0..1	attr	Specifies if the transceiver shall be set to the PWM mode. TRUE: The transceiver shall be switched to PWM mode. FALSE: The transceiver shall work in classic CAN mode.

Table E.22: CanControllerXlConfiguration

Class	**CanControllerXlConfigurationRequirements**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Can::CanTopology			
Note	This element allows the specification of ranges for the CAN XL configuration parameters. These ranges are taken as requirements and have to be respected by the ECU developer.			
Base	*ARObject*			
Aggregated by	*AbstractCanCommunicationControllerAttributes*.canControllerXlRequirements, CanXlProps.canXlConfig Reqs			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
errorSignaling Enabled	Boolean	0..1	attr	Specifies if error signaling shall be enabled. This is not possible when the transceiver is switched to PWM mode (trcvPwmModeEnabled set to TRUE). TRUE: Error signaling shall be enabled. FALSE: Error signaling shall be disabled.
maxNumberOf TimeQuantaPer Bit	Integer	0..1	attr	Maximum number of time quanta in the bit time.
maxPwmL	PositiveInteger	0..1	attr	Specifies the maximum PWM long phase length.
maxPwmO	PositiveInteger	0..1	attr	Specifies the minimum PWM time offset.
maxPwmS	PositiveInteger	0..1	attr	Specifies the maximum PWM short phase length.
maxSample Point	Float	0..1	attr	The max. value of the sample point as a percentage of the total bit time.
maxSyncJump Width	Float	0..1	attr	The max. Synchronization Jump Width value as a percentage of the total bit time. The (Re-)Synchronization Jump Width (SJW) defines how far a resynchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.
maxTrcvDelay Compensation Offset	TimeValue	0..1	attr	Specifies the maximum Transceiver Delay Compensation Offset in seconds. If not specified Transceiver Delay Compensation is disabled.
minNumberOf TimeQuantaPer Bit	Integer	0..1	attr	Minimum number of time quanta in the bit time.
minPwmL	PositiveInteger	0..1	attr	Specifies the minimum PWM long phase length.
minPwmO	PositiveInteger	0..1	attr	Specifies the maximum PWM time offset.
minPwmS	PositiveInteger	0..1	attr	Specifies the minimum PWM short phase length.
minSamplePoint	Float	0..1	attr	The min. value of the sample point as a percentage of the total bit time.

▽

△

Class	CanControllerXlConfigurationRequirements			
minSyncJump Width	Float	0..1	attr	The min. Synchronization Jump Width value as a percentage of the total bit time. The (Re-)Synchronization Jump Width (SJW) defines how far a resynchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.
minTrcvDelay Compensation Offset	TimeValue	0..1	attr	Specifies the minimum Transceiver Delay Compensation Offset in seconds. If not specified Transceiver Delay Compensation is disabled.
trcvPwmMode Enabled	Boolean	0..1	attr	Specifies if the transceiver shall be set to the PWM mode. TRUE: The transceiver shall be switched to PWM mode. FALSE: The transceiver shall work in classic CAN mode.

Table E.23: CanControllerXlConfigurationRequirements

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*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
possibleError	ApplicationError	*	aggr	Application errors that are defined as part of this interface.

Table E.24: ClientServerInterface

Class	Collection			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ElementCollection			
Note	This meta-class specifies a collection of elements. A collection can be utilized to express additional aspects for a set of elements. Note that Collection is an ARElement. Therefore it is applicable e.g. for EvaluatedVariant, even if this is not obvious. Usually the category of a Collection is "SET". On the other hand, a Collection can also express an arbitrary relationship between elements. This is denoted by the category "RELATION" (see also [TPS_GST_00347]). In this case the collection represents an association from "sourceElement" to "targetElement" in the role "role". **Tags:** atp.recommendedPackage=Collections			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
autoCollect	AutoCollectEnum	0..1	attr	This attribute reflects how far the referenced objects are part of the collection. **Tags:** xml.sequenceOffset=20

▽

△

Class	**Collection**			
collected Instance	AtpFeature	*	iref	This instance ref supports the use case that a particular instance is part of the collection. **Tags:** xml.sequenceOffset=60 **InstanceRef implemented by:** AnyInstanceRef
collection Semantics	NameToken	0..1	attr	Provides the ability to express the semantics of a Collection depending on the intended use case. The collectionSemantics is specified as a NameToken which must be agreed by all stakeholders. **Tags:** xml.sequenceOffset=25
element	Identifiable	*	ref	This is an element in the collection. Note that Collection itself is collectable. Therefore collections can be nested. In case of category="RELATION" this represents the target end of the relation. **Tags:** xml.sequenceOffset=40
elementRole	Identifier	0..1	attr	This attribute allows to denote a particular role of the collection. Note that the applicable semantics shall be mutually agreed between the two parties. In particular it denotes the role of element in the context of sourceElement. **Tags:** xml.sequenceOffset=30
sourceElement	Identifiable	*	ref	Only if Category = "RELATION". This represents the source of a relation. **Tags:** xml.sequenceOffset=50
sourceInstance	AtpFeature	*	iref	Only if Category = "RELATION". This represents the source instance of a relation. **Tags:** xml.sequenceOffset=70 **InstanceRef implemented by:** AnyInstanceRef

Table E.25: Collection

Class	***CommConnectorPort*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	The Ecu communication relationship defines which signals, Pdus and frames are actually received and transmitted by this ECU. For each signal, Pdu or Frame that is transmitted or received and used by the Ecu an association between an ISignalPort, IPduPort or FramePort with the corresponding Triggering shall be created. An ISignalPort shall be created only if the corresponding signal is handled by COM (RTE or Signal Gateway). If a Pdu Gateway ECU only routes the Pdu without being interested in the content only a FramePort and an IPduPort needs to be created.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	FramePort, IPduPort, ISignalPort			
Aggregated by	*CommunicationConnector*.ecuCommPortInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Direction	Communication DirectionType	0..1	attr	Communication Direction of the Connector Port (input or output Port).

Table E.26: CommConnectorPort

Enumeration	**CommunicationDirectionType**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication
Note	Describes the communication direction.
Aggregated by	*CommConnectorPort*.communicationDirection, *IEEE1722TpConnection*.communicationDirection, IPSecRule.direction, ISignalIPduGroup.communicationDirection
Literal	***Description***
in	Reception (Input) **Tags:** atp.EnumerationLiteralIndex=0
out	Transmission (Output) **Tags:** atp.EnumerationLiteralIndex=1

Table E.27: CommunicationDirectionType

Class	**CompositeRuleBasedValueSpecification**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This meta-class represents rule-based values for DataPrototypes typed by composite AutosarDataTypes.			
Base	*ARObject*, *AbstractRuleBasedValueSpecification*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValueSpecification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, ConstantSpecification.valueSpec, CryptoServiceKey.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElementCondition.compareValue, FieldSenderComSpec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, SomeipEventDeployment.eventReceptionDefaultValue, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableDataPrototype.initValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument (ordered)	CompositeValueSpecification	*	aggr	This represents the collection of aggregated Value Specifications. The last ValueSpecification in the collection shall be taken to execute the filling rule. **Tags:** xml.sequenceOffset=30
compound Primitive Argument (ordered)	CompositeRuleBasedValueArgument	*	aggr	This represents the collection of aggregated Value Specifications for compound primitive data type. The last ValueSpecification in the collection shall be taken to execute the filling rule. **Tags:** xml.sequenceOffset=35
maxSizeToFill	PositiveInteger	0..1	attr	If a rule is chosen which does not fill until the end, this determines until which size the rule shall fill the values. **Tags:** xml.sequenceOffset=40
rule	Identifier	0..1	attr	This denotes the name of the rule of the RuleBasedValueSpecification. The rule determines the calculation specification according which the arguments are used to calculated the values. **Tags:** xml.sequenceOffset=20

Table E.28: CompositeRuleBasedValueSpecification

Class	CompuConst			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the fact that the value of a computation method scale is constant.			
Base	*ARObject*			
Aggregated by	Compu.compuDefaultValue, CompuScale.compuInverseValue, CompuScaleConstantContents.compuConst			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
compuConstContentType	CompuConstContent	0..1	aggr	This is the actual content of the constant compu method scale. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=10 xml.typeElement=false xml.typeWrapperElement=false

Table E.29: CompuConst

Class	CompuConstTextContent			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the textual content of a scale.			
Base	*ARObject*, *CompuConstContent*			
Aggregated by	CompuConst.compuConstContentType			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
vt	VerbatimString	0..1	attr	This represents a textual constant in the computation method.

Table E.30: CompuConstTextContent

Class	CompuMethod			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to express the relationship between a physical value and the mathematical representation. Note that this is still independent of the technical implementation in data types. It only specifies the formula how the internal value corresponds to its physical pendant. **Tags:** atp.recommendedPackage=CompuMethods			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
compuInternalToPhys	Compu	0..1	aggr	This specifies the computation from internal values to physical values. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=compuInternalToPhys xml.sequenceOffset=80
compuPhysToInternal	Compu	0..1	aggr	This represents the computation from physical values to the internal values. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=compuPhysToInternal xml.sequenceOffset=90

△

Class	**CompuMethod**			
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 E.31: CompuMethod

Class	**CompuScale**			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to specify one segment of a segmented computation method.			
Base	*ARObject*			
Aggregated by	CompuScales.compuScale			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
a2lDisplayText	String	0..1	attr	The value of this attribute shall be taken for generating one display text (specifically the OutVal) within the equivalent of the enclosing `CompuMethod` in A2L.
compuInverse Value	CompuConst	0..1	aggr	This is the inverse value of the constraint. This supports the case that the scale is not reversible per se. **Tags:** xml.sequenceOffset=60
compuScale Contents	CompuScaleContents	0..1	aggr	This represents the computation details of the scale. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=70 xml.typeElement=false xml.typeWrapperElement=false
desc	MultiLanguageOverview Paragraph	0..1	aggr	<desc> represents a general but brief description of the object in question. **Tags:** xml.sequenceOffset=30
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=40
mask	PositiveUnlimitedInteger	0..1	attr	In difference to all the other computational methods every COMPU-SCALE will be applied including the bit MASK. Therefore it is allowed for this type of COMPU-METHOD, that COMPU-SCALES overlap. To calculate the string reverse to a value, the string has to be split and the according value for each substring has to be summed up. The sum is finally transmitted. The processing has to be done in order of the COMPU-SCALE elements. **Tags:** xml.sequenceOffset=35
shortLabel	Identifier	0..1	attr	This element specifies a short name for the particular scale. The name can for example be used to derive a programming language identifier. **Tags:** xml.sequenceOffset=20

▽

△

Class	CompuScale			
symbol	CIdentifier	0..1	attr	The symbol, if provided, is used by code generators to get a C identifier for the CompuScale. The name will be used as is for the code generation, therefore it needs to be unique within the generation context. **Tags:** xml.sequenceOffset=25
upperLimit	Limit	0..1	attr	This specifies the upper limit of a of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=50

Table E.32: CompuScale

Class	ConstantReference			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Instead of defining this value inline, a constant is referenced.			
Base	*ARObject*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValueSpecification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, ConstantSpecification.valueSpec, CryptoServiceKey.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElementCondition.compareValue, FieldSenderComSpec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, SomeipEventDeployment.eventReceptionDefaultValue, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableDataPrototype.initValue			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
constant	ConstantSpecification	0..1	ref	The referenced constant.

Table E.33: ConstantReference

Class	CouplingElement			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	A CouplingElement is used to connect EcuInstances to the VLAN of an EthernetCluster. Coupling Elements can reach from a simple hub to a complex managed switch or even devices with functionalities in higher layers. A CouplingElement that is not related to an EcuInstance occurs as a dedicated single device. **Tags:** atp.recommendedPackage=CouplingElements			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
communicationCluster	EthernetCluster	0..1	ref	This relationship defines to which cluster the CouplingElement belongs.

▽

△

Class	**CouplingElement**			
couplingPort	CouplingPort	*	aggr	Hardware Port of the CouplingElement that is used to connect this CouplingPort to EcuInstances or other CouplingElements. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=couplingPort.shortName, coupling Port.variationPoint.shortLabel vh.latestBindingTime=postBuild
couplingType	CouplingElementEnum	0..1	attr	Describes the coupling type of this CouplingElement.
ecuInstance	EcuInstance	0..1	ref	Optional reference to the ECU where the Coupling Element is located.
firewallRule	StateDependentFirewall	*	ref	Firewall rules defined in the context of a Coupling Element. **Tags:** atp.Status=candidate

Table E.34: CouplingElement

Class	**CouplingPort**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	A CouplingPort is used to connect a CouplingElement with an EcuInstance or two CouplingElements with each other via a CouplingPortConnection. Optionally, the CouplingPort may also have a reference to a macMulticastGroup and a defaultVLAN.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	CouplingElement.couplingPort, EthernetCommunicationController.couplingPort			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
connection Negotiation Behavior	EthernetConnection NegotiationEnum	0..1	attr	Specifies the connection negotiation of the CouplingPort.
couplingPort Details	CouplingPortDetails	0..1	aggr	Defines more details of a CouplingPort in case a more specific configuration is required.
couplingPort Role	CouplingPortRoleEnum	0..1	attr	Defines the role this CouplingPort takes in the context of the CouplingElement.
defaultVlan	EthernetPhysical Channel	0..1	ref	The vLanIdentifier of the referenced VLAN is the Default-PVID (port VLAN ID). A Port VLAN ID is a default VLAN ID that is assigned to an access CouplingPort to designate the VLAN segment to which this port is connected. Also, if a CouplingPort has not been configured with any VLAN memberships, the virtual switch's Port VLAN ID (pvid) becomes the default VLAN ID for the ports connection. This identifier/tag is added for incoming untagged messages at the port (ingress tagging). For outgoing messages with this identifier, the tag is removed at the port (egress untagging, depending on the Vlan Membership.sendActivity).
macLayerType	EthernetMacLayerType Enum	0..1	attr	Specifies the mac layer type of the CouplingPort.
macMulticast Address	MacMulticastGroup	*	ref	Assigns a set of MAC-Multicast-Addresses which are addressable via this CouplingPort. This is a static pre-configuration and further addresses may be learned during runtime.

▽

△

Class	**CouplingPort**			
macSecProps	MacSecProps	*	aggr	Properties to configure MACsec (Media access control security) and the MKA (MACsec Key Agreement) for the CouplingPort (PHY). **Tags:** atp.Status=candidate
physicalLayer Type	EthernetPhysicalLayer TypeEnum	0..1	attr	Specifies the physical layer type of the CouplingPort.
plcaProps	PlcaProps	0..1	aggr	Optional properties for configuration of PLCA (Physical Layer Collision Avoidance) in case 10-BASE-T1S Ethernet is used and PLCA is enabled on the Coupling Port (PHY).
pncMapping	PncMappingIdent	*	ref	Reference to the partial networks this CouplingPort participates in. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=pncMapping
receiveActivity	EthernetSwitchVlan IngressTagEnum	0..1	attr	Defines the handling of frames at the ingress port.
vlan Membership	VlanMembership	*	aggr	Messages of VLANs that are defined here can be communicated via the CouplingPort.
vlanModifier	EthernetPhysical Channel	0..1	ref	All incoming messages at this CouplingPort shall be tagged with this VLAN Id. This tagging is performed regardless whether the message already has a VLAN tag or is untagged, an existing VLAN tag will be overwritten. This feature is XOR with CoupligPort.defaultVlan.
wakeupSleep OnDataline Config	EthernetWakeupSleep OnDatalineConfig	0..1	ref	Optional reference to EthernetWakeupSleepOnDataline Config.

Table E.35: CouplingPort

Class	**CouplingPortConnection**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Connection between two CouplingPorts (firstPort and secondPort) or between a collection of Ports that are all referenced by the portCollection reference.			
Base	*ARObject*			
Aggregated by	EthernetCluster.couplingPortConnection			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
firstPort	CouplingPort	0..1	ref	Reference to the first CouplingPort that is connected via the CouplingPortConnection.
nodePort	CouplingPort	*	ref	Reference to a number of CouplingPorts that are connected via the CouplingPortConnection. This reference shall be used to describe a 10BASE-T1S topology architecture where several CouplingPorts of EthernetCommunicationControllers are connected via one CouplingPortConnection. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=nodePort.couplingPort, nodePort.variation Point.shortLabel vh.latestBindingTime=postBuild
plcaLocalNode Count	PositiveInteger	0..1	attr	Defines the number of communication participants in case 10BASE-T1S and the nodePort reference is used.

▽

△

Class	CouplingPortConnection			
plcaTransmit Opportunity Timer	PositiveInteger	0..1	attr	Timer for the transmission in bit time to evaluate if a Transmission Opportunity is yield or not.
secondPort	CouplingPort	0..1	ref	Reference to the second CouplingPort that is connected via the CouplingPortConnection.

Table E.36: CouplingPortConnection

Class	DataConstr			
Package	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
Note	This meta-class represents the ability to specify constraints on data. **Tags:** atp.recommendedPackage=DataConstrs			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataConstrRule	DataConstrRule	*	aggr	This is one particular rule within the data constraints. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

Table E.37: DataConstr

Class	DataConstrRule			
Package	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
Note	This meta-class represents the ability to express one specific data constraint rule.			
Base	*ARObject*			
Aggregated by	DataConstr.dataConstrRule			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
constrLevel	Integer	0..1	attr	This attribute describes the category of a constraint. One of its functions is in the area of constraint violation, where it can be used from a certain level, to produce error messages. The lower the level, the more stringent the check. Used to distinguish hard or soft limits. **Tags:** xml.sequenceOffset=20
internalConstrs	InternalConstrs	0..1	aggr	Describes the limitations applicable on the internal domain (as opposed to the physical domain). **Tags:** xml.sequenceOffset=40
physConstrs	PhysConstrs	0..1	aggr	Describes the limitations applicable on the physical domain (as opposed to the internal domain). **Tags:** xml.sequenceOffset=30

Table E.38: DataConstrRule

Class	**DataFilter**			
Package	M2::AUTOSARTemplates::CommonStructure::Filter			
Note	Base class for data filters. The type of the filter is specified in attribute dataFilterType. Some of the filter types require additional arguments which are specified as attributes of this class.			
Base	*ARObject*			
Aggregated by	ISignalPort.dataFilter, NonqueuedReceiverComSpec.filter, NonqueuedSenderComSpec.dataFilter, SignalBasedEventElementToISignalTriggeringMapping.filter, SignalBasedFieldToISignalTriggering Mapping.filter, SignalServiceTranslationElementProps.filter, TransmissionModeCondition.dataFilter			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataFilterType	DataFilterTypeEnum	0..1	attr	This attribute specifies the type of the filter.
mask	UnlimitedInteger	0..1	attr	Mask for old and new value.
max	UnlimitedInteger	0..1	attr	Value to specify the upper boundary
min	UnlimitedInteger	0..1	attr	Value to specify the lower boundary
offset	PositiveInteger	0..1	attr	Specifies the initial number of messages to occur before the first message is passed
period	PositiveInteger	0..1	attr	Specifies number of messages to occur before the message is passed again
x	UnlimitedInteger	0..1	attr	Value to compare with

Table E.39: DataFilter

Enumeration	**DataFilterTypeEnum**
Package	M2::AUTOSARTemplates::CommonStructure::Filter
Note	This enum specifies the supported DataFilterTypes.
Aggregated by	DataFilter.dataFilterType
Literal	***Description***
always	No filtering is performed so that the message always passes. **Tags:** atp.EnumerationLiteralIndex=0
maskedNewDiffers MaskedOld	Pass messages where the masked value has changed. (new_value&mask) !=(old_value&mask) new_value: current value of the message old_value: last value of the message (initialized with the initial value of the message, updated with new_value if the new message value is not filtered out) **Tags:** atp.EnumerationLiteralIndex=1
maskedNewDiffers X	Pass messages whose masked value is not equal to a specific value x (new_value&mask) != x new_value: current value of the message **Tags:** atp.EnumerationLiteralIndex=2
maskedNewEquals X	Pass messages whose masked value is equal to a specific value x (new_value&mask) == x new_value: current value of the message **Tags:** atp.EnumerationLiteralIndex=3
never	The filter removes all messages. **Tags:** atp.EnumerationLiteralIndex=4
newIsOutside	Pass a message if its value is outside a predefined boundary. (min > new_value) OR (new_value > max) **Tags:** atp.EnumerationLiteralIndex=5

▽

△

Enumeration	**DataFilterTypeEnum**
newIsWithin	Pass a message if its value is within a predefined boundary. min <= new_value <= max **Tags:** atp.EnumerationLiteralIndex=6
oneEveryN	Pass a message once every N message occurrences. Algorithm: occurrence % period == offset Start: occurrence = 0. Each time the message is received or transmitted, occurrence is incremented by 1 after filtering. Length of occurrence is 8 bit (minimum). **Tags:** atp.EnumerationLiteralIndex=7

Table E.40: DataFilterTypeEnum

Class	***DataPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Base class for prototypical roles of any data type.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*ApplicationCompositeElementDataPrototype*, *AutosarDataPrototype*			
Aggregated by	*AtpClassifier*.atpFeature			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
swDataDefProps	SwDataDefProps	0..1	aggr	This property allows to specify data definition properties which apply on data prototype level. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=swDataDefProps

Table E.41: DataPrototype

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

Table E.42: DataTransformation

Class	**DiagnosticAbstractParameter** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents an abstract base class for modeling a diagnostic parameter.			
Base	*ARObject*			
Subclasses	DiagnosticParameter, DiagnosticParameterElement			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
bitOffset	PositiveInteger	0..1	attr	This represents the bitOffset of the DiagnosticParameter. The value of the bitOffset shall always be interpreted as relative to the start of the enclosing DiagnosticData Identifier, DiagnosticParameterIdentifier, or Diagnostic RoutineSubfunction. **Stereotypes:** atpIdentityContributor **Tags:** atp.Status=candidate
dataElement	DiagnosticDataElement	0..1	aggr	This represents the related dataElement of the Diagnostic Parameter **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataElement.shortName, data Element.variationPoint.shortLabel vh.latestBindingTime=postBuild
parameterSize	PositiveInteger	0..1	attr	This attribute allows for the specification of the parameter size. This information is relevant if there is a gap between one diagnostic parameter and the following diagnostic parameter (or the tail of the telegram). The unit is bit and the values shall be multiples of 8. **Tags:** atp.Status=candidate

Table E.43: DiagnosticAbstractParameter

Class	**DiagnosticAccessPermission**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm			
Note	This represents the specification of whether a given service can be accessed according to the existence of meta-classes referenced by a particular DiagnosticAccessPermission. In other words, this meta-class acts as a mapping element between several (otherwise unrelated) pieces of information that are put into context for the purpose of checking for access rights. **Tags:** atp.recommendedPackage=DiagnosticAccessPermissions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
authentication Enabled Proxy	DiagnosticAuthRole	0..1	aggr	The existence of this aggregation indicates that an authentication is foreseen. The details are clarified by the aggregated class. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=authenticationEnabled
diagnostic Session	DiagnosticSession	*	ref	This represents the associated DiagnosticSessions **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=diagnosticSession
environmental Condition	Diagnostic EnvironmentalCondition	0..1	ref	This represents the environmental conditions associated with the access permission. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=environmentalCondition

△

Class	**DiagnosticAccessPermission**			
securityLevel	DiagnosticSecurityLevel	*	ref	This represents the associated DiagnosticSecurityLevels **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=securityLevel
sovdLock	DiagnosticSovdLock	0..1	ref	This represents the associated SOVD lock. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=sovdLock atp.Status=candidate

Table E.44: DiagnosticAccessPermission

Class	***DiagnosticCondition*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticCondition			
Note	Abstract element for StorageConditions and EnableConditions.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticClearCondition, DiagnosticEnableCondition, DiagnosticStorageCondition			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initValue	Boolean	0..1	attr	Defines the initial status for enable or disable of acceptance/storage of event reports of a diagnostic event. The value is the initialization after power up (before this condition is reported the first time). true: acceptance/storage of a diagnostic event enabled false: acceptance/storage of a diagnostic event disabled

Table E.45: DiagnosticCondition

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 DiagnosticContribution Set. **Tags:** atp.recommendedPackage=DiagnosticContributionSets			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
common Properties	DiagnosticCommon Props	0..1	aggr	This attribute represents a collection of diagnostic properties that are shared among the entire Diagnostic ContributionSet. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=commonProperties

▽

△

Class	**DiagnosticContributionSet**			
element	DiagnosticCommon Element	*	ref	This represents a DiagnosticCommonElement considered in the context of the DiagnosticContributionSet **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=element.diagnosticCommonElement, element.variationPoint.shortLabel vh.latestBindingTime=postBuild
serviceTable	DiagnosticServiceTable	*	ref	This represents the collection of DiagnosticServiceTables to be considered in the scope of this Diagnostic ContributionSet. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serviceTable.diagnosticServiceTable, service Table.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table E.46: DiagnosticContributionSet

Class	**DiagnosticDataTransfer**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::MemoryByAddress			
Note	This represents an instance of the "Data Transfer" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticMemoryByAdresss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMemoryByAddress*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dataTransfer Class	DiagnosticDataTransfer Class	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticDataTransfer in the given context.

Table E.47: DiagnosticDataTransfer

Class	**DiagnosticIndicator**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticIndicator			
Note	Definition of an indicator. **Tags:** atp.recommendedPackage=DiagnosticIndicators			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
type	DiagnosticIndicatorType Enum	0..1	attr	Defines the type of the indicator. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table E.48: DiagnosticIndicator

Class	***DiagnosticMapping*** (abstract)			
Package	M2::AUTOSARTemplates::DiagnosticExtract::DiagnosticMapping			
Note	Abstract element for different kinds of diagnostic mappings.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	DiagnosticEventToDebounceAlgorithmMapping, DiagnosticEventToEnableConditionGroupMapping, DiagnosticEventToOperationCycleMapping, DiagnosticEventToSecurityEventMapping, DiagnosticEventToTroubleCodeUdsMapping, DiagnosticFimAliasEventGroupMapping, DiagnosticFimAliasEventMapping, DiagnosticInhibitSourceEventMapping, DiagnosticMasterToSlaveEventMapping, DiagnosticProvidedDataMapping, DiagnosticSecureCodingMapping, DiagnosticSovdConfigurationDataIdentifierMapping, *DiagnosticSwMapping*, DiagnosticTroubleCodeUdsToClearConditionGroupMapping, DiagnosticTroubleCodeUdsToTroubleCodeObdMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.49: DiagnosticMapping

Class	**DiagnosticParameter**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::CommonDiagnostics			
Note	This meta-class represents the ability to describe information relevant for the execution of a specific diagnostic service, i.e. it can be taken to parameterize the service.			
Base	*ARObject*, *DiagnosticAbstractParameter*			
Aggregated by	DiagnosticDataIdentifier.dataElement, DiagnosticExtendedDataRecord.recordElement, DiagnosticInfoType.dataElement, DiagnosticParameterIdentifier.dataElement, DiagnosticRequestRoutineResults.request, DiagnosticRequestRoutineResults.response, DiagnosticStartRoutine.request, DiagnosticStartRoutine.response, DiagnosticStopRoutine.request, DiagnosticStopRoutine.response			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ident	DiagnosticParameterIdent	0..1	aggr	The aggregation in the role ident provides the ability to make the DiagnosticAbstractParameter identifiable. From the semantical point of view, the AbstractDiagnosticParameter is considered a first-class Identifiable and therefore the aggregation in the role ident shall always exist (until it may be possible to let AbstractDiagnosticParameter directly inherit from Identifiable). **Stereotypes:** atpIdentityContributor
supportInfo	DiagnosticParameterSupportInfo	0..1	aggr	This attribute represents the ability to define which bit of the support info byte is representing this part of the PID.

Table E.50: DiagnosticParameter

Class	***DiagnosticServiceInstance*** (abstract)
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::CommonService
Note	This represents a concrete instance of a diagnostic service.
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

Class	**DiagnosticServiceInstance** (abstract)			
Subclasses	*DiagnosticAuthentication*, DiagnosticClearDiagnosticInformation, DiagnosticClearResetEmissionRelated Info, DiagnosticComControl, DiagnosticControlDTCSetting, DiagnosticCustomServiceInstance, *DiagnosticDataByIdentifier*, DiagnosticDynamicallyDefineDataIdentifier, DiagnosticEcuReset, Diagnostic IOControl, *DiagnosticMemoryByAddress*, DiagnosticReadDTCInformation, DiagnosticReadDataBy PeriodicID, DiagnosticRequestControlOfOnBoardDevice, DiagnosticRequestCurrentPowertrainData, DiagnosticRequestEmissionRelatedDTC, DiagnosticRequestEmissionRelatedDTCPermanentStatus, DiagnosticRequestFileTransfer, DiagnosticRequestOnBoardMonitoringTestResults, DiagnosticRequest PowertrainFreezeFrameData, DiagnosticRequestVehicleInfo, DiagnosticResponseOnEvent, Diagnostic RoutineControl, DiagnosticSecurityAccess, DiagnosticSessionControl			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
access Permission	DiagnosticAccess Permission	0..1	ref	This represents the collection of DiagnosticAccess Permissions that allow for the execution of the referencing DiagnosticServiceInstance.. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=accessPermission
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 DiagnosticService Instance. The subclasses that affected by this pattern implement references to the applicable "class"-role that substantiate this abstract reference. **Stereotypes:** atpAbstract

Table E.51: DiagnosticServiceInstance

Class	**DiagnosticTransferExit**			
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dcm::DiagnosticService::MemoryByAddress			
Note	This represents an instance of the "Transfer Exit" diagnostic service. **Tags:** atp.recommendedPackage=DiagnosticMemoryByAdresss			
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticMemoryByAddress*, *DiagnosticServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
transferExit Class	DiagnosticTransferExit Class	0..1	ref	This reference substantiates that abstract reference in the role serviceClass for this specific concrete class. Thereby, the reference represents the ability to access shared attributes among all DiagnosticTransferExit in the given context.

Table E.52: DiagnosticTransferExit

Class	**DiagnosticTroubleCodeUds**
Package	M2::AUTOSARTemplates::DiagnosticExtract::Dem::DiagnosticTroubleCode
Note	This element is used to describe non OBD-relevant DTCs. **Tags:** atp.recommendedPackage=DiagnosticTroubleCodes
Base	*ARElement*, *ARObject*, *CollectableElement*, *DiagnosticCommonElement*, *DiagnosticTroubleCode*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*
Aggregated by	ARPackage.element

▽

△

Class	**DiagnosticTroubleCodeUds**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
considerPto Status	Boolean	0..1	attr	This attribute describes the affection of the event by the Dem PTO handling. true: the event is affected by the Dem PTO handling. false: the event is not affected by the Dem PTO handling. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
dtcProps	DiagnosticTroubleCode Props	0..1	ref	Defined properties associated with the DemDTC.
eventReadiness Group	EventObdReadiness Group	0..1	aggr	This attribute specifies the Event OBD Readiness group for PID $01 and PID $41 computation. This attribute is only applicable for emission-related ECUs. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=eventReadinessGroup.eventObdReadiness Group, eventReadinessGroup.variationPoint.shortLabel vh.latestBindingTime=postBuild
functionalUnit	PositiveInteger	0..1	attr	This attribute specifies a 1-byte value which identifies the corresponding basic vehicle / system function which reports the DTC. This parameter is necessary for the report of severity information. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
obdDtc Value3Byte	PositiveInteger	0..1	attr	3 Byte OBD DTC value based on the definition from SAE J2012. The existence of this attribute is only required if separated UDS and OBD DTC values are used for SAE J1979-2. If this attribute does not exist, then UDS DTC values are used with J1979-2. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
severity	DiagnosticUdsSeverity Enum	0..1	attr	DTC severity according to ISO 14229-1. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
udsDtcValue	PositiveInteger	0..1	attr	Unique Diagnostic Trouble Code value for UDS. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
wwhObdDtc Class	DiagnosticWwhObdDtc ClassEnum	0..1	attr	This attribute is used to identify (if applicable) the corresponding severity class of an WWH-OBD DTC. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table E.53: DiagnosticTroubleCodeUds

Class	**DltApplication**			
Package	M2::AUTOSARTemplates::LogAndTraceExtract			
Note	This meta-class represents the application from which the log and trace message originates.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	DltEcu.application			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**DltApplication**			
application Description	String	0..1	attr	This attribute can be used to describe the applicationId that is used in the log and trace message in more detail.
applicationId	String	0..1	attr	This attribute identifies the SW-C/BSW module in the log and trace message.
context	DltContext	*	ref	Definition of ContextIds for the Application. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=context.dltContext, context.variation Point.shortLabel vh.latestBindingTime=systemDesignTime

Table E.54: DltApplication

Class	**DltContext**			
Package	M2::AUTOSARTemplates::LogAndTraceExtract			
Note	This meta-class represents the Context that groups Log and Trace Messages that are generated by an application. **Tags:** atp.recommendedPackage=DltContexts			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
context Description	String	0..1	attr	This attribute can be used to describe the contextId that is used in the log and trace message in more detail.
contextId	String	0..1	attr	This attribute is used to group log and trace messages produced by an application to distinguish functionality.
dltMessage	DltMessage	*	ref	Group of Log and Trace Messages assigned to the Dlt Context **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dltMessage.dltMessage, dlt Message.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime

Table E.55: DltContext

Class	**DltEcu**			
Package	M2::AUTOSARTemplates::LogAndTraceExtract			
Note	This element represents an Ecu or Machine that produces logging and tracing information. **Tags:** atp.recommendedPackage=DltEcus			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
application	DltApplication	*	aggr	Application on DltEcu that provides log or trace data. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=application.shortName, application.variation Point.shortLabel vh.latestBindingTime=systemDesignTime
ecuId	String	0..1	attr	This attribute defines the name of the ECU for use within the Dlt protocol.

Table E.56: DltEcu

Class	**DltMessage**			
Package	M2::AUTOSARTemplates::LogAndTraceExtract			
Note	This element defines a DltMessage.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	LogAndTraceMessageCollectionSet.dltMessage			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dltArgument (ordered)	DltArgument	*	aggr	Ordered collection of DltArguments in the DltMessage.
messageId	PositiveInteger	0..1	attr	This attribute defines the unique Id for the DltMessage.
messageLine Number	PositiveInteger	0..1	attr	This attribute describes the position in the source file in which this log message was called.
messageSource File	String	0..1	attr	This attribute describes the source file in which this log message was called.
messageType Info	String	0..1	attr	This attribute describes the message Type
privacyLevel	PrivacyLevel	0..1	aggr	The Privacy Level helps to identify the Log and Trace content towards the degree of privacy to it.

Table E.57: DltMessage

Enumeration	**DoIpEidRetrievalEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation
Note	Enumeration with options to retrieve EID.
Aggregated by	DoIpNetworkConfiguration.eidRetrieval
Literal	***Description***
eidUseApi	API DiagnosticDoIPEntityIdentification is used to retrieve eid **Tags:** atp.EnumerationLiteralIndex=1
eidUseConfigValue	eid is configured manually by DoIpInstantiation.eid **Tags:** atp.EnumerationLiteralIndex=2
eidUseMac	MAC of the network interface is used as eid **Tags:** atp.EnumerationLiteralIndex=0

Table E.58: DoIpEidRetrievalEnum

Class	**EcuInstance**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	ECUInstances are used to define the ECUs used in the topology. The type of the ECU is defined by a reference to an ECU specified with the ECU resource description. **Tags:** atp.recommendedPackage=EcuInstances			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
associated Consumed Provided ServiceInstance Group	ConsumedProvided ServiceInstanceGroup	*	ref	With this reference it is possible to identify which ConsumedProvidedServiceInstanceGroups are applicable for which ECUInstance. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=associatedConsumedProvidedService InstanceGroup.consumedProvidedServiceInstanceGroup, associatedConsumedProvidedServiceInstance Group.variationPoint.shortLabel vh.latestBindingTime=postBuild
associatedPdur IPduGroup	PdurIPduGroup	*	ref	With this reference it is possible to identify which PduR IPdu Groups are applicable for which Communication Connector/ ECU.
channel Synchronous Wakeup	Boolean	0..1	attr	If this parameter is available and set to true, then all available channels will be woken up as soon as at least one channel wakeup occurs. If PNCs are configured, then all PNCs will be requested upon a channel wakeup.
clientIdRange	ClientIdRange	0..1	aggr	Restriction of the Client Identifier for this Ecu to an allowed range of numerical values. The Client Identifier of the transaction handle is generated by the client RTE for inter-Ecu Client/Server communication.
commController	Communication Controller	*	aggr	CommunicationControllers of the ECU. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=commController.shortName, comm Controller.variationPoint.shortLabel vh.latestBindingTime=postBuild
connector	Communication Connector	*	aggr	All channels controlled by a single controller. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=connector.shortName, connector.variation Point.shortLabel vh.latestBindingTime=postBuild
firewallRule	StateDependentFirewall	*	ref	Firewall rules defined in the context of an EcuInstance. **Tags:** atp.Status=candidate
pncNmRequest	Boolean	0..1	attr	Defines if this EcuInstance shall request Nm on all its PhysicalChannels which have Nm variant set to FULL each time a PNC is requested.
pncPrepare SleepTimer	TimeValue	0..1	attr	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.
pnc Synchronous Wakeup	Boolean	0..1	attr	If this parameter is available and set to true then all available PNCs will be woken up as soon as a channel wakeup occurs. This is ensured by adding all PNCs to all channel wakeup sources during upstream mapping.
pnResetTime	TimeValue	0..1	attr	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests in the EIRA and in the ERA.

Table E.59: EcuInstance

Class	**EndToEndTransformationDescription**			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	EndToEndTransformationDescription holds these attributes which are profile specific and have the same value for all E2E transformers.			
Base	*ARObject*, *Describable*, *TransformationDescription*			
Aggregated by	TransformationTechnology.transformationDescription			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clearFromValid ToInvalid	Boolean	0..1	attr	Clear monitoring window on transition from state Valid to state Invalid.
counterOffset	PositiveInteger	0..1	attr	Offset of the counter in the Data[] array in bits.
crcOffset	PositiveInteger	0..1	attr	Offset of the CRC in the Data[] array in bits.
dataIdMode	DataIdModeEnum	0..1	attr	This attribute describes the inclusion mode that is used to include the implicit two-byte Data ID in the one-byte CRC.
dataIdNibble Offset	PositiveInteger	0..1	attr	Offset of the Data ID nibble in the Data[] array in bits.
e2eProfile Compatibility Props	E2EProfileCompatibility Props	0..1	ref	Reference to additional settings for the E2E state machine.
maxDelta Counter	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 Max DeltaCounter is 3, then at the next reception the receiver can accept Counters with values 2, 3 or 4.
maxErrorState Init	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_INIT.
maxErrorState Invalid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_INVALID.
maxErrorState Valid	PositiveInteger	0..1	attr	Maximal number of checks in which ProfileStatus equal to E2E_P_ERROR was determined, within the last Window Size checks, for the state E2E_SM_VALID.
maxNoNewOr RepeatedData	PositiveInteger	0..1	attr	The maximum allowed amount of consecutive failed counter checks.
minOkStateInit	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INIT.
minOkState Invalid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_INVALID.
minOkState Valid	PositiveInteger	0..1	attr	Minimal number of checks in which ProfileStatus equal to E2E_P_OK was determined, within the last WindowSize checks, for the state E2E_SM_VALID.
offset	PositiveInteger	0..1	attr	Offset of the E2E header in the Data[] array in bits.
profileBehavior	EndToEndProfile BehaviorEnum	0..1	attr	Behavior of the check functionality
profileName	NameToken	0..1	attr	Definition of the E2E profile.
syncCounterInit	PositiveInteger	0..1	attr	Number of checks required for validating the consistency of the counter that shall be received with a valid counter (i.e. counter within the allowed lock-in range) after the detection of an unexpected behavior of a received counter.

△

Class	**EndToEndTransformationDescription**			
upperHeader BitsToShift	PositiveInteger	0..1	attr	This attribute describes the number of upper-header bits to be shifted. value = 0 or not present: shift of upper header is NOT performed. value > 0: the E2E Transformer on the protect-side, takes the first upperHeaderBitsToShift bits from the upper buffer (e.g. SOME/IP header part generated by SOME/IP transformer) and shifts them towards the lower bytes and bits within the Data[] for the length of the E2E header (e.g. 12 bytes in case of E2E Profile 4). This means the shift distance is fixed - it depends on the E2E header size - what is configured here is the number of bits that are to be shifted. This option is defined because the Some/IP header generated by SOME/IP transformer shall be, due to compatibility between non-protected and E2E-protected communication, at the same position, which is before E2E header.
windowSizeInit	PositiveInteger	0..1	attr	Size of the monitoring window of state Init for the E2E state machine.
windowSize Invalid	PositiveInteger	0..1	attr	Size of the monitoring window of state Invalid for the E2E state machine.
windowSize Valid	PositiveInteger	0..1	attr	Size of the monitoring window of state Valid for the E2E state machine.

Table E.60: EndToEndTransformationDescription

Class	<<atpVariation>> **EthernetCommunicationController**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Ethernet specific communication port attributes.			
Base	*ARObject*, *CommunicationController*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EcuInstance.commController, MachineDesign.communicationController			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
canXlConfig	AbstractCan Communication Controller	0..1	ref	If the Ethernet frames handled by this Ethernet CommunicationController are to be tunneled through CAN XL, then this reference shall refer to the Abstract CanCommunicationController that aggregates the Can ControllerXlConfiguration of the physical CAN XL channel to be used for tunneling.
couplingPort	CouplingPort	*	aggr	Optional CouplingPort that can be used to connect the ECU to a CouplingElement (e.g. a switch).
macLayerType	EthernetMacLayerType Enum	0..1	attr	Specifies the mac layer type of the ethernet controller.
macUnicast Address	MacAddressString	0..1	attr	Media Access Control address (MAC address) that uniquely identifies each EthernetCommunication Controller in the network.
maximum ReceiveBuffer Length	Integer	0..1	attr	Determines the maximum receive buffer length (frame length) in bytes.
maximum TransmitBuffer Length	Integer	0..1	attr	Determines the maximum transmit buffer length (frame length) in bytes.

▽

△

Class	<<atpVariation>> **EthernetCommunicationController**			
slaveActAs Passive Communication Slave	Boolean	0..1	attr	This attribute specifies if the EcuInstance is acting as a passive communication slave on the connected Physical Channel. This is used for EthernetCommunication Controllers that use Ethernet hardware which supports wake-up and sleep on the network (e.g. Open Alliance TC10 compliant Ethernet hardware).
slaveQualified UnexpectedLink DownTime	TimeValue	0..1	attr	This attribute specifies time when an unexpected link down is evaluated as link down and indicated to the AUTOSAR communication stack.

Table E.61: EthernetCommunicationController

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*			
Aggregated by	*CommunicationCluster*.physicalChannel			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
network Endpoint	NetworkEndpoint	*	aggr	Collection of NetworkEndpoints that are used in the VLan. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=networkEndpoint.shortName
vlan	VlanConfig	0..1	aggr	VLAN Configuration.

Table E.62: EthernetPhysicalChannel

Class	***FibexElement*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore			
Note	ASAM FIBEX elements specifying Communication and Topology.			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*CommunicationCluster*, ConsumedProvidedServiceInstanceGroup, CouplingElement, EcuInstance, EthernetWakeupSleepOnDatalineConfigSet, *Frame*, Gateway, GlobalTimeDomain, ISignal, ISignal Group, ISignalIPduGroup, MachineDesign, NmConfig, *Pdu*, PdurIPduGroup, SecureCommunication PropsSet, ServiceInstanceCollectionSet, SoAdRoutingGroup, SocketConnectionIpduIdentifierSet, *Tp Config*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table E.63: FibexElement

Class	**GlobalTimeDomain**
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime
Note	This represents the ability to define a global time domain. **Tags:** atp.recommendedPackage=GlobalTimeDomains
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*
Aggregated by	ARPackage.element

▽

△

Class	**GlobalTimeDomain**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
debounceTime	TimeValue	0..1	attr	Defines the minimum amount of time between two time sync messages are transmitted.
domainId	PositiveInteger	0..1	attr	This represents the ID of the GlobalTimeDomain used in the network messages sent on behalf of global time management.
gateway	GlobalTimeGateway	*	aggr	A GlobalTimeGateway may exist in the context of a GlobalTimeDomain to actively update the global time information as it is routed from one GlobalTimeDomain to another. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=gateway.shortName, gateway.variation Point.shortLabel vh.latestBindingTime=postBuild
globalTime CorrectionProps	GlobalTimeCorrection Props	0..1	aggr	Defintion of attributes for rate and offset correction.
globalTime Domain Property	AbstractGlobalTime DomainProps	0..1	aggr	Additional properties of the GlobalTimeDomain. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=globalTimeDomainProperty, globalTime DomainProperty.variationPoint.shortLabel vh.latestBindingTime=postBuild
globalTime Master	GlobalTimeMaster	0..1	aggr	This represents the single master of a GlobalTime Domain. A GlobalTimeDomain may have no GlobalTime Domain.master, e.g. when it gets its time from a GPS receiver. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=globalTimeMaster.shortName, globalTime Master.variationPoint.shortLabel vh.latestBindingTime=postBuild
globalTimeSub Domain	GlobalTimeDomain	*	ref	By this means it is possible to create a hierarchy of sub Domains where one global time domain can declare one or more other global time domains as its subDomains. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=globalTimeSubDomain.globalTimeDomain, globalTimeSubDomain.variationPoint.shortLabel vh.latestBindingTime=postBuild
network SegmentId	NetworkSegment Identification	0..1	aggr	Defines the numerical identification of a GlobalTime sub domain.
offsetTime Domain	GlobalTimeDomain	0..1	ref	Reference to a synchronized time domain this offset time domain is based on. The reference source is the offset time domain. The reference target is the synchronized time domain.
pduTriggering	PduTriggering	0..1	ref	This PduTriggering will be taken to transmit the global time information from a GlobalTimeMaster to a the associated GlobalTimeSlaves. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=pduTriggering.pduTriggering, pdu Triggering.variationPoint.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	GlobalTimeDomain			
slave	GlobalTimeSlave	*	aggr	This represents the collections of slaves of the Global TimeDomain. A GlobalTimeDomain may have no Global TimeDomain.slaves, e.g. when it propagates its time directly to sub domains. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=slave.shortName, slave.variationPoint.shortLabel vh.latestBindingTime=postBuild
syncLoss Timeout	TimeValue	0..1	attr	This attribute describes the timeout for the situation that the time synchronization gets lost in the scope of the time domain.

Table E.64: GlobalTimeDomain

Class	***GlobalTimeMaster*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime			
Note	This represents the generic concept of a global time master.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	GlobalTimeCanMaster, GlobalTimeEthMaster, GlobalTimeFrMaster, UserDefinedGlobalTimeMaster			
Aggregated by	GlobalTimeDomain.globalTimeMaster			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Connector	Communication Connector	0..1	ref	The GlobalTimeMaster is bound to the Communication Connector.
immediate ResumeTime	TimeValue	0..1	attr	Defines the minimum time between an "immediate" message and the next periodic message.
isSystemWide GlobalTime Master	Boolean	0..1	attr	If set to TRUE, the GlobalTimeMaster is supposed to act as the root of global time information.
syncPeriod	TimeValue	0..1	attr	This represents the period. Unit: seconds

Table E.65: GlobalTimeMaster

Class	***GlobalTimeSlave*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::GlobalTime			
Note	This represents the generic concept of a global time slave.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	GlobalTimeCanSlave, GlobalTimeEthSlave, GlobalTimeFrSlave, UserDefinedGlobalTimeSlave			
Aggregated by	GlobalTimeDomain.slave			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communication Connector	Communication Connector	0..1	ref	The GlobalTimeSlave is bound to the Communication Connector.
followUp TimeoutValue	TimeValue	0..1	attr	Rx timeout for the follow-up message.
timeLeapFuture Threshold	TimeValue	0..1	attr	Defines the maximum allowed positive difference between the current Local Time Base value and a newly received Global Time Base value.

▽

△

Class	***GlobalTimeSlave*** (abstract)			
timeLeap HealingCounter	PositiveInteger	0..1	attr	Defines the required number of updates to the Time Base where the time difference to the previous received value has to remain within the bounds of timeLeapFuture Threshold and timeLeapPastThreshold until that Time Base is considered healed.
timeLeapPast Threshold	TimeValue	0..1	attr	Defines the maximum allowed negative difference between the current Local Time Base value and a newly received Global Time Base value.

Table E.66: GlobalTimeSlave

Enumeration	**HandleInvalidEnum**
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication
Note	Strategies of handling the reception of invalidValue.
Aggregated by	InvalidationPolicy.handleInvalid, ISignalPort.handleInvalid
Literal	***Description***
dontInvalidate	Invalidation is switched off. **Tags:** atp.EnumerationLiteralIndex=0
external Replacement	Replace a received invalidValue. The replacement value is sourced from the aggregation in the role replaceWith. **Tags:** atp.EnumerationLiteralIndex=1
keep	The application software is supposed to handle signal invalidation on RTE API level either by Data ReceiveErrorEvent or check of error code on read access. **Tags:** atp.EnumerationLiteralIndex=2
replace	Replace a received invalidValue. The replacement value is specified by the initValue. **Tags:** atp.EnumerationLiteralIndex=3

Table E.67: HandleInvalidEnum

Class	**IPduPort**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Connectors reception or send port on the referenced channel referenced by a PduTriggering.			
Base	*ARObject*, *CommConnectorPort*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*CommunicationConnector*.ecuCommPortInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iPduSignal Processing	IPduSignalProcessing Enum	0..1	attr	Definition of the two signal processing modes Immediate and Deferred for both Tx and Rx IPdus.
rxSecurity Verification	Boolean	0..1	attr	This attribute defines the bypassing of signature authentication or MAC verification in the receiving ECU. If not defined or set to true the signature authentication or MAC verification shall be performed for the SecuredIPdu. If set to false the signature authentication or MAC verification shall not be performed for the SecuredIPdu.
timestampRx Acceptance Window	TimeValue	0..1	attr	This attribute is used to define the maximum allowed deviation in seconds from the expected timestamp for which a SecuredIPdu is still deemed authentic. Please note that this attribute is for documentation only to allow the configuration of required freshness value manager and no upstream mapping is defined for it.

▽

△

Class	IPduPort			
useAuthData Freshness	Boolean	0..1	attr	This attribute describes whether a part of AuthenticPdu contained in a SecuredIPdu shall be passed on to the SWC that verifies and generates the Freshness. The part of the Authentic-PDU is defined by the authData FreshnessStartPosition and authDataFreshnessLength.

Table E.68: IPduPort

Class	IPduTiming			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	AUTOSAR COM provides the possibility to define two different TRANSMISSION MODES for each IPdu. The Transmission Mode of an IPdu that is valid at a specific point in time is selected using the values of the signals that are mapped to this IPdu. For each IPdu a Transmission Mode Selector is defined. The Transmission Mode Selector is calculated by evaluating the conditions for a subset of signals (class TransmissionModeCondition in the System Template). The Transmission Mode Selector is defined to be true, if at least one Condition evaluates to true and is defined to be false, if all Conditions evaluate to false.			
Base	*ARObject*, *Describable*			
Aggregated by	ISignalIPdu.iPduTimingSpecification			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
minimumDelay	TimeValue	0..1	attr	Minimum Delay in seconds between successive transmissions of this I-PDU, independent of the Transmission Mode.
transmission Mode Declaration	TransmissionMode Declaration	0..1	aggr	AUTOSAR COM allows configuring statically two different transmission modes for each I-PDU (True and False). The Transmission Mode Selector evaluates the conditions for a subset of signals and decides the transmission mode. It is possible to switch between the transmission modes during runtime.

Table E.69: IPduTiming

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 shall be created for each SystemSignal contained in the SystemSignalGroup. **Tags:** atp.recommendedPackage=ISignals			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	ISignal			
data Transformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignal. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataTransformation.dataTransformation, dataTransformation.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime
dataTypePolicy	DataTypePolicyEnum	0..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.
initValue	ValueSpecification	0..1	aggr	Optional definition of a ISignal's initValue in case the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals. This value can be used to configure the Signal's "Init Value". If a full DataMapping exist for the SystemSignal this information may be available from a configured Sender ComSpec and ReceiverComSpec. In this case the initvalues in SenderComSpec and/or ReceiverComSpec override this optional value specification. Further restrictions apply from the RTE specification.
iSignalProps	ISignalProps	0..1	aggr	Additional optional ISignal properties that may be stored in different files. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=iSignalProps
iSignalType	ISignalTypeEnum	0..1	attr	This attribute defines whether this iSignal is an array that results in a UINT8_N / UINT8_DYN ComSignalType in the COM configuration or a primitive type.
length	UnlimitedInteger	0..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.

▽

△

Class	**ISignal**			
network Representation Props	SwDataDefProps	0..1	aggr	Specification of the actual network representation. The usage of SwDataDefProps for this purpose is restricted to the attributes compuMethod and baseType. The optional baseType attributes "memAllignment" and "byteOrder" shall not be used. The attribute "dataTypePolicy" in the SystemTemplate element defines whether this network representation shall be ignored and the information shall be taken over from the network representation of the ComSpec. If "override" is chosen by the system integrator the network representation can violate against the requirements defined in the PortInterface and in the network representation of the ComSpec. In case that the System Description doesn't use a complete Software Component Description (VFB View) this element is used to configure "ComSignalDataInvalid Value" and the Data Semantics. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=networkRepresentationProps
systemSignal	SystemSignal	0..1	ref	Reference to the System Signal that is supposed to be transmitted in the ISignal.
timeout Substitution Value	ValueSpecification	0..1	aggr	Defines and enables the ComTimeoutSubstituition for this ISignal.
transformation ISignalProps	TransformationISignal Props	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignal specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignals are described in the TransformationTechnology class. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=transformationISignalProps

Table E.70: 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	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**ISignalGroup**			
comBased SignalGroup Transformation	DataTransformation	0..1	ref	Optional reference to a DataTransformation which represents the transformer chain that is used to transform the data that shall be placed inside this ISignalGroup based on the COMBasedTransformer approach. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=comBasedSignalGroupTransformation.data Transformation, comBasedSignalGroup Transformation.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime
iSignal	ISignal	*	ref	Reference to a set of ISignals that shall always be kept together.
systemSignal Group	SystemSignalGroup	0..1	ref	Reference to the SystemSignalGroup that is defined on VFB level and that is supposed to be transmitted in the ISignalGroup.
transformation ISignalProps	TransformationISignal Props	*	aggr	A transformer chain consists of an ordered list of transformers. The ISignalGroup specific configuration properties for each transformer are defined in the TransformationISignalProps class. The transformer configuration properties that are common for all ISignal Groups are described in the TransformationTechnology class. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=transformationISignalProps

Table E.71: 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	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *IPdu*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Pdu*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iPduTiming Specification	IPduTiming	0..1	aggr	Timing specification for Com IPdus (Transmission Modes). This information is mandatory for the sender in a System Extract. This information may be omitted on receivers in a System Extract. atpVariation: The timing of a Pdu can vary. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=iPduTimingSpecification, iPduTiming Specification.variationPoint.shortLabel vh.latestBindingTime=postBuild

▽

△

Class	**ISignalIPdu**			
iSignalToPdu Mapping	ISignalToIPduMapping	*	aggr	Definition of SignalToIPduMappings included in the Signal IPdu. atpVariation: The content of a PDU can be variable. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=iSignalToPduMapping.shortName, iSignalToPduMapping.variationPoint.shortLabel vh.latestBindingTime=postBuild
unusedBit Pattern	Integer	0..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 E.72: ISignalIPdu

Class	**ISignalPort**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Connectors reception or send port on the referenced channel referenced by an ISignalTriggering. If different timeouts or DataFilters for ISignals need to be specified several ISignalPorts may be created.			
Base	*ARObject*, *CommConnectorPort*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*CommunicationConnector*.ecuCommPortInstance			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dataFilter	DataFilter	0..1	aggr	Optional specification of a signal COM filter at the receiver side in case that the System Description doesn't use a complete Software Component Description (VFB View). This supports the inclusion of legacy system signals. If a full DataMapping exist for the SystemSignal this information may be available from a configured ReceiverComSpec. In this case the ReceiverComSpec overrides this optional specification.
ddsQosProfile	DdsCpQosProfile	0..1	ref	Reference to the DDS Qos profile used for this ISignal. **Tags:** atp.Status=candidate
firstTimeout	TimeValue	0..1	attr	• ISignalPort with communicationDirection = in: Optional first timeout value in seconds for the reception of the ISignal. • ISignalPort with communicationDirection = out: Optional first timeout value in seconds for transmission deadline monitoring.
handleInvalid	HandleInvalidEnum	0..1	attr	This attribute defines how invalidation is applied to the ISignals received in the context of this ISignalPort.

▽

△

Class	**ISignalPort**			
timeout	TimeValue	0..1	attr	• ISignalPort with communicationDirection = in: Optional timeout value in seconds for the reception of the ISignal. The attribute value is used to configure the Com Timeout in the COM module. The RTE ignores this attribute. The timeout can also be specified with the NonqueuedReceiverComSpec.aliveTimeout attribute. If a full DataMapping exists for the SystemSignal and the value is available in the configured ReceiverComSpec, then the timeout value in the ReceiverComSpec overrides this optional timeout specification during the creation of the Base Ecu Configuration of the COM module. • ISignalPort with communicationDirection = out: Optional timeout value in seconds for the transmission of the ISignal. The attribute value is used to configure the ComTimeout in the COM module. The RTE ignores this attribute. The timeout can also be specified with the ender ComSpec.transmissionAcknowledge.timeout attribute. If a full DataMapping exists for the SystemSignal and the value is available in the configured SenderComSpec, then the timeout value in the SenderComSpec overrides this optional timeout specification during the creation of the Base Ecu Configuration of the COM module. This attribute can be used in the following cases: • legacy signal where the System Description doesn't use a complete Software Component Description (VFB View) and where the DataMapping is missing. • bus monitoring use cases in which the DataMapping is ignored.

Table E.73: ISignalPort

Class	**ISignalToIPduMapping**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	An ISignalToIPduMapping describes the mapping of ISignals to ISignalIPdus and defines the position of the ISignal within an ISignalIPdu.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ISignalIPdu.iSignalToPduMapping, NmPdu.iSignalToIPduMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iSignal	ISignal	0..1	ref	Reference to a ISignal that is mapped into the ISignal IPdu. Each ISignal contained in the ISignalGroup shall be mapped into an IPdu by an own ISignalToIPduMapping. The references to the ISignal and to the ISignalGroup in an ISignalToIPduMapping are mutually exclusive.
iSignalGroup	ISignalGroup	0..1	ref	Reference to an ISignalGroup that is mapped into the SignalIPdu. If an ISignalToIPduMapping for an ISignal Group is defined, only the UpdateIndicationBitPosition and the transferProperty is relevant. The startPosition and the packingByteOrder shall be ignored. Each ISignal contained in the ISignalGroup shall be mapped into an IPdu by an own ISignalToIPduMapping. The references to the ISignal and to the ISignalGroup in an ISignalToIPduMapping are mutually exclusive.

▽

△

Class	**ISignalToIPduMapping**			
packingByte Order	ByteOrderEnum	0..1	attr	This parameter defines the order of the bytes of the signal and the packing into the SignalIPdu. The byte ordering "Little Endian" (MostSignificantByteLast), "Big Endian" (MostSignificantByteFirst) and "Opaque" can be selected. For opaque data endianness conversion shall be configured to Opaque. The value of this attribute impacts the absolute position of the signal into the SignalIPdu (see the startPosition attribute description). For an ISignalGroup the packingByteOrder is irrelevant and shall be ignored.
startPosition	UnlimitedInteger	0..1	attr	This parameter is necessary to describe the bitposition of a signal within an SignalIPdu. It denotes the least significant bit for "Little Endian" and the most significant bit for "Big Endian" packed signals within the IPdu (see the description of the packingByteOrder attribute). In AUTOSAR the bit counting is always set to "sawtooth" and the bit order is set to "Decreasing". The bit counting in byte 0 starts with bit 0 (least significant bit). The most significant bit in byte 0 is bit 7. Please note that the way the bytes will be actually sent on the bus does not impact this representation: they will always be seen by the software as a byte array. If a mapping for the ISignalGroup is defined, this attribute is irrelevant and shall be ignored.
transferProperty	TransferPropertyEnum	0..1	attr	Defines how the referenced ISignal contributes to the send triggering of the ISignalIPdu.
update IndicationBit Position	UnlimitedInteger	0..1	attr	The UpdateIndicationBit indicates to the receivers that the signal (or the signal group) was updated by the sender. Length is always one bit. The UpdateIndicationBitPosition attribute describes the position of the update bit within the SignalIPdu. For Signals of a ISignalGroup this attribute is irrelevant and shall be ignored. Note that the exact bit position of the updateIndicationBit Position is linked to the value of the attribute packingByte Order because the method of finding the bit position is different for the values mostSignificantByteFirst and most SignificantByteLast. This means that if the value of packingByteOrder is changed while the value of update IndicationBitPosition remains unchanged the exact bit position of updateIndicationBitPosition within the enclosing ISignalIPdu still undergoes a change. This attribute denotes the least significant bit for "Little Endian" and the most significant bit for "Big Endian" packed signals within the IPdu (see the description of the packingByteOrder attribute). In AUTOSAR the bit counting is always set to "sawtooth" and the bit order is set to "Decreasing". The bit counting in byte 0 starts with bit 0 (least significant bit). The most significant bit in byte 0 is bit 7.

Table E.74: ISignalToIPduMapping

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*

▽

△

Class	**ISignalTriggering**			
Aggregated by	*PhysicalChannel*.iSignalTriggering			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iSignal	ISignal	0..1	ref	This reference shall be used if an ISignal is transported on the PhysicalChannel. This reference forms an XOR relationship with the ISignalTriggering-ISignalGroup reference.
iSignalGroup	ISignalGroup	0..1	ref	This reference shall be used if an ISignalGroup is transported on the PhysicalChannel. This reference forms an XOR relationship with the ISignal Triggering-ISignal reference.
iSignalPort	ISignalPort	*	ref	References to the ISignalPort on every ECU of the system which sends and/or receives the ISignal. References for both the sender and the receiver side shall be included when the system is completely defined.

Table E.75: ISignalTriggering

Class	**IamModuleInstantiation**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::IdentityAccessManagement			
Note	This meta-class represents the ability to define a definition of an IAM instantiation. **Tags:** atp.Status=candidate			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
grant	Grant	*	ref	This reference identifies the applicable Grants for this Iam ModuleInstantiation. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=grant atp.Status=candidate
localCom AccessControl Enabled	Boolean	0..1	attr	This switch activates the policy enforcement in Communication Management on local applications. **Tags:** atp.Status=candidate
remoteAccess ControlEnabled	Boolean	0..1	attr	This switch activates the check of the remote subject. **Tags:** atp.Status=candidate

Table E.76: IamModuleInstantiation

Class	***Identifiable*** (abstract)
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	**Identifiable** (abstract)			
Subclasses	ARPackage, *AbstractDoIpLogicAddressProps*, *AbstractEvent*, *AbstractImplementationDataTypeElement*, *AbstractSecurityEventFilter*, *AbstractSecurityIdsmInstanceFilter*, *AbstractServiceInstance*, *AbstractSignalBasedToISignalTriggeringMapping*, AdaptiveSwcInternalBehavior, ApApplicationEndpoint, ApplicationEndpoint, ApplicationError, AppliedStandard, ArtifactChecksum, ArtifactLocator, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, AutosarOperationArgumentInstance, AutosarVariableInstance, *BuildActionEntity*, BuildActionEnvironment, Chapter, CheckpointTransition, ClassContentConditional, ClientIdDefinition, ClientServerOperation, Code, *CollectableElement*, ComManagementMapping, *CommConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEventGroup, CouplingPort, *CouplingPortStructuralElement*, CryptoCertificate, CryptoKeySlot, CryptoProvider, *CryptoServiceMapping*, DataPrototypeGroup, DataTransformation, DdsCpDomain, DdsCpPartition, DdsCpQosProfile, DdsCpTopic, DdsDomainRange, DependencyOnArtifact, *DiagEventDebounceAlgorithm*, DiagnosticAuthTransmitCertificateEvaluation, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticDebounceAlgorithmProps, DiagnosticFunctionInhibitSource, DiagnosticParameterElement, *DiagnosticRoutineSubfunction*, DiagnosticSovdMethodPrimitive, DltApplication, DltArgument, DltMessage, DoIpInterface, DoIpLogicAddress, DoIpRoutingActivation, E2EProfileConfiguration, End2EndEventProtectionProps, End2EndMethodProtectionProps, EndToEndProtection, EthernetWakeupSleepOnDatalineConfig, EventHandler, EventMapping, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeatureMapElement, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FieldMapping, FireAndForgetMethodMapping, FlexrayArTpNode, FlexrayTpPduPool, *FrameTriggering*, GeneralParameter, GlobalSupervision, GlobalTimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HealthChannel*, *HeapUsage*, HwAttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, *IEEE1722TpAcfBus*, *IEEE1722TpAcfBusPart*, IPSecRule, IPv6ExtHeaderFilterList, ISignalToIPduMapping, ISignalTriggering, *IdentCaption*, ImpositionTime, InternalTriggeringPoint, Keyword, LifeCycleState, Linker, MacMulticastGroup, MacSecKayParticipant, McDataInstance, MemorySection, MemoryUsage, MethodMapping, ModeDeclaration, ModeDeclarationMapping, ModeSwitchPoint, NetworkEndpoint, *NmCluster*, *NmNode*, *PackageableElement*, ParameterAccess, PduActivationRoutingGroup, PduToFrameMapping, PduTriggering, PerInstanceMemory, *PersistencyDeploymentElement*, *PersistencyInterfaceElement*, *PhmSupervision*, *PhysicalChannel*, PortGroup, *PortInterfaceMapping*, PossibleErrorReaction, ProcessToMachineMapping, Processor, ProcessorCore, PskIdentityToKeySlotMapping, ResourceConsumption, ResourceGroup, RootSwClusterDesignComponentPrototype, RootSwComponentPrototype, RootSwCompositionPrototype, RptComponent, RptContainer, RptExecutableEntity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptServicePoint, RunnableEntityGroup, *SdgAttribute*, SdgClass, SecOcJobMapping, SecOcJobRequirement, SecureCommunicationAuthenticationProps, *SecureCommunicationDeployment*, SecureCommunicationFreshnessProps, SecurityEventContextProps, *ServiceEventDeployment*, *ServiceFieldDeployment*, ServiceInterfaceElementSecureComConfig, *ServiceMethodDeployment*, *ServiceNeeds*, SignalServiceTranslationEventProps, SignalServiceTranslationProps, SocketAddress, SoftwarePackageStep, SomeipEventGroup, SomeipProvidedEventGroup, SomeipTpChannel, *SpecElementReference*, *StackUsage*, *StateManagementActionItem*, StateManagementActionList, StateManagementStateNotification, *StateManagementStateRequest*, StaticSocketConnection, StructuredReq, SupervisionCheckpoint, SupervisionMode, SupervisionModeCondition, SwGenericAxisParamType, SwServiceArg, SwcServiceDependency, SystemMapping, *TimeBaseResource*, *TimingClock*, TimingClockSyncAccuracy, TimingCondition, *TimingConstraint*, *TimingDescription*, TimingExtensionResource, TimingModeInstance, TlsCryptoCipherSuite, TlsCryptoCipherSuiteProps, TlsJobMapping, Topic1, TpAddress, TraceableTable, TraceableText, *TracedFailure*, *TransformationProps*, TransformationTechnology, Trigger, UcmDescription, UcmRetryStrategy, UcmStep, VariableAccess, VariationPointProxy, VehicleRolloutStep, ViewMap, VlanConfig, WaitPoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=adminData xml.sequenceOffset=-40
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. **Tags:** xml.sequenceOffset=-25

▽

△

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

Table E.77: Identifiable

Class	**ImplementationDataType**			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. **Tags:** atp.recommendedPackage=ImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicArray SizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.

▽

△

Class	**ImplementationDataType**			
isStructWith Optional Element	Boolean	0..1	attr	This attribute is only valid if the attribute category is set to STRUCTURE. If set to true, this attribute indicates that the ImplementationDataType has been created with the intention to define at least one element of the structure as optional.
subElement (ordered)	ImplementationData TypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a Implementation DataType representing a structure. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=subElement.shortName, sub Element.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the Implementation DataType. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=symbolProps.shortName
typeEmitter	NameToken	0..1	attr	This attribute is used to control which part of the AUTOSAR toolchain is supposed to trigger data type definitions.

Table E.78: ImplementationDataType

Class	**ImplementationDataTypeElement**			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	Declares a data object which is locally aggregated. Such an element can only be used within the scope where it is aggregated. This element either consists of further subElements or it is further defined via its swDataDefProps. There are several use cases within the system of ImplementationDataTypes fur such a local declaration: • It can represent the elements of an array, defining the element type and array size • It can represent an element of a struct, defining its type • It can be the local declaration of a debug element.			
Base	*ARObject*, *AbstractImplementationDataTypeElement*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, ImplementationDataType.subElement, ImplementationDataTypeElement.sub Element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arrayImplPolicy	ArrayImplPolicyEnum	0..1	attr	This attribute controls the implementation of the payload of an array. It shall only be used if the enclosing ImplementationDataType constitutes an array.
arraySize	PositiveInteger	0..1	attr	The existence of this attributes (if bigger than 0) defines the size of an array and declares that this Implementation DataTypeElement represents the type of each single array element. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
arraySize Handling	ArraySizeHandling Enum	0..1	attr	The way how the size of the array is handled in case of a variable size array.

▽

△

Class	**ImplementationDataTypeElement**			
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls the meaning of the value of the array size.
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing ImplementationDataTypeElement as optional. This means that, at runtime, the ImplementationDataType Element may or may not have a valid value and shall therefore be ignored. The underlying runtime software provides means to set the CppImplementationDataTypeElement as not valid at the sending end of a communication and determine its validity at the receiving end.
subElement (ordered)	ImplementationData TypeElement	*	aggr	Element of an array, struct, or union in case of a nested declaration (i.e. without using "typedefs"). The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a Implementation DataType representing a structure. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=subElement.shortName, sub Element.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of this ImplementationDataTypeElement. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=swDataDefProps

Table E.79: ImplementationDataTypeElement

Class	**MacMulticastConfiguration**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	References a per cluster globally defined MAC-Multicast-Group.			
Base	*ARObject*, *NetworkEndpointAddress*			
Aggregated by	NetworkEndpoint.networkEndpointAddress			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
macMulticast Group	MacMulticastGroup	0..1	ref	Reference to a macMulticastGroup.

Table E.80: MacMulticastConfiguration

Class	**ModeSwitchInterface**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A mode switch interface declares a ModeDeclarationGroupPrototype to be sent and received. **Tags:** atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
modeGroup	ModeDeclarationGroup Prototype	0..1	aggr	The ModeDeclarationGroupPrototype of this mode interface.

Table E.81: ModeSwitchInterface

Class	**MultiplexedIPdu**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	A MultiplexedPdu (i.e. NOT a COM I-PDU) contains a DynamicPart, an optional StaticPart and a selector Field. In case of multiplexing this IPdu is routed between the Pdu Multiplexer and the Interface Layer. A multiplexer is used to define variable parts within an IPdu that may carry different signals. The receivers of such a IPdu can determine which signalPdus are transmitted by evaluating the selector field, which carries a unique selector code for each sub-part. **Tags:** atp.recommendedPackage=Pdus			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *IPdu*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Pdu*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicPart	DynamicPart	0..1	aggr	According to the value of the selector field some parts of the IPdu have a different layout. In a complete System Description a MultiplexedIPdu shall contain a Dynamic Part. The following use cases support the multiplicity to be 0..1: • If a MultiplexedIPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedIPdu doesn't need to be described in the System Extract/ Ecu Extract. • If a MultiplexedIPdu is received by an ECU which is only interested in the static part of the MultiplexedIPdu then the dynamicPart does not need to be described in the System Extract/Ecu Extract. atpVariation: Content of a multiplexed PDU can vary. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dynamicPart, dynamicPart.variation Point.shortLabel vh.latestBindingTime=postBuild
selectorField ByteOrder	ByteOrderEnum	0..1	attr	This attribute defines the order of the bytes of the selector Field and the packing into the MultiplexedIPdu. Please consider that [constr_3247] and [constr_3223] are restricting the usage of this attribute. In a complete System Description this attribute is mandatory. If a MultiplexedPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedPdu doesn't need to be described in the System Extract/Ecu Extract. To support this use case the multiplicity is set to 0..1.
selectorField Length	Integer	0..1	attr	The size in bits of the selector field shall be configurable in a range of 1-16 bits. In a complete System Description this attribute is mandatory. If a MultiplexedPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedPdu doesn't need to be described in the System Extract/Ecu Extract. To support this use case the multiplicity is set to 0..1.

△

Class	**MultiplexedIPdu**			
selectorField StartPosition	Integer	0..1	attr	This parameter is necessary to describe the position of the selector field within the IPdu. Note that the absolute position of the selectorField in the MultiplexedIPdu is determined by the definition of the selectorFieldByteOrder attribute of the Multiplexed Pdu. If Big Endian is specified, the start position indicates the bit position of the most significant bit in the IPdu. If Little Endian is specified, the start position indicates the bit position of the least significant bit in the IPdu. In AUTOSAR the bit counting is always set to "sawtooth" and the bit order is set to "Decreasing". The bit counting in byte 0 starts with bit 0 (least significant bit). The most significant bit in byte 0 is bit 7. In a complete System Description this attribute is mandatory. If a MultiplexedPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedPdu doesn't need to be described in the System Extract/Ecu Extract. To support this use case the multiplicity is set to 0..1.
staticPart	StaticPart	0..1	aggr	The static part of the multiplexed IPdu is the same regardless of the selector field. The static part is optional. atpVariation: Content of a multiplexed PDU can vary. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=staticPart, staticPart.variationPoint.shortLabel vh.latestBindingTime=postBuild
triggerMode	TriggerMode	0..1	attr	IPduM can be configured to send a transmission request for the new multiplexed IPdu to the PDU-Router because of the trigger conditions/ modes that are described in the TriggerMode enumeration. In a complete System Description this attribute is mandatory. If a MultiplexedPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedPdu doesn't need to be described in the System Extract/Ecu Extract. To support this use case the multiplicity is set to 0..1.
unusedBit Pattern	Integer	0..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. In a complete System Description this attribute is mandatory. If a MultiplexedPdu is received by a Pdu Gateway and is not delivered to the IPduM but routed directly to a bus interface then the content of the MulitplexedPdu doesn't need to be described in the System Extract/Ecu Extract. To support this use case the multiplicity is set to 0..1.

Table E.82: MultiplexedIPdu

Primitive	**NameToken**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This is an identifier as used in xml, e.g. xml-names. Typical usages are, for example, the names of type emitters, protocols, or profiles. For details see NMTOKEN definition on the W3C website (https://www.w3.org/TR/xml/#NT-Nmtoken). Note: Although NameToken supports a wide range of characters, the actually allowed patterns for a certain attribute typed by NameToken may be further restricted by the specification of that attribute. **Tags:** xml.xsd.customType=NMTOKEN-STRING xml.xsd.type=NMTOKEN

Table E.83: NameToken

Class	**NotAvailableValueSpecification**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	This meta-class provides the ability to specify a ValueSpecification to state that the respective element is not available. This ability is needed to support the existence of ApplicationRecordElements where attribute isOptional ist set to the value true.			
Base	*ARObject*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValueSpecification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, ConstantSpecification.valueSpec, CryptoServiceKey.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElementCondition.compareValue, FieldSenderComSpec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, SomeipEventDeployment.eventReceptionDefaultValue, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableDataPrototype.initValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultPattern	PositiveInteger	0..1	attr	The content of this attribute shall be used to initialize gaps in the memory occupied by a structured data type in the case that an NotAvailableValueSpecification is used. Note that this pattern is only applied during initialization!

Table E.84: NotAvailableValueSpecification

Class	**NumericalValueSpecification**
Package	M2::AUTOSARTemplates::CommonStructure::Constants
Note	A numerical ValueSpecification which is intended to be assigned to a Primitive data element. Note that the numerical value is a variant, it can be computed by a formula.
Base	*ARObject*, *ValueSpecification*
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValueSpecification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, ConstantSpecification.valueSpec, CryptoServiceKey.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElementCondition.compareValue, FieldSenderComSpec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, SomeipEventDeployment.eventReceptionDefaultValue, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableDataPrototype.initValue

△

Class	**NumericalValueSpecification**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
value	Numerical	0..1	attr	This is the value itself. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime

Table E.85: NumericalValueSpecification

Class	***PPortComSpec*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes of a provided PortPrototype. This class will contain attributes that are valid for all kinds of provide ports, independent of client-server or sender-receiver communication patterns.			
Base	*ARObject*			
Subclasses	ModeSwitchSenderComSpec, NvProvideComSpec, ParameterProvideComSpec, *SenderComSpec*, ServerComSpec			
Aggregated by	*AbstractProvidedPortPrototype*.providedComSpec, PortPrototypeBlueprint.providedComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.86: 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*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
provided Interface	PortInterface	0..1	tref	The interface that this port provides. **Stereotypes:** isOfType

Table E.87: PPortPrototype

Class	**PRPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This kind of PortPrototype can take the role of both a required and a provided PortPrototype.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*, *AtpBlueprintable*, *Atp Feature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
provided Required Interface	PortInterface	0..1	tref	This represents the PortInterface used to type the PRPort Prototype **Stereotypes:** isOfType

Table E.88: PRPortPrototype

Class	***PackageableElement*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage			
Note	This meta-class specifies the ability to be a member of an AUTOSAR package.			
Base	*ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*ARElement*, EnumerationMappingTable, *FibexElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.89: PackageableElement

Class	***Pdu*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Collection of all Pdus that can be routed through a bus interface.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Subclasses	GeneralPurposePdu, *IPdu*, NmPdu, UserDefinedPdu			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
hasDynamic Length	Boolean	0..1	attr	This attribute defines whether the Pdu has dynamic length (true) or not (false). Please note that the usage of this attribute is restricted by [constr_3448].
length	UnlimitedInteger	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 E.90: Pdu

Class	**PduTriggering**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	The PduTriggering describes on which channel the IPdu is transmitted. The Pdu routing by the PduR is only allowed for subclasses of IPdu. Depending on its relation to entities such channels and clusters it can be unambiguously deduced whether a fan-out is handled by the Pdu router or the Bus Interface. If the fan-out is specified between different clusters it shall be handled by the Pdu Router. If the fan-out is specified between different channels of the same cluster it shall be handled by the Bus Interface.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*PhysicalChannel*.pduTriggering			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
iPdu	Pdu	0..1	ref	Reference to the Pdu for which the PduTriggering is defined. One I-Pdu can be triggered on different channels (PduR fan-out). The Pdu routing by the PduR is only allowed for subclasses of IPdu. Nevertheless is the reference to the Pdu element necessary since the PduTriggering element is also used to specify the sending and receiving connections to Ecu Ports.

▽

△

Class	**PduTriggering**			
iPduPort	IPduPort	*	ref	References to the IPduPort on every ECU of the system which sends and/or receives the I-PDU. References for both the sender and the receiver side shall be included when the system is completely defined.
iSignal Triggering	ISignalTriggering	*	ref	This reference provides the relationship to the ISignal Triggerings that are implemented by the PduTriggering. The reference is optional since no ISignalTriggering can be defined for DCM and Multiplexed Pdus. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=iSignalTriggering.iSignalTriggering, iSignal Triggering.variationPoint.shortLabel vh.latestBindingTime=postBuild
secOcCrypto Mapping	SecOcCryptoService Mapping	0..1	ref	This reference identifies the crypto profile applicable to the usage (send, receive) of the also referenced Secured IPdu. Obviously, this reference is only applicable if the Pdutriggering also references a SecuredIPdu in the role i Pdu.
triggerIPduSend Condition	TriggerIPduSend Condition	*	aggr	Defines the trigger for the Com_TriggerIPDUSend API call. Only if all defined TriggerIPduSendConditions evaluate to true (AND associated) the Com_Trigger IPDUSend API shall be called.

Table E.91: PduTriggering

Class	***PhmStateReference*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Definition of state dependency.			
Base	*ARObject*			
Subclasses	FunctionGroupPhmStateReference			
Aggregated by	SupervisionModeCondition.stateReference			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.92: PhmStateReference

Class	***PhmSupervision*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			
Note	Defines explicitly that NO supervision shall be applied for a set of SupervisionCheckpoints.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AliveSupervision, DeadlineSupervision, LogicalSupervision, NoCheckpointSupervision, NoSupervision			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.93: PhmSupervision

Class	**PortGroup**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Group of ports which share a common functionality , e.g. need specific network resources. This information shall be available on the VFB level in order to delegate it properly via compositions. When propagated into the ECU extract, this information is used as input for the configuration of Services like the Communication Manager. A PortGroup is defined locally in a component (which can be a composition) and refers to the "outer" ports belonging to the group as well as to the "inner" groups which propagate this group into the components which are part of a composition. A PortGroup within an atomic SWC cannot be linked to inner groups.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.portGroup			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
innerGroup	PortGroup	*	iref	Links a PortGroup in a composition to another PortGroup, that is defined in a component which is part of this CompositionSwComponentType. **InstanceRef implemented by:** InnerPortGroupIn CompositionInstanceRef
outerPort	PortPrototype	*	ref	Outer PortPrototype of this AtomicSwComponentType which belongs to the group. A port can belong to several groups or to no group at all. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=outerPort.portPrototype, outerPort.variation Point.shortLabel vh.latestBindingTime=preCompileTime

Table E.94: PortGroup

Class	***PortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/ server communication.
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port.
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port.
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port.
portPrototype Props	PortPrototypeProps	0..1	aggr	This attribute allows for the definition of further qualification of the semantics of a PortPrototype.

△

Class	***PortPrototype*** (abstract)			
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table E.95: PortPrototype

Class	**ProvidedServiceInstance**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ServiceInstances			
Note	Service instances that are provided by the ECU that is connected via the ApplicationEndpoint to a CommunicationConnector.			
Base	*ARObject*, *AbstractServiceInstance*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ApplicationEndpoint.providedServiceInstance, ServiceInstanceCollectionSet.serviceInstance			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
allowedService Consumer	NetworkEndpoint	*	ref	NetworkEndpoints on which the ConsumedService Instances that are communicating with this Provided ServiceInstance are allowed to be located so that the ACL check in the ServiceDiscovery is successful and the connection is allowed to be established. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=allowedServiceConsumer.networkEndpoint, allowedServiceConsumer.variationPoint.shortLabel atp.Status=draft vh.latestBindingTime=postBuild
autoAvailable	Boolean	0..1	attr	Defines that this ProvidedServiceInstance shall be offered by the service discovery at ECU start.
eventHandler	EventHandler	*	aggr	Collection of event groups provided by the Provided ServiceInstance **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=eventHandler.shortName, event Handler.variationPoint.shortLabel vh.latestBindingTime=postBuild
instance Identifier	PositiveInteger	0..1	attr	Instance identifier. Can be used for e.g. service discovery to identify the instance of the service.
loadBalancing Priority	PositiveInteger	0..1	attr	Defines the value to be used for load balancing priority in the service offer. Lower value means higher priority.
loadBalancing Weight	PositiveInteger	0..1	attr	Defines the value to be used for load balancing weight in the service offer. Higher value means higher probability to be chosen.
localUnicast Address	ApplicationEndpoint	0..2	ref	The local address over which the PSI is provided (udp, tcp or both). **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=localUnicastAddress.applicationEndpoint, localUnicastAddress.variationPoint.shortLabel vh.latestBindingTime=postBuild
minorVersion	PositiveInteger	0..1	attr	Minor Version of the Service that is provided by this ProvidedServiceInstance.
priority	PositiveInteger	0..1	attr	Defines the frame priority where values from 0 (best effort) to 7 (highest) are allowed.

▽

△

Class	**ProvidedServiceInstance**			
remoteMulticast Subscription Address	ApplicationEndpoint	*	ref	This reference defines the remote multicast subscribed addresses of service consumers. This reference shall ONLY be used if the remote address of the clients is determined from the configuration and not at runtime. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=remoteMulticastSubscription Address.applicationEndpoint, remoteMulticast SubscriptionAddress.variationPoint.shortLabel vh.latestBindingTime=postBuild
remoteUnicast Address	ApplicationEndpoint	*	ref	This reference defines the remote addresses of service consumers. This reference shall ONLY be used if the remote address of the clients is determined from the configuration and not at runtime. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=remoteUnicastAddress.applicationEndpoint, remoteUnicastAddress.variationPoint.shortLabel vh.latestBindingTime=postBuild
sdServerConfig	SdServerConfig	0..1	aggr	Service Discovery Server configuration. **Tags:** atp.Status=obsolete
sdServerTimer Config	SomeipSdServer ServiceInstanceConfig	0..1	ref	Server specific configuration settings relevant for the SOME/IP service discovery. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=sdServerTimerConfig.someipSdServer ServiceInstanceConfig, sdServerTimerConfig.variation Point.shortLabel vh.latestBindingTime=postBuild
serviceIdentifier	PositiveInteger	0..1	attr	This attribute represents the ability to describe the SOME/IP service ID that is offered.

Table E.96: 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*			
Subclasses	ClientComSpec, ModeSwitchReceiverComSpec, NvRequireComSpec, ParameterRequireComSpec, PersistencyDataRequiredComSpec, *ReceiverComSpec*			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.97: 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*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
required Interface	PortInterface	0..1	tref	The interface that this port requires. **Stereotypes:** isOfType

Table E.98: RPortPrototype

Class	**RecordValueSpecification**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Specifies the values for a record.			
Base	*ARObject*, *CompositeValueSpecification*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValueSpecification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, CompositeRuleBasedValueSpecification.argument, ConstantSpecification.valueSpec, CryptoServiceKey.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElementCondition.compareValue, FieldSenderComSpec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, SomeipEventDeployment.eventReceptionDefaultValue, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableDataPrototype.initValue			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
field (ordered)	ValueSpecification	*	aggr	The value for a single record field. This could also be mapped explicitly to a record element of the data type using the shortName of the ValueSpecification. But this would introduce a relationship to the data type that is too strong. As of now, it is only important that the structure of the data type matches the structure of the Value Specification independently of the shortNames. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=field, field.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table E.99: RecordValueSpecification

Primitive	**Ref**			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes			
Note	This primitive denotes a name based reference. For detailed syntax see the xsd.pattern. • first slash (relative or absolute reference) [optional] • Identifier [required] • a sequence of slashes and Identifiers [optional] This primitive is used by the meta-model tools to create the references. **Tags:** xml.xsd.customType=REF xml.xsd.pattern=/?[a-zA-Z][a-zA-Z0-9_]{0,127}(/[a-zA-Z][a-zA-Z0-9_]{0,127})* xml.xsd.type=string			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
base	Identifier	0..1	attr	This attribute reflects the base to be used for this reference. **Tags:** xml.attribute=true
blueprintValue	String	0..1	attr	This represents a description that documents how the value shall be defined when deriving objects from the blueprint. **Tags:** atp.Status=draft xml.attribute=true
index	PositiveInteger	0..1	attr	This attribute supports the use case to point on specific elements in an array. This is in particular required if arrays are used to implement particular data objects. The counting of array indices starts with the value 0, i.e. the index of the first array element is 0. **Tags:** xml.attribute=true

Table E.100: Ref

Class	**ReferenceValueSpecification**			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Specifies a reference to a data prototype to be used as an initial value for a pointer in the software.			
Base	*ARObject*, *ValueSpecification*			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValueSpecification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, ConstantSpecification.valueSpec, CryptoServiceKey.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElementCondition.compareValue, FieldSenderComSpec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, SomeipEventDeployment.eventReceptionDefaultValue, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableDataPrototype.initValue			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
referenceValue	DataPrototype	0..1	ref	The referenced data prototype.

Table E.101: ReferenceValueSpecification

Class	***Referrable*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	*ARObject*			
Subclasses	*AtpDefinition*, BswDistinguishedPartition, *BswModuleCallPoint*, BswModuleClientServerEntry, BswVariableAccess, CouplingPortTrafficClassAssignment, *CppImplementationDataTypeContextTarget*, *DiagnosticEnvModeElement*, EthernetPriorityRegeneration, ExclusiveAreaNestingOrder, *HwDescriptionEntity*, *ImplementationProps*, ModeTransition, *MultilanguageReferrable*, NmNetworkHandle, PncMappingIdent, *SingleLanguageReferrable*, SoConIPduIdentifier, SocketConnectionBundle, SomeipRequiredEventGroup, TimeSyncServerConfiguration, TpConnectionIdent			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference. **Stereotypes:** atpIdentityContributor **Tags:** xml.enforceMinMultiplicity=true xml.sequenceOffset=-100
shortNameFragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:** xml.sequenceOffset=-90

Table E.102: 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 PPort Prototype) of an AtomicSwComponentType. With this assignment, the role of the service port can be mapped to a specific ServiceNeeds element, so that a tool is able to create the correct connector.			
Base	*ARObject*			
Aggregated by	NvBlockDescriptor.clientServerPort, SwcServiceDependency.assignedPort			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
portPrototype	PortPrototype	0..1	ref	Service PortPrototype used in the assigned role. This PortPrototype shall either belong to the same AtomicSw ComponentType as the SwcInternalBehavior which owns the ServiceDependency or to the same NvBlockSw ComponentType as the NvBlockDescriptor.
role	Identifier	0..1	attr	This is the role of the assigned Port in the given context. The value shall be a shortName of the Blueprint of a Port Interface as standardized in the Software Specification of the related AUTOSAR Service.

Table E.103: RoleBasedPortAssignment

Class	**Sd**			
Package	M2::MSR::AsamHdo::SpecialData			
Note	This class represents a primitive element in a special data group.			
Base	*ARObject*			
Aggregated by	SdgContents.sd			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

△

Class	**Sd**			
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 E.104: Sd

Class	**SdClientConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ObsoleteModel			
Note	Client configuration for Service-Discovery. **Tags:** atp.Status=obsolete atp.recommendedPackage=SdConfigs			
Base	*ARObject*			
Aggregated by	ConsumedEventGroup.sdClientConfig, ConsumedServiceInstance.sdClientConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
capability Record	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service. Capability records shall only be existing if the respective SdClientConfig is composed by a Consumed ServiceInstance (see constr_3260). **Tags:** atp.Status=obsolete
clientService MajorVersion	PositiveInteger	0..1	attr	Major version number of the Service.
clientService MinorVersion	PositiveInteger	0..1	attr	Minor version number of the Service.
initialFind Behavior	InitialSdDelayConfig	0..1	aggr	Controls initial find behavior of clients. **Tags:** atp.Status=obsolete
request ResponseDelay	RequestResponseDelay	0..1	aggr	Maximum/Minimum allowable response delay to entries received by multicast in seconds. **Tags:** atp.Status=obsolete
ttl	PositiveInteger	0..1	attr	TTL for Request and Subscribe messages.

Table E.105: SdClientConfig

Class	**SdServerConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ObsoleteModel			
Note	Server configuration for Service-Discovery. **Tags:** atp.Status=obsolete atp.recommendedPackage=SdConfigs			
Base	*ARObject*			
Aggregated by	EventHandler.sdServerConfig, ProvidedServiceInstance.sdServerConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
capability Record	TagWithOptionalValue	*	aggr	A sequence of records to store arbitrary name/value pairs conveying additional information about the named service. Capability records shall only be existing if the respective SdServerConfig is composed by a Provided ServiceInstance (see constr_3259). **Tags:** atp.Status=obsolete
initialOffer Behavior	InitialSdDelayConfig	0..1	aggr	Controls offer behavior of the server. **Tags:** atp.Status=obsolete
offerCyclicDelay	TimeValue	0..1	attr	Optional attribute to define cyclic offers. Cyclic offer is active, if the delay is set (in seconds).
request ResponseDelay	RequestResponseDelay	0..1	aggr	Maximum/Minimum allowable response delay to entries received by multicast in seconds. **Tags:** atp.Status=obsolete
serverService MajorVersion	PositiveInteger	0..1	attr	Major version number of the Service.
serverService MinorVersion	PositiveInteger	0..1	attr	Minor version number of the Service.
ttl	PositiveInteger	0..1	attr	Time to live. Shall be a positive value (sInt32).

Table E.106: SdServerConfig

Class	**SecOcCryptoServiceMapping**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class has the ability to represent a crypto service mapping for the Pdu-based communication via SecOC.			
Base	*ARObject*, *CryptoServiceMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SystemMapping.cryptoServiceMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authentication	CryptoServicePrimitive	0..1	ref	This reference identifies the applicable crypto primitive for the authentication.
cryptoService Key	CryptoServiceKey	0..1	ref	This reference identifies the applicable crypto key.
cryptoService Queue	CryptoServiceQueue	0..1	ref	This reference identifies the CryptoServiceQueue the processing of this SecOcCryptoServiceMapping shall be performed in.

Table E.107: SecOcCryptoServiceMapping

Class	**SecureCommunicationAuthenticationProps**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication
Note	Authentication properties used to configure SecuredIPdus.
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	**SecureCommunicationAuthenticationProps**			
Aggregated by	SecureCommunicationPropsSet.authenticationProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authInfoTx Length	PositiveInteger	0..1	attr	This attribute defines the length in bits of the authentication code to be included in the payload of the authenticated Pdu.

Table E.108: SecureCommunicationAuthenticationProps

Class	**SecureCommunicationFreshnessProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	Freshness properties used to configure SecuredIPdus.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SecureCommunicationPropsSet.freshnessProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
freshness CounterSync Attempts	PositiveInteger	0..1	attr	This attribute defines the number of Freshness Counter re-synchronization attempts when a verification failed for a Secured I-PDU. If the value is zero, there will be no additional verification attempt to synchronize with a potentially better fitting Freshness Counter value. This attribute is only applicable if useFreshnessTimestamp is FALSE.
freshness TimestampTime PeriodFactor	PositiveInteger	0..1	attr	This attribute defines a factor that specifies the time period for the Freshness Timestamp. It holds a multiplication factor that specifies the concrete meaning of a Freshness Timestamp increment by one on basis of microseconds.
freshnessValue Length	PositiveInteger	0..1	attr	This attribute defines the complete length in bits of the Freshness Value. As long as the key doesn't change the counter shall not overflow. The length of the counter shall be determined based on the expected life time of the corresponding key and frequency of usage of the counter.
freshnessValue TxLength	PositiveInteger	0..1	attr	This attribute defines the length in bits of the Freshness Value to be included in the payload of the Secured I-PDU. This length is specific to the least significant bits of the complete Freshness Counter. If the attribute is 0 no Freshness Value is included in the Secured I-PDU.
useFreshness Timestamp	Boolean	0..1	attr	This attribute specifies whether the Freshness Value is generated through individual Freshness Counters or by a Timestamps. The value is set to TRUE when Timestamps are used.

Table E.109: SecureCommunicationFreshnessProps

Class	**SecuredIPdu**
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication
Note	If useAsCryptographicPdu is not set or set to false this IPdu contains the payload of an Authentic IPdu supplemented by additional Authentication Information (Freshness Counter and an Authenticator). If useAsCryptographicPdu is set to true this IPdu contains the Authenticator for a payload that is transported in a separate message. The separate Authentic IPdu is described by the Pdu that is referenced with the payload reference from this SecuredIPdu. **Tags:** atp.recommendedPackage=Pdus

▽

△

Class	**SecuredIPdu**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FibexElement*, *IPdu*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Pdu*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
authentication Props	SecureCommunication AuthenticationProps	0..1	ref	Reference to authentication properties that are valid for this SecuredIPdu.
dynamic RuntimeLength Handling	Boolean	0..1	attr	Defines whether the length information for handling this SecuredIPdu with SecuredIPdu.useSecuredPdu Header=noHeader is taken from the configuration or from the actually provided length information during runtime. true: SecuredIPdu length information is taken from the actually provided length information during runtime. false: SecuredIPdu length information is taken from the configuration.
freshnessProps	SecureCommunication FreshnessProps	0..1	ref	Reference to freshness properties that are valid for this SecuredIPdu.
payload	PduTriggering	0..1	ref	Reference to a Pdu that will be protected against unauthorized manipulation and replay attacks.
secure Communication Props	SecureCommunication Props	0..1	aggr	Specific configuration properties for this SecuredIPdu.
useAs Cryptographic IPdu	Boolean	0..1	attr	If this attribute is set to true the SecuredIPdu contains the Authentication Information for an AuthenticIPdu that is transmitted in a separate message. The AuthenticIPdu contains the original payload, i.e. the secured data. If this attribute is set to false this SecuredIPdu contains the payload of an Authentic IPdu supplemented by additional Authentication Information.
useSecuredPdu Header	SecuredPduHeader Enum	0..1	attr	This attribute defines the size of the header which is inserted into the SecuredIPdu. If this attribute is set to anything but noHeader, the SecuredIPdu contains the Secured I-PDU Header to indicate the length of the AuthenticIPdu. The AuthenticIPdu contains the original payload, i.e. the secured data.

Table E.110: SecuredIPdu

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*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataElement	VariableDataPrototype	*	aggr	The data elements of this SenderReceiverInterface.
invalidation Policy	InvalidationPolicy	*	aggr	InvalidationPolicy for a particular dataElement
metaDataItem Set	MetaDataItemSet	*	aggr	This aggregation defines fixed sets of meta-data items associated with dataElements of the enclosing Sender ReceiverInterface

Table E.111: SenderReceiverInterface

Class	**ServiceNeeds** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	This expresses the abstract needs that a Software Component or Basic Software Module has on the configuration of an AUTOSAR Service to which it will be connected. "Abstract needs" means that the model abstracts from the Configuration Parameters of the underlying Basic Software.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	BswMgrNeeds, ComMgrUserNeeds, CryptoKeyManagementNeeds, *CryptoNeeds*, CryptoServiceJob Needs, CryptoServiceNeeds, *DiagnosticCapabilityElement*, DltUserNeeds, *DoIpServiceNeeds*, EcuState MgrUserNeeds, ErrorTracerNeeds, FunctionInhibitionAvailabilityNeeds, FunctionInhibitionNeeds, Global SupervisionNeeds, HardwareTestNeeds, IdsMgrCustomTimestampNeeds, IdsMgrNeeds, IndicatorStatus Needs, J1939DcmDm19Support, J1939RmIncomingRequestServiceNeeds, J1939RmOutgoingRequest ServiceNeeds, NvBlockNeeds, SecureOnBoardCommunicationNeeds, SupervisedEntityCheckpoint Needs, SupervisedEntityNeeds, SyncTimeBaseMgrUserNeeds, V2xDataManagerNeeds, V2xFacUser Needs, V2xMUserNeeds, VendorSpecificServiceNeeds			
Aggregated by	BswServiceDependency.serviceNeeds, SwcServiceDependency.serviceNeeds			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.112: ServiceNeeds

Class	**ServiceSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	ServiceSwComponentType is used for configuring services for a given ECU. Instances of this class are only to be created in ECU Configuration phase for the specific purpose of the service configuration. **Tags:** atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtomicSwComponentType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *Atp Type*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *Sw ComponentType*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.113: ServiceSwComponentType

Class	**SocketConnection**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ObsoleteModel			
Note	The SoAd serves as a (De)Multiplexer between different PDU sources and the TCP/IP stack. **Tags:** atp.Status=obsolete			
Base	*ARObject*, *Describable*			
Aggregated by	SoAdConfig.connection, SocketConnectionBundle.bundledConnection			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientIpAddr From Connection Request	Boolean	0..1	attr	If set to true the Server "learns" the client IP address on connection request. This means that the statically configured IP Address of the related client shall be ignored. If set to false the Server only accepts statically configured IP address, e.g. 192.168.1.2. This means that the statically configured IP Address of the Client shall be used.
clientPort	SocketAddress	0..1	ref	Client Port for TCP/UDP connection in an abstract communication sense. The client is the major requester of the communication. Please note that the client may also produce data. **Tags:** atp.Status=obsolete

▽

△

Class	**SocketConnection**			
clientPortFrom Connection Request	Boolean	0..1	attr	If set to true the Server "learns" the client Port on connection request. This means that the statically configured Port of the related client shall be ignored. If set to false the Server only accepts statically configured Port. This means that the statically configured Port of the Client shall be used.
pdu	SocketConnectionIpdu Identifier	*	aggr	PDUs handed over by the PDU Router (Transmission over the Ethernet) or PDUs handed over by SoAd (Reception over Ethernet). Multiple IPdus can be transmitted over one socket connection. **Tags:** atp.Status=obsolete
pduCollection MaxBufferSize	PositiveInteger	0..1	attr	Defines the maximum buffer size in Byte which shall be filled before a socket with Pdu collection enabled shall be transmitted to the lower layer.
pduCollection Timeout	TimeValue	0..1	attr	Defines the time in seconds which shall pass before a socket with Pdu collection enabled shall be transmitted to the lower layer after the first Pdu has been put into the socket buffer.
runtimeIp Address Configuration	RuntimeAddress ConfigurationEnum	0..1	attr	This attribute determines which protocol is used by the client to obtain the IP Address information. If this attribute is not set to none the value determines the service used by the client to obtain the IP Address information for the SocketConnection. If this attribute is set to none the client used the statically configured IP Address information.
runtimePort Configuration	RuntimeAddress ConfigurationEnum	0..1	attr	This attribute determines which protocol is used by the client to obtain the Port information. If this attribute is not set to none the value determines the service used by the client to obtain the Port information for the Socket Connection. If this attribute is set to none the client uses the statically configured Port information.
shortLabel	Identifier	0..1	attr	This attribute specifies an identifying shortName for the SocketConnection. It shall be unique within its context.

Table E.114: SocketConnection

Class	**SocketConnectionIpduIdentifier**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::ObsoleteModel			
Note	An Identifier is required in case of one port per ECU communication where multiple Pdus are transmitted over the same connection. If only one IPdu is transmitted over the connetion this attribute can be ignored. **Tags:** atp.Status=obsolete			
Base	*ARObject*			
Aggregated by	SocketConnection.pdu, SocketConnectionBundle.pdu			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
headerId	PositiveInteger	0..1	attr	If multiple Pdus are transmitted over the same connection this headerId can be used to distinguish between the different Pdus.
pduCollection PduTimeout	TimeValue	0..1	attr	Defines the timeout in seconds the PDU collection shall be transmitted at the latest after this PDU has been put into the buffer.
pduCollection Semantics	PduCollection SemanticsEnum	0..1	attr	Specifies if the referenced PduTriggering shall be collected using a queued (i.e. all PDU instances) or last-is-best (i.e. only the last PDU instance) semantics. If this attribute is not present the behavior of "queued" is assumed.

▽

△

Class	**SocketConnectionIpduIdentifier**			
pduCollection Trigger	PduCollectionTrigger Enum	0..1	attr	Defines whether the referenced Pdu contributes to the triggering of the socket transmission if Pdu collection is enabled for this socket.
pduTriggering	PduTriggering	0..1	ref	Reference to a Pdu that is mapped to a socket connection. **Tags:** atp.Status=obsolete
routingGroup	SoAdRoutingGroup	*	ref	Reference to RoutingGroups that can be enabled or disabled. **Tags:** atp.Status=obsolete

Table E.115: SocketConnectionIpduIdentifier

Class	**SwComponentPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	Role of a software component within a composition.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.component			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
type	SwComponentType	0..1	tref	Type of the instance. **Stereotypes:** isOfType

Table E.116: SwComponentPrototype

Class	***SwConnector*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	The base class for connectors between ports. Connectors have to be identifiable to allow references from the system constraint template.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AssemblySwConnector, DelegationSwConnector, PassThroughSwConnector			
Aggregated by	*AtpClassifier*.atpFeature, CompositionSwComponentType.connector			
Attribute	***Type***	***Mult.***	***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 E.117: 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, swAddr Method, 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, invalid Value • Code generation policy provided by swRecordLayout **Tags:** vh.latestBindingTime=codeGenerationTime
Base	*ARObject*
Aggregated by	*AutosarDataType*.swDataDefProps, CompositeNetworkRepresentation.networkRepresentation, *DataPrototype*.swDataDefProps, DataPrototypeTransformationProps.networkRepresentationProps, DiagnosticDataElement.swDataDefProps, DiagnosticEnvDataElementCondition.swDataDefProps, DltArgument.networkRepresentation, FlatInstanceDescriptor.swDataDefProps, ImplementationDataTypeElement.swDataDefProps, InstantiationDataDefProps.swDataDefProps, ISignal.networkRepresentationProps, McDataInstance.resultingProperties, ParameterAccess.swDataDefProps, PerInstanceMemory.swDataDefProps, *ReceiverComSpec*.networkRepresentation, *SenderComSpec*.networkRepresentation, SomeipDataPrototypeTransformationProps.networkRepresentation, SwPointerTargetProps.swDataDefProps, SwServiceArg.swDataDefProps, SwSystemconst.swDataDefProps, SystemSignal.physicalProps

Attribute	***Type***	***Mult.***	***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

△

Class	<<atpVariation>> **SwDataDefProps**			
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
display Presentation	DisplayPresentation Enum	0..1	attr	This attribute controls the presentation of the related data for measurement and calibration tools.
implementation DataType	AbstractImplementation DataType	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 SwPointerTarget Props), 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 typical alignment of the DataPrototype. If the attribute is not defined the alignment is determined by the swBaseType size and the memoryAllocationKeywordPolicy of the referenced Sw AddrMethod. **Tags:** xml.sequenceOffset=33
swBit Representation	SwBitRepresentation	0..1	aggr	Description of the binary representation in case of a bit variable. **Tags:** xml.sequenceOffset=60
swCalibration Access	SwCalibrationAccess Enum	0..1	attr	Specifies the read or write access by MCD tools for this data object. **Tags:** xml.sequenceOffset=70
swCalprmAxis Set	SwCalprmAxisSet	0..1	aggr	This specifies the properties of the axes in case of a curve or map etc. This is mainly applicable to calibration parameters. **Tags:** xml.sequenceOffset=90
swComparison Variable	SwVariableRefProxy	*	aggr	Variables used for comparison in an MCD process. **Tags:** xml.sequenceOffset=170 xml.typeElement=false

▽

△

Class	<<atpVariation>> **SwDataDefProps**			
swData Dependency	SwDataDependency	0..1	aggr	Describes how the value of the data object has to be calculated from the value of another data object (by the MCD system). **Tags:** xml.sequenceOffset=200
swHostVariable	SwVariableRefProxy	0..1	aggr	Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects. **Tags:** xml.sequenceOffset=220 xml.typeElement=false
swImplPolicy	SwImplPolicyEnum	0..1	attr	Implementation policy for this data object. **Tags:** xml.sequenceOffset=230
swIntended Resolution	Numerical	0..1	attr	The purpose of this element is to describe the requested quantization of data objects early on in the design process. The resolution ultimately occurs via the conversion formula present (compuMethod), which specifies the transition from the physical world to the standardized world (and vice-versa) (here, "the slope per bit" is present implicitly in the conversion formula). In the case of a development phase without a fixed conversion formula, a pre-specification can occur through swIntendedResolution. The resolution is specified in the physical domain according to the property "unit". **Tags:** xml.sequenceOffset=240
swInterpolation Method	Identifier	0..1	attr	This is a keyword identifying the mathematical method to be applied for interpolation. The keyword needs to be related to the interpolation routine which needs to be invoked. **Tags:** xml.sequenceOffset=250
swIsVirtual	Boolean	0..1	attr	This element distinguishes virtual objects. Virtual objects do not appear in the memory, their derivation is much more dependent on other objects and hence they shall have a swDataDependency . **Tags:** xml.sequenceOffset=260
swPointerTarget Props	SwPointerTargetProps	0..1	aggr	Specifies that the containing data object is a pointer to another data object. Note: This atpSplitable property has no atp.Splitkey due to atpVariation (PropertySetPattern). **Stereotypes:** atpSplitable **Tags:** xml.sequenceOffset=280
swRecord Layout	SwRecordLayout	0..1	ref	Record layout for this data object. **Tags:** xml.sequenceOffset=290

▽

△

Class	<<atpVariation>> **SwDataDefProps**			
swRefresh Timing	MultidimensionalTime	0..1	aggr	This element specifies the frequency in which the object involved shall be or is called or calculated. This timing can be collected from the task in which write access processes to the variable run. But this cannot be done by the MCD system. So this attribute can be used in an early phase to express the desired refresh timing and later on to specify the real refresh timing. **Tags:** xml.sequenceOffset=300
swTextProps	SwTextProps	0..1	aggr	the specific properties if the data object is a text object. **Tags:** xml.sequenceOffset=120
swValueBlock Size	Numerical	0..1	attr	This represents the size of a Value Block **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=80
swValueBlock SizeMult (ordered)	Numerical	*	attr	This attribute is used to specify the dimensions of a value block (VAL_BLK) for the case that that value block has more than one dimension. The dimensions given in this attribute are ordered such that the first entry represents the first dimension, the second entry represents the second dimension, and so on. For one-dimensional value blocks the attribute swValue BlockSize shall be used and this attribute shall not exist. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
unit	Unit	0..1	ref	Physical unit associated with the semantics of this data object. This attribute applies if no compuMethod is specified. If both units (this as well as via compuMethod) are specified the units shall be compatible. **Tags:** xml.sequenceOffset=350
valueAxisData Type	ApplicationPrimitive DataType	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 E.118: 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*			
Aggregated by	SwDataDefProps.swPointerTargetProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**SwPointerTargetProps**			
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of the target data type. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=swDataDefProps 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. **Tags:** xml.sequenceOffset=5

Table E.119: SwPointerTargetProps

Class	**SwRecordLayout**			
Package	M2::MSR::DataDictionary::RecordLayout			
Note	Defines how the data objects (variables, calibration parameters etc.) are to be stored in the ECU memory. As an example, this definition specifies the sequence of axis points in the ECU memory. Iterations through axis values are stored within the sub-elements swRecordLayoutGroup. **Tags:** atp.recommendedPackage=SwRecordLayouts			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
swRecord LayoutGroup	SwRecordLayoutGroup	0..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 E.120: SwRecordLayout

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*			
Aggregated by	AdaptiveSwcInternalBehavior.serviceDependency, *AtpClassifier*.atpFeature, SwcInternalBehavior.serviceDependency			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

▽

△

Class	**SwcServiceDependency**			
assignedData	RoleBasedData Assignment	*	aggr	Defines the role of an associated data object of the same component. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=assignedData, assignedData.variation Point.shortLabel vh.latestBindingTime=preCompileTime
assignedPort	RoleBasedPort Assignment	*	aggr	Defines the role of an associated port of the same component. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=assignedPort, assignedPort.variation Point.shortLabel vh.latestBindingTime=preCompileTime
representedPort Group	PortGroup	0..1	ref	This reference specifies an association between the ServiceNeeeds and a PortGroup, for example to request a communication mode which applies for communication via these ports. The referred PortGroup shall be local to this atomic SWC, but via the links between the Port Groups, a tool can evaluate this information such that all the ports linked via this port group on the same ECU can be found.
serviceNeeds	ServiceNeeds	0..1	aggr	The associated ServiceNeeds.

Table E.121: SwcServiceDependency

Class	**SystemSignal**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreCommunication			
Note	The system signal represents the communication system's view of data exchanged between SW components which reside on different ECUs. The system signals allow to represent this communication in a flattened structure, with exactly one system signal defined for each data element prototype sent and received by connected SW component instances. **Tags:** atp.recommendedPackage=SystemSignals			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicLength	Boolean	0..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. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=physicalProps

Table E.122: SystemSignal

Class	**TextValueSpecification**
Package	M2::AUTOSARTemplates::CommonStructure::Constants
Note	The purpose of TextValueSpecification is to define the labels that correspond to enumeration values.
Base	*ARObject*, *ValueSpecification*

▽

△

Class	TextValueSpecification			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValueSpecification.value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameterValue.implInitValue, ConstantSpecification.valueSpec, CryptoServiceKey.developmentValue, DiagnosticEnvDataCondition.compareValue, DiagnosticEnvDataElementCondition.compareValue, FieldSenderComSpec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, MetaDataItem.metaDataItemType, NonqueuedReceiverComSpec.initValue, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.initValue, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireComSpec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, PortDefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, SomeipEventDeployment.eventReceptionDefaultValue, *StateManagementCompareCondition*.compareValue, SwDataDefProps.invalidValue, VariableDataPrototype.initValue			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
value	VerbatimString	0..1	attr	This is the value itself. Note that vt uses the | operator to separate the values for the different bitfield masks in case that the semantics of the related DataPrototype is described by means of a BITFIELD_TEXTTABLE in the associated CompuMethod.

Table E.123: TextValueSpecification

Class	TlsCryptoServiceMapping			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class has the ability to represent a crypto service mapping for the socket-based configuration of Transport Layer Security (TLS).			
Base	*ARObject*, *CryptoServiceMapping*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SystemMapping.cryptoServiceMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
keyExchange	CryptoServicePrimitive	*	ref	This reference identifies the shared(i.e. applicable for each of the aggregated cipher suites) crypto service primitive for the execution of key exchange during the handshake phase.
tlsCipherSuite	TlsCryptoCipherSuite	*	aggr	This aggregation represents the collection of supported cipher suites.
useClient Authentication Request	Boolean	0..1	attr	Defines if client authentication shall be applied for this TLS connection.
useSecurity Extension RecordSize Limit	Boolean	0..1	attr	Defines if the security extension for max_fragment_length shall be supported as defined in IETF RFC 8449, chapter 4.1.

Table E.124: TlsCryptoServiceMapping

Class	TlsPskIdentity
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication
Note	This element is used to describe the pre-shared key shared during the handshake among the communication parties, to establish a TLS connection if the handshake is based on the existence of a pre-shared key.
Base	*ARObject*
Aggregated by	TlsCryptoCipherSuite.pskIdentity

▽

△

Class	**TlsPskIdentity**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
preSharedKey	CryptoServiceKey	0..1	ref	This reference identifies the applicable cryptographic key.
pskIdentity	String	0..1	attr	This attribute provides the key identification.
pskIdentityHint	String	0..1	attr	This attribute provides the identity hint for a pre-shared key.

Table E.125: TlsPskIdentity

Class	***TransformationProps*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::Transformer			
Note	This meta-class represents a abstract base class for transformation settings.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ApSomeipTransformationProps, SOMEIPTransformationProps, UserDefinedTransformationProps			
Aggregated by	TransformationPropsSet.transformationProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table E.126: TransformationProps

Enumeration	**TransportLayerProtocolEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ServiceInstanceManifest::ServiceInstanceDeployment
Note	This enumeration allows to choose a TCP/IP transport layer protocol.
Aggregated by	SomeipEventDeployment.transportProtocol, SomeipMethodDeployment.transportProtocol
Literal	***Description***
tcp	Transmission control protocol **Tags:** atp.EnumerationLiteralIndex=1
udp	User datagram protocol **Tags:** atp.EnumerationLiteralIndex=0

Table E.127: TransportLayerProtocolEnum

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

Table E.128: TriggerInterface

Class	***ValueSpecification*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::Constants			
Note	Base class for expressions leading to a value which can be used to initialize a data object.			
Base	*ARObject*			
Subclasses	*AbstractRuleBasedValueSpecification*, ApplicationValueSpecification, *CompositeValueSpecification*, ConstantReference, NotAvailableValueSpecification, NumericalValueSpecification, ReferenceValue Specification, TextValueSpecification			
Aggregated by	ApplicationAssocMapElementValueSpecification.key, ApplicationAssocMapElementValueSpecification. value, ArrayValueSpecification.element, CalibrationParameterValue.applInitValue, CalibrationParameter Value.implInitValue, ConstantSpecification.valueSpec, CryptoServiceKey.developmentValue, Diagnostic EnvDataCondition.compareValue, DiagnosticEnvDataElementCondition.compareValue, FieldSenderCom Spec.initValue, ISignal.initValue, ISignal.timeoutSubstitutionValue, NonqueuedReceiverComSpec.init Value, NonqueuedReceiverComSpec.timeoutSubstitutionValue, NonqueuedSenderComSpec.initValue, NvProvideComSpec.ramBlockInitValue, NvProvideComSpec.romBlockInitValue, NvRequireComSpec.init Value, ParameterDataPrototype.initValue, ParameterProvideComSpec.initValue, ParameterRequireCom Spec.initValue, PersistencyDataRequiredComSpec.initValue, PersistencyKeyValuePair.initValue, Port DefinedArgumentValue.value, PortPrototypeBlueprintInitValue.value, RecordValueSpecification.field, SomeipEventDeployment.eventReceptionDefaultValue, *StateManagementCompareCondition*.compare Value, SwDataDefProps.invalidValue, VariableDataPrototype.initValue			
Attribute	**Type**	**Mult.**	**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 E.129: ValueSpecification

Class	**VlanConfig**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	VLAN Configuration attributes			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	EthernetPhysicalChannel.vlan			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
vlanIdentifier	PositiveInteger	0..1	attr	A VLAN is identified by this attribute according to IEEE 802.1Q. The allowed values range is from 0..4095.

Table E.130: VlanConfig

Class	**VlanMembership**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::Fibex4Ethernet::EthernetTopology			
Note	Static logical channel or VLAN binding to a switch-port. The reference to an EthernetPhysicalChannel without a VLAN defined represents the handling of untagged frames.			
Base	*ARObject*			
Aggregated by	CouplingPort.vlanMembership			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
defaultPriority	PositiveInteger	0..1	attr	Standard output-priority outgoing Frames will be tagged with. Defines the priority that received frames are assigned together with the VLAN Id (defaultVlan). The values from 0 (best effort) to 7 (highest) are allowed. In case modifyVlan and an already tagged received frame, the actual priority of the received frame is not modified.

△

Class	**VlanMembership**			
dhcpAddress Assignment	DhcpServer Configuration	0..1	aggr	Specifies the IP Address which will be assigned to a DHCP Client at this SwitchPort. If no dhcpAddress Assignment is provided all DHCP-Discover messages received at this Port will be discarded by the DHCP Server.
sendActivity	EthernetSwitchVlan EgressTaggingEnum	0..1	attr	Attribute denotes whether a VLAN tagged ethernet frame will be 1. sent with its VLAN tag (sentTagged) 2. sent without a VLAN tag (sentUntagged) 3. will be dropped at this port (notSent or VLAN not member of this list)
vlan	EthernetPhysical Channel	0..1	ref	References a channel that represents a VLAN or an untagged channel.

Table E.131: VlanMembership

F 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.

F.1 Constraint and Specification Item History of this document according to AUTOSAR Release R17-03 (original version)

F.1.1 Created Constraints in R17-03

Number	Heading
[constr_1473]	No support for PRPortPrototype
[constr_1474]	SwDataDefProps applicable to ImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1475]	ImplementationDataType of category STRING is limited
[constr_1476]	ImplementationDataType of category VECTOR is limited
[constr_1477]	ImplementationDataType of category ASSOCIATIVE_MAP is limited
[constr_1478]	SwDataDefProps applicable to ApplicationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1479]	No support for certain values of ImplementationDataType.category
[constr_1480]	Mutual existence of CompositionDataPrototypeRef.elementInImplDatatype vs. attributes of CompositionDataPrototypeRef.dataPrototype
[constr_1481]	Usage of CompositionDataPrototypeRef in the *AUTOSAR adaptive platform*
[constr_1482]	Mapping of service interfaces vs. mapping of service interface elements
[constr_1483]	Applicability of a ServiceInterface
[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

▽

△

Number	Heading
[constr_1494]	Initial value for event
[constr_1495]	Initial value for field
[constr_1496]	DiagnosticServiceDataMapping.mappedApDataElement shall only refer to specific sub-classes of DataPrototype
[constr_1497]	Attribute optionKind set to commandLineSimpleForm
[constr_1498]	Attribute optionKind set to commandLineShortForm or commandLineLongForm
[constr_1499]	Target SwcServiceDependency of DiagnosticServiceSwMapping.mappedSwcServiceDependencyInExecutable
[constr_1500]	Target SwcServiceDependency of DiagnosticEventPortMapping.swcServiceDependencyInExecutable
[constr_1501]	Target SwcServiceDependency of DiagnosticOperationCyclePortMapping.swcServiceDependencyInExecutable
[constr_1502]	Target SwcServiceDependency of DiagnosticEnableConditionPortMapping.swcServiceDependencyInExecutable
[constr_1503]	Target SwcServiceDependency of DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable
[constr_1504]	Number of Process.modeDependentStartupConfig that refer to the same ModeDeclaration
[constr_1505]	Number of Process.modeDependentStartupConfig that do not refer to a ModeDeclaration
[constr_1507]	PortInterfaceToDataTypeMapping is only applicable to ServiceInterface
[constr_1508]	BaseTypeDirectDefinition.nativeDeclaration shall not be set to the value enum
[constr_3320]	Aggregation of CommunicationConnector by Machine
[constr_3287]	Mandatory information of a ProvidedSomeipServiceInstance
[constr_3288]	IP configuration restriction for unicastNetworkEndpoints
[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_5155]	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_5156]	SomeipEvent.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances

▽

△

Number	Heading
[constr_3308]	SomeipEvent.transportProtocol setting to `tcp` and the impact on `ProvidedSomeipServiceInstance`s
[constr_3309]	SomeipMethod.transportProtocol setting to `udp` and the impact on `ProvidedSomeipServiceInstance`s
[constr_3310]	SomeipMethod.transportProtocol setting to `tcp` and the impact on `ProvidedSomeipServiceInstance`s
[constr_3320]	Aggregation of `CommunicationConnector` by `Machine`
[constr_3349]	Usage of `ApplicationAssocMapDataType` is limited
[constr_3350]	Consistent value of `category` for `AdaptiveAutosarApplication`s 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 `RPortPrototype`s.
[constr_3360]	`RPortPrototypeProps` are related only to TransportLayerIndependentInstanceIds representing a consumer Service Instance.
[constr_3361]	Selective definition of serialization settings.
[constr_3362]	SomeipEvents aggregated by a SomeipField
[constr_3363]	SomeipMethods aggregated by a SomeipField

Table F.1: Added Constraints in original version

F.1.2 Created Specification Items in R17-03

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`

▽

△

Number	Heading
[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
[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

▽

△

Number	Heading
[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.searchIntention
[TPS_MANI_01058]	Ability to create a mapping of ApplicationErrors aggregated in the role possibleError
[TPS_MANI_01059]	Different values of optionKind within a StartupConfig.startupOption
[TPS_MANI_01060]	Use cases for the application of DiagnosticServiceDataMapping
[TPS_MANI_01061]	Requirements on scheduling
[TPS_MANI_01062]	ImplementationDataType to generate a C++ enum
[TPS_MANI_01063]	Sharing of ImplementationDataType with enumeration semantics
[TPS_MANI_03000]	Mapping of AdaptivePlatformServiceInstance to PortPrototypes
[TPS_MANI_03001]	Mapping of AdaptivePlatformServiceInstance to a Machine
[TPS_MANI_03002]	IP configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03003]	ProvidedSomeipServiceInstance Fanout
[TPS_MANI_03004]	IPv4 Multicast event destination address
[TPS_MANI_03005]	IPv4 Multicast address range
[TPS_MANI_03006]	IPv6 Multicast address range
[TPS_MANI_03007]	Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03008]	Tcp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03009]	Tcp and Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03010]	Udp Transport Protocol Configuration in case of IP-Multicast

▽

△

Number	Heading
[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
[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

▽

△

Number	Heading
[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
[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 `DataPrototype`s

▽

△

Number	Heading
[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 F.2: Added Specification Items in original Version

F.2 Constraint and Specification Item History of this document according to AUTOSAR Release R17-10

F.2.1 Added Specification Items in R17-10

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

▽

△

Number	Heading
[TPS_MANI_01091]	Semantics of `CryptoJob`
[TPS_MANI_01092]	Mapping between `CryptoNeed` and `CryptoJob`
[TPS_MANI_01093]	Semantics of `CryptoDriver`
[TPS_MANI_01094]	Scope of `CryptoDriver`
[TPS_MANI_01095]	Semantics of `CryptoKeySlot`
[TPS_MANI_01096]	Semantics of the `CryptoPrimitive`
[TPS_MANI_01097]	Assignment of TLV data ids for data structures with optional members
[TPS_MANI_01098]	Constraints on the definition of an `ImplementationDataType` of `category` `VECTOR`
[TPS_MANI_01099]	Semantics of `ImplementationDataTypeElementExtension`
[TPS_MANI_01100]	Semantics of `Allocator`
[TPS_MANI_01101]	Size-constrained allocation of memory
[TPS_MANI_01102]	Specification of a namespace for an `ImplementationDataType` of `category` `VECTOR`
[TPS_MANI_01103]	Three-level approach to REST modeling
[TPS_MANI_01105]	Semantics of `RestServiceInterface`
[TPS_MANI_01106]	Specification of capabilities for the receiver of `event`s or `field` notifiers
[TPS_MANI_01107]	Specification of capabilities for the sender of `event`s or `field` notifiers
[TPS_MANI_01108]	Specification of capabilities for the caller of a `method`s or `field` setter/getter
[TPS_MANI_01109]	Semantics of `UploadablePackageElement`
[TPS_MANI_01110]	Semantics of `SoftwareCluster`
[TPS_MANI_01111]	Diagnostic Address of a `SoftwareCluster`
[TPS_MANI_01112]	Semantics of `SoftwareClusterDesign`
[TPS_MANI_01113]	Semantics of `SoftwareClusterDesign.diagnosticAddress`
[TPS_MANI_01114]	Relation of `DiagnosticContributionSet` to `SoftwareCluster`
[TPS_MANI_01115]	Specification of executable software within `SoftwareCluster`
[TPS_MANI_01116]	Reference to model elements included in an uploadable software package
[TPS_MANI_01117]	Semantics of `SoftwareClusterDesign.intendedTargetMachine`
[TPS_MANI_01118]	Relation between `SoftwareClusterDesign` and `DiagnosticContributionSet`
[TPS_MANI_01119]	Reference to model elements from `SoftwareClusterDesign`
[TPS_MANI_01120]	Recursive definition of `RestResourceDef`
[TPS_MANI_01121]	Semantics of `RestResourceDef.endpoint`
[TPS_MANI_01122]	Arguments to endpoints
[TPS_MANI_01123]	System Triggered Event
[TPS_MANI_01124]	Semantics of `RestElementDef`
[TPS_MANI_01125]	Properties of REST elements can either be primitive or have array semantics
[TPS_MANI_01126]	Definition of string properties

▽

△

Number	Heading
[TPS_MANI_01127]	Limited support for data semantics in RestAbstractNumericalPropertyDef
[TPS_MANI_01128]	Difference between RestIntegerPropertyDef and RestNumberPropertyDef
[TPS_MANI_01129]	RestObjectRef is only needed for specific implementations of REST-based communication
[TPS_MANI_01130]	Structure of a typical URI for a REST service
[TPS_MANI_01131]	Impact of nested REST resources on the structure of REST URI
[TPS_MANI_01132]	Semantics of CompositionDataPrototypeRef
[TPS_MANI_01133]	Optional element of an event
[TPS_MANI_01134]	Optional element in the context of a method
[TPS_MANI_03110]	Allowed components in system description with category category SOFTWARE_COMPONENT_SYSTEM_DESIGN_DESCRIPTION.
[TPS_MANI_03111]	Mapping between method and operation
[TPS_MANI_03112]	Mapping between an event and a dataElement
[TPS_MANI_03113]	Mapping between a field and elements of Classic Platform PortInterfaces
[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

▽

△

Number	Heading
[TPS_MANI_03135]	Configuration of TLS PSK Identity
[TPS_MANI_03136]	Configuration of requirements for the TLS cryptographic job
[TPS_MANI_03137]	ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are not relevant in case of TLS communication
[TPS_MANI_03138]	SecOC Security Profile
[TPS_MANI_03139]	Standardized SecOC Security Profiles
[TPS_MANI_03140]	Non-Standardized SecOC Security Profiles
[TPS_MANI_03141]	Mapping between SecOcJobRequirement and CryptoJob
[TPS_MANI_03142]	Mapping between TlsJobRequirement and CryptoJob
[TPS_MANI_03143]	Mapping between PresharedKeyIdentity and CryptoKeySlot
[TPS_MANI_03144]	C++ language binding of ImplementationDataTypes of category STRING
[TPS_MANI_03145]	Description of a function group
[TPS_MANI_03146]	Configuration of timeouts for a selected machine state or function group state
[TPS_MANI_03147]	Mapping of a Process to a Machine
[TPS_MANI_03148]	Description of Core affinity
[TPS_MANI_03149]	Definition of a start-up timeout for a Process
[TPS_MANI_03150]	Definition of a termination timeout for a Process
[TPS_MANI_03151]	Default value for termination timeout
[TPS_MANI_03152]	Assignment of a StateDependentStartupConfig to a function group state
[TPS_MANI_03153]	Semantics of ModeDependentStartupConfig.functionGroupMode
[TPS_MANI_03500]	Definition of platform health management checkpoints
[TPS_MANI_03501]	Definition of platform health management supervised entities
[TPS_MANI_03502]	Enabling of PlatformHealthManagementContribution on a Machine
[TPS_MANI_03503]	Applicability of supervision to a specific Process
[TPS_MANI_03504]	Existence of SupervisionEntity
[TPS_MANI_03505]	Existence of PhmCheckpoint
[TPS_MANI_03506]	Optionality of SupervisionEntity and PhmCheckpoint
[TPS_MANI_03508]	Definition of an AliveSupervision for a PhmCheckpoint
[TPS_MANI_03509]	Definition of a CheckpointTransition
[TPS_MANI_03510]	Definition of LogicalSupervision
[TPS_MANI_03511]	Definition of DeadlineSupervision
[TPS_MANI_03512]	Applicability of global supervision to a specific Process
[TPS_MANI_03513]	Collection of SupervisionEntitys into a global supervision
[TPS_MANI_03514]	Expiration tolerance for GlobalSupervisionEntity
[TPS_MANI_03515]	Expiration tolerance for SupervisionEntity
[TPS_MANI_03516]	Condition evaluation for HealthChannelSupervision
[TPS_MANI_03517]	Condition evaluation for HealthChannelExternalMode

▽

△

Number	Heading
[TPS_MANI_03518]	`LogicalExpression` definition
[TPS_MANI_03519]	`Rule` definition
[TPS_MANI_03520]	Execution of `PhmActionList` with `actionListExecution=triggeredOnEvaluation`
[TPS_MANI_03521]	Execution of `PhmActionList` with `actionListExecution=triggeredOnChange`
[TPS_MANI_03522]	Definition of actions for application software
[TPS_MANI_03523]	Definition of actions for Platform Instance
[TPS_MANI_03524]	Definition of actions for Watchdog

Table F.3: Added Specification Items in R17-10

F.2.2 Changed Specification Items in R17-10

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 `PortPrototype`s
[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

▽

△

Number	Heading
[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 F.4: Changed Specification Items in R17-10

F.2.3 Deleted Specification Items in R17-10

Number	Heading
[TPS_MANI_03100]	Transport layer independent TransportLayerIndependentInstanceIds

Table F.5: Deleted Specification Items in R17-10

F.2.4 Added Constraints in R17-10

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

▽

△

Number	Heading
[constr_1537]	Consistent assignment of TLV data ids to arguments of a given ClientServerOperation
[constr_1542]	No nested definition of SoftwareCluster
[constr_1543]	Only one physical address per SoftwareCluster
[constr_3366]	System category for a system description with Adaptive Platform components
[constr_3367]	FieldMapping.notifierDataElement reference
[constr_3368]	FieldMapping.getterOperation reference
[constr_3369]	FieldMapping.setterOperation reference
[constr_3370]	InterfaceMapping shall map all elements of a single ServiceInterface
[constr_3371]	Mutually exclusive existence of FireAndForgetMapping.dataElement reference and FireAndForgetMapping.trigger reference
[constr_3372]	Restriction in usage of ApSomeipTransformationProps.sizeOfStringLengthField
[constr_3374]	method with attribute fireAndForget set to true shall not have any inout or out arguments
[constr_3375]	method with attribute fireAndForget set to true shall not reference an ApplicationError
[constr_3376]	FireAndForgetMapping shall reference only fire and forget methods
[constr_3377]	Restriction of ISignalTriggering references in SignalBasedFieldToISignalTriggeringMapping
[constr_3380]	End2EndEventProtectionProps shall not reference an event and a notifier at the same time
[constr_3387]	Compatibility of PortPrototypes of different ServiceInterfaces
[constr_3388]	Compatibility of events
[constr_3389]	Compatibility of methods
[constr_3390]	Compatibility of fields
[constr_3391]	ServiceInterfaceElementSecureComConfig references to ServiceInterfaceDeployment elements
[constr_3392]	ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are mandatory in case of SecOC communication
[constr_3393]	Usage of shallRunOn and shallNotRunOn references
[constr_3394]	Default value for start-up timeout on the Machine is not configurable
[constr_3395]	TransformationPropsToServiceInterfaceElementMapping is restricted to one single ServiceInterface
[constr_3396]	Number of Process.modeDependentStartupConfig that refer to the same functionGroupMode
[constr_3397]	ModeDependentStartupConfig that refers to a functionGroupMode and to a machineMode
[constr_3398]	ModeDependentStartupConfig that refers to function group modes of different function groups
[constr_3527]	LogicalExpression referenced by one PhmRule

Table F.6: Added Constraints in R17-10

F.2.5 Changed Constraints in R17-10

Number	Heading
[constr_1486]	ImplementationDataType of category STRING and SwBaseType
[constr_1490]	Allowed value of category for reference ProcessToMachineMapping.process.executable
[constr_3290]	Transport Protocol attributes defined for a ProvidedSomeipServiceInstance
[constr_3296]	Transport Protocol attributes defined for a RequiredSomeipServiceInstance
[constr_3307]	SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3308]	SomeipEventDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3309]	SomeipMethodDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3310]	SomeipMethodDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3361]	Selective definition of serialization settings

Table F.7: Changed Constraints in R17-10

F.2.6 Deleted Constraints in R17-10

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 F.8: Deleted Constraints in R17-10

F.3 Constraint and Specification Item History of this document according to AUTOSAR Release R18-03

F.3.1 Added Specification Items in R18-03

Number	Heading
[TPS_MANI_01135]	Semantics of PersistencyKeyValueDatabaseInterface.dataTypeForSerialization
[TPS_MANI_01136]	AutosarDataPrototype is the target of the CompositionDataPrototypeRef
[TPS_MANI_01137]	Applicable use cases for CompositionDataPrototypeRef
[TPS_MANI_01138]	Semantics of PersistencyKeyValueDatabaseInterface.dataElement
[TPS_MANI_01139]	Semantics of PersistencyKeyValueDatabaseInterface.updateStrategy
[TPS_MANI_01140]	Semantics of PersistencyDataElement.updateStrategy
[TPS_MANI_01141]	Semantics of PersistencyFileProxyInterface.updateStrategy
[TPS_MANI_01142]	Semantics of PersistencyFileProxy
[TPS_MANI_01143]	Semantics of PersistencyFileProxy.updateStrategy
[TPS_MANI_01144]	Semantics of PersistencyKeyValuePair
[TPS_MANI_01146]	Initial value for PersistencyKeyValuePair
[TPS_MANI_01147]	Semantics of PersistencyKeyValueDatabase.updateStrategy
[TPS_MANI_01148]	Semantics of PersistencyKeyValuePair.updateStrategy
[TPS_MANI_01149]	Semantics of PersistencyFileArray.file
[TPS_MANI_01150]	Semantics of PersistencyFileArray
[TPS_MANI_01151]	Semantics of PersistencyFileArray.updateStrategy
[TPS_MANI_01152]	Semantics of PersistencyFile.updateStrategy
[TPS_MANI_01154]	PersistencyFileArray.updateStrategy overrides PersistencyFileProxyInterface.updateStrategy
[TPS_MANI_01155]	PersistencyKeyValueDatabase.updateStrategy overrides PersistencyKeyValueDatabaseInterface.updateStrategy
[TPS_MANI_01156]	PersistencyKeyValuePair.updateStrategy overrides PersistencyKeyValueDatabase.updateStrategy
[TPS_MANI_01157]	Semantics of updateStrategy on collection level
[TPS_MANI_01158]	PersistencyFile.updateStrategy overrides PersistencyFileArray.updateStrategy
[TPS_MANI_01159]	Semantics of updateStrategy on element level
[TPS_MANI_01160]	Definition of initial value for PersistencyDataElement
[TPS_MANI_01161]	Impact of values of category on the semantics of SoftwareClusterDesign
[TPS_MANI_01162]	Semantics of SoftwareClusterDesign.dependsOn

▽

△

Number	Heading
[TPS_MANI_01163]	Impact of values of `category` on the semantics of `SoftwareCluster`
[TPS_MANI_01164]	Semantics of `SoftwareCluster.dependsOn`
[TPS_MANI_01165]	Standardized value of `UserDefinedServiceInterfaceDeployment.category`
[TPS_MANI_01166]	Semantics of `CppImplementationDataType`
[TPS_MANI_01167]	`AbstractImplementationDataType`
[TPS_MANI_01168]	Specification of a namespace for a `CppImplementationDataType`
[TPS_MANI_01169]	Support for template data types
[TPS_MANI_01170]	Semantics of `CppTemplateArgument.isVariadicTemplate`
[TPS_MANI_01171]	Modeling of structured data types
[TPS_MANI_01172]	Description of type references in the scope of `CppImplementationDataType`
[TPS_MANI_01173]	Description of type references in the scope of `CppImplementationDataTypeElement`
[TPS_MANI_01174]	Semantics of reference in the role `CppTemplateArgument.templateType`
[TPS_MANI_01175]	Semantics of reference in the role `CppTemplateArgument.allocator`
[TPS_MANI_01176]	Standardized value for attribute `CppImplementationDataType.typeEmitter`
[TPS_MANI_01177]	Semantics of `CppImplementationDataType.typeEmitter`
[TPS_MANI_01178]	Semantics of `RestHttpPortPrototypeMapping.acceptsEncoding`
[TPS_MANI_01179]	Semantics of `PersistencyFileProxy.contentUri`/`PersistencyFile.contentUri` vs. `PersistencyFileArray.uri` and `PersistencyFileProxy.fileName`/`PersistencyFile.fileName`
[TPS_MANI_01180]	Collection of data types that requires serialization support
[TPS_MANI_01181]	Use cases for the application of `DiagnosticServiceSwMapping`
[TPS_MANI_01182]	`PersistencyKeyValuePair.updateStrategy` overrides `PersistencyDataElement.updateStrategy`
[TPS_MANI_01183]	`PersistencyFile.updateStrategy` overrides `PersistencyFileProxy.updateStrategy`
[TPS_MANI_03154]	`ProvidedSomeipServiceInstance` related configuration settings for `event`s
[TPS_MANI_03155]	`ProvidedSomeipServiceInstance` related configuration settings for `method`s
[TPS_MANI_03156]	`RequiredSomeipServiceInstance` related configuration settings for `method`s
[TPS_MANI_03157]	Enabling of data accumulation for upd data transmission
[TPS_MANI_03158]	Configuration of a data accumulation on a `ProvidedServiceInstance` for transmission over udp
[TPS_MANI_03159]	Configuration of a data accumulation on a `RequiredSomeipServiceInstance` for transmission over udp
[TPS_MANI_03160]	Log and Trace configuration options in the Application Manifest
[TPS_MANI_03161]	Log and Trace configuration options in the Service Instance Manifest

▽

△

Number	Heading
[TPS_MANI_03162]	Machine-specific configuration settings for the Log and Trace functional cluster
[TPS_MANI_03163]	Network configuration for Log and Trace messages
[TPS_MANI_03164]	Machine-specific configuration settings for DoIP
[TPS_MANI_03165]	Network configuration for DoIP
[TPS_MANI_03166]	Machine-specific configuration settings for NM module
[TPS_MANI_03167]	Network configuration for Nm
[TPS_MANI_03168]	Configuration of the SOME/IP load balancing option
[TPS_MANI_03169]	CppImplementationDataType with fixed size array semantics
[TPS_MANI_03170]	CppImplementationDataType of category ARRAY
[TPS_MANI_03171]	Value type of a CppImplementationDataType of category ARRAY
[TPS_MANI_03172]	Size of a CppImplementationDataType of category ARRAY
[TPS_MANI_03173]	multidimensional Array
[TPS_MANI_03174]	CppImplementationDataType with variable size array semantics
[TPS_MANI_03175]	CppImplementationDataType of category VECTOR
[TPS_MANI_03176]	Value type of a CppImplementationDataType of category VECTOR
[TPS_MANI_03177]	multidimensional Vector
[TPS_MANI_03178]	CppImplementationDataType of category STRING
[TPS_MANI_03179]	C++ language binding of CppImplementationDataTypes of category STRING
[TPS_MANI_03180]	Definition of Structures
[TPS_MANI_03181]	Definition of members in CppImplementationDataType of category STRUCTURE
[TPS_MANI_03182]	Definition of members in CppImplementationDataTypeElement of category STRUCTURE
[TPS_MANI_03183]	CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03184]	CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03185]	Structure of an CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03186]	Usage of arraySize in case of a Vector
[TPS_MANI_03187]	Definition of enumeration types
[TPS_MANI_03188]	Usage of an Allocator for a CppImplementationDataType of category STRING
[TPS_MANI_03189]	Definition of CppImplementationDataType of category VARIANT
[TPS_MANI_03190]	CppImplementationDataType of category VARIANT
[TPS_MANI_03191]	Definition of type alternatives stored in a VARIANT
[TPS_MANI_03192]	CppImplementationDataType of category VALUE
[TPS_MANI_03193]	CppImplementationDataType or CppImplementationDataTypeElement of category TYPE_REFERENCE
[TPS_MANI_03194]	Function Group State
[TPS_MANI_03195]	Off state in Function Group

▽

△

Number	Heading
[TPS_MANI_03196]	Semantics of `CppImplementationDataTypeElementQualifier.anonymous` attribute
[TPS_MANI_03525]	DDS ServiceInterface binding
[TPS_MANI_03526]	DDS `VariableDataPrototype` binding
[TPS_MANI_03527]	Definition of `ProvidedDdsServiceInstance`
[TPS_MANI_03528]	Definition of `ProvidedDdsEventQosProps`
[TPS_MANI_03529]	Definition of `RequiredDdsServiceInstance`
[TPS_MANI_03530]	Definition of `RequiredDdsEventQosProps`
[TPS_MANI_03531]	`qosProfile` of `ProvidedDdsEventQosProps` is optional
[TPS_MANI_03532]	`qosProfile` of `RequiredDdsEventQosProps` is optional
[TPS_MANI_03533]	`DdsServiceInstanceToMachineMapping`
[TPS_MANI_03534]	Definition of Platform Health Management Health Channel
[TPS_MANI_03535]	Definition of Time Synchronization interaction
[TPS_MANI_03536]	Time Synchronization interaction in a master role
[TPS_MANI_03537]	Time Synchronization interaction in a slave role
[TPS_MANI_03538]	Time Synchronization interaction with a local Time Base
[TPS_MANI_03539]	Definition of Time Bases
[TPS_MANI_03540]	Definition of `PureLocalTimeBase`
[TPS_MANI_03541]	Definition of `SynchronizedSlaveTimeBase`
[TPS_MANI_03542]	Definition of `SynchronizedMasterTimeBase`
[TPS_MANI_03543]	Definition of time sync correction attributes
[TPS_MANI_03544]	Definition of `PlatformHealthManagementContribution`
[TPS_MANI_03545]	Existence of `HealthChannelExternalStatus`
[TPS_MANI_03546]	Definition of reported health status `RPortPrototype`
[TPS_MANI_03547]	Definition of *offset* time domains
[TPS_MANI_03548]	Definition of `TimeSyncPortPrototypeToTimeBaseMapping`
[TPS_MANI_03549]	Usage of `RPortPrototype` for the interaction with Time Synchronization
[TPS_MANI_03550]	Usage of `RPortPrototype` for the interaction with Platform Health Management
[TPS_MANI_03551]	Definition of Time Base kind
[TPS_MANI_03552]	Supervision cycle for `GlobalSupervision`

Table F.9: Added Specification Items in R18-03

F.3.2 Changed Specification Items in R18-03

Number	Heading
[TPS_MANI_01006]	Ordered definition of ServiceInterface.namespace
[TPS_MANI_01008]	Semantics of ExecutableGroup
[TPS_MANI_01009]	Standardized values of ExecutableGroup.category
[TPS_MANI_01013]	Semantics of meta-class ModeDependentStartupConfig
[TPS_MANI_01017]	Relation of startup configuration to resource group
[TPS_MANI_01041]	Startup configuration supports the definition of a launch sequence dependency
[TPS_MANI_01042]	Definition of a linear ImplementationDataType of category VECTOR
[TPS_MANI_01044]	Structure of an ImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_01060]	Use cases for the application of DiagnosticServiceDataMapping
[TPS_MANI_01068]	Semantics of PersistencyFileProxyInterface.maxNumberOfFiles
[TPS_MANI_01069]	Further qualification of properties of PortPrototypes typed by PersistencyKeyValueDatabaseInterfaces
[TPS_MANI_01075]	Specification of redundancy of persistent data
[TPS_MANI_01078]	Semantics of PersistencyPortPrototypeToKeyValueDatabaseMapping
[TPS_MANI_01080]	Semantics of PersistencyPortPrototypeToFileArrayMapping
[TPS_MANI_01097]	Assignment of TLV data ids for data structures with optional members
[TPS_MANI_01100]	Semantics of Allocator
[TPS_MANI_01109]	Semantics of UploadablePackageElement
[TPS_MANI_01112]	Semantics of SoftwareClusterDesign
[TPS_MANI_01113]	Semantics of SoftwareClusterDesign.diagnosticAddress
[TPS_MANI_01116]	Reference to model elements included in an uploadable software package
[TPS_MANI_01117]	Semantics of SoftwareClusterDesign.intendedTargetMachine
[TPS_MANI_01118]	Relation between SoftwareClusterDesign and DiagnosticContributionSet
[TPS_MANI_01119]	Reference to model elements from SoftwareClusterDesign
[TPS_MANI_01133]	Optional element of an event
[TPS_MANI_01134]	Optional element in the context of a method
[TPS_MANI_03001]	Mapping of AdaptivePlatformServiceInstance to a MachineDesign
[TPS_MANI_03002]	IP configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03003]	ProvidedSomeipServiceInstance Fanout
[TPS_MANI_03022]	Context of RequiredSomeipServiceInstance
[TPS_MANI_03110]	Allowed components in system description with category SYSTEM_DESIGN_DESCRIPTION.
[TPS_MANI_03114]	Usage of AssemblySwConnectors in the System Design model
[TPS_MANI_03145]	Description of a function group

▽

△

Number	Heading
[TPS_MANI_03152]	Assignment of a ModeDependentStartupConfig to a function group state
[TPS_MANI_03153]	Semantics of ModeDependentStartupConfig.functionGroupMode
[TPS_MANI_03500]	Definition of Platform Health Management Supervision and Checkpoints
[TPS_MANI_03503]	Applicability of supervision to a specific Process
[TPS_MANI_03505]	Existence of SupervisionCheckpoint
[TPS_MANI_03506]	Optionality of SupervisionCheckpoint
[TPS_MANI_03508]	Definition of an AliveSupervision for a SupervisionCheckpoint
[TPS_MANI_03509]	Definition of a CheckpointTransition
[TPS_MANI_03510]	Definition of LogicalSupervision
[TPS_MANI_03512]	Applicability of global supervision to a specific Process
[TPS_MANI_03513]	Collection of LocalSupervisions into a global supervision
[TPS_MANI_03514]	Expiration tolerance for GlobalSupervision
[TPS_MANI_03515]	Expiration tolerance for LocalSupervision
[TPS_MANI_03516]	Condition evaluation for HealthChannelSupervision
[TPS_MANI_03517]	Condition evaluation for HealthChannelExternalStatus

Table F.10: Changed Specification Items in R18-03

F.3.3 Deleted Specification Items in R18-03

Number	Heading
[TPS_MANI_01031]	Semantics of CompositionDataPrototypeRef
[TPS_MANI_01045]	Process.modeDependentStartupConfig that does not refer to a ModeDeclaration
[TPS_MANI_01132]	Semantics of CompositionDataPrototypeRef
[TPS_MANI_03019]	TTL for SubscribeEventGroupAck Entries
[TPS_MANI_03501]	Definition of platform health management supervised entities
[TPS_MANI_03504]	Existence of SupervisionEntity

Table F.11: Deleted Specification Items in R18-03

F.3.4 Added Constraints in R18-03

Number	Heading
[constr_1546]	Existence of attributes of ServiceInterfaceSubElement
[constr_1547]	Reference from ImplementationDataTypeExtension to ImplementationDataType
[constr_1548]	Reference from ImplementationDataTypeElementExtension to ImplementationDataTypeElement
[constr_1549]	Value of ProcessorCore.coreId
[constr_1550]	Reference from Process to ProcessDesign
[constr_1551]	Existence of CompositionDataPrototypeRef.dataPrototype vs. CompositionDataPrototypeRef.elementInImplDatatype
[constr_1553]	Restriction for ProcessToMachineMapping
[constr_1554]	Restriction regarding PersistencyKeyValuePair.initValue
[constr_1555]	Restriction applicable for PersistencyPortPrototypeToKeyValueDatabaseMapping.portPrototype
[constr_1556]	Restriction applicable for PersistencyPortPrototypeToFileArrayMapping.portPrototype
[constr_1557]	Standardized values of SoftwareClusterDesign.category and SoftwareCluster.category
[constr_1558]	Existence of SoftwareClusterDesign.diagnosticAddress
[constr_1559]	Existence of SoftwareClusterDesign.subSoftwareCluster
[constr_1560]	Usage of SoftwareClusterDesign.requiredARElement
[constr_1561]	Existence of SoftwareClusterDesign.subSoftwareCluster and SoftwareClusterDesign.dependsOn.dependentSoftwareClusterDesign
[constr_1562]	Existence of SoftwareClusterDesign.diagnosticContribution
[constr_1563]	Standardized values of SoftwareClusterDesign.category and SoftwareCluster.category
[constr_1564]	Existence of SoftwareCluster.diagnosticAddress
[constr_1565]	Existence of SoftwareCluster.subSoftwareCluster
[constr_1566]	Usage of SoftwareCluster.containedARElement
[constr_1567]	Existence of SoftwareCluster.subSoftwareCluster and SoftwareCluster.dependsOn.dependentSoftwareCluster
[constr_1568]	Existence of SoftwareCluster.diagnosticExtract
[constr_1569]	Restriction for the scope of RestHttpPortPrototypeMapping.acceptsEncoding
[constr_1570]	Restriction for UserDefinedServiceInterfaceDeployment of category SERVICE_INTERFACE_DEPLOYMENT_IPC
[constr_1571]	CppImplementationDataType is limited
[constr_1572]	Usage of SwDataDefProps.implementationDataType within a CppImplementationDataType
[constr_1573]	CppTemplateArgument.isVariadicTemplate is set to True

▽

△

Number	Heading
[constr_1574]	Number of CppTemplateArguments with isVariadicTemplate set to True
[constr_1575]	Position of CppTemplateArgument with isVariadicTemplate set to True
[constr_1576]	Existence of CppTemplateArgument.templateType vs. CppTemplateArgument.allocator
[constr_1577]	Specification of a nativeDeclaration for a CppImplementationDataType
[constr_1578]	applicable data categories
[constr_1579]	SwDataDefProps applicable to CppImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1580]	Restriction for the usage of RestHttpPortPrototypeMapping.acceptsEncoding
[constr_1581]	Value of fileProxy.fileName
[constr_1582]	PersistencyKeyValuePair.valueDataType shall match to ImplementationDataType for the corresponding PersistencyDataElement
[constr_1585]	Standardized values of attribute DiagnosticServiceSwMapping.category
[constr_1586]	DiagnosticServiceSwMapping.category set to DATA_ELEMENT
[constr_1587]	DiagnosticServiceSwMapping.category set to DATA_IDENTIFIER
[constr_1588]	DiagnosticServiceSwMapping.category set to GENERIC_UDS_SERVICE
[constr_1589]	Value of file.fileName
[constr_3408]	Value range of SomeipEventDeployment.eventId
[constr_3409]	Value range of SomeipMethodDeployment.methodId
[constr_3410]	Value range of SomeipServiceInterfaceDeployment.serviceInterfaceId
[constr_3411]	eventMulticastUdpPort, ipv4MulticastIpAddress and ipv6MulticastIpAddress not relevant for RequiredSomeipServiceInstances
[constr_3412]	OsModuleInstantiation shall have at least one ResourceGroup
[constr_3413]	ModeDependentStartupConfig of a Process is mapped to exactly one ResourceGroup
[constr_3414]	Allowed usage of EthernetNetworkConfiguration attributes
[constr_3415]	Value range of loadBalancingPriority
[constr_3416]	Value range of loadBalancingWeight
[constr_3417]	UserDefinedEventDeployments aggregated by a UserDefinedFieldDeployment
[constr_3418]	UserDefinedMethodDeployments aggregated by a UserDefinedFieldDeployment
[constr_3419]	Allowed usage of EthernetNetworkConfiguration attributes
[constr_3420]	System category for a design description that has one single Adaptive Machine in scope
[constr_3421]	Fibex elements applicable for a MACHINE_DESIGN_EXTRACT
[constr_3422]	CppImplementationDataType of category STRING and SwBaseType
[constr_3423]	ModeDependentStartupConfig of a Process shall reference a functionGroupMode or machineMode
[constr_3424]	ModeDependentStartupConfig shall never reference the functionGroupMode Off

▽

△

Number	Heading
[constr_3425]	Restriction of DoIpInstantiations on a Machine
[constr_3426]	The logTraceFilePath is mandatory in case that logTraceLogMode is set to file
[constr_3427]	The logTraceFilePath is only relevant if logTraceLogMode is set to file
[constr_3428]	Structure shall own at least one element
[constr_3429]	No allocator usage for CppImplementationDataTypes of category VARIANT
[constr_3432]	Allowed subElements for Structures
[constr_3433]	Aggregation of templateArguments for a ARRAY
[constr_3434]	Aggregation of templateArguments for a VECTOR
[constr_3528]	Value range of domainId
[constr_3529]	Value range of serviceInstanceId
[constr_3530]	Mandatory definition of checkpointId
[constr_3531]	Mandatory definition of healthChannelId
[constr_3532]	Mandatory definition of statusId
[constr_3536]	Mandatory definition of supervisedEntityId

Table F.12: Added Constraints in R18-03

F.3.5 Changed Constraints in R18-03

Number	Heading
[constr_1484]	Applicability of ModeDependentStartupConfig.executionDependency
[constr_1507]	PortInterfaceToDataTypeMapping is only applicable to ServiceInterface
[constr_1532]	Consistent assignment of TLV data ids to data structures with optional members
[constr_1537]	Consistent assignment of TLV data ids to arguments of a given ClientServerOperation
[constr_3307]	SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3308]	SomeipEventDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3309]	SomeipMethodDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3310]	SomeipMethodDeployment.transportProtocol setting to tcp and the impact on ProvidedSomeipServiceInstances
[constr_3320]	Aggregation of CommunicationConnector by MachineDesign
[constr_3350]	Consistent value of category for ExecutableGroups referencing an Executable
[constr_3366]	System category for a system design description with Adaptive Platform and Classic Platform content

Table F.13: Changed Constraints in R18-03

F.3.6 Deleted Constraints in R18-03

Number	Heading
[constr_1480]	Mutual existence of CompositionDataPrototypeRef.elementInImplDatatype vs. attributes of CompositionDataPrototypeRef.dataPrototype
[constr_1505]	Number of Process.modeDependentStartupConfig that do not refer to a ModeDeclaration
[constr_1525]	Standardized values of PersistencyFile.category
[constr_1526]	Values of PersistencyFileArray.file.category
[constr_1533]	Applicability of ImplementationDataTypeElementExtension

Table F.14: Deleted Constraints in R18-03

F.4 Constraint and Specification Item History of this document according to AUTOSAR Release R18-10

F.4.1 Added Specification Items in R18-10

Number	Heading
[TPS_MANI_01184]	Definition of optional elements on the level of ApplicationDataType
[TPS_MANI_01185]	Definition of optional elements on the level of CppImplementationDataType
[TPS_MANI_01186]	Definition of the applicable wire type
[TPS_MANI_01187]	Matching pairs of PersistencyFileProxy and PersistencyFile
[TPS_MANI_01188]	Semantics of attribute schedulingPriority
[TPS_MANI_01189]	Software Cluster and DiagnosticContributionSet.category
[TPS_MANI_01190]	Semantics of ApApplicationError
[TPS_MANI_01191]	Modeling of possible errors
[TPS_MANI_01192]	Semantics of ApApplicationErrorDomain
[TPS_MANI_01193]	Combination of ModeDependentStartupConfig.machineMode and ModeDependentStartupConfig.functionGroupMode
[TPS_MANI_01194]	Semantics of PersistencyKeyValueDatabaseInterface.minimumSustainedSize
[TPS_MANI_01195]	Semantics of PersistencyFileProxyInterface.minimumSustainedSize
[TPS_MANI_01196]	Semantics of PersistencyKeyValueDatabase.minimumSustainedSize
[TPS_MANI_01197]	Semantics of PersistencyKeyValueDatabase.maximumAllowedSize
[TPS_MANI_01198]	Semantics of ApApplicationErrorSet
[TPS_MANI_01199]	Semantics of DeterministicClientResourceNeeds
[TPS_MANI_01200]	Semantics of meta-class DeterministicClientResource

▽

△

Number	Heading
[TPS_MANI_01201]	Standardized values for attribute CppTemplateArgument.category
[TPS_MANI_01202]	Semantics of reference SoftwareCluster.moduleInstantiation
[TPS_MANI_01203]	Semantics of DeterministicClient
[TPS_MANI_01204]	Specification of redundancy of persistent data
[TPS_MANI_01205]	Semantics of meta-class PersistencyDeployment
[TPS_MANI_01206]	Modeling of redundancy in the context of PersistencyDeployment
[TPS_MANI_01207]	Standardized values of attribute PersistencyRedundancyCrc.algorithmFamily
[TPS_MANI_01208]	Definition of envionment variables in the scope of a Machine
[TPS_MANI_01209]	Definition of envionment variables in process scope
[TPS_MANI_01210]	Default encoding for all DataPrototypes typed by CppImplementationDataType of category STRING
[TPS_MANI_03197]	Semantics of StdCppImplementationDataType
[TPS_MANI_03198]	Semantics of CustomCppImplementationDataType
[TPS_MANI_03199]	Endpoint protection by SecureComProps
[TPS_MANI_03200]	SecureComProps for udp, tcp and multicast communication
[TPS_MANI_03201]	Semantics of CppTemplateArgument.inplace attribute
[TPS_MANI_03202]	Definition of bitfield types
[TPS_MANI_03203]	Configuration of IPsec
[TPS_MANI_03204]	Definition of IPSecRules
[TPS_MANI_03205]	IPsec connection type
[TPS_MANI_03206]	IPsec AH and ESP protocol configuration
[TPS_MANI_03207]	IPsec Internet Key Exchange protocol configuration
[TPS_MANI_03208]	Protection of AdaptivePlatformServiceInstance by IPsec
[TPS_MANI_03209]	The meaning of MachineDesign.accessControl
[TPS_MANI_03210]	Specification of event specific communication attributes
[TPS_MANI_03211]	Specification of field specific communication attributes
[TPS_MANI_03212]	Specification of initial value for a field
[TPS_MANI_03213]	Semantics of meta-class TlsSecureComProps
[TPS_MANI_03214]	Existence of TlsCryptoCipherSuite.keyExchange vs. TlsSecureComProps.keyExchange
[TPS_MANI_03215]	Semantics of CryptoServiceCertificate
[TPS_MANI_03216]	Existence of TlsCryptoCipherSuite.certificate in the *client* role
[TPS_MANI_03217]	On-the-wire encoding for a chosen string
[TPS_MANI_03218]	Default value for the attribute tcpInitialInactivityTime of meta-class DoIpNetworkConfiguration
[TPS_MANI_03219]	Default value for the attribute tcpGeneralInactivityTime of meta-class DoIpNetworkConfiguration
[TPS_MANI_03220]	Default value for the attribute vehicleAnnouncementCount of meta-class DoIpNetworkConfiguration

▽

△

Number	Heading
[TPS_MANI_03221]	Default value for the attribute `vehicleAnnouncementInterval` of meta-class `DoIpNetworkConfiguration`
[TPS_MANI_03222]	Default value for the attribute `tcpAliveCheckResponseTimeout` of meta-class `DoIpNetworkConfiguration`
[TPS_MANI_03553]	Applicability of health channel to a specific `Process`
[TPS_MANI_03554]	Several `SomeipServiceInstanceToMachineMapping`s with equal settings
[TPS_MANI_03555]	Mix of `SomeipServiceInstanceToMachineMapping` and signal-based communication
[TPS_MANI_03556]	DDS-RPC Service Binding
[TPS_MANI_03557]	DDS `ClientServerOperation` Binding
[TPS_MANI_03558]	DDS `Field` Binding
[TPS_MANI_03559]	Definition of `DdsProvidedServiceInstance.methodQosProps`
[TPS_MANI_03560]	`qosProfile` of `DdsProvidedServiceInstance.methodQosProps` is optional
[TPS_MANI_03561]	Definition of `DdsProvidedServiceInstance.fieldNotifierQosProps`
[TPS_MANI_03562]	`qosProfile` of `DdsProvidedServiceInstance.fieldNotifierQosProps` is optional
[TPS_MANI_03563]	Definition of `DdsProvidedServiceInstance.fieldGetSetQosProps`
[TPS_MANI_03564]	`qosProfile` of `DdsProvidedServiceInstance.fieldGetSetQosProps` is optional
[TPS_MANI_03565]	Definition of `DdsRequiredServiceInstance.methodQosProps`
[TPS_MANI_03566]	`qosProfile` of `DdsRequiredServiceInstance.methodQosProps` is optional
[TPS_MANI_03567]	Definition of `DdsRequiredServiceInstance.fieldNotifierQosProps`
[TPS_MANI_03568]	`qosProfile` of `DdsRequiredServiceInstance.fieldNotifierQosProps` is optional
[TPS_MANI_03569]	Definition of `DdsRequiredServiceInstance.fieldGetSetQosProps`
[TPS_MANI_03570]	`qosProfile` of `DdsRequiredServiceInstance.fieldGetSetQosProps` is optional
[TPS_MANI_03571]	`transportPlugin` for `DdsProvidedServiceInstance`
[TPS_MANI_03572]	`transportPlugin` for `DdsRequiredServiceInstance`

Table F.15: Added Specification Items in R18-10

F.4.2 Changed Specification Items in R18-10

Number	Heading
[TPS_MANI_01001]	Meaning of ServiceInterface
[TPS_MANI_01041]	Startup configuration supports the definition of a launch sequence dependency
[TPS_MANI_01097]	Assignment of TLV data ids
[TPS_MANI_01100]	Semantics of Allocator
[TPS_MANI_01147]	Semantics of PersistencyKeyValueDatabase.updateStrategy
[TPS_MANI_01151]	Semantics of PersistencyFileArray.updateStrategy
[TPS_MANI_01166]	Semantics of CppImplementationDataType
[TPS_MANI_01176]	Standardized value for attribute CppImplementationDataType.typeEmitter
[TPS_MANI_01177]	Semantics of CppImplementationDataType.typeEmitter
[TPS_MANI_01180]	Collection of data types that requires serialization support
[TPS_MANI_03001]	Mapping of AdaptivePlatformServiceInstance to a MachineDesign
[TPS_MANI_03011]	Server Timing configuration for a ProvidedSomeipServiceInstance
[TPS_MANI_03021]	Requirements on the searched minor version from the client's point of view
[TPS_MANI_03025]	Client Timing configuration for a RequiredSomeipServiceInstance
[TPS_MANI_03070]	Size of a length field for a chosen array or map
[TPS_MANI_03103]	Default size for all array and map length fields
[TPS_MANI_03124]	ServiceInterface.event to ISignalTriggering mapping
[TPS_MANI_03125]	ServiceInterface.method to ISignalTriggerings mapping
[TPS_MANI_03126]	ServiceInterface.field mapping to ISignalTriggerings
[TPS_MANI_03134]	Configuration of supported TLS ciphersuites
[TPS_MANI_03137]	ServiceInterfaceElementSecureComConfig is not relevant in case of TLS communication
[TPS_MANI_03157]	Enabling of data collection for upd data transmission
[TPS_MANI_03158]	Configuration of a data collection on a ProvidedServiceInstance for transmission over udp
[TPS_MANI_03165]	Network Interface configuration for DoIP
[TPS_MANI_03170]	CppImplementationDataType of category ARRAY
[TPS_MANI_03173]	Definition of a multidimensional Array
[TPS_MANI_03175]	CppImplementationDataType of category VECTOR
[TPS_MANI_03177]	Definition of a multidimensional Vector
[TPS_MANI_03178]	StdCppImplementationDataType of category STRING
[TPS_MANI_03179]	C++ language binding of StdCppImplementationDataTypes of category STRING
[TPS_MANI_03180]	Definition of Structures
[TPS_MANI_03181]	Definition of members in StdCppImplementationDataType of category STRUCTURE

▽

△

Number	Heading
[TPS_MANI_03184]	CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03185]	Structure of an CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03187]	Definition of enumeration types
[TPS_MANI_03193]	CppImplementationDataType of category TYPE_REFERENCE
[TPS_MANI_03196]	Semantics of CppImplementationDataTypeElementQualifier.inplace attribute
[TPS_MANI_03503]	Applicability of checkpoints to a specific Process
[TPS_MANI_03512]	Applicability of global supervision without Process context
[TPS_MANI_03516]	Condition evaluation for HealthChannelSupervision
[TPS_MANI_03518]	PhmLogicalExpression definition
[TPS_MANI_03519]	PhmRule definition
[TPS_MANI_03520]	Execution of PhmActionList with actionListExecution=triggeredOnEvaluation
[TPS_MANI_03521]	Execution of PhmActionList with actionListExecution=triggeredOnChange
[TPS_MANI_03522]	Definition of actions for application software
[TPS_MANI_03523]	Definition of actions for Platform Instance
[TPS_MANI_03524]	Definition of actions for Watchdog
[TPS_MANI_03526]	DDS VariableDataPrototype binding
[TPS_MANI_03527]	Definition of DdsProvidedServiceInstance
[TPS_MANI_03528]	Definition of DdsProvidedServiceInstance.eventQosProps
[TPS_MANI_03529]	Definition of DdsRequiredServiceInstance
[TPS_MANI_03530]	Definition of DdsRequiredServiceInstance.eventQosProps
[TPS_MANI_03531]	qosProfile of DdsProvidedServiceInstance.eventQosProps is optional
[TPS_MANI_03532]	qosProfile of DdsRequiredServiceInstance.eventQosProps is optional
[TPS_MANI_03533]	DdsServiceInstanceToMachineMapping
[TPS_MANI_03552]	Supervision cycle for GlobalSupervision

Table F.16: Changed Specification Items in R18-10

F.4.3 Deleted Specification Items in R18-10

Number	Heading
[TPS_MANI_01008]	Semantics of ExecutableGroup
[TPS_MANI_01009]	Standardized values of ExecutableGroup.category
[TPS_MANI_01018]	ImplementationDataType of category VECTOR

▽

△

Number	Heading
[TPS_MANI_01028]	ImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_01029]	Usage of ImplementationDataType
[TPS_MANI_01030]	ImplementationDataType of category STRING
[TPS_MANI_01042]	Definition of a linear ImplementationDataType of category VECTOR
[TPS_MANI_01043]	Definition of a rectangular ImplementationDataType of category VECTOR
[TPS_MANI_01044]	Structure of an ImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_01055]	Definition of application-level errors
[TPS_MANI_01056]	Semantics of ApplicationError.errorContext
[TPS_MANI_01058]	Ability to create a mapping of ApplicationErrors aggregated in the role possibleError
[TPS_MANI_01062]	ImplementationDataType to generate a C++ enum
[TPS_MANI_01063]	Sharing of ImplementationDataType with enumeration semantics
[TPS_MANI_01074]	Specification of encryption of persistent data
[TPS_MANI_01075]	Specification of redundancy of persistent data
[TPS_MANI_01077]	Specification of file encryption
[TPS_MANI_01082]	Eligibility of DataPrototypes for the definition of optionality
[TPS_MANI_01083]	Optionality is supported for ApplicationDataType as well as ImplementationDataType
[TPS_MANI_01084]	Optionality for a DataPrototype typed by an ApplicationDataType
[TPS_MANI_01085]	Definition of optionality for a DataPrototype typed by an ImplementationDataType
[TPS_MANI_01087]	Interaction with crypto software
[TPS_MANI_01088]	Semantics of CryptoNeed
[TPS_MANI_01089]	Relation between CryptoNeed and PortPrototype
[TPS_MANI_01090]	Modeling of crypto software as a platform module
[TPS_MANI_01091]	Semantics of CryptoJob
[TPS_MANI_01092]	Mapping between CryptoNeed and CryptoJob
[TPS_MANI_01093]	Semantics of CryptoDriver
[TPS_MANI_01094]	Scope of CryptoDriver
[TPS_MANI_01095]	Semantics of CryptoKeySlot
[TPS_MANI_01096]	Semantics of the CryptoPrimitive
[TPS_MANI_01098]	Constraints on the definition of an ImplementationDataType of category VECTOR
[TPS_MANI_01099]	Semantics of ImplementationDataTypeElementExtension
[TPS_MANI_01101]	Size-constrained allocation of memory
[TPS_MANI_01102]	Specification of a namespace for an ImplementationDataType of category VECTOR
[TPS_MANI_01133]	Optional element of an event
[TPS_MANI_01134]	Optional element in the context of a method

▽

△

Number	Heading
[TPS_MANI_03121]	Signal-based VariableDataPrototype binding
[TPS_MANI_03122]	Signal-based Field binding
[TPS_MANI_03123]	Signal-based ClientServerOperation binding
[TPS_MANI_03135]	Configuration of TLS PSK Identity
[TPS_MANI_03136]	Configuration of requirements for the TLS cryptographic job
[TPS_MANI_03141]	Mapping between SecOcJobRequirement and CryptoJob
[TPS_MANI_03142]	Mapping between TlsJobRequirement and CryptoJob
[TPS_MANI_03143]	Mapping between PresharedKeyIdentity and CryptoKeySlot
[TPS_MANI_03144]	C++ language binding of ImplementationDataTypes of category STRING
[TPS_MANI_03182]	Definition of members in CppImplementationDataTypeElement of category STRUCTURE

Table F.17: Deleted Specification Items in R18-10

F.4.4 Added Constraints in R18-10

Number	Heading
[constr_1593]	Completeness of the existence of a set of TlvDataIdDefinition.tlvArguments
[constr_1594]	Consistent assignment of TLV data ids to ApplicationRecordDataType
[constr_1595]	Consistent assignment of TLV data ids to CppImplementationDataType or CppImplementationDataTypeElement
[constr_1596]	Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to ArgumentDataPrototype
[constr_1597]	Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to ApplicationRecordElement
[constr_1598]	Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to CppImplementationDataTypeElement
[constr_1599]	TlvDataIdDefinition referencing ArgumentDataPrototype
[constr_1600]	TlvDataIdDefinition referencing ApplicationRecordElement
[constr_1601]	TlvDataIdDefinition referencing CppImplementationDataTypeElement
[constr_1603]	Completeness of the existence of a set of TlvDataIdDefinition.tlvRecordElements
[constr_1604]	Completeness of the existence of a set of TlvDataIdDefinition.tlvSubElements
[constr_1605]	Standardized values of attribute Executable.category
[constr_1606]	Processes with mutual ExecutionDependencys
[constr_1613]	File name of matching pairs of PersistencyFileProxy and PersistencyFile
[constr_1614]	Existence of attribute TransformationPropsToServiceInterfaceElementMapping.transformationProps.sessionHandling

▽

△

Number	Heading
[constr_1615]	Existence of attribute SomeipDataPrototypeTransformationProps.someip-TransformationProps.sessionHandling
[constr_1618]	Ability to shut down
[constr_1619]	Ability to restart
[constr_1620]	Value of schedulingPriority if schedulingPolicy is set to scheduling-PolicyFifo or schedulingPolicyRoundRobin
[constr_1621]	Value of schedulingPriority if StartupConfig.schedulingPolicy is set to schedulingPolicyOther
[constr_1625]	Existence of reference ApApplicationError.errorDomain
[constr_1627]	Supported value range for attribute ApApplicationErrorDomain.value
[constr_1628]	Definition of static length field sizes in case of TLV usage
[constr_1629]	Identical sizes of length fields in case of TLV usage
[constr_1630]	No definition of length field sizes on DataPrototype level in case of TLV usage
[constr_1658]	Number of DiagnosticTroubleCodeUdsToClearConditionGroupMapping elements per DiagnosticTroubleCodeUds
[constr_1659]	Restriction for the usage of CppImplementationDataTypeElementQualifier.inplace
[constr_1660]	Restriction for the usage of CppTemplateArgument.inplace
[constr_1661]	Multiplicity of OsModuleInstantiation.resourceGroup
[constr_1663]	Standardized values of attribute DiagnosticServiceDataIdentifier-PortMapping.category
[constr_1664]	Unique ApApplicationError.shortName
[constr_1665]	Unique ApApplicationError.errorCode
[constr_1666]	References from PersistencyPortPrototypeToKeyValueDatabaseMapping to PersistencyKeyValueDatabase
[constr_1667]	References from PersistencyPortPrototypeToFileArrayMapping to Per-sistencyFileArray
[constr_1668]	Allowed combinations of PersistencyRedundancyCrc.length and algo-rithmFamily
[constr_1673]	Existence of attributes hasGetter, hasSetter, and hasNotifier
[constr_1674]	Suported encoding of StdCppImplementationDataType of category STRING
[constr_1675]	Existence of attribute ApSomeipTransformationProps.stringEncoding
[constr_1676]	Consistency of references shallRunOn and shallNotRunOn
[constr_1677]	Mutual exclusive existence of references shallRunOn and shallNotRunOn
[constr_1678]	Allowed values for attribute ApSomeipTransformationProps.stringEncoding
[constr_3443]	Specification of a namespace for a StdCppImplementationDataType
[constr_3446]	CppTemplateArgument with allocator reference and the inplace flag
[constr_3447]	ApSomeipTransformationProps.sizeOfArrayLengthField that equals 0
[constr_3462]	CppTemplateArgument.templateType reference to StdCppImplementa-tionDataType of category STRUCTURE and the inplace flag
[constr_3485]	UDP endpoint using DTLS can only serve provided or required service instances exclusively

▽

△

Number	Heading
[constr_3486]	TCP endpoint using TLS can only serve provided or required service instances exclusively.
[constr_3487]	TCP endpoint can only serve provided or required service instances exclusively
[constr_3492]	`DoIpInstantiation.logicalAddress` shall be defined as member in the `DoIpRequestConfiguration`
[constr_3493]	Applicable attributes for standardized E2E Profiles
[constr_3494]	Mandatory Machine States
[constr_3495]	Supported value range for attribute `DoIpInstantiation.eid`
[constr_3496]	Supported value range for attribute `DoIpInstantiation.gid`
[constr_3497]	Supported value range for attribute `DoIpInstantiation.maxRequestBytes`
[constr_3498]	Supported value range for attribute `DoIpInstantiation.logicalAddress`
[constr_3499]	Supported value range for attribute `DoIpRequestConfiguration.startAddress`
[constr_3537]	`LocalSupervision` referenced once in the context of a `GlobalSupervision`
[constr_3538]	Only one `ServiceInstanceToMachineMapping` per technology and `CommunicationConnector`
[constr_3539]	Only one `AliveSupervision` per `SupervisionCheckpoint`
[constr_3540]	`SupervisionCheckpoint` in supervision graph
[constr_3541]	`qosProfile` mandatory for `DdsProvidedServiceInstance`
[constr_3542]	`qosProfile` mandatory for `DdsRequiredServiceInstance`
[constr_3543]	At least one `transportPlugin` definition required for each `DdsProvidedServiceInstance`
[constr_3544]	At least one `transportPlugin` definition required for each `DdsRequiredServiceInstance`
[constr_5000]	Supported value range for attribute `DoIpRequestConfiguration.endAddress`
[constr_5001]	Usage of `DoIpNetworkConfiguration.eidUseMac`
[constr_5002]	Supported values of `ServiceInstanceToMachineMapping.category`
[constr_5003]	Existence of `TlsCryptoCipherSuite.certificate` in the *server* role
[constr_5004]	Mapping of a `Process` to a `Machine` is mandatory in the Execution Manifest

Table F.18: Added Constraints in R18-10

F.4.5 Changed Constraints in R18-10

Number	Heading
[constr_1490]	Allowed value of `category` for reference `ProcessToMachineMapping.process.executable`
[constr_1551]	Existence of `DataPrototypeInServiceInterfaceRef.dataPrototype` vs. `DataPrototypeInServiceInterfaceRef.elementInImplDatatype`

▽

△

Number	Heading
[constr_1572]	Usage of SwDataDefProps.implementationDataType within a CppImplementationDataType
[constr_1573]	CppTemplateArgument.isVariadicTemplate is set to True
[constr_1582]	PersistencyKeyValuePair.valueDataType shall match to AbstractImplementationDataType for the corresponding PersistencyDataElement
[constr_1585]	Standardized values of attribute DiagnosticServiceSwMapping.category
[constr_1589]	Value of file.fileName
[constr_3375]	method with attribute fireAndForget set to true shall not reference an ApApplicationError
[constr_3392]	ServiceInterfaceElementSecureComConfig.dataId and ServiceInterfaceElementSecureComConfig.freshnessValueId are mandatory in case of SecOC communication
[constr_3433]	Aggregation of templateArguments for an ARRAY
[constr_3434]	Aggregation of templateArguments for a VECTOR
[constr_3527]	PhmLogicalExpression referenced by one PhmRule
[constr_3528]	Value range of domainId
[constr_3529]	Value range of serviceInstanceId

Table F.19: Changed Constraints in R18-10

F.4.6 Deleted Constraints in R18-10

Number	Heading
[constr_1474]	SwDataDefProps applicable to ImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1475]	ImplementationDataType of category STRING is limited
[constr_1476]	ImplementationDataType of category VECTOR is limited
[constr_1477]	ImplementationDataType of category ASSOCIATIVE_MAP is limited
[constr_1479]	No support for certain values of ImplementationDataType.category
[constr_1484]	Applicability of ModeDependentStartupConfig.executionDependency
[constr_1485]	No subElement for ImplementationDataType of category STRING
[constr_1486]	ImplementationDataType of category STRING and SwBaseType
[constr_1487]	Number of subElements of an ImplementationDataType of category ASSOCIATIVE_MAP
[constr_1491]	Semantics of ServiceInterface.possibleError
[constr_1493]	ArgumentDataPrototype referenced in the role ApplicationError.errorContext
[constr_1495]	Initial value for field

▽

△

Number	Heading
[constr_1506]	ImplementationDataType of category VECTOR shall not define dynamicArraySizeProfile
[constr_1508]	BaseTypeDirectDefinition.nativeDeclaration shall not be set to the value enum
[constr_1522]	Semantics of ClientServerOperation.possibleError
[constr_1527]	ImplementationDataTypeElement finally referenced as the target element in the context of an ImplementationDataTypeElementInAutosarDataPrototypeRef
[constr_1528]	Definition of optionality for multiple DataPrototypes typed by the same AutosarDataType
[constr_1529]	Standardized values of CryptoNeed.category
[constr_1530]	Standardized values of CryptoPrimitive.algorithmFamily and CryptoKeySlot.algorithmFamily
[constr_1531]	Standardized values of CryptoPrimitive.algorithmMode
[constr_1532]	Consistent assignment of TLV data ids to data structures with optional members
[constr_1537]	Consistent assignment of TLV data ids to arguments of a given ClientServerOperation
[constr_1546]	Existence of attributes of ServiceInterfaceSubElement
[constr_1547]	Reference from ImplementationDataTypeExtension to ImplementationDataType
[constr_1548]	Reference from ImplementationDataTypeElementExtension to ImplementationDataTypeElement
[constr_1577]	Specification of a nativeDeclaration for a CppImplementationDataType
[constr_1587]	DiagnosticServiceSwMapping.category set to DATA_IDENTIFIER
[constr_1588]	DiagnosticServiceSwMapping.category set to GENERIC_UDS_SERVICE
[constr_3293]	Mandatory information of a RequiredSomeipServiceInstance
[constr_3303]	ANY not allowed for SomeipServiceInterfaceDeployment.serviceInterfaceVersion
[constr_3350]	Consistent value of category for ExecutableGroups referencing an Executable
[constr_3377]	Restriction of ISignalTriggering references in SignalBasedFieldToISignalTriggeringMapping
[constr_3422]	CppImplementationDataType of category STRING and SwBaseType
[constr_3428]	Structure shall own at least one element
[constr_3432]	Allowed subElements for Structures

Table F.20: Deleted Constraints in R18-10

F.5 Constraint and Specification Item History of this document according to AUTOSAR Release R19-03

F.5.1 Added Specification Items in R19-03

Number	Heading
[TPS_MANI_01211]	Specification of executable software within SoftwareClusterDesign
[TPS_MANI_01212]	Usage of attribute typeEmitter in the context of a CustomCppImplementationDataType
[TPS_MANI_01213]	Semantics of meta-class StrongRevisionLabelString
[TPS_MANI_01214]	Semantics of SoftwareCluster.conflictsTo
[TPS_MANI_01215]	Semantics of meta-class SoftwareActivationDependencyFormula
[TPS_MANI_01216]	Semantics of meta-class SoftwareActivationDependencyFormulaPart
[TPS_MANI_01217]	Semantics of metaclass SoftwareActivationDependencyCompareCondition
[TPS_MANI_01218]	Cryptographic signature of SoftwareCluster
[TPS_MANI_01219]	License of software in included SoftwareCluster
[TPS_MANI_01220]	Release notes of software in included SoftwareCluster
[TPS_MANI_01221]	Semantics of meta-class SoftwarePackage
[TPS_MANI_01222]	Cryptographic signature of SoftwarePackage
[TPS_MANI_01223]	Semantics of attribute SoftwarePackage.packagerId
[TPS_MANI_01224]	Actions taken after installation of a SoftwarePackage
[TPS_MANI_01225]	Actions taken during installation of a SoftwarePackage
[TPS_MANI_01226]	Machine-specific configuration settings for the UCM module
[TPS_MANI_01227]	Semantics of attribute UcmModuleInstantiation.identifier
[TPS_MANI_01228]	Semantics of meta-class ProcessDesign
[TPS_MANI_01229]	Pre-allocation of a given ProcessDesign on a specific MachineDesign
[TPS_MANI_01230]	Semantics of DiagnosticProvidedDataMapping
[TPS_MANI_01231]	GrantDesign references ProcessDesign
[TPS_MANI_01232]	Semantics of meta-class ComOfferServiceGrantDesign
[TPS_MANI_01233]	Semantics of meta-class ComFindServiceGrantDesign
[TPS_MANI_01234]	Semantics of ComFieldGrantDesign
[TPS_MANI_01235]	Semantics of ComEventGrantDesign
[TPS_MANI_01236]	Semantics of ComMethodGrantDesign
[TPS_MANI_01237]	Semantics of meta-class ComFieldGrant
[TPS_MANI_01238]	Semantics of meta-class ComMethodGrant
[TPS_MANI_01239]	Semantics of meta-class ComEventGrant
[TPS_MANI_01240]	Semantics of meta-class ComOfferServiceGrant
[TPS_MANI_01241]	Semantics of meta-class ComFindServiceGrant

△

Number	Heading
[TPS_MANI_01242]	PortInterfaces used for communication with the AUTOSAR DiagnosticManager
[TPS_MANI_01243]	Semantics of DiagnosticDataIdentifierInterface
[TPS_MANI_01244]	Semantics of DiagnosticDataElementInterface
[TPS_MANI_01245]	Semantics of DiagnosticDataIdentifierGenericInterface
[TPS_MANI_01246]	Semantics of DiagnosticMonitorInterface
[TPS_MANI_01247]	Semantics of DiagnosticDTCInformationInterface
[TPS_MANI_01248]	Semantics of DiagnosticEventInterface
[TPS_MANI_01249]	Semantics of DiagnosticConditionInterface
[TPS_MANI_01250]	Semantics of DiagnosticIndicatorInterface
[TPS_MANI_01251]	Semantics of DiagnosticSecurityLevelInterface
[TPS_MANI_01252]	Semantics of DiagnosticServiceValidationInterface
[TPS_MANI_01253]	Semantics of DiagnosticOperationCycleInterface
[TPS_MANI_01254]	Semantics of DiagnosticGenericUdsInterface
[TPS_MANI_01255]	Semantics of DiagnosticGenericUdsInterface
[TPS_MANI_01256]	AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticIndicatorInterface
[TPS_MANI_01257]	AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticConditionInterface
[TPS_MANI_01258]	AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticGenericUdsInterface
[TPS_MANI_01259]	Mapping of DiagnosticClearCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01260]	Mapping of DiagnosticIndicator to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01261]	Mapping of DiagnosticMemoryDestination to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01262]	Mapping of DiagnosticSecurityLevel to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01263]	Mapping of DiagnosticDataIdentifier to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01264]	Mapping of DiagnosticServiceInstance to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01265]	Semantics of DiagnosticDownloadInterface and DiagnosticDownloadInterface
[TPS_MANI_01266]	Mapping of DiagnosticServiceInstance for upload/download to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_03223]	Existence of CppImplementationDataType
[TPS_MANI_03224]	Modeling of a Partial Network Cluster
[TPS_MANI_03225]	References to vlans in PncMapping
[TPS_MANI_03226]	Collection of partialNetworks and vlans in NmNetworkHandle

Table F.21: Added Specification Items in R19-03

F.5.2 Changed Specification Items in R19-03

Number	Heading
[TPS_MANI_01012]	Formal modeling of application startup behavior
[TPS_MANI_01013]	Semantics of meta-class StateDependentStartupConfig
[TPS_MANI_01017]	Relation of startup configuration to resource group
[TPS_MANI_01041]	Startup configuration supports the definition of a launch sequence dependency
[TPS_MANI_01046]	Semantics of StateDependentStartupConfig.functionGroupState
[TPS_MANI_01049]	Mapping of DiagnosticOperationCycle to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01050]	Mapping of DiagnosticEnableCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01051]	Mapping of DiagnosticStorageCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01136]	AutosarDataPrototype is the target of the DataPrototypeInServiceInterfaceRef
[TPS_MANI_01137]	Applicable use cases for DataPrototypeInServiceInterfaceRef
[TPS_MANI_01164]	Semantics of SoftwareCluster.dependsOn
[TPS_MANI_01177]	Semantics of CppImplementationDataType.typeEmitter
[TPS_MANI_01207]	Standardized values of attribute PersistencyRedundancyCrc.algorithmFamily
[TPS_MANI_03070]	Size of a length field for a chosen array or map
[TPS_MANI_03071]	Size of a length field for a chosen structure
[TPS_MANI_03072]	Size of a length field for a chosen union
[TPS_MANI_03073]	Alignment of a dynamic DataPrototype
[TPS_MANI_03074]	Size of a type selector field for a chosen union
[TPS_MANI_03075]	Byte Order of chosen DataPrototype in the serialized data stream
[TPS_MANI_03116]	Size of a length field for a chosen string
[TPS_MANI_03127]	Usage of End2EndEventProtectionProps
[TPS_MANI_03128]	Usage of same dataId in case of Multi-Binding
[TPS_MANI_03152]	Assignment of a StateDependentStartupConfig to a function group state
[TPS_MANI_03187]	Definition of enumeration types
[TPS_MANI_03190]	CppImplementationDataType of category VARIANT
[TPS_MANI_03202]	Definition of bitfield types
[TPS_MANI_03217]	On-the-wire encoding for a chosen string

Table F.22: Changed Specification Items in R19-03

F.5.3 Deleted Specification Items in R19-03

Number	Heading
[TPS_MANI_01038]	Diagnostic software mapping on the *AUTOSAR adaptive platform*
[TPS_MANI_01170]	Semantics of `CppTemplateArgument.isVariadicTemplate`
[TPS_MANI_01181]	Use cases for the application of `DiagnosticServiceSwMapping`
[TPS_MANI_01193]	Combination of `ModeDependentStartupConfig.machineMode` and `ModeDependentStartupConfig.functionGroupMode`
[TPS_MANI_03066]	Description of machine states
[TPS_MANI_03153]	Semantics of `ModeDependentStartupConfig.functionGroupMode`

Table F.23: Deleted Specification Items in R19-03

F.5.4 Added Constraints in R19-03

Number	Heading
[constr_1687]	Definition of machine state
[constr_1688]	`StateDependentStartupConfig` shall only refer to function group states of the same function group
[constr_1689]	Modeling of a startup dependency between different `Process`es
[constr_1690]	`SoftwareCluster` shall only be referenced by a single `SoftwarePackage`.
[constr_1691]	`UcmModuleInstantiation.identifier` shall be unique
[constr_1692]	Value of `schedulingPriority`
[constr_1693]	Relation of `Executable`, `ProcessDesign`, and `Process`
[constr_1695]	Semantics of a `Grant` depends on the existence of `IamModuleInstantiation`
[constr_1696]	`ClientServerOperation` aggregated by `DiagnosticRoutineInterface`
[constr_1697]	Restriction for `ClientServerOperation` aggregated by a `DiagnosticDataIdentifierInterface` or `DiagnosticDataElementInterface`
[constr_1698]	Target `SwcServiceDependency` of `DiagnosticClearConditionPortMapping.swcServiceDependencyInExecutable`
[constr_1699]	Target `SwcServiceDependency` of `DiagnosticIndicatorPortMapping.swcServiceDependencyInExecutable`
[constr_1700]	Target `SwcServiceDependency` of `DiagnosticMemoryDestinationPortMapping.swcServiceDependencyInExecutable`
[constr_1701]	Target `SwcServiceDependency` of `DiagnosticSecurityLevelPortMapping.swcServiceDependencyInExecutable`
[constr_1702]	Target `SwcServiceDependency` of `DiagnosticServiceDataIdentifierPortMapping.swcServiceDependencyInExecutable`
[constr_1703]	Target `SwcServiceDependency` of `DiagnosticGenericUdsPortMapping.swcServiceDependencyInExecutable`
[constr_1704]	Target `SwcServiceDependency` of `DiagnosticUploadDownloadPortMapping.swcServiceDependencyInExecutable`

▽

△

Number	Heading
[constr_5033]	Compatibility of data types with `category` `VALUE`
[constr_5034]	Compatibility of data types with `category` `BOOLEAN`
[constr_5035]	Compatibility of data types with `category` `STRING`
[constr_5036]	Compatibility of data types with `category` `ARRAY`
[constr_5037]	Compatibility of data types with `category` `ARRAY` with `variableSize`
[constr_5038]	Compatibility of data types with `category` `ARRAY` with `fixedSize`
[constr_5039]	Compatibility of data types with `category` `STRUCTURE`
[constr_5040]	Compatibility of `ApplicationRecordDataType` and `CppImplementationDataType` that both represent an `Optional Element Structure`
[constr_5041]	Compatibility of data types with `category` `ASSOCIATIVE_MAP`
[constr_5042]	No data type mapping for `CppImplementationDataType` of `category` `VARIANT`
[constr_5043]	Forbidden mappings to `CppImplementationDataType`
[constr_5044]	`DataTypeMap` for composite data types
[constr_5045]	Only one `SomeipServiceDiscovery` configuration per VLAN is allowed
[constr_5046]	Usage of `DoIpNetworkConfiguration.eidUseMac`
[constr_5047]	Supported values of `ServiceInstanceToMachineMapping.category`
[constr_5048]	Existence of `TlsCryptoCipherSuite.certificate` in the *server* role

Table F.24: Added Constraints in R19-03

F.5.5 Changed Constraints in R19-03

Number	Heading
[constr_1481]	Usage of `DataPrototypeInServiceInterfaceRef` in the *AUTOSAR adaptive platform*
[constr_1500]	Target `SwcServiceDependency` of `DiagnosticEventPortMapping.swcServiceDependencyInExecutable`
[constr_1501]	Target `SwcServiceDependency` of `DiagnosticOperationCyclePortMapping.swcServiceDependencyInExecutable`
[constr_1502]	Target `SwcServiceDependency` of `DiagnosticEnableConditionPortMapping.swcServiceDependencyInExecutable`
[constr_1503]	Target `SwcServiceDependency` of `DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable`
[constr_1551]	Existence of `DataPrototypeInServiceInterfaceRef.dataPrototype` vs. `DataPrototypeInServiceInterfaceRef.elementInImplDatatype`
[constr_1567]	Existence of `SoftwareCluster.subSoftwareCluster` and `SoftwareCluster.dependsOn/conflictsTo`
[constr_1595]	Consistent assignment of TLV data ids to `CppImplementationDataType` or `CppImplementationDataTypeElement`

▽

△

Number	Heading
[constr_1606]	Processes with mutual ExecutionDependencys
[constr_1615]	Existence of attribute SomeipDataPrototypeTransformationProps.someip-TransformationProps.sessionHandling
[constr_1618]	Ability to shut down
[constr_1619]	Ability to restart
[constr_3396]	Number of Process.stateDependentStartupConfig that refer to the same functionGroupState
[constr_3413]	StateDependentStartupConfig of a Process is mapped to exactly one ResourceGroup
[constr_3421]	Fibex elements applicable for a System of category MACHINE_DESIGN_EXTRACT
[constr_3423]	StateDependentStartupConfig of a Process shall reference a functionGroupState
[constr_3424]	StateDependentStartupConfig shall never reference the functionGroupState Off
[constr_3447]	ApSomeipTransformationProps.sizeOfArrayLengthField that equals 0

Table F.25: Changed Constraints in R19-03

F.5.6 Deleted Constraints in R19-03

Number	Heading
[constr_1499]	Target SwcServiceDependency of DiagnosticServiceSwMapping.mappedSwcServiceDependencyInExecutable
[constr_1504]	Number of Process.modeDependentStartupConfig that refer to the same machineMode
[constr_1573]	CppTemplateArgument.isVariadicTemplate is set to True
[constr_1574]	Number of CppTemplateArguments with isVariadicTemplate set to True
[constr_1575]	Position of CppTemplateArgument with isVariadicTemplate set to True
[constr_1585]	Standardized values of attribute DiagnosticServiceSwMapping.category
[constr_1586]	DiagnosticServiceSwMapping.category set to DATA_ELEMENT
[constr_1620]	Value of schedulingPriority if schedulingPolicy is set to schedulingPolicyFifo or schedulingPolicyRoundRobin
[constr_1621]	Value of schedulingPriority if StartupConfig.schedulingPolicy is set to schedulingPolicyOther
[constr_1663]	Standardized values of attribute DiagnosticServiceDataIdentifierMapping.category
[constr_3380]	End2EndEventProtectionProps shall not reference an event and a notifier at the same time
[constr_3397]	ModeDependentStartupConfig that refers to a functionGroupMode and to a machineMode

▽

△

Number	Heading
[constr_3398]	ModeDependentStartupConfig that refers to function group modes of different function groups
[constr_3494]	Mandatory Machine States
[constr_3531]	Mandatory definition of healthChannelId
[constr_3536]	Mandatory definition of supervisedEntityId
[constr_5001]	Usage of DoIpNetworkConfiguration.eidUseMac
[constr_5002]	Supported values of ServiceInstanceToMachineMapping.category
[constr_5003]	Existence of TlsCryptoCipherSuite.certificate in the *server* role

Table F.26: Deleted Constraints in R19-03

F.6 Constraint and Specification Item History of this document according to AUTOSAR Release R19-11

F.6.1 Added Specification Items in R19-11

Number	Heading
[TPS_MAIN_01281]	Usage of meta-class RecoveryViaApplicationAction
[TPS_MANI_01267]	Semantics of attribute SoftwareClusterDesign.dependsOn
[TPS_MANI_01268]	Semantics of attribute SoftwareClusterDesign.conflictsTo
[TPS_MANI_01269]	Specification of boundaries for resource consumption
[TPS_MANI_01270]	Reference from TransformationPropsToServiceInterfaceElementMapping to TlvDataIdDefinitionSet
[TPS_MANI_01271]	Semantics of Executable.loggingBehavior
[TPS_MANI_01272]	Duplicate entries in logTraceLogMode
[TPS_MANI_01273]	Support for trusted Platform
[TPS_MANI_01274]	System category for a design description that has one single Adaptive Machine in scope
[TPS_MANI_01275]	Semantics of meta-class ServiceInstanceToSwClusterDesignPortPrototypeMapping
[TPS_MANI_01276]	Semantics of CompositionRPortToExecutableRPortMapping and CompositionPPortToExecutablePPortMapping
[TPS_MANI_01277]	Definition of a start-up timeout for a StartupConfig of a Process
[TPS_MANI_01278]	Definition of a termination timeout for a StartupConfig of a Process
[TPS_MANI_01279]	Semantics of Executable.reportingBehavior
[TPS_MANI_01280]	Semantics of meta-class PhmRecoveryActionInterface
[TPS_MANI_01282]	Semantics of reference CompositionPortToExecutablePortMapping.processDesign

▽

△

Number	Heading
[TPS_MANI_01283]	Semantics of meta-class RawDataStreamInterface
[TPS_MANI_01284]	Granularity of meta-class RawDataStreamGrantDesign
[TPS_MANI_01285]	Purpose of meta-class RawDataStreamDeployment
[TPS_MANI_01286]	Semantics of attribute RawDataStreamMethodDeployment.callTimeout
[TPS_MANI_01287]	Semantics of RawDataStreamMapping
[TPS_MANI_01288]	Impact of the SoftwarePackage on the value of function group states on the target platform
[TPS_MANI_01289]	Order of function group states is relevant
[TPS_MANI_01290]	VehiclePackage names affected UCMs
[TPS_MANI_01291]	Identification of an actual UCM in the context of an update campaign
[TPS_MANI_01292]	Definition of fallback-order for UCM master
[TPS_MANI_01294]	Update campaign depends on driver's acceptance
[TPS_MANI_01295]	Semantics of VehicleRolloutStep
[TPS_MANI_01296]	Ordered execution of rollout steps in an update campaign
[TPS_MANI_01297]	Semantics of meta-class UcmStep
[TPS_MANI_01298]	No ordering of VehicleRolloutStep.ucmProcessing
[TPS_MANI_01299]	Aggregation of SoftwarePackageSteps at UcmStep
[TPS_MANI_01300]	Semantics of reference SoftwarePackageStep.transfer.transfer
[TPS_MANI_01301]	Semantics of aggregation SoftwarePackageStep.transfer
[TPS_MANI_01302]	Semantics of reference SoftwarePackageStep.process
[TPS_MANI_01303]	Semantics of reference SoftwarePackageStep.preActivate
[TPS_MANI_01304]	Semantics of reference SoftwarePackageStep.verify
[TPS_MANI_01305]	Semantics of attribute SoftwarePackageStep.activationSwitch
[TPS_MANI_01306]	Simultaneous existence of attributes SoftwarePackageStep.transfer and SoftwarePackageStep.process
[TPS_MANI_01307]	Semantics of meta-class EthernetRawDataStreamGrant
[TPS_MANI_03227]	Usage of ephemeral ports
[TPS_MANI_03228]	Usage of End2EndMethodProtectionProps
[TPS_MANI_03229]	Usage of same End2EndMethodProtectionProps.dataId in case of Multi-Binding
[TPS_MANI_03230]	Sharing timers for ProvidedSomeipServiceInstance
[TPS_MANI_03231]	Sharing timers for RequiredSomeipServiceInstance
[TPS_MANI_03232]	Definition of general IPsec configuration settings
[TPS_MANI_03233]	IPsec mode
[TPS_MANI_03234]	IPsec AH and ESP CipherSuites
[TPS_MANI_03573]	Definition of no minimum deadline supervision
[TPS_MANI_03574]	Definition of no maximum deadline supervision
[TPS_MANI_03575]	Definition of no minimum alive supervision
[TPS_MANI_03576]	Definition of no maximum alive supervision

▽

△

Number	Heading
[TPS_MANI_03577]	`headerId` required for signal-service-translation
[TPS_MANI_03578]	Signal-based `ServiceInterface` binding over `Ethernet`
[TPS_MANI_03579]	Signal-based `ServiceEventDeployment` over `Ethernet`
[TPS_MANI_03580]	Service offer at startup
[TPS_MANI_03581]	Service find at startup
[TPS_MANI_03582]	Service find for required signal
[TPS_MANI_03583]	Service subscribe for required signal
[TPS_MANI_03584]	Definition of transmission triggers for translations with different sources
[TPS_MANI_03585]	Processing order of COM-Stack features
[TPS_MANI_03586]	No transmission trigger for translations with different sources
[TPS_MANI_03587]	Transmission trigger for translations with different sources
[TPS_MANI_03588]	Full translation before transmission triggering
[TPS_MANI_03589]	Reception data filter of COM-Stack
[TPS_MANI_03590]	Transfer properties and transmission modes of COM-Stack
[TPS_MANI_03591]	`SomeipEventDeployment.serializer` equals `signalBased`
[TPS_MANI_03592]	`ISignal` invalidation of COM-Stack
[TPS_MANI_03593]	`handleInvalid` = `dontInvalidate` behavior of COM-Stack
[TPS_MANI_03594]	`handleInvalid` = `replace` behavior of COM-Stack
[TPS_MANI_03595]	Update Bit support for `ISignal`
[TPS_MANI_03596]	Update Bit support for `ISignalIPdu`
[TPS_MANI_03597]	Support for `MultiplexedIPdu`
[TPS_MANI_03598]	Expected check period of E2E-Protected payload
[TPS_MANI_03599]	Expected update period of E2E-Protected payload
[TPS_MANI_03600]	Signal-service-translation of E2E protected payload
[TPS_MANI_03601]	Signal-service-translation of E2E protected payload - timeout handling
[TPS_MANI_03602]	Signal-service-translation of E2E protected payload - error handling
[TPS_MANI_03603]	Service-signal-translation of E2E protected payload
[TPS_MANI_03604]	Service-signal-translation of E2E protected payload - timeout handling
[TPS_MANI_03605]	Service-signal-translation of E2E protected payload - error handling
[TPS_MANI_03606]	Service offer for provided signal
[TPS_MANI_03607]	Handling of safe signal-service-translation in one `Executable`
[TPS_MANI_03608]	Support for safe signal-service-translation and service-signal-translation
[TPS_MANI_03609]	Support for safe signal-service-translation with same or different E2E profiles
[TPS_MANI_03610]	1:n mapping for E2E protected data
[TPS_MANI_03611]	E2E protected target out of E2E protected sources
[TPS_MANI_03612]	Sufficient ASIL level of translation software
[TPS_MANI_03614]	No translation of not OK E2E protected data out of several sources
[TPS_MANI_03615]	`SomeipEventDeployment.serializer` equals `someip`

▽

△

Number	Heading
[TPS_MANI_03616]	Semantic versioning of `ServiceInterface.majorVersion` and `ServiceInterface.minorVersion`
[TPS_MANI_03617]	Version mapping between `ServiceInterface` and `ServiceInterfaceDeployment`
[TPS_MANI_03618]	Usage of `RequiredSomeipServiceInstance.blacklistedVersion`
[TPS_MANI_03619]	SOME/IP Service search for `requiredMinorVersion`
[TPS_MANI_03620]	Service discovery control
[TPS_MANI_03621]	Data filter inside the signal-service-translation

Table F.27: Added Specification Items in R19-11

F.6.2 Changed Specification Items in R19-11

Number	Heading
[TPS_MANI_01032]	Usage of ServiceInterfaceMapping
[TPS_MANI_01057]	Semantics of RPortPrototypeProps.searchIntention
[TPS_MANI_01164]	Semantics of SoftwareCluster.dependsOn
[TPS_MANI_01196]	Semantics of PersistencyDeployment.minimumSustainedSize
[TPS_MANI_01197]	Semantics of PersistencyDeployment.maximumAllowedSize
[TPS_MANI_01214]	Semantics of SoftwareCluster.conflictsTo
[TPS_MANI_01215]	Semantics of meta-class SoftwareActivationDependencyFormula
[TPS_MANI_01216]	Semantics of meta-class SoftwareActivationDependencyFormula-Part
[TPS_MANI_01217]	Semantics of metaclass SoftwareActivationDependencyCompare-Condition
[TPS_MANI_01249]	Semantics of DiagnosticConditionInterface
[TPS_MANI_01255]	Semantics of DoIP DiagnosticPortInterfaces
[TPS_MANI_03004]	IPv4 Multicast event destination address
[TPS_MANI_03010]	Udp Transport Protocol Configuration in case of IP-Multicast
[TPS_MANI_03061]	IPv6 Multicast event destination address
[TPS_MANI_03114]	Usage of AssemblySwConnectors in the System Design model
[TPS_MANI_03130]	Standardized E2EProfileConfiguration.profileName values
[TPS_MANI_03160]	Log and Trace configuration options in the Execution Manifest
[TPS_MANI_03161]	Log and Trace configuration options in the Service Instance Manifest
[TPS_MANI_03167]	Network configuration for Nm
[TPS_MANI_03205]	IPsec policy
[TPS_MANI_03206]	IPsec AH and ESP protocol configuration
[TPS_MANI_03207]	IPsec Internet Key Exchange protocol configuration
[TPS_MANI_03216]	Existence of TlsCryptoCipherSuite.certificate and TlsCryptoCipherSuite.pskIdentity in the *client* role

Table F.28: Changed Specification Items in R19-11

F.6.3 Deleted Specification Items in R19-11

Number	Heading
[TPS_MANI_01051]	Mapping of DiagnosticStorageCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01052]	Semantics of RPortPrototypeProps.portInstantiationBehavior
[TPS_MANI_01162]	Semantics of SoftwareClusterDesign.dependsOn
[TPS_MANI_03120]	Signal-based ServiceInterface binding

▽

△

Number	Heading
[TPS_MANI_03146]	Configuration of timeouts for a selected machine state or function group state
[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_03550]	Usage of RPortPrototype for the interaction with Platform Health Management

Table F.29: Deleted Specification Items in R19-11

F.6.4 Added Constraints in R19-11

Number	Heading
[constr_1705]	Target of reference SoftwareActivationDependencyCompareCondition.softwareActivationDependency
[constr_1707]	Eligible subclasses of HeapUsage in the context of StateDependentStartupConfig.resourceConsumption
[constr_1708]	Combination of CppImplementationDataTypeElement.isOptional and CppImplementationDataTypeElementQualifier.inplace
[constr_1709]	Applicability of attribute PersistencyRedundancyEnum.redundantPerKey
[constr_1710]	Consistency of values of attributes PersistencyInterface.redundancy and PersistencyRedundancyHandling.scope
[constr_1723]	ProvidedSomeipServiceInstance shall be unique in respect of serviceInstanceId, serviceInterfaceId and majorVersion
[constr_1727]	Qualified combinations of PortPrototypes and PhmSupervisedEntityInterface on application software level
[constr_1728]	Qualified combinations of PortPrototypes and PhmHealthChannelInterface on application software level
[constr_1729]	Qualified combinations of PortPrototypes and PhmRecoveryActionInterface on application software level
[constr_1730]	Restriction regarding the modeling of the PhmRecoveryActionInterface.recovery
[constr_1731]	Value of UcmDescription.identifier in the scope of a VehiclePackage
[constr_1732]	Existence of attribute activationSwitch set to True in the context of the enclosing UcmStep
[constr_1733]	Simultaneous existence of SoftwarePackageStep.preActivate and SoftwarePackageStep.verify
[constr_1734]	Restriction for attribute SoftwarePackageStep.activationSwitch
[constr_1736]	Multiplicity of reference LogicalSupervision.initialCheckpoint
[constr_1737]	Multiplicity of reference LogicalSupervision.finalCheckpoint
[constr_1738]	Multiplicity of reference GlobalSupervision.localSupervision
[constr_1739]	Multiplicity of aggregation LocalSupervision.transition

▽

△

Number	Heading
[constr_1740]	Multiplicity of reference LogicalSupervision.transition
[constr_1742]	Multiplicity of reference SupervisionCheckpoint.phmCheckpoint
[constr_3550]	Existence of ServiceInstanceToSignalMapping for an event with signal-Based serialization
[constr_3551]	Full mapping of target ISignalGroup
[constr_3552]	Full mapping of target event
[constr_3553]	Existence of ServiceInstanceToSignalMapping for an field with signal-Based serialization
[constr_3554]	E2E protection configuration check
[constr_3555]	No support for useAsCryptographicIPdu is true
[constr_3556]	Unique transport layer mapping
[constr_3557]	Mandatory majorVersion at SomeipServiceInterfaceDeployment.servi-ceInterfaceVersion
[constr_3558]	RequiredSomeipServiceInstance.blacklistedVersion is restricted to the usage of minorVersion
[constr_3561]	minimumMinorVersion and RequiredSomeipServiceInstance.required-MinorVersion value
[constr_3562]	Existence of NonqueuedReceiverComSpec.filter
[constr_5052]	SOME/IP ServiceInstances of the same serviceInterface on one Machine
[constr_5056]	Restriction of CompositionSwComponentType.connector usage in AP
[constr_5057]	PassThroughSwConnector and ServiceInterfaceMapping
[constr_5102]	Usage of remote port ranges in IPSecRule is not allowed
[constr_5103]	Usage of local port ranges in IPSecRule is not allowed

Table F.30: Added Constraints in R19-11

F.6.5 Changed Constraints in R19-11

Number	Heading
[constr_1561]	Existence of SoftwareClusterDesign.subSoftwareCluster and Soft-wareClusterDesign.dependsOn.dependentSoftwareClusterDesign
[constr_1567]	Existence of SoftwareCluster.subSoftwareCluster and SoftwareClus-ter.dependsOn/conflictsTo
[constr_1570]	Restriction for UserDefinedServiceInterfaceDeployment of category SERVICE_INTERFACE_DEPLOYMENT_IPC
[constr_1579]	SwDataDefProps applicable to CppImplementationDataTypes exclusive to the *AUTOSAR adaptive platform*
[constr_1630]	No definition of length field sizes on DataPrototype level in case of TLV usage
[constr_3375]	method with attribute fireAndForget set to true shall not reference an ApAppli-cationError

▽

△

Number	Heading
[constr_3419]	Allowed usage of `UdpNmNetworkConfiguration` attributes
[constr_3426]	The `logTraceFilePath` is mandatory in case that `logTraceLogMode` is set to `file`
[constr_3427]	The `logTraceFilePath` is only relevant if `logTraceLogMode` is set to `file`
[constr_3493]	Applicable attributes for standardized E2E Profiles
[constr_5048]	Existence of `TlsCryptoCipherSuite.certificate` and `TlsCryptoCipherSuite.pskIdentity` in the *server* role

Table F.31: Changed Constraints in R19-11

F.6.6 Deleted Constraints in R19-11

Number	Heading
[constr_1503]	Target `SwcServiceDependency` of `DiagnosticStorageConditionPortMapping.swcServiceDependencyInExecutable`
[constr_3387]	Compatibility of `PortPrototype`s of different `ServiceInterface`s
[constr_3388]	Compatibility of `event`s
[constr_3389]	Compatibility of `method`s
[constr_3390]	Compatibility of `field`s
[constr_3411]	`eventMulticastUdpPort`, `ipv4MulticastIpAddress` and `ipv6MulticastIpAddress` not relevant for `RequiredSomeipServiceInstance`s
[constr_3420]	System `category` for a design description that has one single Adaptive `Machine` in scope

Table F.32: Deleted Constraints in R19-11

F.7 Constraint and Specification Item History of this document according to AUTOSAR Release R20-11

F.7.1 Added Specification Items in R20-11

Number	Heading
[TPS_MANI_01308]	`Process` is not designed for re-usability
[TPS_MANI_01309]	Semantics of attribute `CppImplementationDataType.headerFile`
[TPS_MANI_01310]	Semantics of `SoftwareClusterDesign.dependsOn`
[TPS_MANI_01311]	Handling of manufacturer checks

▽

△

Number	Heading
[TPS_MANI_01312]	Handling of supplier checks
[TPS_MANI_01313]	Definition of updateStrategy on element level
[TPS_MANI_01314]	Further qualification of properties of PortPrototypes typed by PersistencyKeyValueStorageInterfaces
[TPS_MANI_01315]	PersistencyKeyValuePair.initValue overrides PersistencyDataRequiredComSpec.initValue
[TPS_MANI_01316]	Existence of ServiceInstanceToPortPrototypeMapping.processDesign
[TPS_MANI_01317]	Existence of ServiceInstanceToPortPrototypeMapping.process
[TPS_MANI_01319]	Modeling of redundancy in the context of PersistencyInterface
[TPS_MANI_01320]	Definition of redundancy on interface level may be overruled in deployment
[TPS_MANI_01321]	Semantics of meta-class PersistencyDeploymentElement
[TPS_MANI_01322]	Semantics of meta-class PersistencyPortPrototypeToDeploymentMapping
[TPS_MANI_01323]	Matching pairs of PersistencyDataElement and PersistencyKeyValuePair
[TPS_MANI_01324]	Semantics of E2E attributes in ClientComSpec
[TPS_MANI_01325]	Semantics of E2E attributes in ServerComSpec
[TPS_MANI_01326]	Generic Mapping to a DiagnosticServiceInstance on the AUTOSAR Adaptive Platform
[TPS_MANI_01327]	Value of EndToEndTransformationComSpecProps.disableEndToEndCheck vs. value of EndToEndTransformationComSpecProps.disableEndToEndStateMachine
[TPS_MANI_01328]	Standardized values for attribute StartupConfig.schedulingPolicy
[TPS_MANI_01329]	Reference to model elements in different SoftwareClusters
[TPS_MANI_01330]	Definition of machine function group
[TPS_MANI_01331]	Standardized values of attribute SoftwareCluster.category
[TPS_MANI_01332]	Semantics of DiagnosticEcuResetInterface
[TPS_MANI_01333]	Attribute NotAvailableValueSpecification.defaultPattern is not applicable
[TPS_MANI_01334]	Semantics of StartupConfig.terminationBehavior
[TPS_MANI_01335]	Semantics of SoftwareClusterDependencyFormula.category
[TPS_MANI_01336]	Two use cases for using the DiagnosticEventPortMapping
[TPS_MANI_01337]	Standardized values for attribute Process.functionClusterAffiliation
[TPS_MANI_01338]	Semantics of SecurityEventReportToSecurityEventDefinitionMapping
[TPS_MANI_01339]	Existence of the SecurityEventReportToSecurityEventDefinitionMapping is motivated by the AUTOSAR methodology
[TPS_MANI_01340]	Semantics of SecurityEventReportInterface
[TPS_MANI_01341]	Security events that are actually reported by a local IdsM
[TPS_MANI_01342]	Semantics of SecurityEventMapping

▽

△

Number	Heading
[TPS_MANI_03235]	Usage of ApSomeipTransformationProps.sessionHandling
[TPS_MANI_03236]	Mapping of ProvidedSomeipServiceInstance to different PPortPrototypes
[TPS_MANI_03237]	Transport Protocol attributes defined for a RequiredSomeipServiceInstance
[TPS_MANI_03238]	Definition of ComMethodGrantDesign.remoteSubject
[TPS_MANI_03239]	Definition of ComEventGrantDesign.remoteSubject
[TPS_MANI_03240]	Modeling of a remote peer in case of TLS-based secure channel
[TPS_MANI_03241]	Modeling of relevant TlsSecureComProps for TlsIamRemoteSubject
[TPS_MANI_03242]	Modeling of a remote peer in case of IPsec-based secure channel
[TPS_MANI_03244]	Modeling of a remote peer in case of a general IP communication
[TPS_MANI_03245]	Definition of ComMethodGrant.remoteSubjects on server side
[TPS_MANI_03246]	Definition of ComMethodGrant.remoteSubjects on client side
[TPS_MANI_03247]	Definition of ComEventGrant.remoteSubjects on provider side
[TPS_MANI_03248]	Definition of ComEventGrant.remoteSubjects on receiver side
[TPS_MANI_03249]	Definition of ComFieldGrant.remoteSubjects on provider side
[TPS_MANI_03250]	Definition of ComFieldGrant.remoteSubjects on client side
[TPS_MANI_03251]	Definition of ComFieldGrantDesign.remoteSubject
[TPS_MANI_03252]	Usage of same End2EndMethodProtectionProps.sourceId in case of Multi-Binding
[TPS_MANI_03253]	Interaction with crypto software
[TPS_MANI_03254]	Modeling of application that uses and modifies a Crypto Key
[TPS_MANI_03255]	Modeling of Key Manager application that manages a Crypto Key that is used by Stack Services
[TPS_MANI_03256]	Modeling of application that accesses a Crypto Certificate
[TPS_MANI_03257]	Modeling of application that accesses a Crypto Provider
[TPS_MANI_03258]	Modeling of application designed as trust-master
[TPS_MANI_03259]	Linking of Crypto Certificate to a Crypto Key Slot
[TPS_MANI_03260]	Semantics of meta-class CryptoModuleInstantiation
[TPS_MANI_03261]	Support of CryptoProviders
[TPS_MANI_03262]	Semantics of CryptoProviderToPortPrototypeMapping
[TPS_MANI_03263]	Assignment of CryptoKeySlots to CryptoProviders
[TPS_MANI_03264]	Semantics of CryptoKeySlotToPortPrototypeMapping
[TPS_MANI_03265]	Support of CryptoCertificates
[TPS_MANI_03266]	Semantics of CryptoCertificateToCryptoKeySlotMapping
[TPS_MANI_03267]	Semantics of CryptoCertificateToPortPrototypeMapping
[TPS_MANI_03268]	Semantics of FunctionalClusterInteractsWithFunctionalClusterMapping
[TPS_MANI_03269]	Semantics of ComCertificateToCryptoCertificateMapping
[TPS_MANI_03270]	Semantics of ComKeyToCryptoKeySlotMapping

▽

△

Number	Heading
[TPS_MANI_03271]	Semantics of ComSecOcToCryptoKeySlotMapping
[TPS_MANI_03272]	Semantics of PersistencyDeploymentToCryptoKeySlotMapping
[TPS_MANI_03273]	Semantics of PersistencyDeploymentElementToCryptoKeySlotMapping
[TPS_MANI_03274]	Configuration of log and trace message source
[TPS_MANI_03275]	Configuration of log and trace message source on design level
[TPS_MANI_03276]	Semantics of PersistencyDeploymentToDltLogChannelMapping
[TPS_MANI_03622]	DDS Transport Protocols are up to the stack implementer
[TPS_MANI_03623]	Usage of checkpointId in application code
[TPS_MANI_03624]	Usage of statusId in application code
[TPS_MANI_03625]	Consistency of HealthChannelExternalReportedStatus.statusId and PhmHealthChannelStatus.statusId
[TPS_MANI_03626]	Consistency of SupervisionCheckpoint.phmCheckpoint and PhmCheckpoint.checkpointId
[TPS_MANI_03627]	No signal-service-translation for methods
[TPS_MANI_03628]	Standardized values of ServiceInterface.category
[TPS_MANI_03629]	Relation of ServiceInstanceToSignalMapping and CommunicationConnector
[TPS_MANI_03630]	Semantics of triggersRecoveryNotification
[TPS_MANI_03631]	Semantics of meta-class PhmHealthChannelRecoveryNotificationInterface
[TPS_MANI_03632]	Semantics of TimeBaseProviderToPersistencyMapping

Table F.33: Added Specification Items in R20-11

F.7.2 Changed Specification Items in R20-11

Number	Heading
[TPS_MANI_01061]	Requirements on scheduling
[TPS_MANI_01065]	Purpose of PersistencyKeyValueStorageInterface
[TPS_MANI_01067]	Purpose of PersistencyFileStorageInterface
[TPS_MANI_01068]	Semantics of PersistencyFileStorageInterface.maxNumberOfFiles
[TPS_MANI_01073]	Semantics of PortPrototype typed by PersistencyKeyValueStorageInterface
[TPS_MANI_01078]	Semantics of PersistencyPortPrototypeToKeyValueStorageMapping
[TPS_MANI_01079]	Semantics of PersistencyKeyValueStorage
[TPS_MANI_01080]	Semantics of meta-class PersistencyPortPrototypeToFileStorageMapping

▽

△

Number	Heading
[TPS_MANI_01081]	Semantics of PortPrototype typed by PersistencyFileStorageInterface
[TPS_MANI_01106]	Specification of intentions for the receiver of events or field notifiers
[TPS_MANI_01107]	Specification of intentions for the sender of events or field notifiers
[TPS_MANI_01108]	Specification of intentions for the caller of a methods or field setter/getter
[TPS_MANI_01135]	Semantics of PersistencyKeyValueStorageInterface.dataTypeForSerialization
[TPS_MANI_01138]	Semantics of PersistencyKeyValueStorageInterface.dataElement
[TPS_MANI_01139]	Semantics of PersistencyInterface.updateStrategy
[TPS_MANI_01140]	Semantics of PersistencyInterfaceElement.updateStrategy
[TPS_MANI_01142]	Semantics of PersistencyFileElement
[TPS_MANI_01144]	Semantics of PersistencyKeyValuePair
[TPS_MANI_01147]	Semantics of PersistencyDeployment.updateStrategy
[TPS_MANI_01148]	Semantics of PersistencyDeploymentElement.updateStrategy
[TPS_MANI_01149]	Semantics of PersistencyFileStorage.file
[TPS_MANI_01150]	Semantics of PersistencyFileStorage
[TPS_MANI_01155]	PersistencyDeployment.updateStrategy overrides PersistencyInterface.updateStrategy
[TPS_MANI_01156]	PersistencyDeploymentElement.updateStrategy overrides PersistencyDeployment.updateStrategy
[TPS_MANI_01157]	Semantics of updateStrategy on collection level
[TPS_MANI_01159]	Semantics of updateStrategy on element level
[TPS_MANI_01160]	Definition of initial value for PersistencyDataElement
[TPS_MANI_01164]	Semantics of SoftwareCluster.dependsOn
[TPS_MANI_01176]	Standardized value for attribute CppImplementationDataType.typeEmitter
[TPS_MANI_01177]	Semantics of attribute CppImplementationDataType.typeEmitter
[TPS_MANI_01179]	Semantics of PersistencyFileElement.contentUri/PersistencyFile.contentUri vs. PersistencyFileStorage.uri and PersistencyFileElement.fileName/PersistencyFile.fileName
[TPS_MANI_01180]	Collection of data types that requires serialization support
[TPS_MANI_01182]	Value of PersistencyDeploymentElement.updateStrategy overrides PersistencyInterfaceElement.updateStrategy
[TPS_MANI_01187]	Matching pairs of PersistencyFileElement and PersistencyFile
[TPS_MANI_01194]	Semantics of PersistencyInterface.minimumSustainedSize
[TPS_MANI_01196]	Semantics of PersistencyDeployment.minimumSustainedSize
[TPS_MANI_01197]	Semantics of PersistencyDeployment.maximumAllowedSize
[TPS_MANI_01204]	Specification of redundancy of persistent data
[TPS_MANI_01206]	Modeling of redundancy in the context of PersistencyDeployment
[TPS_MANI_01207]	Standardized values of attribute PersistencyRedundancyChecksum.algorithmFamily

▽

△

Number	Heading
[TPS_MANI_01213]	Semantics of meta-class StrongRevisionLabelString
[TPS_MANI_01214]	Semantics of SoftwareCluster.conflictsTo
[TPS_MANI_01215]	Semantics of meta-class SoftwareClusterDependencyFormula
[TPS_MANI_01216]	Semantics of meta-class SoftwareClusterDependencyFormulaPart
[TPS_MANI_01217]	Semantics of meta-class SoftwareClusterDependencyCompareCondition
[TPS_MANI_01263]	Mapping of DiagnosticDataIdentifier or DiagnosticDataElement to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01272]	Duplicate entries in logTraceLogMode
[TPS_MANI_01280]	Semantics of meta-class PhmSupervisionRecoveryNotificationInterface
[TPS_MANI_01284]	Granularity of meta-class RawDataStreamGrantDesign
[TPS_MANI_01285]	Purpose of meta-class RawDataStreamDeployment
[TPS_MANI_01300]	Semantics of reference SoftwarePackageStep.transfer.transfer
[TPS_MANI_03059]	RequiredSomeipServiceInstance.requiredServiceInstanceId
[TPS_MANI_03111]	Mapping between method and operationlocated in a ClientServerInterface
[TPS_MANI_03113]	Mapping between a field and elements of Classic Platform PortInterfaces
[TPS_MANI_03130]	Standardized E2EProfileConfiguration.profileName values
[TPS_MANI_03145]	Description of a function group
[TPS_MANI_03160]	Further configuration options in DltLogChannel
[TPS_MANI_03163]	Network configuration for Log and Trace messages
[TPS_MANI_03165]	Network Interface configuration for DoIP
[TPS_MANI_03194]	Function Group State
[TPS_MANI_03195]	Off state in Function Group
[TPS_MANI_03207]	IPsec Internet Key Exchange protocol configuration
[TPS_MANI_03209]	The meaning of MachineDesign.accessControl
[TPS_MANI_03516]	Status for HealthChannelSupervision
[TPS_MANI_03517]	Evaluation of HealthChannelExternalStatus
[TPS_MANI_03535]	Definition of Time Synchronization interaction
[TPS_MANI_03536]	Time Synchronization interaction in a provider role
[TPS_MANI_03537]	Time Synchronization interaction in a consumer role
[TPS_MANI_03539]	Definition of Time-Base Resources
[TPS_MANI_03541]	Definition of SynchronizedTimeBaseConsumer
[TPS_MANI_03542]	Definition of SynchronizedTimeBaseProvider
[TPS_MANI_03543]	Definition of time sync correction attributes
[TPS_MANI_03546]	Definition of reported health status RPortPrototype
[TPS_MANI_03548]	Definition of TimeSyncPortPrototypeToTimeBaseMapping
[TPS_MANI_03549]	Usage of PortPrototype for the interaction with Time Synchronization

▽

△

Number	Heading
[TPS_MANI_03551]	Definition of Time Base kind
[TPS_MANI_03556]	DDS-RPC Service Binding
[TPS_MANI_03557]	DDS `ClientServerOperation` Binding
[TPS_MANI_03558]	DDS `Field` Binding
[TPS_MANI_03598]	Expected check period of E2E-Protected payload
[TPS_MANI_03599]	Expected update period of E2E-Protected payload
[TPS_MANI_03612]	Sufficient ASIL level of translation software
[TPS_MANI_03617]	Version mapping between `ServiceInterface` and `ServiceInterfaceDeployment`

Table F.34: Changed Specification Items in R20-11

F.7.3 Deleted Specification Items in R20-11

Number	Heading
[TPS_MAIN_01281]	Usage of meta-class `RecoveryViaApplicationAction`
[TPS_MANI_01015]	Semantics of meta-class `StartupOption`
[TPS_MANI_01059]	Different values of `optionKind` within a `StartupConfig.startupOption`
[TPS_MANI_01069]	Further qualification of properties of `PortPrototype`s typed by `PersistencyKeyValueDatabaseInterface`s
[TPS_MANI_01141]	Semantics of `PersistencyFileProxyInterface.updateStrategy`
[TPS_MANI_01143]	Semantics of `PersistencyFileProxy.updateStrategy`
[TPS_MANI_01151]	Semantics of `PersistencyFileArray.updateStrategy`
[TPS_MANI_01152]	Semantics of `PersistencyFile.updateStrategy`
[TPS_MANI_01154]	`PersistencyFileArray.updateStrategy` overrides `PersistencyFileProxyInterface.updateStrategy`
[TPS_MANI_01158]	`PersistencyFile.updateStrategy` overrides `PersistencyFileArray.updateStrategy`
[TPS_MANI_01163]	Impact of values of `category` on the semantics of `SoftwareCluster`
[TPS_MANI_01183]	`PersistencyFile.updateStrategy` overrides `PersistencyFileProxy.updateStrategy`
[TPS_MANI_01195]	Semantics of `PersistencyFileProxyInterface.minimumSustainedSize`
[TPS_MANI_01224]	Actions taken after installation of a `SoftwarePackage`
[TPS_MANI_01264]	Mapping of `DiagnosticServiceInstance` to `PortPrototype`(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01266]	Mapping of `DiagnosticServiceInstance` for upload/download to `PortPrototype`(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01267]	Semantics of attribute `SoftwareClusterDesign.dependsOn`

▽

△

Number	Heading
[TPS_MANI_01268]	Semantics of attribute SoftwareClusterDesign.conflictsTo
[TPS_MANI_01283]	Semantics of meta-class RawDataStreamInterface
[TPS_MANI_01286]	Semantics of attribute RawDataStreamMethodDeployment.callTimeout
[TPS_MANI_03065]	Hardware resources of the machine
[TPS_MANI_03161]	Log and Trace configuration options in the Service Instance Manifest
[TPS_MANI_03518]	PhmLogicalExpression definition
[TPS_MANI_03519]	PhmRule definition
[TPS_MANI_03520]	Execution of PhmActionList with actionListExecution=triggeredOnEvaluation
[TPS_MANI_03521]	Execution of PhmActionList with actionListExecution=triggeredOnChange
[TPS_MANI_03522]	Definition of actions for application software
[TPS_MANI_03523]	Definition of actions for Platform Instance
[TPS_MANI_03524]	Definition of actions for Watchdog
[TPS_MANI_03538]	Time Synchronization interaction with a local Time Base
[TPS_MANI_03540]	Definition of PureLocalTimeBase
[TPS_MANI_03559]	Definition of DdsProvidedServiceInstance.methodQosProps
[TPS_MANI_03560]	qosProfile of DdsProvidedServiceInstance.methodQosProps is optional
[TPS_MANI_03563]	Definition of DdsProvidedServiceInstance.fieldGetSetQosProps
[TPS_MANI_03564]	qosProfile of DdsProvidedServiceInstance.fieldGetSetQosProps is optional
[TPS_MANI_03565]	Definition of DdsRequiredServiceInstance.methodQosProps
[TPS_MANI_03566]	qosProfile of DdsRequiredServiceInstance.methodQosProps is optional
[TPS_MANI_03569]	Definition of DdsRequiredServiceInstance.fieldGetSetQosProps
[TPS_MANI_03570]	qosProfile of DdsRequiredServiceInstance.fieldGetSetQosProps is optional
[TPS_MANI_03571]	transportPlugin for DdsProvidedServiceInstance
[TPS_MANI_03572]	transportPlugin for DdsRequiredServiceInstance

Table F.35: Deleted Specification Items in R20-11

F.7.4 Added Constraints in R20-11

Number	Heading
[constr_1743]	CppImplementationDataType.headerFile vs. CppImplementationDataType.typeEmitter
[constr_1744]	Definition of process state In the context of the ExecutionDependency
[constr_1746]	Mutual exclusive existence of PersistencyInterface.redundancy and PersistencyInterface.redundancyHandling
[constr_1747]	Completeness of the SoftwareCluster.version
[constr_1748]	Existence of references TlvDataIdDefinition.tlvArgument, TlvDataIdDefinition.tlvRecordElement, and TlvDataIdDefinition.tlvSubElement
[constr_1751]	Value of PersistencyRedundancyMOutOfN.n and PersistencyRedundancyMOutOfN.m
[constr_1764]	Counterpart of PhmCheckpoint
[constr_1765]	Diagnostic Services eligible for DiagnosticServiceGenericMapping
[constr_1769]	Existence of ProcessArgument.argument
[constr_1770]	Value of ProvidedSomeipServiceInstance.serviceInstanceId
[constr_1784]	Restriction for the reference to UploadableExclusivePackageElement
[constr_1785]	Restriction regarding the reference into another SoftwareCluster
[constr_1786]	Restriction to use functionGroup in terms of SoftwareCluster
[constr_1787]	Restricted use of function groups in the context of a SoftwareCluster
[constr_1788]	Restriction to SoftwareCluster of category PLATFORM_CORE
[constr_1789]	Scope of machine function group
[constr_1809]	Global supervision restricted to one function group
[constr_3563]	Mandatory topic name values
[constr_3564]	Consistency between DDS Service Interface Deployment and Provided DDS Service Instance
[constr_3565]	Consistency between DDS Service Interface Deployment and Required DDS Service Instance
[constr_3568]	No support for cross PlatformHealthManagementContribution references
[constr_3569]	Applicability of attribute invalidValue on CppImplementationDataType of category TYPE_REFERENCE
[constr_3612]	Multiplicity of references recoveryNotification, recoveryAction, and process at RecoveryNotificationToPPortPrototypeMapping
[constr_3613]	Reference to a PhmSupervisionRecoveryNotificationInterface in the context of a HealthChannelSupervision
[constr_3614]	Reference to a PhmHealthChannelRecoveryNotificationInterface in the context of a HealthChannelExternalStatus
[constr_3619]	Mandatory references of TimeBaseProviderToPersistencyMapping
[constr_5115]	Search for a specific SOME/IP ServiceInstance and for all SOME/IP ServiceInstances over the same RPortPrototype

▽

△

Number	Heading
[constr_5155]	SomeipServiceInstanceToMachineMapping only supports a single Address Family
[constr_5156]	SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_5161]	RequiredSomeipServiceInstance that is mapped by a SomeipServiceInstanceToMachineMapping without a configured tcpPort and udpPort
[constr_5227]	Mandatory elements of UdpNmCluster
[constr_5228]	Partial Networking timing constraint
[constr_5230]	Existence of attribute E2EProfileCompatibilityProps.transitToInvalidExtended is mandatory for each E2EProfileConfiguration
[constr_5238]	CryptoKeySlotAllowedModification.restrictUpdate and the relationship to maxNumberOfAllowedUpdates
[constr_5239]	Predefined values for CryptoKeySlotContentAllowedUsage.allowedKeyslotUsage
[constr_5240]	Restriction applicable for CryptoProviderToPortPrototypeMapping.portPrototype
[constr_5241]	Restriction applicable for CryptoKeySlotToPortPrototypeMapping.portPrototype
[constr_5242]	Restriction applicable for CryptoCertificateToPortPrototypeMapping.portPrototype
[constr_5243]	Restriction of LogAndTraceInstantiation.dltEcuId attribute value
[constr_10002]	Only one mapping per PortPrototype
[constr_10003]	Restriction for the existence of DiagnosticServiceDataIdentifierPortMapping.diagnosticDataIdentifier vs. DiagnosticServiceDataIdentifierPortMapping.diagnosticDataElement
[constr_10004]	Consistency of DiagnosticServiceGenericMapping for PortPrototype typed by DiagnosticDataIdentifierGenericInterface
[constr_10007]	Existence of ProcessExecutionError.executionError
[constr_10008]	Value of ProcessExecutionError.executionError
[constr_10010]	Usage of attribute category in a SoftwareClusterDependencyFormula
[constr_10011]	Definition of sub-software-cluster
[constr_10021]	Existence of IdsmModuleInstantiation
[constr_10022]	Restriction for SecurityEventMapping.process.securityEvent.id w.r.t SecurityEventMapping.id
[constr_10023]	Mandatory content of any functionGroup

Table F.36: Added Constraints in R20-11

F.7.5 Changed Constraints in R20-11

Number	Heading
[constr_1490]	Allowed value for Executable.category if ProcessToMachineMapping references a NonOsModuleInstantiation
[constr_1500]	Target SwcServiceDependency of DiagnosticEventPortMapping.swcServiceDependencyInExecutable
[constr_1507]	PortInterfaceToDataTypeMapping is only applicable to ServiceInterface or PersistencyKeyValueStorageInterface
[constr_1555]	Restriction applicable for PersistencyPortPrototypeToKeyValueStorageMapping.portPrototype
[constr_1556]	Restriction applicable for PersistencyPortPrototypeToFileStorageMapping.portPrototype
[constr_1564]	Existence of SoftwareCluster.diagnosticAddress
[constr_1566]	Usage of SoftwareCluster.containedARElement
[constr_1568]	Existence of SoftwareCluster.diagnosticExtract
[constr_1581]	Value of fileElement.fileName
[constr_1589]	Value of file.fileName
[constr_1613]	File name of matching pairs of PersistencyFileElement and PersistencyFile
[constr_1659]	Restriction for the usage of CppImplementationDataTypeElementQualifier.inplace
[constr_1666]	References from PersistencyPortPrototypeToKeyValueStorageMapping to PersistencyKeyValueStorage
[constr_1667]	References from PersistencyPortPrototypeToFileStorageMapping to PersistencyFileStorage
[constr_1668]	Allowed combinations of PersistencyRedundancyChecksum.length and algorithmFamily
[constr_1673]	Existence of attributes hasGetter, hasSetter, and hasNotifier
[constr_1710]	Consistency of values of attributes PersistencyInterface.redundancy and PersistencyRedundancyHandling.scope
[constr_1729]	Qualified combinations of PortPrototypes and PhmSupervisionRecoveryNotificationInterface / PhmHealthChannelRecoveryNotificationInterface on State Management software level
[constr_3305]	Value of attribute SomeipEventDeployment.eventId shall be unique
[constr_3306]	Value of attribute methodId shall be unique per SomeipServiceInterfaceDeployment
[constr_3356]	Restriction in usage of ApSomeipTransformationProps.alignment
[constr_3414]	Allowed usage of PlatformModuleEthernetEndpointConfiguration attributes
[constr_3421]	Fibex elements applicable for a System of category MACHINE_DESIGN_EXTRACT
[constr_3426]	The logTraceFilePath is mandatory in case that logTraceLogMode is set to file
[constr_3427]	The logTraceFilePath is only relevant if logTraceLogMode is set to file

▽

△

Number	Heading
[constr_3493]	Applicable attributes for standardized E2E Profiles
[constr_3552]	Full mapping of target `event`
[constr_3554]	E2E protection configuration check
[constr_5052]	`ProvidedSomeipServiceInstance`s of the same serviceInterface on one Machine

Table F.37: Changed Constraints in R20-11

F.7.6 Deleted Constraints in R20-11

Number	Heading
[constr_1481]	Usage of `DataPrototypeInServiceInterfaceRef` in the *AUTOSAR adaptive platform*
[constr_1497]	Attribute `optionKind` set to `commandLineSimpleForm`
[constr_1498]	Attribute `optionKind` set to `commandLineShortForm` or `commandLineLongForm`
[constr_1524]	Standardized values of `PersistencyFileProxyInterface.category`
[constr_1534]	Existence of `DiagnosticSoftwareClusterProps`
[constr_1542]	No nested definition of `SoftwareCluster`
[constr_1561]	Existence of `SoftwareClusterDesign`.`subSoftwareCluster` and `SoftwareClusterDesign`.`dependsOn`.`dependentSoftwareClusterDesign`
[constr_1563]	Standardized values of `SoftwareClusterDesign.category` and `SoftwareCluster.category`
[constr_1565]	Existence of `SoftwareCluster`.`subSoftwareCluster`
[constr_1567]	Existence of `SoftwareCluster`.`subSoftwareCluster` and `SoftwareCluster`.`dependsOn`/`conflictsTo`
[constr_1615]	Existence of attribute `SomeipDataPrototypeTransformationProps`.`someipTransformationProps`.`sessionHandling`
[constr_1687]	Definition of machine state
[constr_1703]	Target `SwcServiceDependency` of `DiagnosticGenericUdsPortMapping`.`swcServiceDependencyInExecutable`
[constr_1704]	Target `SwcServiceDependency` of `DiagnosticUploadDownloadPortMapping`.`swcServiceDependencyInExecutable`
[constr_1705]	Target of reference `SoftwareActivationDependencyCompareCondition`.`softwareActivationDependency`
[constr_1709]	Applicability of attribute `PersistencyRedundancyEnum`.`redundantPerKey`
[constr_1730]	Restriction regarding the modeling of the `PhmRecoveryActionInterface`.`recovery`
[constr_3296]	Transport Protocol attributes defined for a `RequiredSomeipServiceInstance`
[constr_3297]	`SomeipServiceInstanceToMachineMapping` only supports a single Address Family

▽

△

Number	Heading
[constr_3307]	SomeipEventDeployment.transportProtocol setting to udp and the impact on ProvidedSomeipServiceInstances
[constr_3412]	OsModuleInstantiation shall have at least one ResourceGroup
[constr_3527]	PhmLogicalExpression referenced by one PhmRule
[constr_3543]	At least one transportPlugin definition required for each DdsProvidedServiceInstance
[constr_3544]	At least one transportPlugin definition required for each DdsRequiredServiceInstance
[constr_3556]	Unique transport layer mapping

Table F.38: Deleted Constraints in R20-11

F.8 Constraint and Specification Item History of this document according to AUTOSAR Release R21-11

F.8.1 Added Specification Items in R21-11

Number	Heading
[TPS_MANI_01343]	Relation between Function Group states and NmNetworkHandle
[TPS_MANI_01344]	Actions taken after installation of a SoftwarePackage
[TPS_MANI_01345]	Ability to attach checksums to SoftwareCluster
[TPS_MANI_01346]	No formal definition of checksum algorithm
[TPS_MANI_01347]	Definition of a DiagnosticDataElement used in the context of a DID obtained from a PPortPrototype typed by a DiagnosticDataElementInterface
[TPS_MANI_01348]	Definition of a DiagnosticDataElement used in the context of a DID obtained from a PPortPrototype typed by a DiagnosticDataIdentifierInterface
[TPS_MANI_01349]	Configuration of diagnostic-related properties of a SoftwareCluster
[TPS_MANI_01350]	Semantics of DiagnosticServiceValidationConfiguration
[TPS_MANI_01351]	Reporting the status of a DiagnosticEvent on the *AUTOSAR adaptive platform*
[TPS_MANI_01352]	Definition of DiagnosticServiceValidationMapping
[TPS_MANI_01353]	Semantics of DiagnosticExternalAuthenticationInterface
[TPS_MANI_01354]	Rationale for the existence of meta-class AbstractRawDataStreamEthernetCredentials
[TPS_MANI_01355]	Definition of local Ethernet credentials
[TPS_MANI_01356]	Definition of remote server's Ethernet credentials
[TPS_MANI_01357]	Definition of remote client's Ethernet credentials

▽

△

Number	Heading
[TPS_MANI_01358]	Restriction for the configuration of diagnostic debouncing
[TPS_MANI_01359]	Semantics of DiagnosticAuthenticationInterface
[TPS_MANI_01360]	Creation of two diagnostic mappings the fulfill different roles in the context of authenticating a diagnostic client
[TPS_MANI_01361]	Support the authentication request of a diagnostic client
[TPS_MANI_01362]	Support the responding to an authentication request of a diagnostic client
[TPS_MANI_01363]	Semantics of DiagnosticComControlInterface
[TPS_MANI_02384]	DltLogSink with category DLT_LOGSINK_REMOTE
[TPS_MANI_02385]	DltLogSink with category DLT_LOGSINK_DLT
[TPS_MANI_02386]	DltLogSink with category DLT_LOGSINK_FILE
[TPS_MANI_02387]	DltLogSink with category DLT_LOGSINK_CONSOLE
[TPS_MANI_02388]	Semantics of DltLogSinkToPortPrototypeMapping
[TPS_MANI_03277]	ServiceInterfaceElementMappings for a subset of elements of a single ServiceInterface
[TPS_MANI_03278]	Usage of ApSomeipTransformationProps.byteOrder
[TPS_MANI_03279]	Priority of Nm messages
[TPS_MANI_03280]	Semantics of ApApplicationEndpoint
[TPS_MANI_03281]	Port specific TCP configuration settings
[TPS_MANI_03282]	Assignment of a Dlt Ecu Identifier to a LogAndTraceInstantiation
[TPS_MANI_03283]	Standardized values for attribute DltLogSink.category
[TPS_MANI_03284]	Semantics of LogAndTraceInterface
[TPS_MANI_03285]	Semantics of PortPrototype.logAndTraceMessageCollectionSet
[TPS_MANI_03286]	Assignment of DltApplication to a Process
[TPS_MANI_03287]	ServiceInterface.trigger to ISignalTriggering mapping
[TPS_MANI_03288]	ApSomeipTransformationProps for triggers
[TPS_MANI_03289]	Semantics of ServiceInterfaceTriggerMapping
[TPS_MANI_03290]	Semantics of ComTriggerGrantDesign
[TPS_MANI_03291]	Semantics of ServiceInterface.trigger
[TPS_MANI_03633]	Semantics of several supervisionMode references for one supervision
[TPS_MANI_03635]	Determination of translation direction
[TPS_MANI_03636]	Definition of ServiceInterfaceElementSecureComConfig in the context of signal/service translation
[TPS_MANI_03637]	Ignored attributes of ServiceInterfaceElementSecureComConfig in the context of signal/service translation
[TPS_MANI_03638]	Mapping of MultiplexedIPdu
[TPS_MANI_03639]	Mapping of MultiplexedIPdu static part
[TPS_MANI_03640]	Mapping of MultiplexedIPdu dynamic part
[TPS_MANI_03641]	No mapping of MultiplexedIPdu selector field value
[TPS_MANI_03642]	Exactly one alternative defined for mapped event / field during runtime

▽

△

Number	Heading
[TPS_MANI_03643]	Translation-`Executable category`
[TPS_MANI_03644]	Supported reception data filters
[TPS_MANI_03645]	Applicability of a filter to the `scoped filtered element`
[TPS_MANI_03646]	Definition of transmission triggers for `signal-service-translation`
[TPS_MANI_03647]	Full translation before transmission triggering
[TPS_MANI_03648]	Transmission trigger for `signal-service-translation`
[TPS_MANI_03649]	No transmission trigger for `signal-service-translation`
[TPS_MANI_03650]	Definition of `DdsServiceInstanceResourceIdentifierTypeEnum`
[TPS_MANI_03651]	Mandatory `SupervisionCheckpoint.process` reference
[TPS_MANI_03652]	Signal/service translation header format
[TPS_MANI_03653]	Signal/service translation header endianness
[TPS_MANI_03654]	Definition of `Partial SOME/IP Header` ID
[TPS_MANI_03655]	Definition of `No Message Header` - implicit SOME/IP Message Identification
[TPS_MANI_03656]	Definition of `No Message Header` - implicit Arbitrary Message Identification
[TPS_MANI_03657]	Signal-service-translation SOME/IP Service Discovery Find
[TPS_MANI_03658]	Service-signal-translation SOME/IP Service Discovery Offer
[TPS_MANI_03659]	No `signal-service-translation` SOME/IP Service Discovery Find
[TPS_MANI_03660]	No `signal-service-translation` SOME/IP Service Discovery Offer
[TPS_MANI_03661]	Configuration of Governance in DDS Security
[TPS_MANI_03662]	Configuration of Topic access rules

Table F.39: Added Specification Items in R21-11

F.8.2 Changed Specification Items in R21-11

Number	Heading
[TPS_MANI_01001]	Meaning of `ServiceInterface`
[TPS_MANI_01048]	Retrieving the status of a `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_01111]	Diagnostic Address of a `SoftwareCluster`
[TPS_MANI_01176]	Standardized value for attribute `CppImplementationDataType.typeEmitter`

▽

△

Number	Heading
[TPS_MANI_01239]	Semantics of meta-class ComEventGrant
[TPS_MANI_01244]	Semantics of DiagnosticDataElementInterface
[TPS_MANI_01259]	Mapping of DiagnosticClearCondition to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01260]	Mapping of DiagnosticIndicator to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01261]	Mapping of DiagnosticMemoryDestination to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01262]	Mapping of DiagnosticSecurityLevel to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01263]	Mapping of DiagnosticDataIdentifier or DiagnosticDataElement to PortPrototype(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01274]	Standardized System Category Definitions for the Adaptive Platform
[TPS_MANI_01287]	Semantics of RawDataStreamMapping
[TPS_MANI_01326]	Generic Mapping to a DiagnosticServiceInstance on the AUTOSAR Adaptive Platform
[TPS_MANI_01331]	Standardized values of attribute SoftwareCluster.category
[TPS_MANI_03007]	Udp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03008]	Tcp Transport Protocol Configuration for ProvidedSomeipServiceInstance
[TPS_MANI_03010]	Udp Transport Protocol Configuration in case of IP-Multicast
[TPS_MANI_03038]	Purpose of ServiceEventDeployment
[TPS_MANI_03101]	SOME/IP serialization
[TPS_MANI_03109]	TransformationProps on the level of DataPrototypes overwrites TransformationProps settings on the level of a ServiceInterface
[TPS_MANI_03114]	Usage of AssemblySwConnectors in the System Design model
[TPS_MANI_03130]	Standardized values of the attribute E2EProfileConfiguration.profileName
[TPS_MANI_03131]	Non-Standardized values of attribute E2EProfileConfiguration.profileName
[TPS_MANI_03163]	Network configuration for Log and Trace messages
[TPS_MANI_03165]	Network Interface configuration for DoIP
[TPS_MANI_03185]	Structure of a CppImplementationDataType of category ASSOCIATIVE_MAP
[TPS_MANI_03225]	References to VLANs in PncMapping
[TPS_MANI_03227]	Usage of ephemeral ports
[TPS_MANI_03252]	Usage of same End2EndMethodProtectionProps.sourceId in case of Multi-Binding
[TPS_MANI_03274]	Configuration of log and trace sinks
[TPS_MANI_03276]	Semantics of PersistencyDeploymentToDltLogSinkMapping
[TPS_MANI_03512]	Applicability of global supervision without Process context

▽

△

Number	Heading
[TPS_MANI_03513]	Collection of supervisions into a `GlobalSupervision`
[TPS_MANI_03514]	Expiration tolerance for `SupervisionMode`
[TPS_MANI_03515]	Failure tolerance for `AliveSupervision`
[TPS_MANI_03527]	Definition of `DdsProvidedServiceInstance`
[TPS_MANI_03529]	Definition of `DdsRequiredServiceInstance`
[TPS_MANI_03577]	Definition of `Arbitrary Message Header` ID
[TPS_MANI_03585]	Processing order of COM-Stack features
[TPS_MANI_03589]	Reception data filter of COM-Stack
[TPS_MANI_03590]	Transfer properties and transmission modes of COM-Stack
[TPS_MANI_03592]	`ISignal` invalidation of COM-Stack
[TPS_MANI_03593]	`handleInvalid` = `dontInvalidate` behavior of COM-Stack
[TPS_MANI_03594]	`handleInvalid` = `replace` behavior of COM-Stack
[TPS_MANI_03597]	Support for `MultiplexedIPdu`
[TPS_MANI_03600]	Signal-service-translation of E2E protected payload
[TPS_MANI_03601]	Signal-service-translation of E2E protected payload - timeout handling
[TPS_MANI_03602]	Signal-service-translation of E2E protected payload - error handling
[TPS_MANI_03603]	Service-signal-translation of E2E protected payload
[TPS_MANI_03604]	Service-signal-translation of E2E protected payload - timeout handling
[TPS_MANI_03605]	Service-signal-translation of E2E protected payload - error handling
[TPS_MANI_03607]	Handling of safe `signal/service translation` in one `Executable`
[TPS_MANI_03608]	Support for safe `signal/service translation`
[TPS_MANI_03609]	Support for safe `signal/service translation` with same or different E2E profiles
[TPS_MANI_03621]	Data filtering during the `signal-service-translation`
[TPS_MANI_03627]	No `signal/service translation` for methods

Table F.40: Changed Specification Items in R21-11

F.8.3 Deleted Specification Items in R21-11

Number	Heading
[TPS_MANI_01014]	Semantics of meta-class `StartupConfigSet`
[TPS_MANI_01037]	Diagnostic data mapping on the *AUTOSAR adaptive platform*
[TPS_MANI_01054]	Definition of the queue length of an `event` or `field` notifier
[TPS_MANI_01060]	Use cases for the application of `DiagnosticServiceDataMapping`
[TPS_MANI_01103]	Three-level approach to REST modeling
[TPS_MANI_01105]	Semantics of `RestServiceInterface`

▽

△

Number	Heading
[TPS_MANI_01120]	Recursive definition of RestResourceDef
[TPS_MANI_01121]	Semantics of RestResourceDef.endpoint
[TPS_MANI_01122]	Arguments to endpoints
[TPS_MANI_01123]	System Triggered Event
[TPS_MANI_01124]	Semantics of RestElementDef
[TPS_MANI_01125]	Properties of REST elements can either be primitive or have array semantics
[TPS_MANI_01126]	Definition of string properties
[TPS_MANI_01127]	Limited support for data semantics in RestAbstractNumericalPropertyDef
[TPS_MANI_01128]	Difference between RestIntegerPropertyDef and RestNumberPropertyDef
[TPS_MANI_01129]	RestObjectRef is only needed for specific implementations of REST-based communication
[TPS_MANI_01130]	Structure of a typical URI for a REST service
[TPS_MANI_01131]	Impact of nested REST resources on the structure of REST URI
[TPS_MANI_01178]	Semantics of RestHttpPortPrototypeMapping.acceptEncoding
[TPS_MANI_01256]	AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticIndicatorInterface
[TPS_MANI_01257]	AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticConditionInterface
[TPS_MANI_01258]	AdaptiveApplicationSwComponentType offers a PPortPrototype typed by DiagnosticGenericUdsInterface
[TPS_MANI_01272]	Duplicate entries in logTrageLogMode
[TPS_MANI_01288]	Impact of the SoftwarePackage on the value of function group states on the target platform
[TPS_MANI_01289]	Order of function group states is relevant
[TPS_MANI_01303]	Semantics of reference SoftwarePackageStep.preActivate
[TPS_MANI_01304]	Semantics of reference SoftwarePackageStep.verify
[TPS_MANI_01311]	Handling of manufacturer checks
[TPS_MANI_01312]	Handling of supplier checks
[TPS_MANI_01336]	Two use cases for using the DiagnosticEventPortMapping
[TPS_MANI_03075]	Byte Order of chosen DataPrototype in the serialized data stream
[TPS_MANI_03160]	Further configuration options in DltLogChannel
[TPS_MANI_03275]	Configuration of log and trace message source on design level
[TPS_MANI_03552]	Supervision cycle for GlobalSupervision
[TPS_MANI_03584]	Definition of transmission triggers for translations with different sources
[TPS_MANI_03586]	No transmission trigger for translations with different sources
[TPS_MANI_03587]	Transmission trigger for translations with different sources
[TPS_MANI_03588]	Full translation before transmission triggering
[TPS_MANI_03596]	Update Bit support for ISignalIPdu

Table F.41: Deleted Specification Items in R21-11

F.8.4 Added Constraints in R21-11

Number	Heading
[constr_3623]	SupervisionCheckpoints in the context of a GlobalSupervision
[constr_3624]	At least one Supervision defined in the context of a GlobalSupervision
[constr_3625]	DeadlineSupervision referencing CheckpointTransition in the context of a GlobalSupervision
[constr_3626]	LogicalSupervision referencing CheckpointTransition in the context of a GlobalSupervision
[constr_3627]	Existence of SupervisionModeCondition.stateReference
[constr_3628]	Reference to Function Group State from a SupervisionModeCondition
[constr_3629]	Identical Function Group in the scope of a GlobalSupervision
[constr_3630]	GlobalSupervision and Process relation
[constr_3631]	Global supervision restricted to one Function Group
[constr_3632]	Supervision of a Supervised Entity Instance in the scope of a Function Group State
[constr_3633]	Mandatory attributes of AliveSupervision
[constr_3634]	Multiplicity of CheckpointTransition.source and CheckpointTransition.target
[constr_3635]	Mandatory attributes of DeadlineSupervision
[constr_3636]	Consistent ISignal communication direction in and RequiredApServiceInstance
[constr_3637]	Consistent ISignal communication direction out and ProvidedApServiceInstance
[constr_3639]	Existence of SupervisionMode.expiredSupervisionTolerance
[constr_3640]	Existence of SupervisionMode.modeCondition
[constr_3641]	Allowed combinations of ServiceInterfaceDeployment, AdaptivePlatformServiceInstance, ServiceInstanceToMachineMapping
[constr_3642]	Restriction of aggregation of PortPrototypeProps to the Adaptive Platform
[constr_3643]	No filter support for service-signal-translation direction
[constr_3644]	No transmissionTrigger support for service-signal-translation direction
[constr_3645]	discoveryType mandatory for DdsProvidedServiceInstance
[constr_3646]	resourceIdentifierType mandatory for DdsProvidedServiceInstance
[constr_3647]	resourceIdentifierType value for USER_DATA QoS-based discovery
[constr_3648]	discoveryType mandatory for DdsRequiredServiceInstance
[constr_3649]	Consistent SupervisionCheckpoint.process reference
[constr_3650]	headerId required in case of Arbitrary Message Header
[constr_3674]	Existence of NoSupervision.targetPhmSupervisedEntity
[constr_3675]	Existence of NoSupervision.process
[constr_3676]	Exclusive usage of NoSupervision

▽

△

Number	Heading
[constr_3677]	ComGrants referencing DDS Service Instances
[constr_3678]	Existence of attributes for DdsSecureComProps
[constr_3679]	Existence of attributes for DdsSecureGovernance
[constr_3680]	Existence of attributes for DdsTopicAccessRule
[constr_3681]	Supported values of DdsTopicAccessRule.dataProtectionKind
[constr_3682]	Values of DdsDomainRange.min and DdsDomainRange.max
[constr_3683]	Attributes referencing DdsTopicAccessRule
[constr_3684]	Mutual exclusivity of Secure Communication Properties
[constr_5250]	Protection of AdaptivePlatformServiceInstances of the same ServiceInterfaceDeployment
[constr_5260]	UDP endpoint using DTLS CLIENT role can only serve required service instances
[constr_5261]	TCP endpoint using TLS CLIENT role can only serve required service instances
[constr_5275]	Existence of LogAndTraceInstantiation.dltEcu
[constr_5276]	Existence of LogAndTraceInstantiation.logSink
[constr_5277]	applicable DltLogSink categories
[constr_5278]	DltLogSink with category DLT_LOGSINK_REMOTE is only allowed to be referenced by DltLogSinkToPortPrototypeMapping
[constr_5279]	DltLogSink with category DLT_LOGSINK_DLT is only allowed to be referenced by LogAndTraceInstantiation
[constr_5280]	Existence of DltLogSink.defaultLogThreshold
[constr_5281]	Existence of DltLogSink.defaultTraceState
[constr_5282]	Existence of DltLogSinkToPortPrototypeMapping.process
[constr_5283]	Existence of DltLogSinkToPortPrototypeMapping.dltLogSink
[constr_5284]	Existence of DltLogSinkToPortPrototypeMapping.dltContext
[constr_5285]	Existence of PortPrototype references in DltLogSinkToPortPrototypeMapping
[constr_5286]	Restriction applicable for DltLogSinkToPortPrototypeMapping. rPortPrototype
[constr_5287]	Restriction applicable for DltLogSinkToPortPrototypeMapping. pPortPrototype
[constr_5288]	Existence of process reference in DltApplicationToProcessMapping
[constr_5289]	Existence of dltApplication reference in DltApplicationToProcessMapping
[constr_5290]	PPortPrototype is not allowed to be typed by LogAndTraceInterface
[constr_5291]	Allowed usage of PortPrototype.logAndTraceMessageCollectionSet
[constr_5292]	Assigned dltSessionId shall be consistent for the same PortPrototype
[constr_5316]	Allowed ServiceEventDeployment.trigger references
[constr_5317]	ServiceEventDeployment not allowed to reference an event and a trigger at the same time
[constr_5318]	Existence of ServiceInstanceToSignalMapping for an trigger with signalBased serialization

▽

△

Number	Heading
[constr_10029]	ServiceInterfaceDeployment shall cover all elements of the corresponding ServiceInterfaceDeployment
[constr_10030]	Existence of DiagnosticDataIdentifierInterface.read
[constr_10031]	Existence of DiagnosticRoutineInterface.start
[constr_10035]	Completeness of the PersistencyDeployment.version
[constr_10037]	Existence of attribute TagWithOptionalValue.sequenceOffset in the context of attribute capabilityRecord owned by ProvidedSomeipServiceInstance, RequiredSomeipServiceInstance, SdServerConfig, SdClientConfig, or AbstractServiceInstance
[constr_10046]	Value of PersistencyRedundancyMOutOfN.n
[constr_10047]	Restriction for the applicability of DiagnosticMonitorPortMapping
[constr_10048]	Existence of reference from DiagnosticMonitorPortMapping to DiagnosticEvent
[constr_10049]	Restriction for the applicability of DiagnosticEventPortMapping
[constr_10050]	Restriction for the applicability of DiagnosticOperationCyclePortMapping
[constr_10051]	Existence of reference from DiagnosticOperationCyclePortMapping to DiagnosticOperationCycle
[constr_10052]	Restriction for the applicability of DiagnosticEnableConditionPortMapping
[constr_10053]	Existence of reference from DiagnosticEnableConditionPortMapping to DiagnosticEnableCondition
[constr_10054]	Restriction for the applicability of DiagnosticClearConditionPortMapping
[constr_10055]	Existence of reference from DiagnosticClearConditionPortMapping to DiagnosticClearCondition
[constr_10056]	Restriction for the applicability of DiagnosticIndicatorPortMapping
[constr_10057]	Restriction for the applicability of DiagnosticMemoryDestinationPortMapping
[constr_10058]	Restriction for the applicability of DiagnosticSecurityLevelPortMapping
[constr_10059]	Existence of reference from DiagnosticSecurityLevelPortMapping to DiagnosticSecurityLevel
[constr_10060]	PortInterface of PPortPrototype referenced by DiagnosticDataPortMapping
[constr_10061]	Mapping to DiagnosticDataIdentifierInterface, DiagnosticDataElementInterface, or DiagnosticDataIdentifierGenericInterface
[constr_10062]	DiagnosticServiceInstances that can be mapped by a DiagnosticServiceGenericMapping
[constr_10063]	Possible values for DiagnosticServiceValidationMapping.category
[constr_10064]	Existence of DiagnosticServiceValidationMapping. pPortPrototypeInExecutable
[constr_10065]	Validity of DiagnosticServiceValidationConfiguration. manufacturerValidationOrder
[constr_10066]	Validity of DiagnosticServiceValidationConfiguration. supplierValidationOrder

▽

△

Number	Heading
[constr_10069]	Existence of SoftwareClusterDiagnosticProps.powerDownTime
[constr_10070]	Value of RequiredSomeipServiceInstance.requiredServiceInstanceId
[constr_10076]	Existence of RawDataStreamEthernetUdpCredentials.udpPort
[constr_10077]	Existence of ipV4Address and ipV6Address within AbstractRawDataStreamEthernetCredentials
[constr_10078]	Existence of RawDataStreamEthernetTcpUdpCredentials.tcpPort and udpPort
[constr_10079]	Existence of EthernetRawDataStreamMapping.localTcpPort and localUdpPort
[constr_10080]	Existence of initial values for PersistencyFileElement
[constr_10081]	Existence of initial values in the definition of PersistencyDataRequiredComSpec
[constr_10082]	Existence of initial values for PersistencyFile
[constr_10083]	Existence of initial values for PersistencyKeyValuePair
[constr_10086]	Existence of unicastUdpCredentials and multicastCredentials in the context of a EthernetRawDataStreamServerMapping
[constr_10090]	Existence of ProcessToMachineMapping.persistencyCentralStorageURI
[constr_10092]	Restriction for the applicability of DiagnosticAuthenticationPortMapping
[constr_10093]	Existence of reference from DiagnosticAuthenticationPortMapping to DiagnosticAuthentication
[constr_10094]	Restriction for the applicability of DiagnosticExternalAuthenticationPortMapping
[constr_10095]	Existence of reference from DiagnosticExternalAuthenticationPortMapping to DiagnosticAuthentication

Table F.42: Added Constraints in R21-11

F.8.5 Changed Constraints in R21-11

Number	Heading
[constr_1482]	Mapping of service interfaces vs. mapping of service interface elements
[constr_1564]	Existence of SoftwareCluster.diagnosticProps.diagnosticAddress
[constr_1598]	Scope of the uniqueness of the value of TlvDataIdDefinition.id for references to CppImplementationDataTypeElement
[constr_1601]	TlvDataIdDefinition referencing CppImplementationDataTypeElement
[constr_1604]	Completeness of the existence of a set of TlvDataIdDefinition.tlvImplementationDataTypeElements

▽

△

Number	Heading
[constr_1697]	Restriction for ClientServerOperation aggregated by a DiagnosticDataIdentifierInterface or DiagnosticDataElementInterface
[constr_1748]	Existence of references TlvDataIdDefinition.tlvArgument, TlvDataIdDefinition.tlvRecordElement, and TlvDataIdDefinition.tlvImplementationDataTypeElement
[constr_1751]	Value of PersistencyRedundancyMOutOfN.m
[constr_1764]	Counterpart of PhmCheckpoint
[constr_1770]	Value of ProvidedSomeipServiceInstance.serviceInstanceId
[constr_3288]	IP configuration restriction for unicastNetworkEndpoints
[constr_3395]	TransformationPropsToServiceInterfaceElementMapping is restricted to one single ServiceInterface
[constr_3421]	Fibex elements applicable for a System of category MACHINE_DESIGN_EXTRACT
[constr_3485]	UDP endpoint using DTLS SERVER role can only serve provided service instances
[constr_3486]	TCP endpoint using TLS SERVER role can only serve provided service instances
[constr_3493]	Applicable attributes for standardized E2E Profiles
[constr_3539]	Only one AliveSupervision per SupervisionCheckpoint
[constr_3540]	SupervisionCheckpoint in supervision graph
[constr_3541]	qosProfile mandatory for DdsProvidedServiceInstance
[constr_3542]	qosProfile mandatory for DdsRequiredServiceInstance
[constr_3555]	No support for useAsCryptographicIPdu is true
[constr_5047]	Supported values of TlsSecureComProps.category
[constr_5052]	ProvidedSomeipServiceInstances of the same serviceInterface on one Machine
[constr_5056]	Restriction of sub-class of CompositionSwComponentType.connector
[constr_5057]	PassThroughSwConnector and ServiceInterfaceMapping
[constr_5155]	SomeipServiceInstanceToMachineMapping only supports a single Address Family
[constr_10002]	Only one mapping per PortPrototype
[constr_10003]	Restriction for the existence of DiagnosticDataPortMapping.diagnosticDataIdentifier vs. DiagnosticDataPortMapping.diagnosticDataElement

Table F.43: Changed Constraints in R21-11

F.8.6 Deleted Constraints in R21-11

Number	Heading
[constr_1496]	DiagnosticServiceDataMapping.mappedApDataElement shall only refer to specific sub-classes of DataPrototype
[constr_1500]	Target SwcServiceDependency of DiagnosticEventPortMapping. swcServiceDependencyInExecutable
[constr_1501]	Target SwcServiceDependency of DiagnosticOperationCyclePortMapping. swcServiceDependencyInExecutable
[constr_1502]	Target SwcServiceDependency of DiagnosticEnableConditionPortMapping. swcServiceDependencyInExecutable
[constr_1569]	Restriction for the scope of RestHttpPortPrototypeMapping. acceptEncoding
[constr_1580]	Restriction for the usage of RestHttpPortPrototypeMapping. acceptEncoding
[constr_1698]	Target SwcServiceDependency of DiagnosticClearConditionPortMapping. swcServiceDependencyInExecutable
[constr_1699]	Target SwcServiceDependency of DiagnosticIndicatorPortMapping. swcServiceDependencyInExecutable
[constr_1700]	Target SwcServiceDependency of DiagnosticMemoryDestinationPortMapping. swcServiceDependencyInExecutable
[constr_1701]	Target SwcServiceDependency of DiagnosticSecurityLevelPortMapping. swcServiceDependencyInExecutable
[constr_1702]	Target SwcServiceDependency of DiagnosticServiceDataIdentifierPortMapping. swcServiceDependencyInExecutable
[constr_1733]	Simultaneous existence of SoftwarePackageStep.preActivate and SoftwarePackageStep.verify
[constr_1738]	Multiplicity of reference GlobalSupervision.localSupervision
[constr_1739]	Multiplicity of aggregation localSupervision.transition
[constr_1744]	Definition of process state In the context of the ExecutionDependency
[constr_1765]	Diagnostic Services eligible for DiagnosticServiceGenericMapping
[constr_1809]	Global supervision restricted to one function group
[constr_3366]	System category for a system design description with Adaptive Platform and Classic Platform content
[constr_3426]	The logTraceFilePath is mandatory in case that logTrageLogMode is set to file
[constr_3427]	The logTraceFilePath is only relevant if logTrageLogMode is set to file
[constr_3537]	localSupervision referenced once in the context of a GlobalSupervision

▽

△

Number	Heading
[constr_3562]	Existence of NonqueuedReceiverComSpec.filter
[constr_5243]	Restriction of LogAndTraceInstantiation.dltEcuId attribute value
[constr_10004]	Consistency of DiagnosticServiceGenericMapping for PortPrototype typed by DiagnosticDataIdentifierGenericInterface

Table F.44: Deleted Constraints in R21-11

F.9 Constraint and Specification Item History of this document according to AUTOSAR Release R22-11

F.9.1 Added Specification Items in R22-11

Number	Heading
[TPS_MANI_01365]	Semantics of attribute NmHandleToFunctionGroupStateMapping.mappingDirection set to functionGroupStateToNmHandle
[TPS_MANI_01366]	Semantics of attribute NmHandleToFunctionGroupStateMapping.mappingDirection set to nmHandleInactiveToFunctionGroupState
[TPS_MANI_01367]	Semantics of attribute NmHandleToFunctionGroupStateMapping.mappingDirection set to nmHandleActiveToFunctionGroupState
[TPS_MANI_01369]	Semantics of reference SoftwareClusterDesign.requiredARElement
[TPS_MANI_01370]	Semantics of ExecutableImplementationProps
[TPS_MANI_01371]	PersistencyDeployment.redundancyHandling does not exist
[TPS_MANI_01372]	Properties of data obtained from the application software
[TPS_MANI_01373]	Semantics of DiagnosticRequestFileTransferInterface
[TPS_MANI_01374]	Semantics of UcmToTimeBaseResourceMapping
[TPS_MANI_01375]	Semantics of meta-class UcmRetryStrategy
[TPS_MANI_01376]	Semantics of meta-class UcmMasterModuleInstantiation
[TPS_MANI_01377]	Semantics of meta-class UcmSubordinateModuleInstantiation
[TPS_MANI_01378]	Semantics of meta-class PersistencyKeyValueDataTypeMapping
[TPS_MANI_01379]	Semantics of FunctionalClusterInteractsWithPersistencyDeploymentMapping
[TPS_MANI_01380]	Usage of PersistencyDeployment
[TPS_MANI_01381]	Semantics of meta-class ArtifactLocator
[TPS_MANI_01382]	Location of artifact that contains executable code represented by Executable
[TPS_MANI_01383]	Semantics of ExecutableLoggingImplementationProps
[TPS_MANI_01384]	Semantics of StateManagementRequestRule
[TPS_MANI_01387]	Semantics of meta-class StateManagementActionList

▽

△

Number	Heading
[TPS_MANI_01388]	Semantics of the meta-class StateManagementStateMachineActionItem
[TPS_MANI_01389]	Semantics of meta-class StateManagementSyncActionItem
[TPS_MANI_01390]	Semantics of meta-class StateManagementSetFunctionGroupStateActionItem
[TPS_MANI_01391]	Semantics of StateManagementStateRequest
[TPS_MANI_01392]	Semantics of StateManagementStateNotification
[TPS_MANI_01393]	Initialization of a data object typed by a CppImplementationDataType of category VARIANT
[TPS_MANI_01394]	Support for the authorization of SOVD clients
[TPS_MANI_01395]	Support for the proximity challenge in the context of SOVD
[TPS_MANI_01396]	Semantics of meta-class DiagnosticSovdLock
[TPS_MANI_01397]	Standardized values of DiagnosticDataIdentifier.category of the usage in SOVD context
[TPS_MANI_01398]	Definition of an *SOVD data group*
[TPS_MANI_01399]	Modeling of the configuration of Service-oriented Vehicle Diagnostics
[TPS_MANI_01400]	Secure communication between SOVD Gateway and SOVD Server
[TPS_MANI_01401]	Support for TLS in vehicle-internal SOVD communication
[TPS_MANI_01402]	Identification of the SOVD Server
[TPS_MANI_01403]	External communication of the SOVD gateway
[TPS_MANI_01404]	Secure communication between SOVD gateway and SOVD Client
[TPS_MANI_01405]	Semantics of sub-classes of SoftwareClusterDiagnosticAddress
[TPS_MANI_01406]	Semantics of DeterministicSyncMaster
[TPS_MANI_01407]	Semantics of abstract base class DeterministicSyncVerificationPolicy
[TPS_MANI_01408]	Semantics of reference DeterministicClient.deterministicSyncMaster
[TPS_MANI_01409]	Semantics of meta-class DeterministicSyncMasterToTimeBaseConsumerMapping
[TPS_MANI_01410]	Semantics of attribute SoftwareCluster.installationBehavior
[TPS_MANI_01835]	Semantics of StateManagementRequestTrigger.rule
[TPS_MANI_01836]	Semantics of StateManagementRequestError.rule
[TPS_MANI_01863]	Supported value range for attribute ApApplicationErrorDomain.value
[TPS_MANI_03292]	Semantic of SomeipServiceInstanceToMachineMapping.tcpPort
[TPS_MANI_03293]	Semantic of SomeipServiceInstanceToMachineMapping.udpPort
[TPS_MANI_03294]	Semantics of DoIpNetworkConfiguration.eidRetrieval
[TPS_MANI_03295]	Semantics of FirewallStateSwitchInterface.firewallStateMachine
[TPS_MANI_03296]	Machine-specific configuration settings for Firewall
[TPS_MANI_03297]	Semantics of AdaptiveFirewallToPortPrototypeMapping
[TPS_MANI_03298]	Semantics of StateDependentFirewall

▽

△

Number	Heading
[TPS_MANI_03299]	Semantics of `FirewallRuleProps`
[TPS_MANI_03300]	Semantics of `FirewallRule`
[TPS_MANI_03301]	Semantics of `FirewallRule.refillAmount` and `FirewallRule.bucketSize`
[TPS_MANI_03302]	Firewall rules on Data Link Layer
[TPS_MANI_03303]	Filtering of packets with a single MAC address and MAC address range
[TPS_MANI_03304]	Firewall rules on Network Layer
[TPS_MANI_03305]	Filtering of packets with a single IP address and IP address range
[TPS_MANI_03306]	Firewall rules on Transport Layer
[TPS_MANI_03307]	Byte-Pattern Firewall rule
[TPS_MANI_03308]	Deep inspection of SOME/IP SD messages
[TPS_MANI_03309]	Deep inspection of SOME/IP messages
[TPS_MANI_03310]	Deep inspection of DoIP messages
[TPS_MANI_03311]	Deep inspection of DDS messages
[TPS_MANI_03312]	Static configuration of remote peer addresses for a `ProvidedSomeipServiceInstance`
[TPS_MANI_03313]	Semantics of `SomeipRemoteUnicastConfig.eventGroup` reference
[TPS_MANI_03314]	Static configuration of a remote peer address for a `RequiredSomeipServiceInstance`
[TPS_MANI_03315]	Semantics of `SomeipRemoteMulticastConfig`
[TPS_MANI_03316]	Semantics of `SomeipRemoteMulticastConfig.eventGroup`
[TPS_MANI_03317]	MACsec configuration
[TPS_MANI_03318]	MAC Security Key Agreement Entity configuration
[TPS_MANI_03319]	Standardized values for the attribute `cipherSuite` of meta-class `MacSecCipherSuiteConfig`
[TPS_MANI_03320]	Semantics of `MacSecCipherSuiteConfig.cipherSuitePriority`
[TPS_MANI_03321]	Semantics of `FirewallStateSwitchInterface`
[TPS_MANI_03663]	Ignored references of `ServiceInstanceToMachineMapping` in the context of signal/service translation
[TPS_MANI_03664]	`ServiceInterfaceElementSecureComConfig.securedRxVerification`
[TPS_MANI_03665]	`ServiceInterface.method` with `fireAndForget` equals `true` to `ISignalTriggering`s mapping
[TPS_MANI_03666]	Service Discovery for multi-event `ISignalIPdu`s
[TPS_MANI_03667]	Mapping from one `ISignalIPdu` to several `event`s in case of `signal-service-translation`
[TPS_MANI_03668]	Supported type of Ethernet tunneling through CAN XL on Adaptive Platform
[TPS_MANI_03669]	Ethernet tunneling through CAN XL
[TPS_MANI_03670]	Definition of `NoCheckpointSupervision`
[TPS_MANI_03671]	Periodic storage of time base value
[TPS_MANI_03672]	Definition of `RecoveryNotification` for an `HealthChannel`

▽

△

Number	Heading
[TPS_MANI_03673]	`RecoveryNotification` referenced by several `HealthChannel`. `recoveryNotification`s

Table F.45: Added Specification Items in R22-11

F.9.2 Changed Specification Items in R22-11

Number	Heading
[TPS_MANI_01111]	Diagnostic Address of a `SoftwareCluster`
[TPS_MANI_01114]	Relation of `DiagnosticContributionSet` to `SoftwareCluster`
[TPS_MANI_01164]	Semantics of `SoftwareCluster.dependsOn`
[TPS_MANI_01179]	Semantics of `PersistencyFileElement.contentUri`/ `PersistencyFile.contentUri` vs. `PersistencyDeployment`. `deploymentUri` and `PersistencyFileElement.fileName`/ `PersistencyFile.fileName`
[TPS_MANI_01203]	Semantics of `DeterministicClient`
[TPS_MANI_01226]	Machine-specific configuration settings for the UCM module
[TPS_MANI_01255]	Semantics of DoIP `DiagnosticPortInterface`s
[TPS_MANI_01269]	Specification of boundaries for resource consumption
[TPS_MANI_01295]	Semantics of `VehicleRolloutStep`
[TPS_MANI_01297]	Semantics of meta-class `UcmStep`
[TPS_MANI_01349]	Configuration of diagnostic-related properties of a `SoftwareCluster`
[TPS_MANI_01360]	Creation of two diagnostic mappings the fulfill different roles in the context of authenticating a diagnostic client
[TPS_MANI_01361]	Support the authentication request of a diagnostic client
[TPS_MANI_01362]	Convey the Authentication state of a diagnostic client to the diagnostic server instance
[TPS_MANI_03013]	Repetition Phase configuration for a `ProvidedSomeipServiceInstance`
[TPS_MANI_03014]	Main Phase configuration for a `ProvidedSomeipServiceInstance`
[TPS_MANI_03015]	TTL for Offer Service Entries
[TPS_MANI_03027]	Repetition Phase configuration for a `RequiredSomeipServiceInstance`
[TPS_MANI_03115]	Mapping between a fire and forget `method` and elements of Classic Platform `PortInterface`s
[TPS_MANI_03178]	`StdCppImplementationDataType` of `category STRING`
[TPS_MANI_03285]	Semantics of `LTMessageCollectionToPortPrototypeMapping`
[TPS_MANI_03505]	Existence of `SupervisionCheckpoint`
[TPS_MANI_03546]	Definition of reported health status `RPortPrototype`
[TPS_MANI_03618]	Usage of `RequiredSomeipServiceInstance.blocklistedVersion`
[TPS_MANI_03654]	Definition of `Partial SOME/IP Header` ID

▽

△

Number	Heading
[TPS_MANI_03659]	No `signal-service-translation` SOME/IP Service Discovery Find
[TPS_MANI_03660]	No `signal-service-translation` SOME/IP Service Discovery Offer

Table F.46: Changed Specification Items in R22-11

F.9.3 Deleted Specification Items in R22-11

Number	Heading
[TPS_MANI_01100]	Semantics of `Allocator`
[TPS_MANI_01113]	Semantics of `SoftwareClusterDesign.diagnosticAddress`
[TPS_MANI_01161]	Impact of values of `category` on the semantics of `SoftwareClusterDesign`
[TPS_MANI_01271]	Semantics of `Executable.loggingBehavior`
[TPS_MANI_01305]	Semantics of attribute `SoftwarePackageStep.activationSwitch`
[TPS_MANI_01310]	Semantics of `SoftwareClusterDesign.dependsOn`
[TPS_MANI_01335]	Semantics of `SoftwareClusterDependencyFormula.category`
[TPS_MANI_03028]	TTL for Find Service Entries
[TPS_MANI_03126]	`ServiceInterface.field` mapping to `ISignalTriggering`s
[TPS_MANI_03506]	Optionality of `SupervisionCheckpoint`
[TPS_MANI_03623]	Usage of `checkpointId` in application code

Table F.47: Deleted Specification Items in R22-11

F.9.4 Added Constraints in R22-11

Number	Heading
[constr_3690]	`DdsServiceInterfaceDeployment.serviceInterfaceId` value shall not conflict with topic-based service discovery
[constr_3691]	Existence of `ServiceInterfaceElementSecureComConfig.securedRxVerification`
[constr_3692]	`DataPrototypeInServiceInterfaceInstanceRef.targetDataPrototype` in the context of a `SignalBasedFireAndForgetMethodToISignalTriggeringMapping`
[constr_3693]	`EthernetCommunicationConnector.category` is set to `CAN_XL`
[constr_3694]	Existence of `canXlConfig` vs. `canXlConfigReqs`
[constr_3709]	`AliveSupervision.terminatingCheckpoint` required for self terminating `Process`es

▽

△

Number	Heading
[constr_3710]	Process referenced by AliveSupervision.terminatingCheckpoint
[constr_3711]	AliveSupervision.terminatingCheckpointTimeoutUntilTermination
[constr_3712]	Exclusive usage of NoCheckpointSupervision
[constr_3715]	Reference in the role SomeipEventGroup.event
[constr_3719]	RecoveryNotification referenced either by HealthChannelExternalStatus or HealthChannelSupervision
[constr_3720]	Upper multiplicity of reference in the role ComGrantDesign.remoteSubject
[constr_3721]	Upper multiplicity of reference in the role EthernetCommunicationConnector. unicastNetworkEndpoint
[constr_3722]	Upper multiplicity of reference in the role EthernetCommunicationConnector. canXlProps
[constr_3723]	Upper multiplicity of reference in the role MachineDesign.tcpIpProps
[constr_3724]	Upper multiplicity of reference in the role MachineDesign.tcpIpIcmpProps
[constr_3725]	Upper multiplicity of reference in the role MachineDesign.ethIpProps
[constr_3727]	Upper multiplicity of reference in the role SoftwareClusterDesign. intendedTargetMachine
[constr_3728]	Upper multiplicity of reference in the role IdsPlatformInstantiation. networkInterface
[constr_3729]	Upper multiplicity of reference in the role LogAndTraceInstantiation. timeBaseResource
[constr_3730]	Upper multiplicity of reference in the role HealthChannel. recoveryNotification
[constr_3731]	Upper multiplicity of reference in the role ProcessDesign.executable
[constr_3732]	Upper multiplicity of reference in the role Process.executable
[constr_3733]	Upper multiplicity of aggregation in the role methodMapping
[constr_3734]	Upper multiplicity of reference in the role DoIpNetworkConfiguration. networkConfiguration
[constr_5324]	MachineDesign.communicationController aggregation restriction
[constr_5332]	Mandatory multicast endpoint in case of multicastThreshold different from 0
[constr_5333]	No multicast in case of TCP
[constr_5338]	ProvidedSomeipServiceInstance shall offer all SomeipEventGroups for subscription
[constr_5339]	SomeipEventGroups of a SomeipServiceInterfaceDeployment shall be referenced at most once from a RequiredSomeipServiceInstance that instantiates the SomeipServiceInterfaceDeployment
[constr_5343]	Usage of DoIpNetworkConfiguration.eidRetrieval
[constr_5347]	Supported value range for attribute SecOcSecureComProps. authenticationVerifyAttempts
[constr_5348]	Mandatory initialMode in ModeDeclarationGroup that is referenced by StateDependentFirewall
[constr_5349]	Mandatory defaultAction in StateDependentFirewall
[constr_5350]	Mandatory action in FirewallRuleProps

▽

△

Number	Heading
[constr_5351]	FirewallRule is allowed to aggregate at most one protocol subelement
[constr_5352]	DdsRule.submessageType value restriction
[constr_5353]	DdsRule.readerEntityId and DdsRule.writerEntityId value restriction
[constr_5355]	SomeipServiceInstanceToMachineMapping with configured remote peer addresses shall not mix ProvidedSomeipServiceInstances and RequiredSomeipServiceInstances
[constr_5356]	RequiredSomeipServiceInstance is allowed to have only a single statically configured remote peer as service provider
[constr_5357]	SomeipRemoteMulticastConfig shall only be used on required side
[constr_5358]	AdaptiveFirewallToPortPrototypeMapping.rPortPrototype restriction
[constr_6815]	Existence of CppTemplateArgument.templateType for CppImplementationDataType of category STRING
[constr_10098]	Relation of MachineDesign.pnResetTimer and UdpNmCluster.nmMsgCycleTime
[constr_10101]	Attribute NmHandleToFunctionGroupStateMapping.mappingDirection is set to nmHandleActiveToFunctionGroupState or nmHandleInactiveToFunctionGroupState
[constr_10102]	Existence of initial values for PersistencyKeyValuePair
[constr_10103]	Existence of initial values for PersistencyFile
[constr_10105]	Existence of UcmRetryStrategy.maximumNumberOfRetries
[constr_10106]	Existence of UcmRetryStrategy.retryIntervalTime
[constr_10107]	Existence of the attribute UcmMasterModuleInstantiation.blockInconsistent
[constr_10108]	Existence of the attribute UcmMasterModuleInstantiation.serviceBusy
[constr_10109]	Existence of the attribute UcmMasterModuleInstantiation.updateSessionRejected
[constr_10110]	Existence of UcmSubordinateModuleInstantiation on a Machine
[constr_10111]	Existence of attribute DiagnosticAuthentication.authenticationTimeout
[constr_10113]	Restriction for the existence of ExecutableLoggingImplementationProps
[constr_10114]	Existence of attributes of DiagnosticEnvDataElementCondition if the reference in the role pPortPrototype exists
[constr_10124]	Multiplicity of attribute ApplicationAssocMapDataType.key
[constr_10125]	Multiplicity of attribute ApplicationAssocMapDataType.value
[constr_10126]	Multiplicity of attribute ApplicationAssocMapElementValueSpecification.key
[constr_10127]	Multiplicity of attribute ApplicationAssocMapElementValueSpecification.value
[constr_10128]	Multiplicity of attribute CppImplementationDataTypeElementQualifier.typeReference
[constr_10129]	Multiplicity of attribute Field.hasGetter
[constr_10130]	Multiplicity of attribute Field.hasSetter
[constr_10131]	Multiplicity of attribute Field.hasNotifier

▽

△

Number	Heading
[constr_10132]	Multiplicity of attribute ApApplicationError.errorCode
[constr_10133]	Multiplicity of attribute ApApplicationErrorDomain.value
[constr_10134]	Multiplicity of reference in the role PortInterfaceToDataTypeMapping. dataTypeMappingSet
[constr_10135]	Multiplicity of reference in the role PortInterfaceToDataTypeMapping. portInterface
[constr_10136]	Multiplicity of reference in the role ServiceInterfaceMapping. compositeServiceInterface
[constr_10137]	Multiplicity of reference in the role ServiceInterfaceMapping. sourceServiceInterface
[constr_10138]	Multiplicity of reference in the role ServiceInterfaceEventMapping. sourceEvent
[constr_10139]	Multiplicity of reference in the role ServiceInterfaceEventMapping. targetEvent
[constr_10140]	Multiplicity of reference in the role ServiceInterfaceFieldMapping. sourceField
[constr_10141]	Multiplicity of reference in the role ServiceInterfaceFieldMapping. targetField
[constr_10142]	Multiplicity of reference in the role ServiceInterfaceMethodMapping. sourceMethod
[constr_10143]	Multiplicity of reference in the role ServiceInterfaceMethodMapping. targetMethod
[constr_10144]	Multiplicity of reference in the role PersistencyRedundancyChecksum. algorithmFamily
[constr_10145]	Multiplicity of reference in the role PersistencyRedundancyChecksum.length
[constr_10146]	Multiplicity of reference in the role PersistencyRedundancyMOutOfN.m
[constr_10147]	Multiplicity of reference in the role PersistencyRedundancyMOutOfN.n
[constr_10148]	Multiplicity of reference in the role PersistencyFileElement.contentUri
[constr_10149]	Multiplicity of reference in the role PersistencyFileElement.fileName
[constr_10150]	Multiplicity of reference in the role SynchronizedTimeBaseProviderInterface.timeBaseKind
[constr_10151]	Multiplicity of reference in the role PhmCheckpoint.checkpointId
[constr_10152]	Multiplicity of reference in the role FieldSenderComSpec.initValue
[constr_10153]	Multiplicity of reference in the role PersistencyDataRequiredComSpec. dataElement
[constr_10154]	Multiplicity of reference in the role ProcessDesignToMachineDesignMapping. processDesign
[constr_10155]	Multiplicity of reference in the role ComOfferServiceGrantDesign. providedServicePort
[constr_10156]	Multiplicity of reference in the role ComFindServiceGrantDesign. requiredServicePort
[constr_10157]	Multiplicity of reference in the role ComFieldGrantDesign.field
[constr_10158]	Multiplicity of reference in the role ComFieldGrantDesign.role

▽

△

Number	Heading
[constr_10159]	Multiplicity of reference in the role ComEventGrantDesign.event
[constr_10160]	Multiplicity of reference in the role ComTriggerGrantDesign.trigger
[constr_10161]	Multiplicity of reference in the role ComMethodGrantDesign.method
[constr_10162]	Multiplicity of reference in the role DiagnosticClearConditionPortMapping.clearCondition
[constr_10163]	Multiplicity of reference in the role DiagnosticIndicatorPortMapping.indicator
[constr_10164]	Multiplicity of reference in the role DiagnosticMemoryDestinationPortMapping.memoryDestination
[constr_10165]	Multiplicity of reference in the role DiagnosticDataPortMapping.process
[constr_10166]	Multiplicity of attribute DiagnosticProvidedDataMapping.dataProvider
[constr_10167]	Multiplicity of attribute SomeipServiceDiscovery.someipServiceDiscoveryPort
[constr_10169]	Multiplicity of reference in the role Machine.machineDesign
[constr_10170]	Multiplicity of attribute Machine.trustedPlatformExecutableLaunchBehavior
[constr_10171]	Multiplicity of attribute Machine.processor
[constr_10172]	Multiplicity of attribute Processor.core
[constr_10173]	Multiplicity of attribute ProcessorCore.coreId
[constr_10174]	Multiplicity of the reference in the role ProcessToMachineMapping.process
[constr_10175]	Multiplicity of attribute StateDependentStartupConfig.resourceGroup
[constr_10176]	Multiplicity of attribute StateDependentStartupConfig.startupConfig
[constr_10177]	Multiplicity of attribute PersistencyDeployment.updateStrategy
[constr_10178]	Multiplicity of the reference in the role PersistencyPortPrototypeToDeploymentMapping.process
[constr_10179]	Multiplicity of attribute PersistencyKeyValuePair.valueDataType
[constr_10180]	Multiplicity of the reference in the role PersistencyPortPrototypeToKeyValueStorageMapping.keyValueStorage
[constr_10182]	Multiplicity of the reference in the role PersistencyPortPrototypeToFileStorageMapping.fileStorage
[constr_10183]	Multiplicity of attribute PersistencyFile.fileName
[constr_10184]	Multiplicity of the reference in the role SynchronizedTimeBaseConsumer.networkTimeConsumer
[constr_10185]	Multiplicity of the reference in the role SynchronizedTimeBaseProvider.networkTimeProvider
[constr_10186]	Multiplicity of attribute DoIpInstantiation.entityStatusMaxByteFieldUse
[constr_10187]	Multiplicity of attribute DoIpInstantiation.gidInvalidityPattern
[constr_10188]	Multiplicity of attribute DoIpInstantiation.logicalAddress
[constr_10189]	Multiplicity of attribute DoIpInstantiation.maxRequestBytes
[constr_10190]	Multiplicity of attribute DoIpInstantiation.vinInvalidityPattern

▽

△

Number	Heading
[constr_10191]	Multiplicity of attribute DoIpNetworkConfiguration.isActivationLineDependent
[constr_10192]	Multiplicity of attribute DoIpNetworkConfiguration.maxInitialVehicleAnnouncementTime
[constr_10193]	Multiplicity of attribute DoIpNetworkConfiguration.maxTesterConnections
[constr_10194]	Multiplicity of attribute DoIpNetworkConfiguration.networkInterfaceId
[constr_10195]	Multiplicity of attribute DoIpNetworkConfiguration.vehicleIdentificationSyncStatus
[constr_10197]	Multiplicity of attribute DoIpRequestConfiguration.endAddress
[constr_10198]	Multiplicity of attribute DoIpRequestConfiguration.requestType
[constr_10199]	Multiplicity of attribute DoIpRequestConfiguration.startAddress
[constr_10200]	Multiplicity of attribute UcmModuleInstantiation.identifier
[constr_10201]	Multiplicity of of the reference in the role ComGrant.serviceInstance
[constr_10202]	Multiplicity of attribute ComFieldGrant.role
[constr_10203]	Multiplicity of the reference in the role ComFieldGrant.serviceDeployment
[constr_10204]	Multiplicity of the reference in the role ComMethodGrant.serviceDeployment
[constr_10205]	Multiplicity of the reference in the role ComEventGrant.serviceDeployment
[constr_10206]	Multiplicity of the reference in the role ComOfferServiceGrant.serviceInstance
[constr_10207]	Multiplicity of the reference in the role CryptoProviderToPortPrototypeMapping.cryptoProvider
[constr_10208]	Multiplicity of the reference in the role CryptoProviderToPortPrototypeMapping.process
[constr_10209]	Multiplicity of the reference in the role CryptoKeySlotToPortPrototypeMapping.keySlot
[constr_10210]	Multiplicity of the reference in the role CryptoKeySlotToPortPrototypeMapping.process
[constr_10211]	Multiplicity of the reference in the role CryptoCertificateToCryptoKeySlotMapping.cryptoCertificate
[constr_10212]	Multiplicity of attribute SomeipServiceInterfaceDeployment.serviceInterfaceId
[constr_10213]	Multiplicity of attribute SomeipServiceInterfaceDeployment.serviceInterfaceVersion
[constr_10214]	Multiplicity of attribute SomeipEventGroup.eventGroupId
[constr_10215]	Multiplicity of attribute SomeipEventDeployment.eventId
[constr_10216]	Multiplicity of attribute SomeipEventDeployment.transportProtocol
[constr_10217]	Multiplicity of the attribute DdsServiceInterfaceDeployment.serviceInterfaceId
[constr_10218]	Multiplicity of reference in the role ProvidedSomeipServiceInstance.sdServerConfig
[constr_10219]	Multiplicity of attribute ProvidedSomeipServiceInstance.serviceInstanceId
[constr_10220]	Multiplicity of attribute SomeipProvidedEventGroup.multicastThreshold

▽

△

Number	Heading
[constr_10221]	Multiplicity of reference in the role RequiredSomeipServiceInstance.sdClientConfig
[constr_10222]	Multiplicity of the reference in the role SomeipRequiredEventGroup.sdClientEventGroupTimingConfig
[constr_10223]	Multiplicity of attribute DdsServiceInstanceProps.domainId
[constr_10224]	Multiplicity of reference in the role DdsEventQosProps.event
[constr_10225]	Multiplicity of reference in the role DdsFieldQosProps.field
[constr_10226]	Multiplicity of attribute E2EProfileConfiguration.profileName
[constr_10227]	Multiplicity of attribute SecOcJobRequirement.secOcJobSemantic
[constr_10228]	Multiplicity of attribute SignalBasedFieldToISignalTriggeringMapping.dataPrototypeInServiceInterfaceRef
[constr_10229]	Multiplicity of reference in the role SignalBasedMethodToISignalTriggeringMapping.method
[constr_10230]	Multiplicity of attribute SignalServiceTranslationEventProps.safeTranslation
[constr_10231]	Multiplicity of attribute SignalServiceTranslationEventProps.secureTranslation
[constr_10232]	Multiplicity of reference in the role PersistencyDeploymentToCryptoKeySlotMapping.persistencyDeployment
[constr_10233]	Multiplicity of the reference in the role SoftwareCluster.vendorSignature
[constr_10234]	Multiplicity of attribute SoftwareCluster.version
[constr_10235]	Multiplicity of attribute SoftwareCluster.vendorId
[constr_10236]	Multiplicity of attribute SoftwareClusterDiagnosticAddress.addressSemantics
[constr_10237]	Multiplicity of attribute SoftwareClusterDependencyCompareCondition.compareType
[constr_10240]	Multiplicity of attribute SoftwarePackage.actionType
[constr_10241]	Multiplicity of attribute SoftwarePackage.compressedSoftwarePackageSize
[constr_10242]	Multiplicity of attribute SoftwarePackage.minimumSupportedUcmVersion
[constr_10243]	Multiplicity of attribute SoftwarePackage.packagerId
[constr_10244]	Multiplicity of reference in the role SoftwarePackage.packagerSignature
[constr_10245]	Multiplicity of reference in the role SoftwarePackage.softwareCluster
[constr_10246]	Multiplicity of attribute SoftwarePackage.uncompressedSoftwareClusterSize
[constr_10247]	Multiplicity of reference in the role VehiclePackage.packagerSignature
[constr_10248]	Multiplicity of reference in the role UcmDescription.identifier
[constr_10249]	Multiplicity of reference in the role VehicleDriverNotification.approvalRequired
[constr_10250]	Multiplicity of reference in the role VehicleDriverNotification.notificationState
[constr_10251]	Multiplicity of the reference in the role ServiceFieldDeployment.field

▽

△

Number	Heading
[constr_10252]	Multiplicity of attribute SignalBasedEventElementToISignalTriggeringMapping.dataPrototypeInServiceInterfaceRef
[constr_10253]	Multiplicity of attribute SoftwareClusterDependencyCompareCondition.considerBuildNumber
[constr_10254]	Multiplicity of attribute SoftwareClusterDependencyCompareCondition.version
[constr_10255]	Multiplicity of attribute SignalServiceTranslationProps.serviceControl
[constr_10256]	Multiplicity of reference in the role SoftwarePackageStoring.storing
[constr_10365]	Existence of PersistencyDeployment.deploymentUri
[constr_10366]	Possible multiplicities of PersistencyDeployment.deploymentUri
[constr_10367]	Condition for the multiplicity of attribute PersistencyDeployment.deploymentUri
[constr_10374]	Existence of the attribute UcmSubordinateModuleInstantiation.verifyUpdate
[constr_10375]	Existence of the attribute UcmSubordinateModuleInstantiation.prepareUpdate
[constr_10376]	Existence of the attribute UcmSubordinateModuleInstantiation.prepareRollback
[constr_10377]	Completeness of the modeling of PersistencyKeyValueDataTypeMapping
[constr_10378]	PersistencyKeyValueDataTypeMapping references AbstractImplementationDataType in the role currentDataType
[constr_10379]	PersistencyKeyValueDataTypeMapping references ApplicationDataType in the role currentDataType
[constr_10380]	Target of ArtifactLocator.representedModelElement
[constr_10381]	Existence of attribute ArtifactLocator.uri
[constr_10382]	Existence of attribute ArtifactLocator.representedModelElement
[constr_10384]	PortInterface used for trigger state requests
[constr_10385]	PortInterface used for error state requests
[constr_10386]	Existence of references StateManagementStateMachineActionItem.start and stop
[constr_10387]	Consistency of StateManagementSetFunctionGroupStateActionItem.portPrototype and StateManagementSetFunctionGroupStateActionItem.setFunctionGroupState
[constr_10388]	Restriction for a PortInterface used for state switch notifications
[constr_10389]	Existence of attribute StateManagementFunctionGroupSwitchNotificationInterface.modeGroup
[constr_10390]	Existence of attribute StateManagementStateRequest.stateRequestPort
[constr_10391]	Existence of attribute StateManagementStateNotification.notificationPort
[constr_10392]	Existence of attribute StateManagementRequestRule.formula

▽

△

Number	Heading
[constr_10393]	Existence of reference in the role StateManagementRequestRule.nextState
[constr_10394]	Existence of attribute StateManagementCompareCondition.compareType
[constr_10395]	Existence of attribute StateManagementCompareCondition.compareValue
[constr_10396]	Existence of reference in the role StateManagementTriggerCompareRule. assumedCurrentState
[constr_10397]	Existence of reference in the role StateManagementSetFunctionGroupStateActionItem.portPrototype
[constr_10398]	Existence of reference in the role StateManagementSetFunctionGroupStateActionItem. setFunctionGroupState
[constr_10399]	Allowed interval of the "index" field according to the initialization rule for data object typed by a CppImplementationDataType of category VARIANT
[constr_10400]	Existence of SovdServerInstantiation.componentQualifier
[constr_10401]	Existence of SovdGatewayLocalEndpointTcpConfig.tcpPort
[constr_10402]	Existence of SovdGatewayEthernetCredentials.ipv4Address vs. SovdGatewayEthernetCredentials.ipv6Address
[constr_10403]	Existence of SovdGatewayEthernetCredentials.udpPort
[constr_10404]	Existence of SoftwareClusterSovdAddress.componentQualifier
[constr_10405]	Existence of reference in the role StateManagementActionList. affectedState
[constr_10406]	Existence of DeterministicSyncMOutOfN.numberOfConnectedClients
[constr_10407]	Existence of DeterministicSyncMOutOfN.minimumNumberOfRequests
[constr_10408]	Existence of DeterministicSyncMasterToTimeBaseConsumerMapping. deterministicSyncMaster
[constr_10409]	Existence of DeterministicSyncMasterToTimeBaseConsumerMapping. timeBaseConsumer
[constr_10410]	Value of SoftwareCluster.installationBehavior for a SoftwareCluster of category PLATFORM_CORE
[constr_10411]	Existence of ExecutionDependency and references to Function Group States

Table F.48: Added Constraints in R22-11

F.9.5 Changed Constraints in R22-11

Number	Heading
[constr_1543]	Only one physical address per SoftwareCluster. diagnosticDeploymentProps
[constr_1554]	Restriction regarding attribute PersistencyKeyValuePair.initValue
[constr_1785]	Restriction regarding the reference into another SoftwareCluster

▽

△

Number	Heading
[constr_3305]	Value of attribute SomeipEventDeployment.eventId shall be unique
[constr_3371]	Mutually exclusive existence of FireAndForgetMethodMapping.dataElement reference and FireAndForgetMethodMapping.trigger reference
[constr_3376]	FireAndForgetMethodMapping shall reference only fire and forget methods
[constr_3558]	RequiredSomeipServiceInstance.blocklistedVersion is restricted to the usage of minorVersion
[constr_3632]	Supervision of a Supervised Entity Instance in the scope of a Function Group State
[constr_3639]	Existence of SupervisionMode.expiredSupervisionTolerance
[constr_3676]	Exclusive usage of NoSupervision
[constr_5035]	Compatibility of data types with category STRING
[constr_5291]	Allowed usage of LTMessageCollectionToPortPrototypeMapping.rPortPrototype
[constr_10002]	Only one mapping per PortPrototype
[constr_10062]	DiagnosticServiceInstances that can be mapped by a DiagnosticServiceGenericMapping
[constr_10069]	Existence of SoftwareClusterDiagnosticDeploymentProps.powerDownTime
[constr_10090]	Existence of ProcessToMachineMapping.persistencyCentralStorageURI

Table F.49: Changed Constraints in R22-11

F.9.6 Deleted Constraints in R22-11

Number	Heading
[constr_1535]	Existence of DiagnosticSoftwareClusterProps in the context of a DiagnosticContributionSet
[constr_1557]	Standardized values of SoftwareClusterDesign.category and SoftwareCluster.category
[constr_1558]	Existence of SoftwareClusterDesign.diagnosticAddress
[constr_1559]	Existence of SoftwareClusterDesign.subSoftwareCluster
[constr_1562]	Existence of SoftwareClusterDesign.diagnosticContribution
[constr_1564]	Existence of SoftwareCluster.diagnosticProps.diagnosticAddress
[constr_1568]	Existence of SoftwareCluster.diagnosticExtract
[constr_1627]	Supported value range for attribute ApApplicationErrorDomain.value
[constr_1674]	Supported encoding of StdCppImplementationDataType of category STRING
[constr_1707]	Eligible subclasses of HeapUsage in the context of StateDependentStartupConfig.resourceConsumption

▽

△

Number	Heading
[constr_1732]	Existence of attribute `activationSwitch` set to `True` in the context of the enclosing `UcmStep`
[constr_1734]	Restriction for attribute `SoftwarePackageStep`.`activationSwitch`
[constr_5046]	Usage of `DoIpNetworkConfiguration`.`eidUseMac`
[constr_5280]	Existence of `DltLogSink`.`defaultLogThreshold`
[constr_10010]	Usage of attribute `category` in a SoftwareClusterDependencyFormula
[constr_10011]	Definition of sub-software-cluster

Table F.50: Deleted Constraints in R22-11

F.9.7 Added Advisories in R22-11

Number	Heading
[advisory_01000]	Existence of the reference `MethodMapping`.`clientServerOperation`
[advisory_01001]	Existence of reference `MethodMapping`.`method`
[advisory_01002]	Existence of reference `EventMapping`.`dataElement`
[advisory_01003]	Existence of reference `EventMapping`.`event`
[advisory_01004]	Existence of reference `FieldMapping`.`field`
[advisory_01005]	Existence of references `FieldMapping`.`getterOperation`, `setterOperation`, and `notifierDataElement`
[advisory_01006]	Existence of reference `FireAndForgetMethodMapping`.`method`
[advisory_01007]	Existence of references `FireAndForgetMethodMapping`.`dataElement` and `trigger`
[advisory_01008]	Multiplicity of reference in the role `RootSwClusterDesignComponentPrototype`.`applicationType`
[advisory_01009]	`PersistencyKeyValueDataTypeMapping`.`currentDataType` shall refer to a data type used in the context of the `PersistencyKeyValueStorageInterface`
[advisory_02000]	Define a `ServiceInstanceToSignalMapping` per member of an `AdaptivePlatformServiceInstance`
[advisory_02001]	No explicit `signal/service translation` for `ISignalGroup`s
[advisory_02002]	`ServiceInterface`.`field` mapping to `ISignalTriggering`s for primitive data type `field`s
[advisory_02003]	`ServiceInterface`.`field` mapping to `ISignalTriggering`s for composite data type `field`s

Table F.51: Added Advisories in R22-11

F.9.8 Changed Advisories in R22-11

none

F.9.9 Deleted Advisories in R22-11

none

F.10 Constraint and Specification Item History of this document according to AUTOSAR Release R23-11

F.10.1 Added Specification Items in R23-11

Number	Heading
[TPS_MANI_01411]	Initialization of a `DataPrototype` typed by a `StdCppImplementationDataType` of `category` `VALUE` that represents a Boolean
[TPS_MANI_01412]	Default value for startup timeout
[TPS_MANI_01413]	Semantics of `UploadableDeploymentElement`
[TPS_MANI_01414]	Semantics of `UploadableDesignElement`
[TPS_MANI_01415]	Semantics of attribute `DiagnosticMultipleResourcePortMapping.overrideId`
[TPS_MANI_01416]	Reporting the status of multiple `DiagnosticEvent`s in the context of one `PortInterface` on the *AUTOSAR adaptive platform*
[TPS_MANI_01417]	Retrieving the status of multiple `DiagnosticEvent`s to `PortPrototype`(s) on the *AUTOSAR adaptive platform*
[TPS_MANI_01418]	Mapping of multiple `DiagnosticCondition`s to a `PortPrototype` on the *AUTOSAR adaptive platform*
[TPS_MANI_01419]	Semantics of `NetworkManagementPortInterface`
[TPS_MANI_01420]	Support for the transport of bulk data in the context of SOVD
[TPS_MANI_01421]	Support for service validation in the context of SOVD
[TPS_MANI_01422]	Support for update procedures in the context of SOVD
[TPS_MANI_01423]	Support for configuration in the context of SOVD
[TPS_MANI_01424]	Semantics of `DiagnosticSovdProximityChallengePortMapping`
[TPS_MANI_01425]	Semantics of `DiagnosticSovdAuthorizationPortMapping`
[TPS_MANI_01426]	Mapping of `DiagnosticSovdBulkData` to `PortPrototype` on the *AUTOSAR adaptive platform*
[TPS_MANI_01427]	Mapping of `DiagnosticSovdUpdate` to `PortPrototype` on the *AUTOSAR adaptive platform*
[TPS_MANI_01428]	Mapping of `DiagnosticSovdServiceInstance` to `PortPrototype` on the *AUTOSAR adaptive platform*
[TPS_MANI_01429]	Mapping of `DiagnosticSovdConfiguration` to `PortPrototype` on the *AUTOSAR adaptive platform*
[TPS_MANI_01430]	Mapping of `DiagnosticSovdConfigurationParameter` to `DiagnosticDataIdentifier` on the *AUTOSAR adaptive platform*
[TPS_MANI_01431]	Definition of SOVD service instance

▽

△

Number	Heading
[TPS_MANI_01432]	Semantics of `DiagnosticSovdMethodPrimitive`
[TPS_MANI_01433]	Semantics of attribute `OsModuleInstantiation`. `osArtiAdapterLaunchBehavior`
[TPS_MANI_01434]	Diagnostic Address defined in the context of a `DiagnosticCommonProps`
[TPS_MANI_01435]	Semantics of meta-class `DiagnosticExternalAuthenticationIdentification`
[TPS_MANI_01436]	Relation of `StateManagementStateMachineActionItem` to internal state machine instances
[TPS_MANI_01437]	Standardized values of attribute `StateManagementStateNotification`. `stateMachine.category`
[TPS_MANI_01438]	Existence of `StateManagementStateMachineActionItem`. `overrideInitialState`
[TPS_MANI_01439]	Semantics of meta-class `IdsmTimestampProviderInterface`
[TPS_MANI_01440]	Using the `IdsmTimestampProviderMapping` for the provision of timestamps in the context of the *Intrusion Detection System Management*
[TPS_MANI_01441]	Using the `IdsmContextProviderMapping` for the provision of context in the context of the *Intrusion Detection System Management*
[TPS_MANI_01442]	Semantics of meta-class `IdsmContextProviderInterface`
[TPS_MANI_01443]	Semantics of meta-class `StateManagementNmActionItem`
[TPS_MANI_01444]	Semantics of attribute `StateManagementSleepActionItem.sleepTime`
[TPS_MANI_01445]	Semantics of meta-class `StateManagementSleepActionItem`
[TPS_MANI_01446]	Semantics of meta-class `SmInteractsWithNmMapping`
[TPS_MANI_01447]	Semantics of meta-class `NmInteractsWithSmMapping`
[TPS_MANI_03322]	Semantics of meta-class `ComTriggerGrant`
[TPS_MANI_03323]	Support for TLS in DoIP communication
[TPS_MANI_03324]	Semantics of `FunctionalClusterToSecurityEventDefinitionMapping`
[TPS_MANI_03325]	`DltLogSink` with `category` `DLT_LOGSINK_ARTI`

Table F.52: Added Specification Items in R23-11

F.10.2 Changed Specification Items in R23-11

Number	Heading
[TPS_MANI_01109]	Semantics of `UploadablePackageElement`
[TPS_MANI_01119]	Reference to model elements from `SoftwareClusterDesign`
[TPS_MANI_01137]	Possible use cases for the usage of `DataPrototypeInServiceInterfaceRef`
[TPS_MANI_01157]	Semantics of `updateStrategy` on collection level

▽

△

Number	Heading
[TPS_MANI_01159]	Semantics of `updateStrategy` on element level
[TPS_MANI_01191]	Modeling of possible errors
[TPS_MANI_01210]	Default encoding for all `DataPrototype`s typed by `CppImplementationDataType` of `category` `STRING`
[TPS_MANI_01239]	Semantics of meta-class `ComEventGrant`
[TPS_MANI_01274]	Standardized System Category Definitions for the Adaptive Platform
[TPS_MANI_01347]	Definition of a `DiagnosticDataElement` used in the context of a DID obtained from a `PPortPrototype` typed by a `DiagnosticDataElementInterface`
[TPS_MANI_01348]	Definition of a `DiagnosticDataElement` used in the context of a DID obtained from a `PPortPrototype` typed by a `DiagnosticDataIdentifierInterface`
[TPS_MANI_01362]	Convey the Authentication state of a diagnostic client to the diagnostic server instance
[TPS_MANI_01397]	Standardized values of `DiagnosticDataIdentifier.category` of the usage in SOVD context
[TPS_MANI_01405]	Semantics of sub-classes of `SoftwareClusterDiagnosticAddress`
[TPS_MANI_03101]	SOME/IP serialization
[TPS_MANI_03102]	UserDefined serialization
[TPS_MANI_03103]	Default size for all array and map 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 `DataPrototype`s
[TPS_MANI_03108]	Default Byte Order for all `DataPrototype`s
[TPS_MANI_03117]	Default size for all string length fields
[TPS_MANI_03283]	Standardized values for attribute `DltLogSink.category`
[TPS_MANI_03299]	Semantics of `FirewallRuleProps`
[TPS_MANI_03546]	Definition of reported health status `RPortPrototype`
[TPS_MANI_03628]	Standardized values of `ServiceInterface.category`

Table F.53: Changed Specification Items in R23-11

F.10.3 Deleted Specification Items in R23-11

Number	Heading
[TPS_MANI_01111]	Diagnostic Address of a `SoftwareCluster`
[TPS_MANI_01199]	Semantics of `DeterministicClientResourceNeeds`
[TPS_MANI_01200]	Semantics of meta-class `DeterministicClientResource`

▽

△

Number	Heading
[TPS_MANI_01203]	Semantics of DeterministicClient
[TPS_MANI_01233]	Semantics of meta-class ComFindServiceGrantDesign
[TPS_MANI_01241]	Semantics of meta-class ComFindServiceGrant
[TPS_MANI_01330]	Definition of machine Function Group
[TPS_MANI_01406]	Semantics of DeterministicSyncMaster
[TPS_MANI_01407]	Semantics of abstract base class DeterministicSyncVerificationPolicy
[TPS_MANI_01408]	Semantics of reference DeterministicClient. deterministicSyncMaster
[TPS_MANI_01409]	Semantics of meta-class DeterministicSyncMasterToTimeBaseConsumerMapping
[TPS_MANI_03125]	ServiceInterface.method to ISignalTriggerings mapping
[TPS_MANI_03235]	Usage of ApSomeipTransformationProps.sessionHandling
[TPS_MANI_03288]	ApSomeipTransformationProps for triggers

Table F.54: Deleted Specification Items in R23-11

F.10.4 Added Constraints in R23-11

Number	Heading
[constr_5366]	Allowed target of ComEventGrant.serviceDeployment reference
[constr_5367]	Allowed target of ComTriggerGrant.serviceDeployment reference
[constr_5368]	Multiplicity of the reference in the role ComTriggerGrant.serviceDeployment
[constr_5372]	SecureComProps for a PlatformModuleEthernetEndpointConfiguration that contains a UDP configuration
[constr_5373]	SecureComProps for a PlatformModuleEthernetEndpointConfiguration that contains a TCP configuration
[constr_5381]	Modeling of Security Event reports by FunctionalCluster shall not be done via ProcessToMachineMapping
[constr_5386]	Executable.traceSwitchConfiguration
[constr_5387]	Existence of Executable.traceSwitchConfiguration.traceMessage
[constr_5388]	Existence of Executable.traceSwitchConfiguration.traceSwitch
[constr_5392]	Assignment of the same event to several SomeipEventGroups is forbidden in case one of the SomeipEventGroups has the multicastThreshold set to to a value greater than 1
[constr_6905]	CppTemplateArgument.allocator
[constr_10416]	Aggregation of optional templateArgument that defines an Allocator for a VECTOR
[constr_10417]	Existence of attributes of CppImplementationDataTypes depending on the category

▽

△

Number	Heading
[constr_10420]	Restriction for the existence of initial values for PersistencyDataElement
[constr_10425]	Existence of initial values for PersistencyDataElement
[constr_10426]	Multiplicity of attribute PersistencyDeploymentElement.updateStrategy
[constr_10427]	Multiplicity of attribute PersistencyInterfaceElement.updateStrategy
[constr_10428]	Existence of attribute UcmModuleInstantiation.identifier in subclasses
[constr_10429]	Existence of attribute VehicleRolloutStep. violatedSafetyConditionBehavior
[constr_10430]	Existence of attribute Machine.defaultApplicationTimeout
[constr_10431]	Existence of attribute Machine.defaultApplicationTimeout. enterTimeoutValue
[constr_10432]	Existence of attribute Machine.defaultApplicationTimeout. exitTimeoutValue
[constr_10436]	Restriction for the applicability of DiagnosticMultipleMonitorPortMapping
[constr_10437]	Restriction for the applicability of DiagnosticMultipleEventPortMapping
[constr_10438]	Restriction for the applicability of DiagnosticMultipleConditionPortMapping
[constr_10441]	Restriction for NetworkHandlePortMapping.pPortPrototypeInExecutable
[constr_10442]	Restriction for the applicability of DiagnosticSovdProximityChallengePortMapping
[constr_10443]	Restriction for the applicability of DiagnosticSovdProximityChallengeInterface
[constr_10444]	Existence of DiagnosticSovdProximityChallengePortMapping
[constr_10445]	Existence of the reference in the role DiagnosticSovdProximityChallengePortMapping. pPortPrototypeInExecutable
[constr_10446]	Existence of the reference in the role DiagnosticSovdProximityChallengePortMapping.process
[constr_10447]	Restriction for the applicability of DiagnosticSovdAuthorizationPortMapping
[constr_10448]	Restriction for the applicability of DiagnosticSovdAuthorizationInterface
[constr_10449]	Existence of DiagnosticSovdAuthorizationPortMapping
[constr_10450]	Existence of the reference in the role DiagnosticSovdAuthorizationPortMapping. pPortPrototypeInExecutable
[constr_10451]	Existence of the reference in the role DiagnosticSovdAuthorizationPortMapping.process
[constr_10452]	Restriction for the applicability of DiagnosticSovdBulkDataPortMapping
[constr_10453]	Restriction for the applicability of DiagnosticSovdBulkDataInterface
[constr_10454]	Uniqueness of reference from DiagnosticSovdBulkDataPortMapping to DiagnosticSovdBulkData
[constr_10455]	Existence of the reference in the role DiagnosticSovdBulkDataPortMapping. serviceInstance

▽

△

Number	Heading
[constr_10456]	Existence of the reference in the role DiagnosticSovdBulkDataPortMapping.pPortPrototypeInExecutable
[constr_10457]	Existence of the reference in the role DiagnosticSovdBulkDataPortMapping.process
[constr_10458]	Restriction for the applicability of DiagnosticSovdUpdatePortMapping
[constr_10459]	Restriction for the applicability of DiagnosticSovdUpdateInterface
[constr_10460]	Uniqueness of reference from DiagnosticSovdUpdatePortMapping to DiagnosticSovdUpdate
[constr_10461]	Existence of the reference in the role DiagnosticSovdUpdatePortMapping.serviceInstance
[constr_10462]	Existence of the reference in the role DiagnosticSovdUpdatePortMapping.pPortPrototypeInExecutable
[constr_10463]	Existence of the reference in the role DiagnosticSovdUpdatePortMapping.process
[constr_10464]	Restriction for the applicability of DiagnosticSovdServiceValidationPortMapping
[constr_10465]	Restriction for the applicability of DiagnosticSovdServiceValidationInterface
[constr_10466]	Existence of the reference in the role DiagnosticSovdServiceValidationPortMapping.pPortPrototypeInExecutable
[constr_10467]	Existence of the reference in the role DiagnosticSovdServiceValidationPortMapping.process
[constr_10468]	Restriction for the applicability of DiagnosticSovdConfigurationPortMapping
[constr_10469]	Restriction for the applicability of DiagnosticSovdConfigurationInterface
[constr_10470]	Uniqueness of reference from DiagnosticSovdConfigurationPortMapping to DiagnosticSovdConfiguration
[constr_10471]	Existence of the reference in the role DiagnosticSovdConfigurationPortMapping.serviceInstance
[constr_10472]	Existence of the reference in the role DiagnosticSovdConfigurationPortMapping.pPortPrototypeInExecutable
[constr_10473]	Existence of the reference in the role DiagnosticSovdConfigurationPortMapping.process
[constr_10474]	Uniqueness of reference from DiagnosticSovdConfigurationDataIdentifierMapping to DiagnosticSovdConfigurationParameter
[constr_10475]	Existence of the reference in the role DiagnosticSovdConfigurationDataIdentifierMapping.serviceInstance
[constr_10476]	Existence of the reference in the role DiagnosticSovdConfigurationDataIdentifierMapping.dataIdentifier
[constr_10477]	Existence of DiagnosticSovdLog

▽

△

Number	Heading
[constr_10478]	Existence of DiagnosticSovdUpdate
[constr_10479]	Restriction on values of attribute DiagnosticSovdMethodPrimitive.category aggregated in the role put in the context of SOVD Update
[constr_10480]	Restriction on value of attribute DiagnosticSovdMethodPrimitive.category aggregated in the role get in the context of SOVD Update
[constr_10481]	Restriction on values of attribute DiagnosticSovdMethodPrimitive.category aggregated in the role get in the context of SOVD Log
[constr_10482]	Restriction on values of attribute DiagnosticSovdMethodPrimitive.category aggregated in the roles put and delete in the context of SOVD Log
[constr_10483]	Applicable values of baseTypeEncoding in the context of the definition of a DiagnosticDataElement
[constr_10484]	Existence of FunctionalClusterInteractsWithPersistencyDeploymentMapping.contractVersion
[constr_10485]	Existence of PersistencyInterface.contractVersion
[constr_10486]	Existence of PersistencyKeyValueDataTypeMapping.previousContractVersion
[constr_10487]	Only one physical address per DiagnosticCommonProps
[constr_10488]	Existence of attribute DiagnosticCommonProps.authenticationTimeout
[constr_10489]	Existence of StateManagementModuleInstantiation in the context of a Machine
[constr_10490]	DiagnosticDataElement shall be directly or indirectly the target of a reference owned by a subclass of DiagnosticMapping
[constr_10491]	Type of PPortPrototype referenced in the role IdsmTimestampProviderMapping.pPortPrototypeInExecutable
[constr_10492]	PPortPrototype typed by a IdsmTimestampProviderInterface
[constr_10493]	Existence of IdsmTimestampProviderMapping.idsPlatformInstantiation
[constr_10494]	Existence of IdsmTimestampProviderMapping.pPortPrototypeInExecutable
[constr_10495]	Existence of IdsmTimestampProviderMapping.process
[constr_10496]	Type of PPortPrototype referenced in the role IdsmContextProviderMapping.pPortPrototypeInExecutable
[constr_10497]	PPortPrototype typed by a IdsmContextProviderInterface
[constr_10498]	Existence of IdsmContextProviderMapping.idsPlatformInstantiation
[constr_10499]	Existence of IdsmContextProviderMapping.pPortPrototypeInExecutable
[constr_10500]	Existence of IdsmContextProviderMapping.process
[constr_10501]	Access to time stamps for *Intrusion Detection System Management*
[constr_10508]	Existence of the attribute UcmSubordinateModuleInstantiation.maxAvailablePersistencyStorageSpace
[constr_10509]	Existence of attribute PersistencyDeployment.maximumAllowedSize
[constr_10510]	Existence of attribute PersistencyDeployment.minimumSustainedSize

▽

△

Number	Heading
[constr_10511]	Reference StateManagementStateRequest.stateRequestPort shall not exist if StateManagementStateRequest is referenced from NmInteractsWithSmMapping
[constr_10512]	Restriction for the value of StateManagementSleepActionItem.sleepTime
[constr_10513]	Existence of the reference in the role SmInteractsWithNmMapping. actionItem
[constr_10514]	Existence of the reference in the role SmInteractsWithNmMapping. nmNetworkHandle
[constr_10515]	Existence of the reference in the role NmInteractsWithSmMapping. stateRequest
[constr_10516]	Existence of the reference in the role NmInteractsWithSmMapping. nmNetworkHandle
[constr_11000]	SomeipEventDeployment.eventReceptionDefaultValue shall not be specified in the physical domain

Table F.55: Added Constraints in R23-11

F.10.5 Changed Constraints in R23-11

Number	Heading
[constr_1675]	Existence of attribute ApSomeipTransformationProps.stringEncoding
[constr_3395]	TransformationPropsToServiceInterfaceElementMapping is restricted to one single ServiceInterface
[constr_3434]	Aggregation of templateArguments for a VECTOR
[constr_3550]	Existence of ServiceInstanceToSignalMapping for an event with signalBased serialization
[constr_3551]	Full mapping of target ISignalGroup
[constr_3552]	Full mapping of target event
[constr_3553]	Existence of ServiceInstanceToSignalMapping for an field with signalBased serialization
[constr_3555]	No support for SecuredIPdu.useAsCryptographicIPdu set to true
[constr_3636]	Consistent ISignal communication direction in and RequiredApServiceInstance
[constr_3637]	Consistent ISignal communication direction out and ProvidedApServiceInstance
[constr_5052]	ProvidedSomeipServiceInstances of the same serviceInterface on one Machine
[constr_5318]	Existence of ServiceInstanceToSignalMapping for an trigger with signalBased serialization
[constr_5351]	FirewallRule is allowed to aggregate at most one protocol subelement
[constr_5352]	DdsRule.submessageType value restriction

▽

△

Number	Heading
[constr_5353]	DdsRule.readerEntityId and DdsRule.writerEntityId value restriction
[constr_5358]	AdaptiveFirewallToPortPrototypeMapping.rPortPrototype restriction
[constr_10048]	Existence of reference from DiagnosticMonitorPortMapping to DiagnosticEvent
[constr_10148]	Multiplicity of reference in the role PersistencyFileElement.contentUri
[constr_10386]	Existence of references StateManagementStateMachineActionItem. startStateMachine and stopStateMachine
[constr_10390]	Existence of reference StateManagementStateRequest.stateRequestPort

Table F.56: Changed Constraints in R23-11

F.10.6 Deleted Constraints in R23-11

Number	Heading
[constr_1543]	Only one physical address per SoftwareCluster. diagnosticDeploymentProps
[constr_1572]	Usage of SwDataDefProps.implementationDataType within a CppImplementationDataType
[constr_1614]	Existence of attribute TransformationPropsToServiceInterfaceElementMapping. transformationProps.sessionHandling
[constr_1789]	Scope of machine Function Group
[constr_3394]	Default value for start-up timeout on the Machine is not configurable
[constr_3733]	Upper multiplicity of aggregation in the role ServiceInstanceToSignalMapping.methodMapping
[constr_10095]	Existence of reference from DiagnosticExternalAuthenticationPortMapping to DiagnosticAuthentication
[constr_10111]	Existence of attribute DiagnosticAuthentication.authenticationTimeout
[constr_10156]	Multiplicity of reference in the role ComFindServiceGrantDesign. requiredServicePort
[constr_10229]	Multiplicity of reference in the role SignalBasedMethodToISignalTriggeringMapping.method
[constr_10406]	Existence of DeterministicSyncMOutOfN.numberOfConnectedClients
[constr_10407]	Existence of DeterministicSyncMOutOfN.minimumNumberOfRequests
[constr_10408]	Existence of DeterministicSyncMasterToTimeBaseConsumerMapping. deterministicSyncMaster
[constr_10409]	Existence of DeterministicSyncMasterToTimeBaseConsumerMapping. timeBaseConsumer

Table F.57: Deleted Constraints in R23-11

F.10.7 Added Advisories in R23-11

none

F.10.8 Changed Advisories in R23-11

Number	Heading
[advisory_02000]	Define a `ServiceInstanceToSignalMapping` per member of an `AdaptivePlatformServiceInstance`
[advisory_02001]	No explicit `signal/service translation` for `ISignalGroup`s
[advisory_02002]	`ServiceInterface.field` mapping to `ISignalTriggering`s for primitive data type `field`s
[advisory_02003]	`ServiceInterface.field` mapping to `ISignalTriggering`s for composite data type `field`s

Table F.58: Changed Advisories in R23-11

F.10.9 Deleted Advisories in R23-11

none

G Splitable Elements in the Scope of this Document

This chapter contains a table of all model elements stereotyped ≪atpSplitable≫ in the scope of this document.

Each entry in the table consists of the identification of the specific model element itself and the applicable value of the tagged value atp.Splitkey.

For more information about the concept of splitable model elements and how these shall be treated please refer to [6].

Name of splitable element	*Splitkey*
AdaptiveApplicationSwComponentType.internalBehavior	internalBehavior.shortName, internal Behavior.variationPoint.shortLabel
CryptoModuleInstantiation.certificateToKeySlotMapping	certificateToKeySlotMapping
CryptoModuleInstantiation.cryptoCertificate	cryptoCertificate.shortName
CryptoModuleInstantiation.cryptoProvider	cryptoProvider.shortName
CryptoProvider.keySlot	keySlot.shortName
DiagnosticAuthenticationPortMapping.process	process
DiagnosticClearConditionGroup.clearCondition	clearCondition
DiagnosticClearConditionPortMapping.process	process
DiagnosticDataPortMapping.process	process
DiagnosticExternalAuthenticationPortMapping.process	process
DiagnosticIndicatorPortMapping.process	process
DiagnosticMemoryDestinationPortMapping.process	process
DiagnosticMonitorPortMapping.process	process
DiagnosticMultipleConditionPortMapping.process	process
DiagnosticMultipleEventPortMapping.process	process
DiagnosticMultipleMonitorPortMapping.process	process
DiagnosticSecurityLevelPortMapping.process	process
DiagnosticServiceGenericMapping.process	process
DiagnosticServiceValidationMapping.process	process
DiagnosticSovdAuthorizationPortMapping.process	process
DiagnosticSovdBulkDataPortMapping.process	process
DiagnosticSovdConfigurationPortMapping.process	process
DiagnosticSovdProximityChallengePortMapping.process	process
DiagnosticSovdUpdatePortMapping.process	process
GlobalSupervision.supervisionMode	supervisionMode.shortName
IamModuleInstantiation.grant	grant
IdsmModuleInstantiation.reportableSecurityEvent	reportableSecurityEvent
IdsPlatformInstantiation.timeBase	timeBase.timeBaseResource, timeBase.variation Point.shortLabel
Machine.environmentVariable	environmentVariable
Machine.moduleInstantiation	moduleInstantiation.shortName
Machine.secureCommunicationDeployment	secureCommunicationDeployment.shortName
MachineDesign.communicationConnector	communicationConnector.shortName
MachineDesign.communicationController	communicationController.shortName
MachineDesign.serviceDiscoveryConfig	serviceDiscoveryConfig

△

Name of splitable element	*Splitkey*
NmNetworkHandle.partialNetwork	partialNetwork
PlatformHealthManagementContribution.checkpoint	checkpoint.shortName
PlatformHealthManagementContribution.globalSupervision	globalSupervision.shortName
PlatformHealthManagementContribution.healthChannel	healthChannel.shortName
PlatformHealthManagementContribution.supervisionModeCondition	supervisionModeCondition.shortName
Process.securityEvent	securityEvent
ServiceInstanceToPortPrototypeMapping.process	process
ServiceInterface.event	event.shortName, event.variationPoint.shortLabel
ServiceInterface.field	field.shortName, field.variationPoint.shortLabel
ServiceInterface.method	method.shortName, method.variationPoint.short Label
ServiceInterface.trigger	trigger.shortName, trigger.variationPoint.shortLabel
ServiceInterfaceDeployment.eventDeployment	eventDeployment.shortName, event Deployment.variationPoint.shortLabel
ServiceInterfaceDeployment.fieldDeployment	fieldDeployment.shortName, field Deployment.variationPoint.shortLabel
ServiceInterfaceDeployment.methodDeployment	methodDeployment.shortName, method Deployment.variationPoint.shortLabel
SoftwareCluster.artifactChecksum	artifactChecksum.shortName, artifactChecksum.uri
SoftwareCluster.conflictsTo	conflictsTo
SoftwareCluster.containedARElement	containedARElement
SoftwareCluster.containedPackageElement	containedPackageElement
SoftwareCluster.dependsOn	dependsOn
SoftwareCluster.moduleInstantiation	moduleInstantiation
SoftwareClusterDesign.containedProcess	containedProcess
SoftwareClusterDesign.diagnosticContribution	diagnosticContribution
SoftwareClusterDesign.requiredARElement	requiredARElement
SoftwareClusterDesign.requiredDesignElement	requiredDesignElement
SoftwareClusterDesign.requiredFibexElement	requiredFibexElement
SomeipDataPrototypeTransformationProps.networkRepresentation	networkRepresentation
StateManagementModuleInstantiation.actionItemList	actionItemList.shortName
StateManagementModuleInstantiation.notification	notification.shortName
StateManagementModuleInstantiation.request	request.shortName

Table G.1: Usage of splitable elements

H Variation Points in the Scope of this Document

This chapter contains a table of all model elements stereotyped ≪atpVariation≫ in the scope of this document.

Each entry in the table consists of the identification of the model element itself and the applicable value of the tagged value vh.latestBindingTime.

For more information about the concept of variation points and how model elements that contain variation points shall be treated please refer to [6].

Variation Point	*Latest Binding Time*
AdaptiveApplicationSwComponentType.internalBehavior	preCompileTime
CppImplementationDataType.arraySize	preCompileTime
IdsPlatformInstantiation.timeBase	systemDesignTime
ServiceInterface.event	blueprintDerivationTime
ServiceInterface.field	blueprintDerivationTime
ServiceInterface.method	blueprintDerivationTime
ServiceInterface.trigger	blueprintDerivationTime
ServiceInterfaceDeployment.eventDeployment	blueprintDerivationTime
ServiceInterfaceDeployment.fieldDeployment	blueprintDerivationTime
ServiceInterfaceDeployment.methodDeployment	blueprintDerivationTime

Table H.1: Usage of variation points