

Document Title	Application Interfaces User Guide
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	442
Document Classification	Auxiliary

Document Version	1.1.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History			
Date	Version	Changed by	Change Description
09.12.2011	1.1.0	AUTOSAR Administration	<ul style="list-style-type: none"> • Description of Categories of model elements created • Synchronization of Update of XML package structure especially regarding Port Blueprints • Synchronization to updates of AUTOSAR meta model • Description of Naming conventions for connectors
11.04.2011	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

This specification 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 specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification 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 specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Advice for users

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1	Purpose of this document.....	5
1.1	Document Overview	5
2	Introduction to Application Interfaces Table.....	6
2.1	Structural overview of Domains in AI Table.....	8
3	AUTOSAR Methodology.....	9
3.1	Overview on available documents.....	10
3.1.1	Software Component Template [1].....	10
3.1.2	Standardization Template [2]	10
3.1.3	Generic Structure Template [4]	10
3.1.4	AI Specification [6]	10
3.1.5	Modeling Guide for Application Interfaces [9].....	10
3.1.6	AUTOSAR Methodology [10]	10
3.1.7	Explanation of Application Interfaces for Domain Body Comfort [11]..	11
3.1.8	Explanation of Application Interfaces for Domain Powertrain [12].....	11
3.1.9	Explanation of Application Interfaces for Domain Chassis [13].....	11
3.1.10	Explanation of Application Interfaces for Domain Occupant and Pedestrian Safety [14]	11
3.1.11	Explanation of Application Interfaces for Domain Multimedia, Telematics, Human Machine Interface [15]	11
4	Metamodel representation of AI Table.....	12
4.1	Category of Model Elements	12
4.1.1	STANDARD	12
4.1.2	BLUEPRINT	12
4.1.3	EXAMPLE	13
4.2	Meta model diagrams and the AI Table.....	13
4.2.1	Composition	13
4.2.1.1	Multiple Instantiation	15
4.2.2	Blueprint Mapping & BlueprintMappingSet.....	18
4.2.3	PortPrototypes	19
4.2.3.1	PortPrototypeBlueprints	19
4.2.3.2	BlueprintMapping of PortPrototypeBlueprints.....	20
4.2.4	PortInterfaces.....	20
4.2.4.1	Sender Receiver Communication	21
4.2.4.2	Client Server Communication	23
4.2.5	DataTypes.....	24
4.2.5.1	ApplicationDataTypes.....	25
4.2.5.1.1	Application Primitive Data Types	26
4.2.5.1.2	Application Composite Data Types	27
4.2.5.1.3	ApplicationArrayDataType	28
4.2.5.1.4	ApplicationRecordDataType	29
4.2.6	Physical Units.....	30
4.2.7	Computation Methods	31
5	Structure of Application Interfaces (AI) Table	32
5.1	Main sheets of the AI Table.....	32
5.1.1	Sheet 04_Keywords	32

5.1.2	Sheet 05_TopLevel	33
5.1.3	Sheets 050xxxxx	35
5.1.4	Sheet 06_Interfaces_DataElements (Sender-Receiver-Interface)	37
5.1.5	Sheet 06_Interface_ClientServer	39
5.1.6	Sheet 07_DataTypes_ContinuousValue	41
5.1.7	Sheet 08_DataTypes_Enumeration	41
5.1.8	Sheet 09_DataTypes_Array	43
5.1.9	Sheet 11_DataTypes_Record	44
5.1.10	Sheet 13_Units.....	46
5.1.11	Sheet 15_Redirected_Ports	47
5.2	Complete List of all Sheets of the AI Table	48
6	Relationship between AI Table data and XML Output	51
6.1	Overview	51
6.1.1	Dependencies of XML Generation	51
6.1.2	Contents of Generated XML	51
6.1.2.1	File Distribution	52
6.1.3	Schema Structure	52
6.2	Common Elements.....	54
6.2.1	Package Structure.....	54
6.2.2	References.....	58
6.2.3	Instance References	59
6.2.4	Type References.....	59
6.2.5	Descriptions	59
6.3	Component Types	61
6.3.1	Composition Types	62
6.3.2	Ports.....	63
6.3.3	Components.....	63
6.3.4	Connectors.....	64
6.4	PortPrototypeBlueprints	66
6.5	PortInterfaces	67
6.5.1	Sender-Receiver-Interface	67
6.5.2	Client-Server-Interface	68
6.6	PortInterface Blueprints.....	70
6.7	Data Types.....	70
6.7.1	Continuous Value Types	71
6.7.2	Enumeration Types	73
6.7.3	Array Types.....	75
6.7.4	Record Types.....	76
6.8	Units	77
7	References	79
7.1	Standard documents	79
7.2	Auxiliary documents	79

1 Purpose of this document

AUTOSAR aims at the delivery of functionality through communicating Software-Components, which can be placed nearly arbitrarily on a network of ECUs. To ensure the interoperability of Software-Components from different sources (i.e. vendors), the interfaces of these should be unified.

The content of the AI Table [8] is the specification of interfaces of several automotive domains. The composition of these domains will establish the Top level domain inside the table. The goal is to define and publish stable and widely accepted application interfaces.

This document aims at explaining all relevant details about the AI Table especially for users, who have to maintain the standardized application interfaces. Experienced users can skip the chapter AUTOSAR Methodology. Some sections contain extract from other AUTOSAR documents. In case of differences in the contents then the original AUTOSAR documents are valid.

1.1 Document Overview

This document gives an overview of the methodological background of the application interfaces. This document also gives an overview of the content of the 'Application Interface Table', the top level (inter-domain level) and included domains (Body, Powertrain, Chassis, Occupant and Pedestrian Safety, Multimedia, Telematics, Human Machine Interface). It also describes the structure of the AI Table (realized in an Excel table) and explains how to handle it.

Abbreviations List

Abbreviation	Meaning
.arxml	Autosar Extensible Markup Language File
AI Table	Application Interface Table
Bugzilla	Tool for change request management
CPU	Central Processing Unit
ECU	Electronic Control Unit
Excel	Microsoft spreadsheet-application
MS	Milestone
RTE	Run-Time Environment
SPEM	Software Process Engineering meta-model
SVN	Subversion (version control system)
SW-C	Software Component
SWC	Software Component
SW	Software
VB	Visual Basic
VFB	Virtual Function Bus
WP	Work package
XML	Extensible Markup Language
XSD	XML Schema Definition
HMI	Human Machine Interface

2 Introduction to Application Interfaces Table

The application interface table (AI Table) is the user interface dedicated to manage all the data which define the application interfaces (see Figure 1). This is implemented in a tool (Excel) with validation and work product output generation macros (e.g. VB scripts). Input to the tool is based on AUTOSAR defined methodology meta-model data. The output of the tool is a XML model, which is conform to the XSD and follows the semantics defined in the template. The AUTOSAR XML file (called .ARXML file) contains detailed information in a structured format of all the standardized application interfaces data. The xml file contains the definitions of the application interfaces transferred to a commonly readable format, which can be the input for e.g. authoring tools at the development units of SW developing companies.

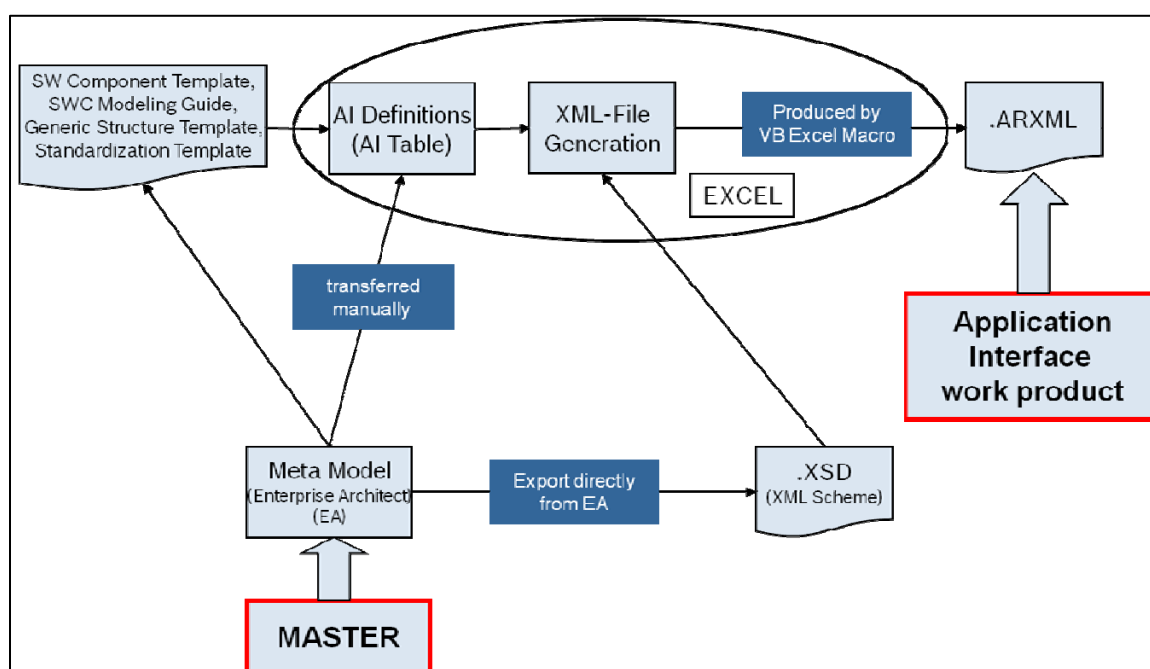


Figure 1: The AI Table Process

The AI Table enables the manipulation of AI definitions in order to produce the outcome: the xml file for data exchange.

The SWC template [1] tells us what can be modeled. The Standardization Template [2] explains and supports the blueprint approach.

The AI Table description (i.e. XML) tells us what is being modeled. The AI Table definitions follow the modeling guide defined by SWC modeling guide [9]. The following Figure 2 shows the main structure of SW composition and their decomposition into components.

In order to standardize application interfaces a number of SW components are described within the AI Table decomposed to the domains and their main functions. Nevertheless, these components must be seen as examples only (at present for Release 4.0 only). They are not part of the standard; but they are necessary in order to specify the port / port prototypes in a consistent way. Each port / port prototype needs a connection to a component to be specified in a proper way.

The structure within Figure 2 illustrates the decomposition of Software-Compositions using the AUTOSAR Software Component Template [1].

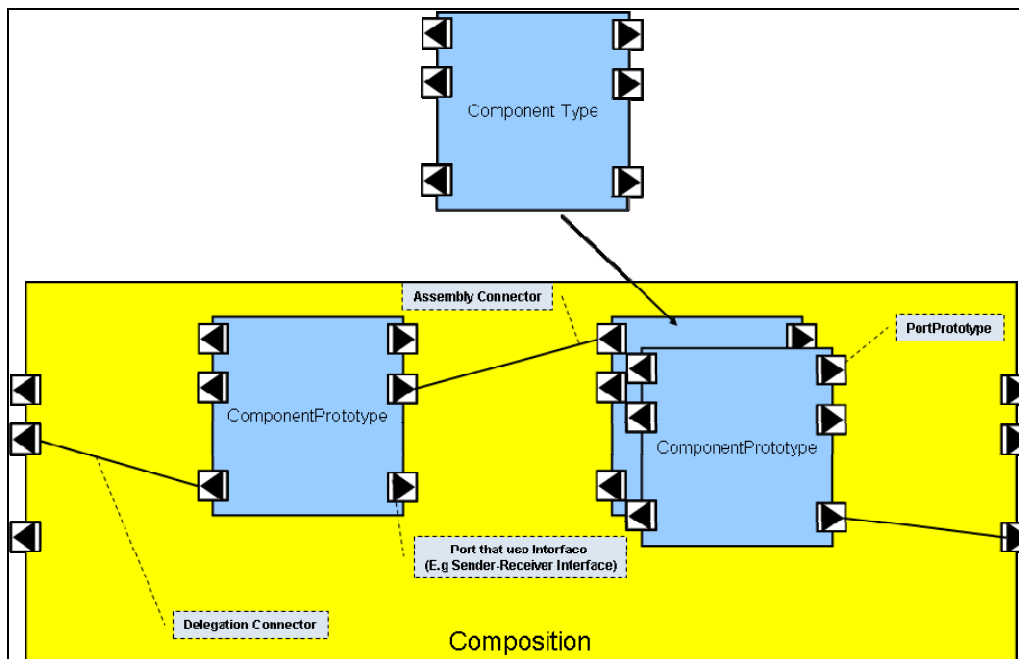


Figure 2: Decomposition of a component using the concepts defined in the software component template

Note: The yellow blocks in the figure represent the AI Table columns with yellow color, which are the composition that is decomposed into other compositions / components. These component types and prototypes are described within the same AI Table sheet in blue colored columns (see blue blocks in the figure). A SwComponentPrototype implements the usage of a SwComponentType in a specific role.

SwComponentPrototypes are only used for implementing SwComponentTypes in a specific role, i.e. they are used to instantiate the SwComponentType.

Example: a SwComponentPrototype "LeftDoorControl" fulfills the role of implementing the SwComponentType "DoorControl" for the left door of a vehicle while the SwComponentPrototype "RightDoorControl" fulfills the role of the SwComponentType "DoorControl" for the right door.

The AI Table is an Excel table containing a number of work sheets. Within the sheets the following main information that are application interface relevant are handled:

- Compositions; main compositions are from domains (1) Body, (2) Powertrain, (3) Chassis, (4) Occupant and Pedestrian Safety, (5) multimedia and telematics and human machine interface
- Components
- Ports
- Port interfaces and its VariableDataPrototypes
- Data types for VariableDataPrototypes
- Units
- Instances of component types

- Keywords

For a detailed list of sheets provided by the table refer to Chapter 5.

2.1 Structural overview of Domains in AI Table

Currently the AI Table contains the specification of a number of different automotive domains. The outcome of each domain including inter-domain connections can be identified within the sheets of the AI Table:

- Interdomain level (Top level) Sheet: 0500
- Body Sheet: 0501*
- Powertrain Sheet: 0502*
- Chassis Sheet: 0503*
- Occupant and pedestrian safety Sheet: 0504*
- Multimedia, Telematics, Human Machine Interface (HMI) Sheet: 0505*

Although the table is structured following the domains, the resulting decomposition into components/compositions is not a mandatory architecture for AUTOSAR compliant vehicle architectures. The AI Table shows components/compositions as examples for explanation of standardized ports and port interfaces. The top level composition is a dummy composition required to represent the inter domain ports in the VFB view.

Further explanations on domain details can be found in chapter 3.1.7 to 3.1.11 and further referenced documents.

3 AUTOSAR Methodology

AUTOSAR requires a formal technical approach for some steps of system development. This approach is called the “AUTOSAR Methodology”. The AUTOSAR Methodology is neither a complete process description nor a business model and “roles” and “responsibilities” are not defined in this methodology. Furthermore, it does not prescribe a precise order in which activities should be carried out. The methodology is a mere work-product flow: it defines the dependencies of activities on work-products.

During system design, the software components and the hardware have to be selected, and overall system constraints have to be identified. AUTOSAR intends to ease the formal description of these initial system design decisions via the information exchange format and the use of templates. So defining the System Configuration Input means to fill out or edit the appropriate templates. AUTOSAR methodology allows for a high degree of reuse in this context. In any case, this editing is assumed to be supported by editing tools. Here is a brief description of activities:

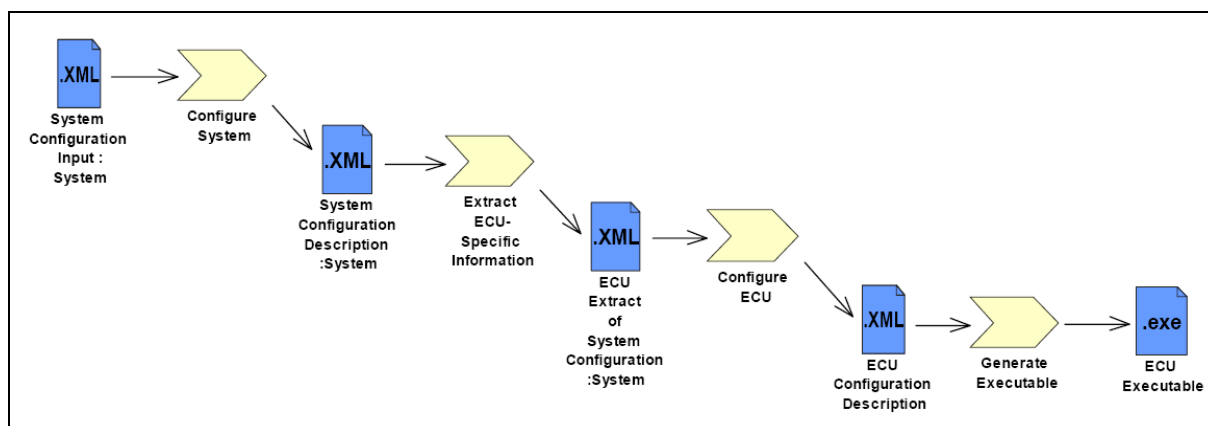


Figure 3: Overview AUTOSAR Methodology

- **Configure System:** mainly maps the software components to the ECUs with regard to resources and timing requirements.

The SW-component description, system constraints description and ECU resources description are required to configure the system.

The AI Table output along the internal behavior defines the SW-component description.

The output of this activity is the System Configuration Description. This description includes all system information (e.g. bus mapping, topology) and the mapping of which software component is located on which ECU.

- **Extract ECU-Specific Information:** extracts the information from the System Configuration Description needed for a specific ECU. This is then placed in the ECU Extract of System Configuration.

- **Configure ECU:** adds all necessary information for implementation like task scheduling, necessary Basic Software modules, configuration of the Basic Software, assignment of runnable entities to tasks, etc. The result of the activity Configure ECU

is included in the ECU Configuration Description, which collects all information that is local to a specific ECU. The runnable software to this specific ECU can be built from this information.

- **Generate Executable:** an executable is generated based on the configuration of the ECU described in the ECU Configuration Description. This step typically involves generating code (e.g. for the RTE and the Basic Software), compiling code (compiling generated code or compiling software-components available as source-code) and linking everything together into an executable.

Nevertheless, the implementation of a software component is more or less independent from ECU configuration.

The general concepts of this chapter are an extract of the detailed AUTOSAR Methodology [10].

3.1 Overview on available documents

For detailed information, following documents are available.

3.1.1 Software Component Template [1]

This document provides introductory description and rationale for the part of the AUTOSAR meta-model relevant for the definition of Software Components.

3.1.2 Standardization Template [2]

This document is intended to support the delivery of standardized model elements by AUTOSAR. This document also refines the blueprint approach for standardization.

3.1.3 Generic Structure Template [4]

This document acts as a supplement for the formal definition provided by the AUTOSAR meta model. This document provides the introductory description and rationale for the parts of the AUTOSAR meta model relevant for all AUTOSAR templates.

3.1.4 AI Specification [6]

This is the output of the standardization of Application Interfaces. The output is delivered as a set of .arxml files.

3.1.5 Modeling Guide for Application Interfaces [9]

This document gives guidelines and conventions on using the AUTOSAR model elements in order to build AUTOSAR systems. It does not contain guidelines for the AUTOSAR meta-model.

3.1.6 AUTOSAR Methodology [10]

See above.

3.1.7 Explanation of Application Interfaces for Domain Body Comfort [11]

The document explains design decisions and boundary conditions that lead to the Application Interfaces of the domain Body and Comfort.

3.1.8 Explanation of Application Interfaces for Domain Powertrain [12]

The document explains design decisions and boundary conditions that lead to the Application Interfaces of the domain Powertrain.

3.1.9 Explanation of Application Interfaces for Domain Chassis [13]

The document explains design decisions and boundary conditions that lead to the Application Interfaces of the domain Chassis.

3.1.10 Explanation of Application Interfaces for Domain Occupant and Pedestrian Safety [14]

The document explains design decisions and boundary conditions that lead to the Application Interfaces of the domain Occupant and pedestrian safety.

3.1.11 Explanation of Application Interfaces for Domain Multimedia, Telematics, Human Machine Interface [15]

The document explains design decisions and boundary conditions that lead to the Application Interfaces of the domain Multimedia, Telematics, Human Machine Interface.

To find these documents refer to the table at the end of this document (See Chapter 7).

4 Metamodel representation of AI Table

This section describes the relation between meta-model implementation (AUTOSAR Meta Model [7]) and representation within the AI Table [8].

The AUTOSAR meta-model conceptually is defined as 'M2' level, which describes the entities called software components and ports. The relations between those entities as well as their semantics are part of this model.

4.1 Category of Model Elements

All Application Interface model elements are classified into three different categories. They are;

- STANDARD
- BLUEPRINT
- EXAMPLE

4.1.1 STANDARD

All elements, which can be used as they are defined by just including them in the project, belong to the category STANDARD. These elements need no modifications before their use in the projects.

Elements of category STANDARD are;

- DataConstraints
- PhysicalDimensions
- Units

4.1.2 BLUEPRINT

Blueprints are the pre-definition of model elements which form the basis for further modeling. Blueprints are model elements from which other model elements can be derived by copying. These elements are not complete in all aspects. They act as a template for projects to create the real elements.

Elements of category BLUEPRINT are;

- ApplicationDataTypes
- CompuMethods
- PortInterfaces
- PortPrototypeBlueprints

Rules for naming Prototypes derived from BLUEPRINTs

AUTOSAR Standardization will use rules for creating shortNames for prototypes derived from blueprints, i.e. the recommendation below is mandatory for AUTOSAR standardization work.

- Recommendation in case of single usage: <ShortName>
- Recommendation in case of multiple usage: <ShortName>{<Keyword>}0..n"

4.1.3 EXAMPLE

Elements that shall not be standardized, but are helpful for understanding are created as category EXAMPLE. These act as help for the users to actually create their project specific implementations. The elements of category EXAMPLE represent one out of many possible ways of implementation.

Elements of category EXAMPLE are;

- SwComponentTypes
- ApplicationDataTypes
- CompuMethods
- PortInterfaces

ApplicationDataTypes, CompuMethods and PortInterfaces categorized as examples are the derived elements from their blueprints and are not additional elements.

4.2 Meta model diagrams and the AI Table

This section describes the AUTOSAR meta-model (M2) diagrams and their relationship with the AI Table contents to implement the application software-component.

The following diagrams correspond to R4.0 of the AUTOSAR meta-model.

4.2.1 Composition

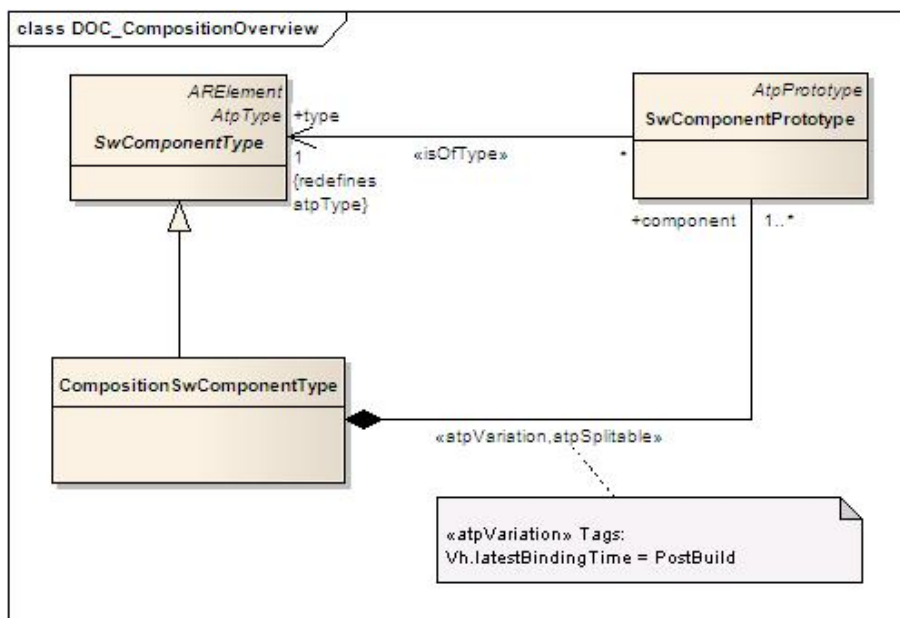


Figure 4: Composition

The purpose of an AUTOSAR CompositionSwComponentType is to allow the encapsulation of specific functionality by aggregating existing software-components. Since a CompositionSwComponentType is also a SwComponentType, it may be aggregated again in further CompositionSwComponentTypes. This recursive relation is formally expressed in Figure 4.

It is important to understand that while compositions allow for (sub-) system abstraction, they are solely an architectural element for the implementation of model scalability. They simply group existing software-components and thereby take away complexity when viewing or designing logical system architecture.

Meta Model Reference:

M2::AUTOSARTemplates::SWComponentTemplate::Composition [1]

AI Table Reference:

AUTOSAR_ApplicationInterfaces.xls [8]

Work Sheet Name: Compositions

Sheet name	Defined Composition	Used Component Types
05010501_Seat	Seat	MgrOfSeat SeatAdj SeatAxis
0501050101_SeatAxis	SeatAxis	SeatAxisA SeatAxis SeatAxisActrAdpr
050106_ExteriorLight	ExtrLi	ExtrLiMgr FlashMgr LiAdprAut AdprComrg AdprHomeCmngAndHomeLvng HdlampLvlgMgr ActrOfHdlampLvlg

Figure 5: part of sheet 'Compositions'

Example:

In the case of Exterior light Composition found in sheet 050106_ExteriorLight: the "ExtrLi" which is a CompositionSwComponent type, composed of different component types like ExtrLiMgr, FlashMgr, LiAdprAut, AdprComrg, AdprHomeCmngAndHomeLvng, HdlampLvlgMgr, ActrOfHdlampLvlg etc...

AI Table Reference:

AUTOSAR_ApplicationInterfaces.xls [8]

Work Sheet Name: Instances

Composition(Sheet name)	Component Type	Number of Instances	Instance names
0501050101_SeatAxis	SeatAxisActrAdpr	1	SeatAxisActrAdpr
050106_ExteriorLight	ExtrLiMgr	1	ExtrLiMgr
050106_ExteriorLight	FlashMgr	1	FlashMgr
050106_ExteriorLight	LiAdprAut	1	LiAdprAut
050106_ExteriorLight	AdprComrg	1	AdprComrg
050106_ExteriorLight	AdprHomeCmngAndHomeLvng	1	AdprHomeCmngAndHomeLvng
050106_ExteriorLight	HdlampLvlgMgr	1	HdlampLvlgMgr
050106_ExteriorLight	ActrOfHdlampLvlg	2	ActrOfHdlampLvlgLe ActrOfHdlampLvlgRi

Figure 6: part of sheet 'Instances'

Example: The component type ExtrLiMgr is not decomposed further into component types therefore; it is just a component type. The component prototype ExtrLiMgr found in 050106_ExteriorLight (cell AB2 shown in Figure 7) is of the type ExtrLiMgr (component type). Here the type and prototype have the same shortname.

The component prototypes ActrOfHdlampLvlgLe and ActrOfHdlampLvlgRi found in 050106_ExteriorLight (cell BF2 shown in Figure 7) is of the type ActrOfHdlampLvlg (component type). Here the type and prototype have different shortname and the type is instantiated twice.

	A	B	C	D	E	F	Z	AA	AB	AC	AD	AE	AF	BF	BG	BH	BI	BJ					
1	PortInterface Shortlame	Shortname of Port	Longname of Port	Description of Port	Initiator WP	Milestone	ExtrLi	ExtrLiMgr	ActrOfHdlampLvlg														
2								ExtrLiMgr	ActrOfHdlampLvlgLe, ActrOfHdlampLvlgRi														
3							S3MS4	S3MS4	S3MS4														
4								ExteriorLightManager	HeadLampLevelingActuator														
5								Master for Exterior Light	Actuator SW-C for Head Lamp Leveling														
6								10.1	10.1														

Figure 7: Sheet 050106_ExteriorLight - Decomposition Components

There can be arbitrary numbers of SwComponentPrototypes that refer to specific SwComponentTypes created. Note that CompositionSwComponentType also aggregates the abstract meta-class SwConnector for connection of the SwComponentPrototypes belonging to each other.

4.2.1.1 Multiple Instantiation

When designing a system it is often the case that elements in the runtime space share the same structure. A well-known example domain is object-oriented programming, where objects instantiated from the same class all have the same structure specified by that class. The ability to specify a structure once and then use it in multiple places in the design is called as multiple instantiation.

The same concept is used in the AI Table for definition of SwComponentType for the components which exists multiple times in a domain.

In the example shown below, the SoftwareComponents WshrFrnt, WshrRe and WshrHdlamp are created from the SwComponentType Wshr.

	A	B	AR	AS	AT	AV	AW	AX	AY	BA	BB	BC	BD	BF	BG	BH	BI	BK		
1	PortInterface Shortlame	Shortname of Port	Wipr	Wshr	Wshr	Wshr														
2			Frnt, WiprR	WshrFrnt	WshrRe	WshrHdlamp														
3			S1MS4	S1MS4	S1MS4	S1MS4														
4			WiprFront, WiprRear	WasherFront, WasherRear, WasherHeadLamp	WasherFront, WasherRear, WasherHeadLamp	WasherFront, WasherRear, WasherHeadLamp														
5			Wiper module commands the wipers of the	Washer module commands the washers of the	Washer module commands the washers of the	Washer module commands the washers of the														
6			10.1	10.1	10.1	10.1														
7			R	core cond opt	IV	P	R	core cond opt	IV	P	R	core cond opt	IV	P	R	core cond opt	IV	P		

Figure 8: Sheet 050108_WiperWasher – Example of Multiple Instantiation

All the SwComponentPrototypes instantiated from the SwComponentType will have the same properties of the SwComponentType, in other words for example if SwComponentType is defined with 2 Provider PortPrototypes and 3 Receiver PortPrototypes then all the instances of the SwComponentType will have same number of Provider and Receiver PortPrototypes.

If two SwComponentTypes are connected to each other and both the SwComponentTypes are multiply instantiated then it is possible that the connections between these SwComponentTypes are ambiguous. However due to limitations in the AI Table macros it is currently not feasible to cover all possible model scenarios.

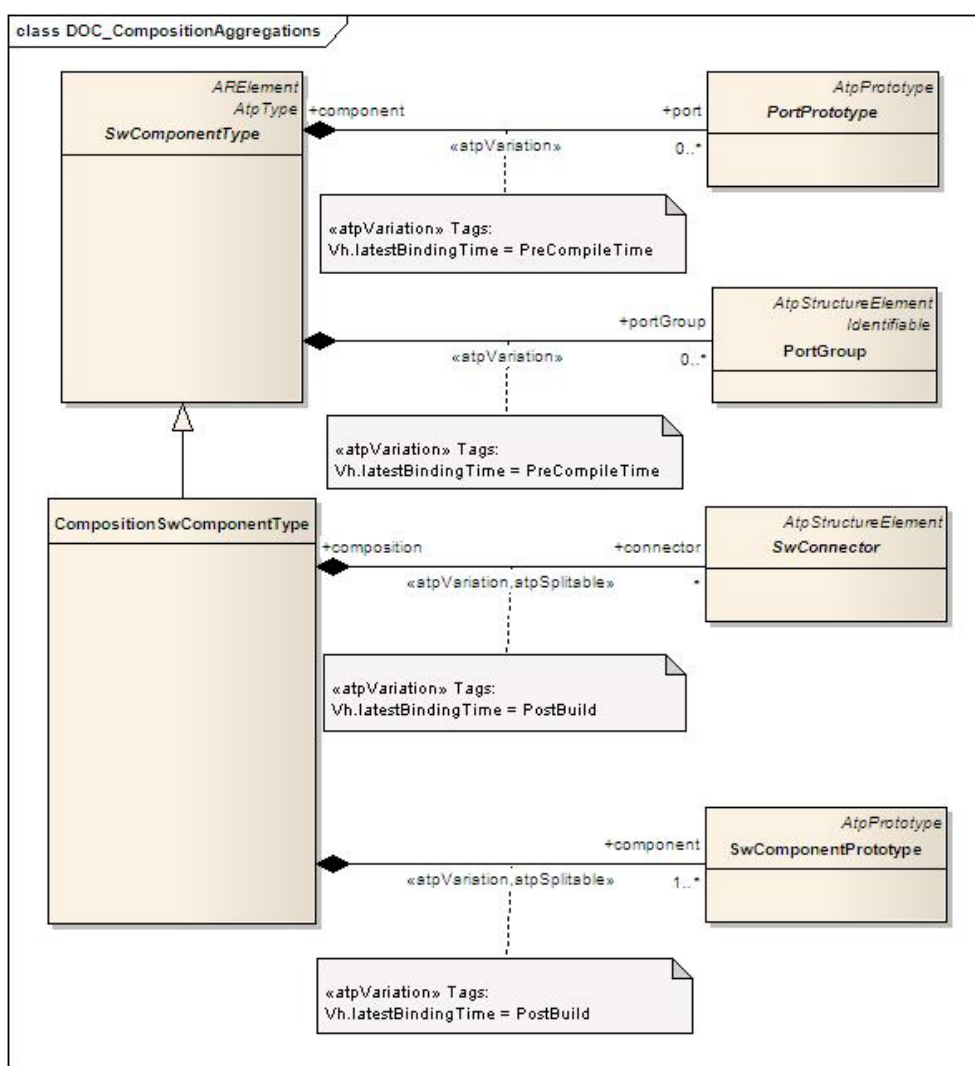


Figure 9: Composition – Aggregations

Meta Model Reference:

M2::AUTOSARTemplates::SWComponentTemplate:: Components [1]

AI Table Reference:

AUTOSAR_ApplicationInterfaces.xls [8]

Work Sheet Name: 050XXXXX Sheets

Note that being a SwComponentType, a CompositionSwComponentType also exposes PortPrototypes to the outside world. However, the PortPrototypes are only delegated and do not play the same role as PortPrototypes attached to AtomicSwComponentTypes (AtomicSwComponentTypes encapsulate the implementation of their functionality and behavior and merely expose well defined connection points, called PortPrototypes, to the outside world.). For more details, refer to the SW Component Template [1]

CompositionSwComponentTypes contain two types of SwConnectors.

1. AssemblySwConnectors to interconnect PortPrototypes of SwComponentPrototypes that are part of the CompositionSwComponentType
2. DelegationSwConnectors to connect from "inner" PortPrototypes to delegated "outer" PortPrototypes.

In the case that the outer PortPrototype is referenced by multiple DelegationSwConnectors, the semantic is the multiplication of the AssemblySwConnectors referencing the outer PortPrototypes.

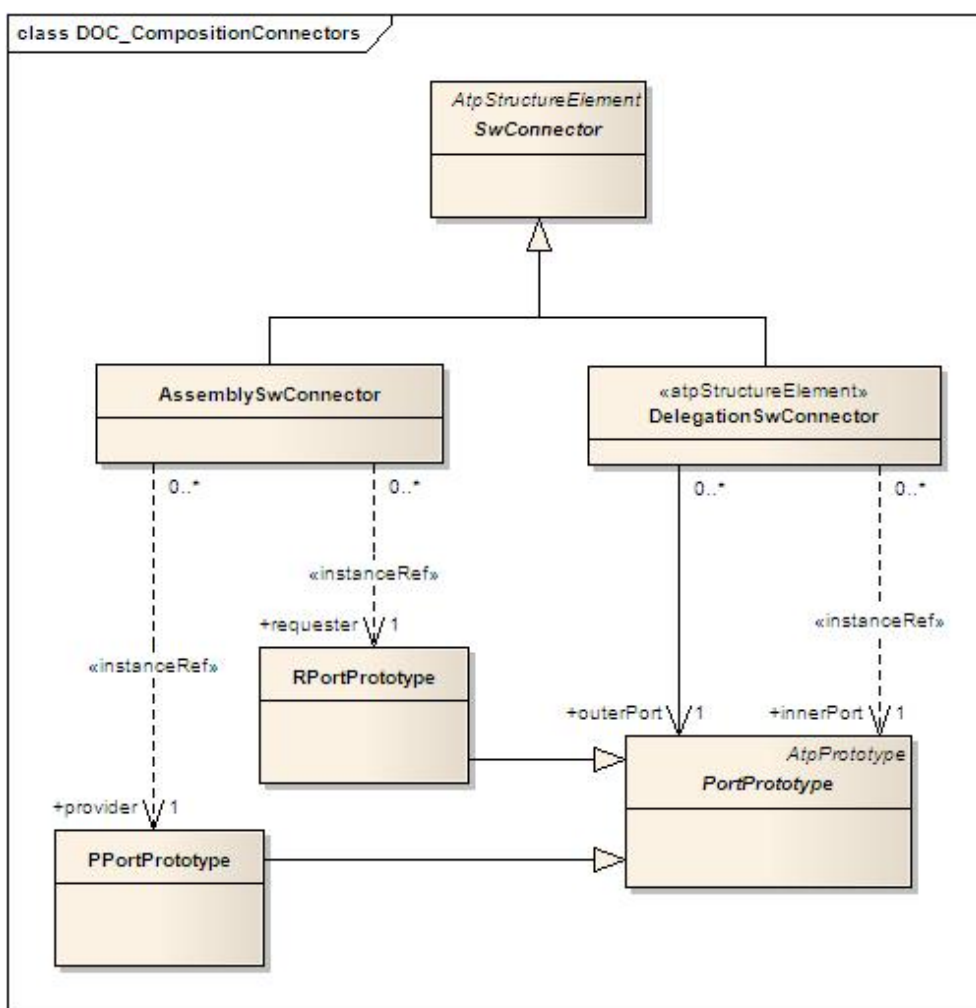


Figure 10: Composition - Connectors

Example: In the case of Exterior light decomposition, “ExtrLi” software component prototype is the composite type which provides an “outer” PPortPrototype “TrlrSts”, which is delegated from “inner” PPortPrototype of software component prototype “ExtrLiAdprTrlr”. The same PPortPrototype is connected to RPortPrototype of “ExtrLiAdprReLe” and “ExtrLiAdprReRi” through an assembly connector prototype.

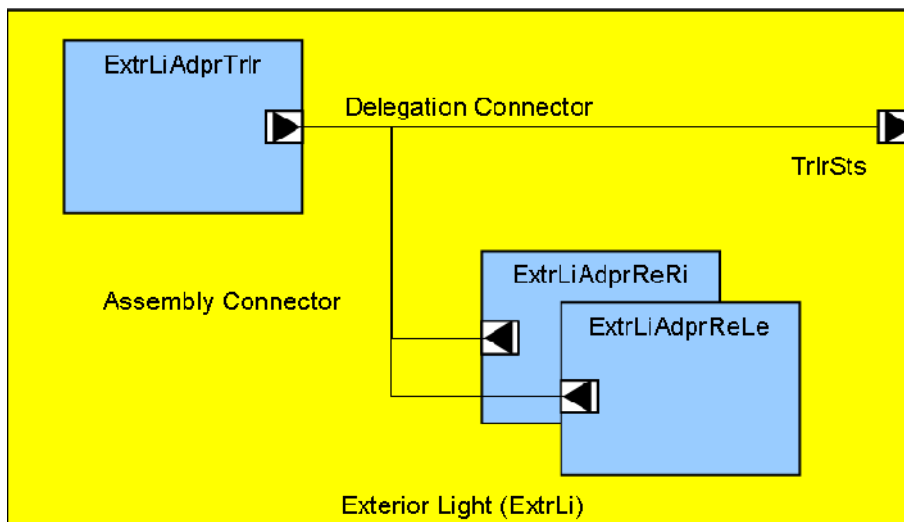


Figure 11 : Exterior Light Decomposition example

4.2.2 Blueprint Mapping & BlueprintMappingSet

Blueprint mapping acts as reference between the blueprint and the derived element. Blueprint mapping identifies the relationship between the blueprinted element and the actual blueprint. It also validates the derived element against the blueprint. Aggregation of these BlueprintMappings is a BlueprintMappingSet. The Figure below shows the BlueprintMapping and BlueprintMappingSet.

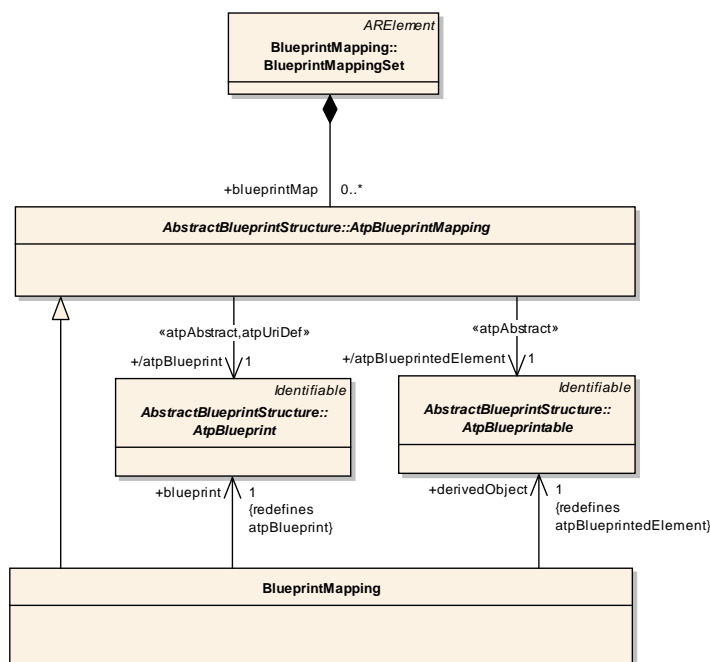


Figure 12: BlueprintMapping & BlueprintMappingSet

4.2.3 PortPrototypes

PortPrototypes also referred as Ports in some places are the well-defined connection points for communication between different software-components. A PortPrototype is either required type or provided type. A require-port (in technical terms: RPortPrototype) requires certain services or data, while a provide-port (or PPortPrototype) on the other hand provides those services or data.

Two SwComponentPrototypes are eventually connected by hooking up a PPortPrototype of one SwComponentPrototype to a compatible RPortPrototype of the other SwComponentPrototype.

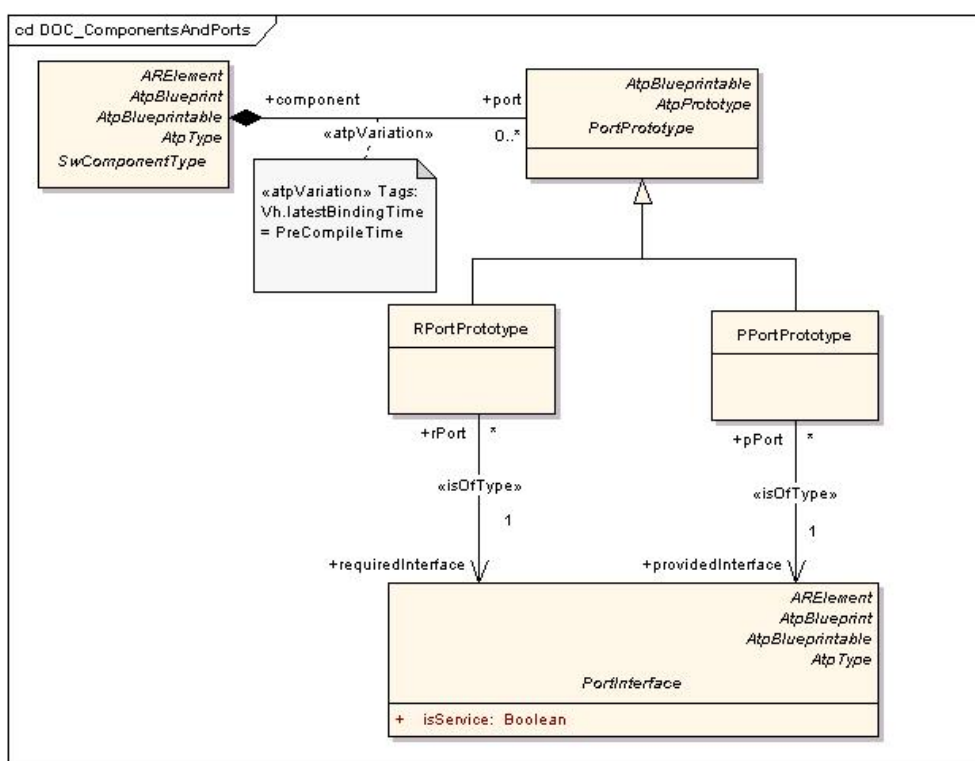


Figure 13: PortPrototypes

4.2.3.1 PortPrototypeBlueprints

PortPrototypeBlueprint is an ARElement and acts as a blueprint for the creation of PortPrototypes. A user can pick a specific PortPrototypeBlueprint and create PortPrototype out of it.

PortPrototypeBlueprint does not have any relation to the SwComponentType. The PortPrototypeBlueprints are not explicitly represented in the AI Table, they can be found as a separate package in the XML.

PortPrototypeBlueprints can be seen as a library, from which the user can choose a PortPrototypeBlueprint as a template to create PortPrototype. Consequently, the PortPrototypeBlueprints are just a collection of PortPrototypes without any architectural relation. As soon as a PortPrototypeBlueprint is attached to a SwComponentPrototype, it becomes a PortPrototype.

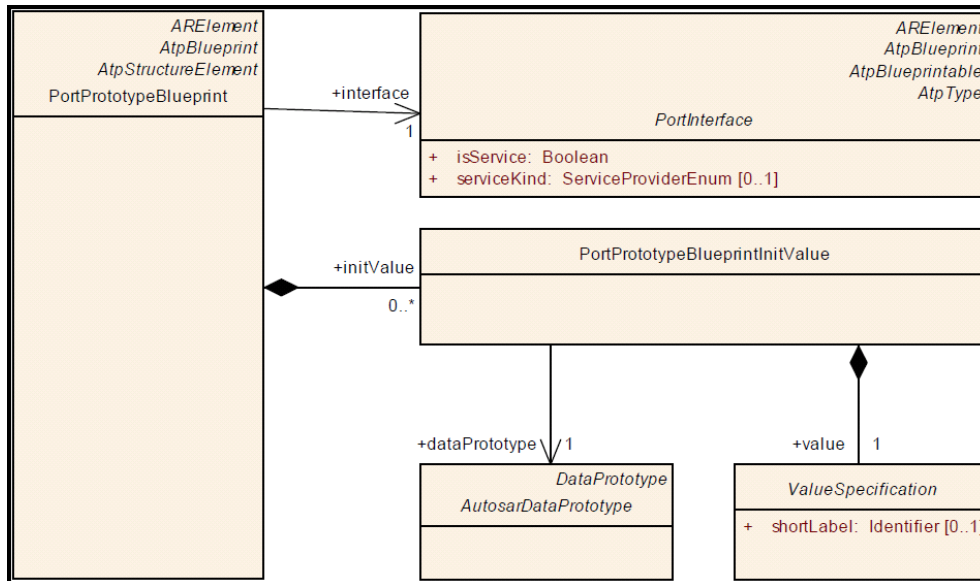


Figure 14: PortPrototypeBlueprints

4.2.3.2 BlueprintMapping of PortPrototypeBlueprints

The process of creating a PortPrototype from the available PortPrototypeBlueprints is called BlueprintMapping. BlueprintMapping is demonstrated in Figure 12. The mapping between PortPrototypes and PortPrototypeBlueprints can be found in the BlueprintMappingSet "PortPrototypeBlueprintMappings" within the package "BlueprintMappingSets_Example".

4.2.4 PortInterfaces

The PortInterface defines the kind of information transported between two PortPrototypes.

PortInterfaces are used to support a design-by-contract workflow, i.e. they provide a means to formally verify structural and dynamic compatibility between software-components. In other words, PortInterfaces represent a pivotal point in the AUTOSAR concept.

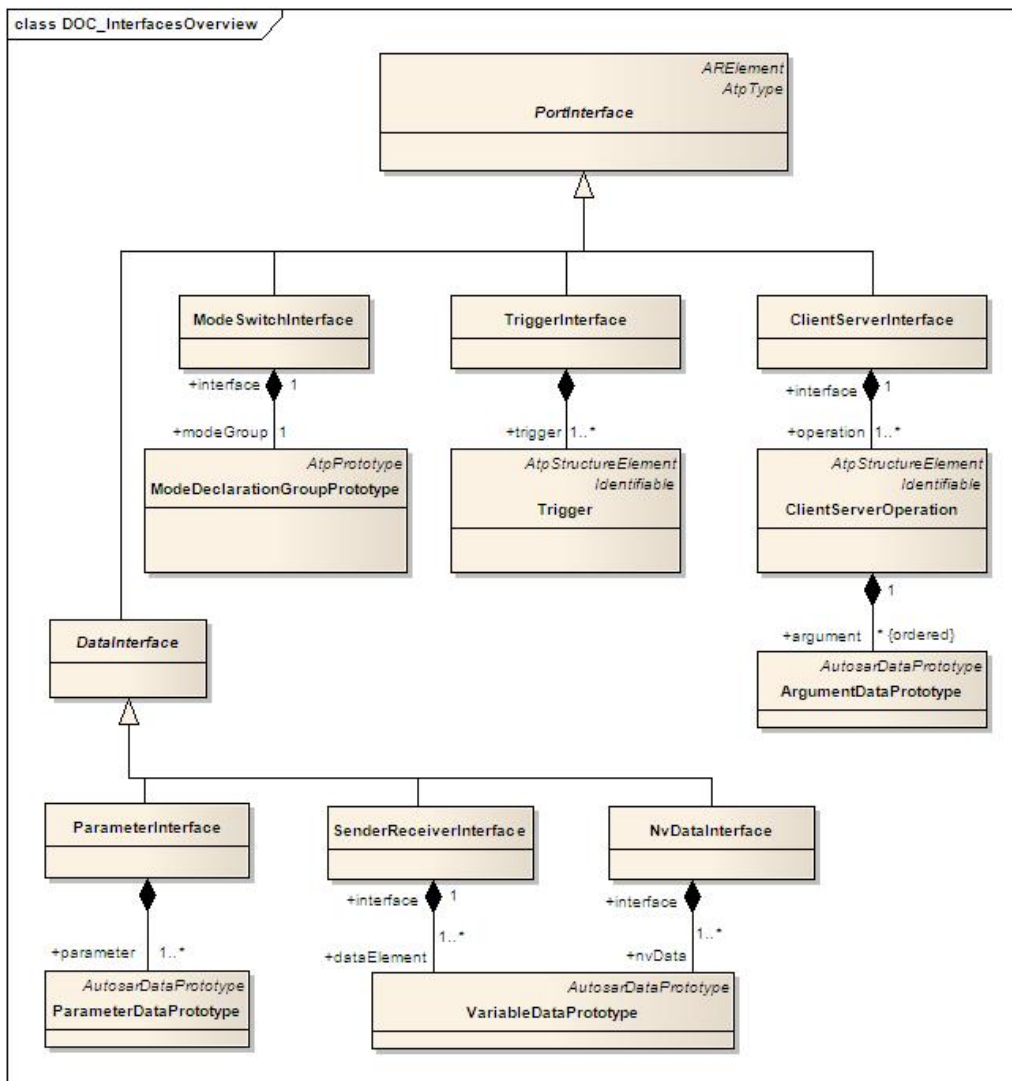


Figure 15: Interfaces Overview

Meta Model Reference:

M2::AUTOSARTemplates::SWComponentTemplate::PortInterface [1]

BlueprintMapping is the process of creating PortInterfaces from the PortInterfaceBlueprints. This is demonstrated in Figure 12. The mappings between PortInterfaces and PortInterfaceBlueprints can be found in the BlueprintMappingSet "PortInterfaceBlueprintMappings" within the package "BlueprintMappingSets_Example".

4.2.4.1 Sender Receiver Communication

SenderReceiverInterfaces allow for the specification of the typically asynchronous communication pattern where a sender provides data that is required by one or more receivers. While the actual communication takes place via the respective PortPrototypes, a SenderReceiverInterface allows for formally describing what kind of information is sent and received.

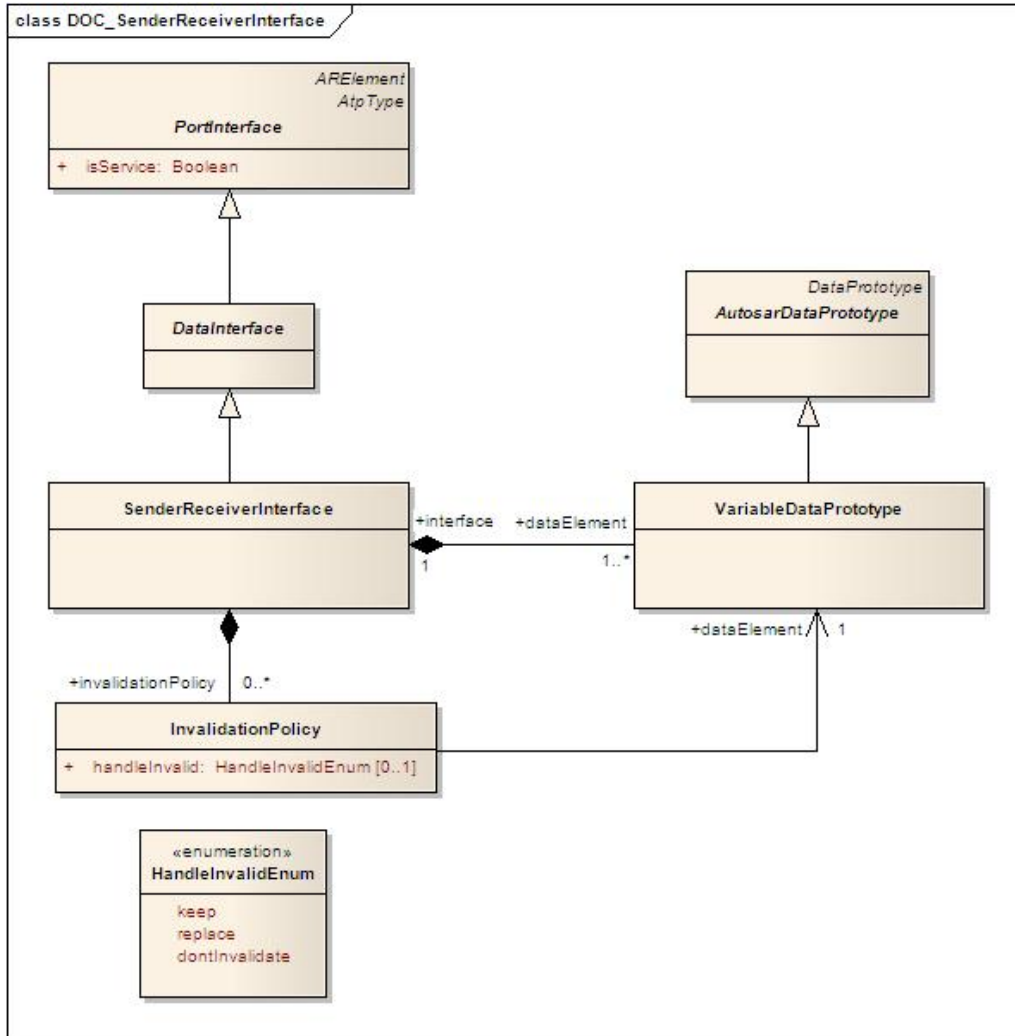


Figure 16: Sender Receiver Interface

A SenderReceiverInterface declares a number of data elements (VariableDataPrototype) to be sent and received. A SenderReceiverInterface focuses on the description of information items represented by VariableDataPrototypes. A VariableDataPrototype aggregated in the role of dataElement represents an atomic piece of information transmitted among PortPrototypes typed by a SenderReceiverInterface.

AI Table Reference:

AUTOSAR_ApplicationInterfaces.xls [8]
 Work Sheet Name: 06_Interface_DataElements

SenderReceiver Interface ShortName	Long Name	Initiator WP	Milestone	Description	Reference	1st data element			
						Name	Type	Description	Queuein
LockgFbReq1	LockingFeedbackRequest	10.1	S3MS4	Request special blinking for feedback that the locking system was unlocked, locked or safed by user request. The blink sequence is represented by a separate interface.	CL	Sts	LockActvn1		
TrlrSts1	TrailerStatus1	ID/10.1	S3MS4	Detection of presence of a trailer by switch type detection. Information: Trailer attached 0 means 'not present' 1 means 'present'		TrlrSts	Boolean	0 means 'not present' 1 means 'present'	

Figure 17: Sheet 06_Interface_DataElements – Example of SenderReceiver Interface

Example: “TrlrSts1” is a SenderReceiver interface that has one data element “TrlrSts” of Type “Boolean”.

4.2.4.2 Client Server Communication

The underlying semantics of a client/server communication is that a client may initiate the execution of an operation by a server that supports the operation. The server executes the operation and immediately provides the client with the result (synchronous operation call) or else the client checks for the completion of the operation by itself (asynchronous operation call).

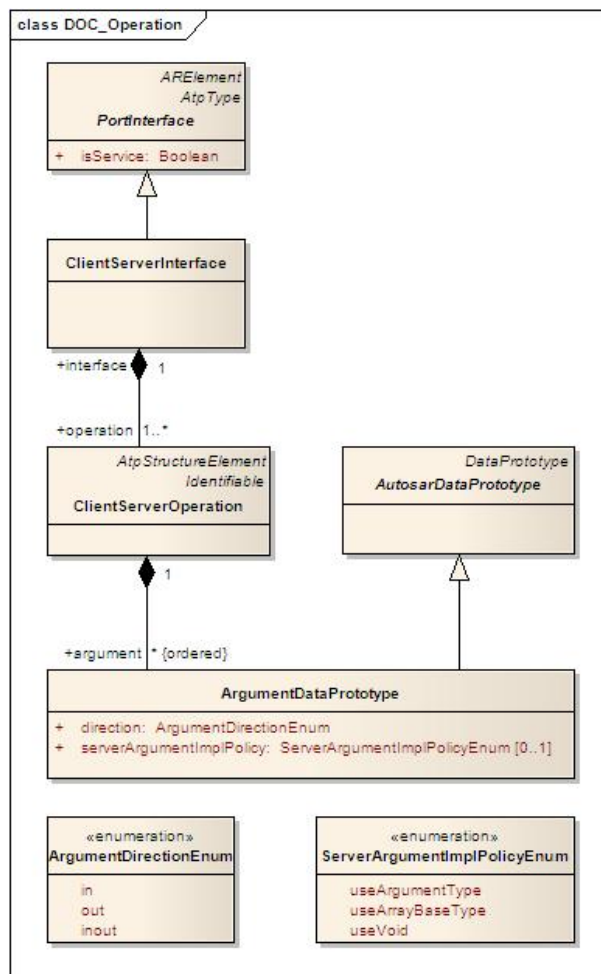


Figure 18: ClientServer Interface

ClientServerInterface, therefore to some extent is a counterpart to the SenderReceiverInterface. Instead of defining pieces of information to be transferred among software-components, a ClientServerInterface defines a collection of ClientServerOperations.

As depicted in Figure 18, a ClientServerInterface is composed of ClientServerOperations, i.e. a ClientServerOperation cannot be reused in the context of a different ClientServerInterface. A ClientServerOperation consists of 0 to many ArgumentDataPrototypes. The latter may be

- passed to the operation
- passed to and returned from the operation
- Returned from the operation.

AI Table Reference: AUTOSAR_ApplicationInterfaces.xls [8]
 Work Sheet Name: 06_Interface_ClientServer

ClientServerInterface Shortname	Long Name	Initiator WP	Milestone	Description	Preference	OperationName		
						Shortname	Long Name	Description
TrsmRatGear1	Transmission: Gear ratio for a given gear	10.2	S4MS3	Returns the gear ratio for a given gear Theoretical transmission ratio		GetTrsmRatGear	Returns the gear ratio for a given gear	Returns the gear ratio for a given gear

Figure 19: Sheet 06_InterfaceClientServer – Example of ClientServer Interface, showing operation

Argument1					Argument2				
ArgumentName Shortname	Long Name	Description	DataType	III/OUT/IIOUT	ArgumentName Shortname	Long Name	Description	DataType	III/OUT/IIOUT
Gear	Gear for which the ratio should be returned	Gear for which the ratio should be returned	Nr4	IN	Rat	Gear ratio of given gear	Gear ratio of given gear	Fac1	OUT

Figure 20: Sheet 06_InterfaceClientServer – Example of ClientServer Interface, showing arguments

Example: In the above screen shots, a client/server interface “TrsmRatGear1” is defined to return the gear ratio for a given gear, the client requests the operation ‘GetTrsmRatGear’ with input argument ‘Gear’. The function call returns the output argument ‘Rat’.

4.2.5 DataTypes

It is possible to describe data provided by a software component from the application as well as implementation point of view. The common concept behind this is expressed by the abstract meta-class AutosarDataType, from which an ApplicationDataType and an ImplementationDataType are derived.

Figure 21 shows a summary of the basic meta-classes used for the definition of AutosarDataTypes.

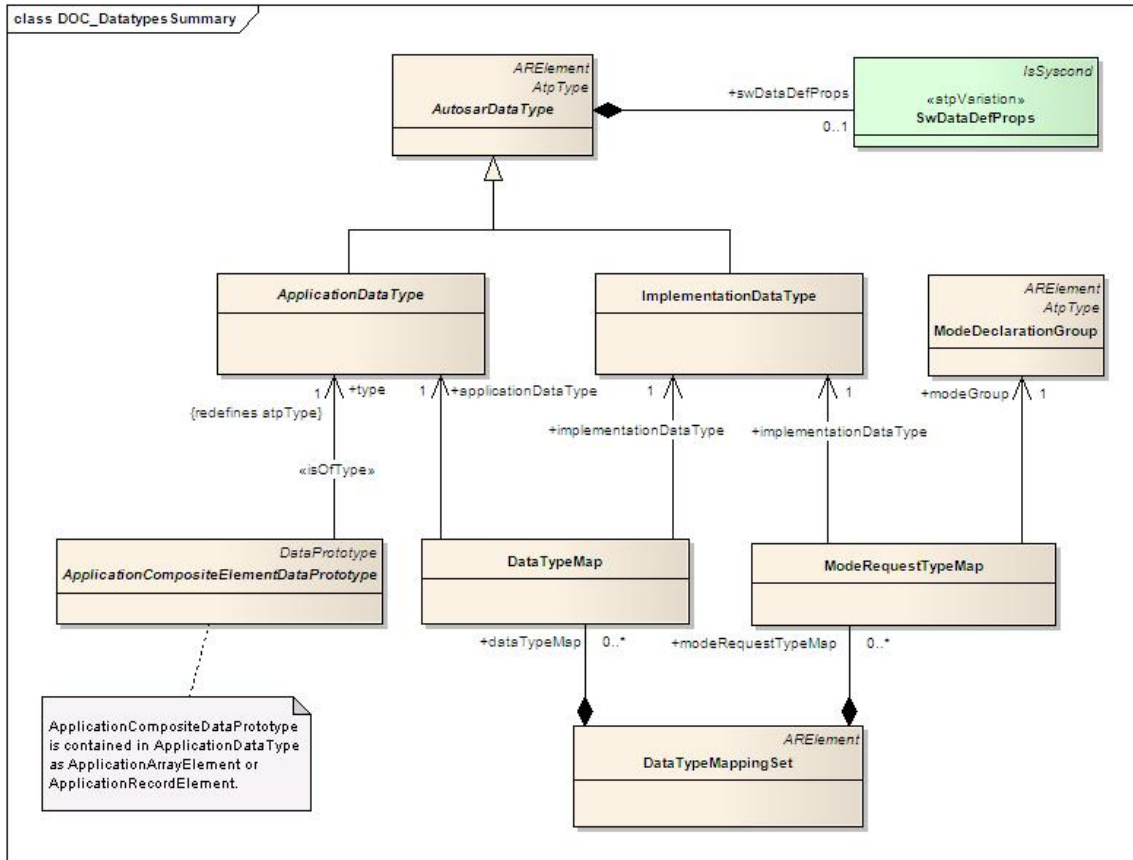


Figure 21: DataTypes Overview

An **ApplicationDataType** can be composed (in form of a record or an array) of elements which themselves are typed by another **ApplicationDataType**. This is expressed by the meta-class **ApplicationCompositeElementDataPrototype**, which is shown in

Figure 21 for completeness. An **ImplementationDataType** can be composed too, but in this case no type/prototype concept has been applied.

4.2.5.1 ApplicationDataTypes

The abstract meta-class **ApplicationDataType** is further derived into an **ApplicationPrimitiveDataType** and an **ApplicationCompositeDataType**. Like any **AutosarDataType**, the primitive and composite types on application level are characterized by its category and its **SwDataDefProps**. For a given category, only a limited set of attributes of the **SwDataDefProps** make sense.

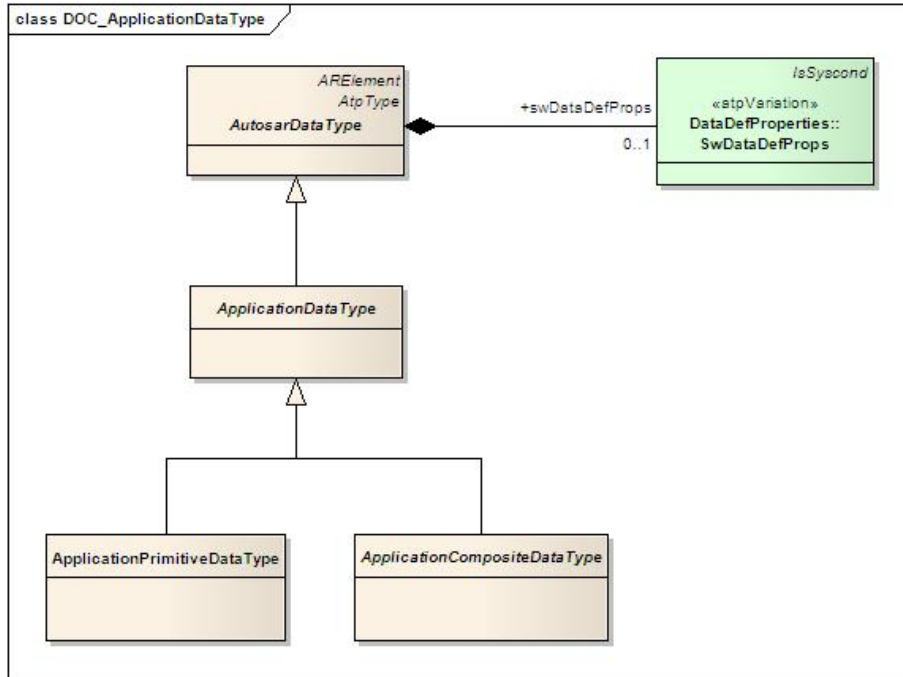


Figure 22: Application Data Type

4.2.5.1.1 Application Primitive Data Types

This chapter defines the primitive application data types that may be used for the data prototypes of PortInterfaces or composite application data types.

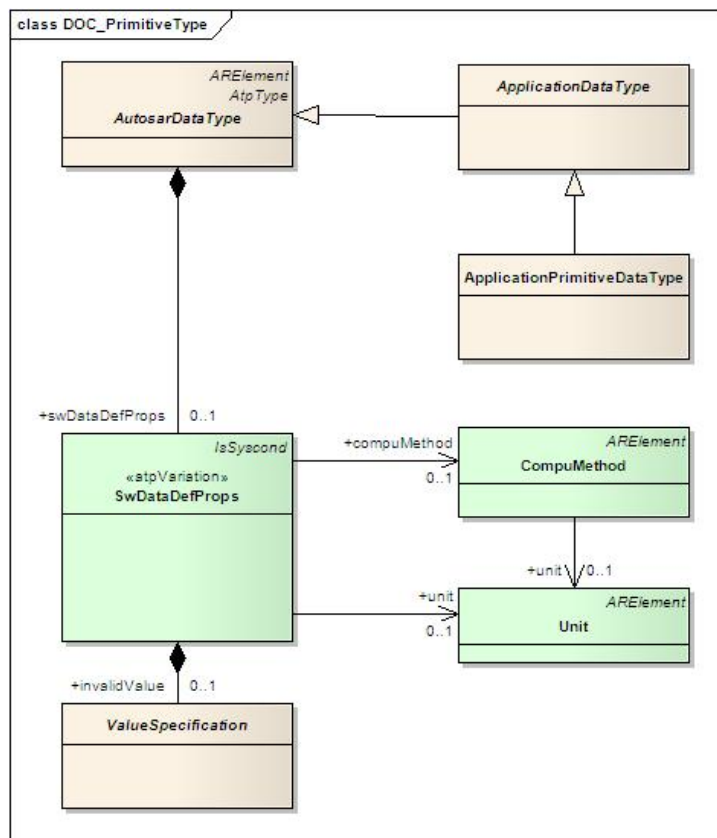


Figure 23: Data Types Primitive

Meta Model Reference:

M2::AUTOSAR Templates::SWComponentTemplate::DataType::DataTypes [1]

AI Table Reference: AUTOSAR_ApplicationInterfaces.xls [8]

Work Sheet Name: 07_DataTypes_ContinuousValue

Short name	Long name	Description	Initiator WP	Minimal Bits Size recommended	Resolution	Physical Lower Limit	Physical Upper Limit	Offset	Unit
Perc8	Percent8	Generic percent data type 8	10.4	UInt16	0.00031	-5.00000	15.00000	5.00000	%

Figure 24: Sheet 07_DataTypes_ContinuousValue – Example of a Continuous Value DataType

Example: In the above screen shot, a ContinuousValue DataType Perc8 is defined. The resolution, physical lower and upper limits as well as the offset and unit of this DataType are also shown in the screen shot.

AI Table Reference: AUTOSAR_ApplicationInterfaces.xls [8]

Work Sheet Name: 08_DataTypes_Enumeration

Data Type Name	Long Name	Description	Initiator WP	Milestone	Minimal Number of Bits	Enumeration values			
						value	name	comment	is boolean
TrsmTyp1		Information on standard transmission.	10.2	S1MS4		0 Mt		manual transmission	
TrsmTyp1		Other transmission types on value 5 - 15	10.2	S1MS4	3	1 At		automatic transmission	
TrsmTyp1			10.2	S1MS4		2 Amt		direct shift/ automated Mt	
TrsmTyp1			10.2	S1MS4		3 Cvt		continuously variable transmission	
TrsmTyp1		Information about the state of the clutch	10.2	S1MS4		4 Dct		twin-clutch gearbox or dual or double clutch transmission (D	

Figure 25: Sheet 08_DataTypes_Enumeration – Example of Enumeration DataType

Example: In the above screen shot, an Enumeration DataType TrsmTyp1 is defined. All permissible values of the Enumeration DataType are also included in the AI Table.

4.2.5.1.2 Application Composite Data Types

The meta-classes ApplicationArrayDataType and ApplicationRecordDataType provide the means to define composite DataTypes. Such a composite DataType is required, if the application software wants to have access to the individual elements of the composite as well as to do operations with the whole composite, e.g. wants to communicate the complete record or array in a single transaction.

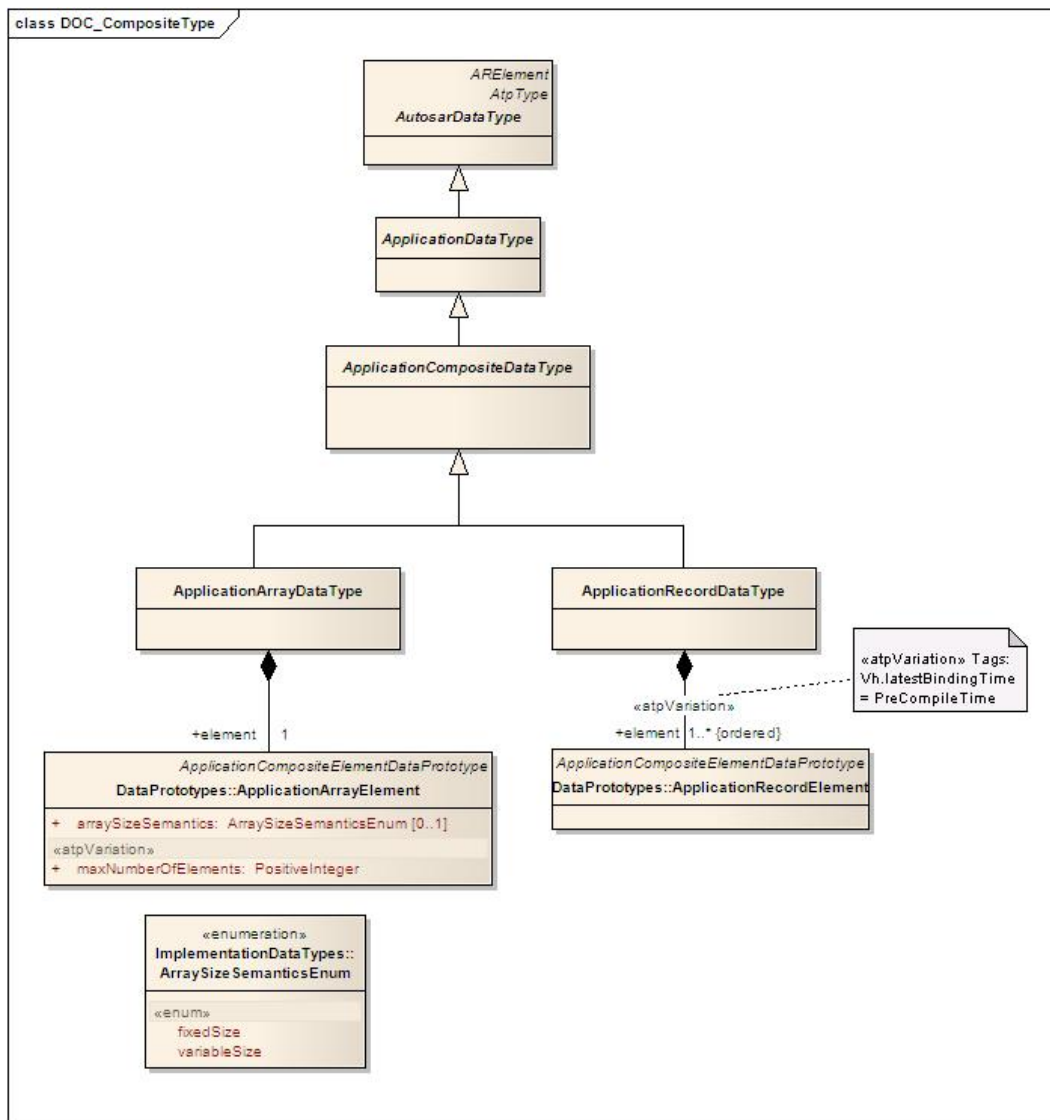


Figure 26: Data Types Composite

It is possible to use a combination of ApplicationArrayDataType and ApplicationRecordDataType, so that an ApplicationArrayDataType could be defined as ApplicationRecordElement of an ApplicationRecordDataType and in the same manner, an ApplicationRecordDataType could be used as the base type of an ApplicationArrayDataType. The creation of nested ApplicationComposite DataTypes is also possible.

4.2.5.1.3 ApplicationArrayDataType

An ApplicationArrayDataType may contain maxNumberOfElements of ApplicationArrayElements. Each of these ApplicationArrayElements must have the same type. When referring to an element of an array within the software-component descriptions, the element index runs from 0 to (maxNumberOfElements-1).

AI Table Reference: AUTOSAR_ApplicationInterfaces.xls [8]
 Work Sheet Name: 09_DataTypes_Array

Data Type Name	Long Name	Description	Initiator WP	Milestone	Type Name	Number of Element
TirePPerWhl1	Tire Pressure Per Wheel 1	Tire pressures at wheels. Convention is: - Index 0 = Front Left - Index 1 = Front Right - Index 2 = Rear Left - Index 3 = Rear Right - Index 4 = Spare Wheel	10.3	S5MS4	P1	5
TireTPerWhl1	Tire Temperature Per Wheel 1	Tire temperatures. Convention is: - Index 0 = Front Left - Index 1 = Front Right - Index 2 = Rear Left - Index 3 = Rear Right - Index 4 = Spare Wheel	10.3	S5MS4	T3	5

Figure 27: Sheet 09_DataTypes_Array – Examples of Array DataType

Example: The standard variables of DataType array used in application component development are listed here for reference. Array DataType “TirePPerWhl1” contains 5 elements, each of DataType ‘P1’.

4.2.5.1.4 ApplicationRecordDataType

A declaration of ApplicationRecordDataType describes a nonempty set of objects, each of which has a unique identifier with respect to the ApplicationRecordDataType and each has its own ApplicationDataType. The shortName of each ApplicationRecordElement within the scope of an ApplicationRecordDataType must be unique.

AI Table Reference: AUTOSAR_ApplicationInterfaces.xls [8]

Work Sheet Name: 11_DataTypes_Record

Record Type Name	Long Name	Description	Initiator WP	Milestone	Number of element				Reference
						Name	Type Name	Comment	
LockgCenSts1	Locking Central Status	Status of the Central Locking. The status undefined is valid if there are no feedback switches in the latch and the ECU lost the information of actual status	10.1	S5MS4					
LockgCenSts1			10.1	S5MS4	2	LockgSt TrigSrc	LockSts2 LockTrigSrc1		CL CL
DiagcLock1	Diagnostic Lock	Diagnostic of the status of the locking. States if the lock is working or not.	10.1	S5MS4	2	Lock Diagc	LockActvn1 OnOff1		CL CL
DiagcChdProtn1	Diagnostic Child Protection	Diagnostic of the status of the child protection. States if the child protection is working or not.	10.1	S5MS4	2	ChdProtn Diagc	ChdProtnCmd1 OnOff1		CL CL

Figure 28: Sheet 11_DataTypes_Record – Example of Record DataType

Example: The “IndcrTurnSeq1” is a record type variable to represent the turn indicator sequence. The record DataType ‘IndcrTurnSeq1’ contains 5 elements’, each element represented in a unique row defined with a corresponding DataType.

4.2.6 Physical Units

An important part of the semantics associated with a DataType is its physical dimension. Units are used to augment the value with additional information like m/s or liter. This is necessary for a correct interpretation of the physical value for input and output processes. The unit involves information about its physical dimensions.

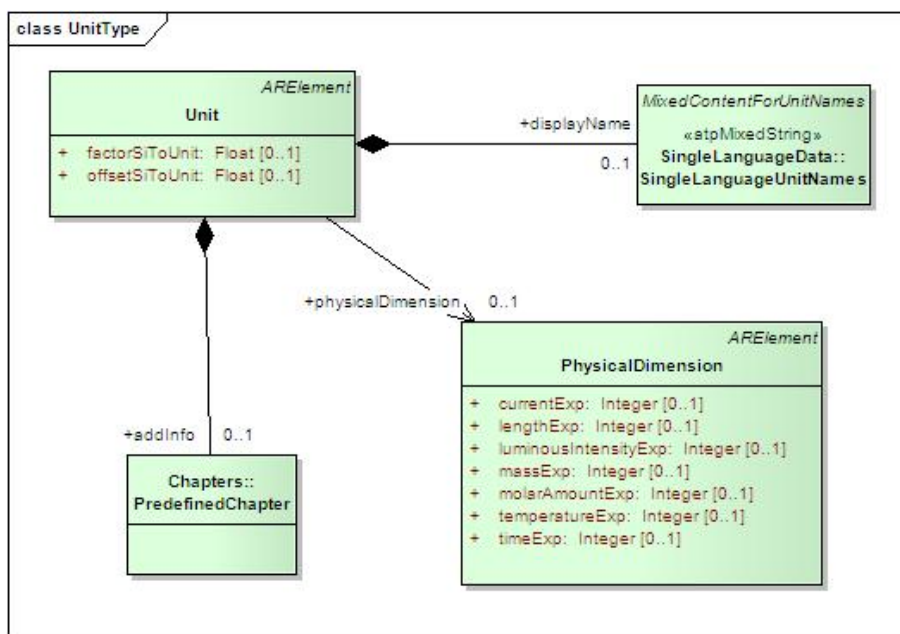


Figure 29: Units

The unit references one physical dimension. If the physical dimensions of two units are identical, a conversion between them is possible.

Meta Model Reference:

M2::AUTOSARTemplates::SWComponentTemplate::DataType::Units [1]

AI Table Reference: AUTOSAR_ApplicationInterfaces.xls [8]

Work Sheet Name: 13_Units

Unit Name (short name)	Long Name	Unit Display Name	Description	Physical Dimension							FACTOR-SI-TO-UNIT	OFFSET-SI-TO-UNIT	
				electrical current	luminous intensity	time	mass	amount or substance	thermodynamic temperature	length			
Perc	Percent	%	a percentage is a way of expressing a number as a fraction of 100 (no SI unit).									100	0
PercPerSec	Percent Per Second	%/s	time-derivative of percent			-1						100	0
DegCgrd	Degree Centigrade	degC	temperature, no SI unit, (degC = Kelvin - 273.15)						1			1	-273.15
MtrRecpr	Meter Reciprocal	1/m	SI derived unit of wave number for electromagnetic fields or curvature as reciprocal of curve radius							-1		1	0
Bar	Bar	bar	pressure, no SI unit; 1 bar = 100,000 Pascal (Pa)			-2	1			-1		0.00001	0
Hz	Hertz	Hz	SI derived unit of frequency			-1						1	0

Figure 30: Sheet 13_Units – Example of Unit

Example: The unit represented as 'DegCgrd', belonging to the physical dimension of 'thermodynamic temperature'.

4.2.7 Computation Methods

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.

5 Structure of Application Interfaces (AI) Table

The AI Table is referenced as [8].

5.1 Main sheets of the AI Table

A brief overview of the sheets can be found in 02_General_Purposes of the AI Table. The following section gives a detailed explanation of the structure with graphical representations. The essential sheets of the AI Table are listed below; a complete list can be referenced in sub-section 5.2.

These are the sheets relevant for modifying contents of the AI Table for standardized application interfaces without taking into account administrative sheets. Administrative sheets contain the results of consistency checks and they are filled in automatically by the macros. The sheets might have to be checked by the user after consistency check macros are executed.

Sheet number	Content
04_Keywords	Definition of keywords
05... compositions, components	Definition of compositions and components along with its ports
06_Interface_DataElements 06_Interface_ClientServer	Definition of sender/receiver interface, client server interface
07_DataTypes_ContinuousValue	Definition of DataTypes for continuous values
08_DataTypes_Enumeration	Definition of DataTypes for enumerations
09_DataTypes_Array	Definition of DataTypes for arrays
11_DataTypes_Record	Definition of DataTypes for records
13_Units	Definition of units
15_Redirected_Ports	Definition of redirected ports

In the following sections, all the main categories of the sheets are explained in detail.

Note: All the figures and screenshots shown in the following sections are only examples and they may not match exactly to the AI Table content.

5.1.1 Sheet 04_Keywords

The “04_Keywords” work sheet of the AI Table contains the Keywords and abbreviations for them. The columns “Keyword”, “Keyword Abbreviation” usage context list and “usecase” are used to define standard Keywords (e.g. Accept), keywords abbreviation (e.g. Acpt) which are commonly agreed in AUTOSAR. These defined keyword abbreviations are used to define short names of standardized Meta model elements (e.g. Port, Port interface) in the AI Table. A sample definition is shown in Figure 31.

C	D	E	F	G	H	I	J	K	L	M
Keyword	Keyword Abbreviation	DefinitionEnglish	Milestone	Initiator WP	Mean / Environment / Device	Action / Physical Type	Condition / Qualifier	Index	Preposition	Use Case (short names)
Acceleration	A	This keyword is used to give the physical value of acceleration. It is the value of gaining speed.	S0MS4	ID		X				WhlAVertBas
Accelerator	Accr		S0MS4	ID	X					AccrPedIRat
Accept	Acpt		S2MS4	ID		X				TirePmonAcptOfWhisNew
Access	Acs		S0MS4	ID		X				AcsKeyls

Figure 31: Sheet 04_Keywords – Example of Keywords

5.1.2 Sheet 05_TopLevel

The TopLevel worksheet contains inter-domain PortPrototypes and PortInterface connection matrix. Port interface short name, port short name, port long name, port description of each of the entries are present in columns with headings “Port interface shortname” “Shortname of Port”, “Long name of Port” and “Description of port” respectively. After these columns administrative columns (Initiator WP and Milestone¹) and consistency check results matrix columns are present (For consistency check understanding refer sheet 102_User_Documentation in AI Table). After the administrative columns, communication related information is provided in 'transmissionAcknowledgement Timeout' and 'canInvalidate' columns. 'TransmissionAcknowledgement Timeout' column specifies the number of seconds before an error is reported or in case of allowed redundancy, the value is sent again. 'canInvalidate' column provides the status whether the component can actively invalidate data. The information 'TransmissionAcknowledgement Timeout' and 'canInvalidate' are not created in the generated XML file.

After the grey marked 'consistency check' columns, yellow marked 'TopLvl' column is present. This yellow marked column has no functional meaning in this sheet. Each domain is represented in the following columns. The short name of composite composition type (e.g. Body) and short name of composite composition prototype (e.g. Body) are mentioned in separate rows for each domain column.

Each domain column contains 5 sub columns

Provider port (with headline 'P')

Receiver port (with headline 'R')

Existence of port (with headline "core cond opt")

Initial value (with headline IV) and

Description (with headline components specific description)

For each entry of port interface and its corresponding port, a inter domain link (data exchange connection) is established with a 'X' in column 'P' or 'R' as required.

¹ The quality of data visible within the AI Table is always with a quality indicator based on a combination taken from step numbers and milestones like SxMSy (where x could be 0, 1, 2, 3 ... and y could be 1, 2, 3, 4). Milestones field needs to be changed manually always after a review or after any modification to the model elements. Milestone of the PortInterface cannot be higher than the milestone of the DataType. Similarly, the milestone of the port cannot be higher than the milestone of the PortInterface.

Sender/Receiver/Interface Shortname	Long Name	Initiator WP	Milestone	Description	1st data element		
					Name	Type	Description
TrsmTqDfTrfAct1	Transfer Case Actual	10.3	S3MS2	Actual theoretical max transfer Torque Transfer Case Differential as realised by the Actuator. Sign convention: positive to the right or front.	TrsmTqDfTrfAct	Tq13	Actual theoretic Torque Transfer Differential as r
TrsmTqDfTrfMax1	Transfer Case Maximum torque. ABS value of torque.	10.3	S3MS2	Absolute Value of the Maximum Torque at the Transfer Case Differential	TrsmTqDfTrfMax	Tq14	Actuator Absolute Value Torque at the Tr Differential

Figure 33: Sheet 06_Interface_DataElements – Interface Specification

Short name	Initiator WP	Minimal Bits Size recommended	Resolution	Physical Lower Limit	Physical Upper Limit	Offset	Unit	Milestone	no description	
									no	description
Tq13	10.3	Sint15	1.00000	-10000.00000	10000.0000	0.00000	Nm	S2MS4		
Tq14	10.3	Uint14	1.00000	0.00000	10000.0000	0.00000	Nm	S2MS4		
Tq2	10.3	Sint16	1.00000	-20000.00000	20000.0000	0.00000	Nm	S1MS4		
Tq3	10.3	Uint13	4.00000	0.00000	20000.0000	0.00000	Nm	S1MS4		
Tq4	10.3	Uint15	4.00000	0.00000	80000.0000	0.00000	Nm	S1MS4		
TqGrdt1	10.3	Uint16	16.00000	0.00000	1000000.0000	0.00000	Nm/s	S1MS4		
DampnCoeff1	10.3	Sint15	1.00000	-10000.00000	10000.0000	0.00000	Ns/m	S1MS1		

Figure 34: Sheet 07_DataTypes_ContinuousValue - ContinuousValue DataType definition

Unit Name (short name)	Long Name	Unit Display Name	Description	Physical Dimension							FACTOR-SI-TO-UNIT	OFFSET-SI-TO-UNIT	
				electric current	luminous intensity	time	mass	amount of substance	thermodynamic temperature	length			
MtrPerSec	Meter Per Second	m/s	SI derived unit of velocity									1	0
MtrPerSecSqd	Meter Per Second Squared	m/s ²	SI derived unit of acceleration									1	0
Nwt	Newton	N	SI derived unit of force									1	0
NwtMtr	Newton Meter	Nm	SI derived unit of moment of force (torque = kg m ² /s ²)									1	0
NwtMtrPerSec	Newton Meter Per Second	Nm/s	SI derived unit of time-derivative of torque									1	0
Rad	Radian	rad	SI derived unit of plane angle (dimensionless)									1	0
RadPerSec	Radian Per Second	rad/s	SI derived unit of angular velocity									1	0
RadPerSecSqd	Radian Per Second Squared	rad/s ²	SI derived unit of angular acceleration									1	0

Figure 35: Sheet 13_Units - Unit Definition

In above diagrams the Interface assignment information flow in AI Table is shown by Red rectangle boxes.

In ideal case of standardization, there should not be any open ports, but in current release of the AI Table some connections are left open. So in “Top Level” open ports are allowed although it is not a desired situation. Open ports are those ports which could be defined as provider “P” only or receiver “R” only and there is no closed connection established between SWCs across domains.

5.1.3 Sheets 050xxxxx

These worksheet contents are very similar to the contents explained in the “TopLevel” worksheet. For each of the domains a worksheet exists with the format “050x_<Domain Name / Composite composition>” (e.g. 0501_Body) which contains the intra domain software composition data exchange matrix. Following this first worksheet the domain can have several worksheets “050xy_<Domain composition type name (Decomposed composition)>” (e.g. 050101_CentralLocking) will have the intra sub compositions / SW components data exchange matrix. For all the domains

worksheet naming and organization is arranged in the manner as explained above with example.

Below diagram is example for the for “Body” Domain worksheet, in this diagram for simplicity limited information is shown

Below diagram shows the “Body” domain composition work sheets

PortInterface Shortname	Shortname of Port	Initiator WP	Consistency Check			Body	LockgCen	IntrLi									
							LockgCen	IntrLi									
						S3MS4	S3MS4	S3MS4									
							CentralLocking	InteriorLight									
				Central Locking	Interior Light												
			0	18	10	10.1			10.1								
			multiple Provider	no Provider	no Receiver	P	R	P	R	core cond opt	IV	P	R	core cond opt	IV	P	
WshngCmd1	ActvnOfWshrFmt	10.1															
SeatFixaSts1	SeatFixaStsOf[Seat]	ID/10.4				X											
 DomSide1 / 0501_Body / 050101_CentralLocking / 050102_InteriorLight / 050103_MirrorAdjustment / 050104_MirrorTinting / 050105_SeatAdjustmen 																	

Figure 36: Sheet 0501_Body – Body Domain Composition Worksheet

The first worksheet of each domain represents the Top level interface matrix for that domain. Corresponding worksheets of the domain are containing information about port and port interface connection matrix for each of the SW composition / component.

Each of the worksheets contains the following information

Port interface short name, port short name, port long name, port description are present in columns with headings “Port interface shortname” “Shortname of Port”, “Long name of Port” and “Description of port” respectively. After these columns administrative columns and consistency check results matrix columns are present (For consistency check understanding refer sheet 102_User_Documentation in AI Table). In headline row each domain shortname of component / composition type and shortname of component / composition prototype are mentioned in different columns depending upon the number of component / composition.

For each entry of port interface and its corresponding port, a inter domain link (data exchange connection) is established with a ‘X’ in column ‘P’ or ‘R’ as required. As per AUTOSAR definition there should be a unique provider, resulting in only one column “P” marked with “X” per entry. As data can be received by more than one port, several “R” columns can be marked with “X” per entry.

The column “core cond opt” was defined to represent the use case for the handling of variants in the AI; i.e. specifies the existence of the PortPrototype. This information is not yet created in the XML file.

The ‘IV’ column is used to specify the initial value in case the sending component is not yet initialized. If the sender also specifies an init value the receiver’s value will be used. This information is not created in the XML file generated by the macro.

The 'components specific description' is used to specify any additional remarks for the ports which will be considered for the description/introduction [Refer [1]] attribute for the PortPrototype in the XML model.

In some use cases, the P or R columns in the sheet could have the short name of a port written directly in the cell instead of a 'X'. This could happen if the short names of the ports are not identical in different domains or due to modeling of the component due to multiple instantiation.

For every port, its corresponding port interface is defined with the data elements (for SenderReceiver interface) or arguments for operations (for Client-Server interface).

Details of their definition and specification are available in the sheets 06_Interface_DataElements and 06_Interface_ClientServer as part of the interface definition.

Example diagrams for data information flow across other worksheets are available in Chapter 5.1.2.

5.1.4 Sheet 06_Interfaces_DataElements (Sender-Receiver-Interface)

This work book sheet contains all the Sender-Receiver-interfaces referenced in any of worksheets 05xx domain / composition. Each of the Sender-Receiver interfaces can be used in all the 05xx domain / composition worksheets more than once. One interface can also be used for different ports.

In this sheet, PortInterface shortname and PortInterface longname of each of the interfaces are present in columns with headings "SenderReceiverInterface ShortName" and "Long Name" respectively. After these columns administrative data columns and the "Description" column are present. The "Description" column contains the description of the PortInterface. For each of the interfaces at least one data element (Variable Data Prototype) has to be specified. It is also allowed to specify more than one data elements. At the moment in current AI Table a maximum of 6 data elements is used, but if required more data elements can be added. Every data element has data element name, DataType, description of the data element, queuing information and signal quality information in separate columns with headings "Name", "Type", "Description", "Queuing" and "Signal Qualifier" respectively. The 'Queuing' column indicates the way the DataElement must be processed at the receiver's side. TRUE: elements added to a queue in FIFO data structure; FALSE: last is best semantics applies. This information is not exported yet in the XML model and currently not used. The column 'Signal Qualifier' provides additional information on the quality of the signal and its representation. This information is not exported yet in the XML model, and its usage is still under discussion. Data type for each data element can be of continuous value or enumeration or array or record DataTypes, these are defined in the worksheets "07_DataTypes_ContinuousValue", "08_DataTypes_Enumeration", "09_DataTypes_Array", "11_DataTypes_Record" respectively.

In addition, this worksheet has columns marked with grey color headings which are used for consistency check results.

The diagram below shows the data information flow between the interfaces worksheet 06_Interfaces_DataElements and the DataType worksheets (e.g. 07_DataTypes_ContinuousValue) marked with red rectangular boxes.

A	B	C	D	G		H	I	J	K
SenderReceiverInterface ShortName	Long Name	Initiator WP	Milestone	1st data element					
				Name	Type	Description	Queueing	Signal Qualifier	
ActrOcoptPedSftyAdjPosn1	OccupentPedestrianSftyActuatorAdjustPosition	10.4	S2MS1	Posn	Perc4	0%: minimum position of actuator 100%: maximum			extended
ActrOcoptPedSftyAdjStfn1	OccupentPedestrianSftyActuatorAdjustStiffness	10.4	S2MS1	Stfn	Perc4	0%: minimum stiffness of actuator (airbag)			extended

Figure 37: Sheet 06_Interface_DataElements – AI Table Sender-Receiver-interfaces specification

A	B	C	D	E	F	G	H	I	J
Short name	Long name	Description	Initiator WP	Minimal Bits Size recommended	Resolution	Physical Lower Limit	Physical Upper Limit	Offset	Unit
Perc2	Percent2	Generic percent data type 2	10.3	UInt8	0.50000	0.00000	100.00000	0.00000	%
Perc4	Percent4	Generic percent data type 4	ID	UInt7	1.00000	0.00000	100.00000	0.00000	%

Figure 38: Sheet 07_DataTypes_ContinuousValue – AI Table ContinuousValue DataType definition

Unit Name (short name)	Long Name	Unit Display Name	Description	Physical Dimension							FACTOR SI-TO-UNIT	OFFSET SI-TO-UNIT	
				electric current	luminous intensity	time	mass	amount of substance	thermodynamic temperature	length			
Perc	Percent	%	a percentage is a way of expressing a number as a fraction of 100 (no SI unit)									100	0
PercPerSec	Percent Per Second	%s	time-derivative of percent									100	0
DegCgrd	Degree Centigrade	degC	temperature, no SI unit, (degC = Kelvin - 273.15)						1			1	-273.15
MtrRecpr	Meter Reciprocal	1m	SI derived unit of wave number for electromagnetic fields or curvature as reciprocal of curve radius									1	0

Figure 39: Sheet 13_Units - Unit definition

Since PortInterfaces are designed to support reusability, it is recommended to reuse already defined SenderReceiver PortInterfaces for PortPrototypes with same kind of information to be transported.

Figure 40 demonstrates the reusability of PortInterfaces. In the example shown below the PortInterface BodyRollAg1 is used by the PortPrototypes BodyRollAgAbsItEstimd, BodyRollAgRelEstimd and BodyRollAgRelMeasd. These PortPrototypes are received by the SW-Component Esc. The PortPrototypes BodyRollAgAbsItEstimd and BodyRollAgRelEstimd are provided by the SW-Component Susp and the PortPrototype BodyRollAgRelEstimd is provided by the SW-Component ChassisSnsr.

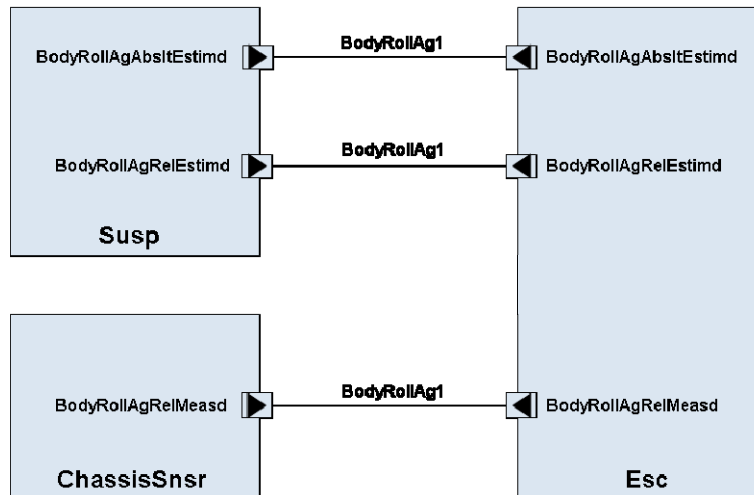


Figure 40: PortInterface reusability example

Similar to PortInterfaces, it is recommended to reuse already defined DataType

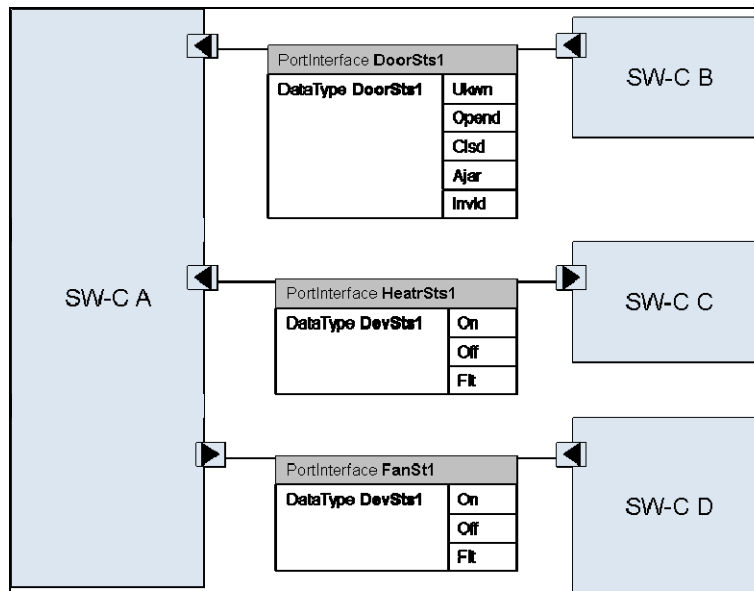


Figure 41: Reusability of DataTypes

Note that some rules in the SW-C and System Modeling Guide [9], such as NR044 and NR048, are defined in order to enable the reusability of the DataType.

5.1.5 Sheet 06_Interface_ClientServer

This worksheet contains all the ClientServer interfaces referenced in any of worksheets 05xx composite compositions / decomposed composition. Each of the defined ClientServer interfaces can be used in all the 05xx domain / composition worksheets more than once. One interface can also be used for different ports.

In this sheet, Port interface short name and Port interface long name of each of the interfaces are present in columns with headings “ClientServerInterface ShortName” and “Long Name” respectively. After these columns administrative data columns and the “Description” column are present. The “Description” column contains the description of the PortInterface. After these columns the operation names of the ClientServer interface are defined. This contains information, operation short name,

operation long name and operation description as defined in separate columns with headings “ShortName”, “Long Name” and “Description” respectively. One interface consists of multiple operations that will be specified in different rows. For each operation it is allowed to specify any number of arguments. For simplifying AI Table in current state any defined interface can specify maximum three arguments. Every argument has an argument short name, argument long name, argument description, argument DataType, type of the argument (input, output and input as well as output) in separate columns with headings “ArgumentName Shortname”, “Long Name”, “Description”, “DataType” and “IN/OUT/INOUT” respectively. The Data type for each element can be of continuous value or enumeration or array or record types, these are defined in the worksheets “07_DataTypes_ContinuousValue”, “08_DataTypes_Enumeration”, “09_DataTypes_Array”, “11_DataTypes_Record” respectively.

In addition, this work sheet contains columns marked with grey color headings, which are used for consistency check results

The diagram below shows the data information flow between the interfaces worksheet 06_Interface_ClientServer and the DataType worksheets marked with red rectangular boxes.

ClientServer Interface ShortName	Long Name	Initiator WP	Milestone	Description	Reference	OperationName			Argument1				
						ShortName	Long Name	Description	ArgumentName Shortname	Long Name	Description	DataType	IN/OUT/INOUT
TrsmRatGear1	Gear ratio for a given gear	10.2	S3MS4	Returns the gear ratio for a given gear Theoretical transmission ratio $i = \frac{\text{nttransmission_in}}{\text{nttransmission_out}}$ transmission_in = after converter transmission_out = gearbox out The gear ratio means the theoretical/physical ratio belonging to each gear and not any actual measured value (proposal for Continuously Variable Transmission(CVT): if there is a wide range for gear states, this value could deliver a theoretical value). Negative values: Reverse driving direction. Without considering the: 1.) axle ratio 2.) converter ratio 3.) High/Low-Range ratio REMARK: Default value after reset is 1.0		GetTrsmRatGear	Returns the gear ratio for a given gear	Returns the gear ratio for a given gear	Gear	Gear for which the ratio should be returned	Gear for which the ratio should be returned	Nr4	IN

Figure 42 : Sheet 06_Interface_ClientServer – AI Table ClientServer interfaces specification

Short name	Long name	Description	Initiator WP	Minimal Bits Size recommended	Resolution	Physical Lower Limit	Physical Upper Limit	Offset	Unit	Reference	Milestone
Ti2	Time 2	Generic data type for time	10.3	Uln8	0.10000	0.00000	3.1000	0.00000	s	ACC	S2MS4
U1	Voltage 1	Generic data type for voltage	D/10.1	Uln8	0.10000	0.00000	25.2000	0.00000	V	CL	S1MS4
Nr4	Number 4	Generic data type for number	10.2	Sim8	1.00000	-128.00000	127.0000	0.00000	-		S2MS4
IM2	Mass 2	Generic data type for mass	10.4	Uln8	1.00000	0.00000	250.0000	0.00000	kg	occupant weight	S3MS4

Figure 43 : Sheet 07_DataTypes_ContinuousValue – AI Table ContinuousValue DataType definition

The unit for the DataType Nr4 is ‘NoUnit’ and is represented as ‘-’. The DataType Nr4 is only a number and does not represent any physical quantity hence there is no unit.

Unit Name (short name)	Long Name	Unit Display Name	Description	Physical Dimension							FACTOR-SI-TO-UNIT	OFFSET-SI-TO-UNIT	Initiator VP	Milestone	
				electrical current	luminous intensity	time	mass	amount of substance	thermodynamic temperature	length					
MtrPerSecCubd	Meter Per Second Cubed	m/s ³	jerk (derived from SI units), also called jolt (esp. in British English), surge or lurch, is the rate of change of acceleration, more precisely, the derivative of acceleration with respect to time, the second derivative of velocity, or the third derivative of displacement. Unit for dimensionless continuous value data types like numbers, factors and coefficients												
NoUnit	No Unit	.													

Figure 44: Sheet 13_Units - AI Table Units Definition

Since PortInterfaces are designed to support reusability, it is recommended to reuse already defined ClientServer PortInterfaces for PortPrototypes with the same kind of information to be transported.

5.1.6 Sheet 07_DataTypes_ContinuousValue

This worksheet will have DataTypes with different resolution defined to be utilized in any of the port interfaces defined in any of worksheets 06xx or in complex DataTypes.

In this sheet, Data type short name, Data type long name and DataType description of each of the DataType definition entry are present in columns with headings “Short Name”, “Long Name” and “Description” respectively. Following the administrative data columns resolution and range details of the DataTypes are defined. These columns will have information on minimum number of bits requirement, resolution, physical lower and higher limits, offset value and physical unit are defined, with headings “Minimal Bits Size recommended”, “Resolution”, “Physical Lower Limit”, “Physical Upper Limit”, “Offset” and “Unit” respectively. The Minimal Bits Size recommended is calculated from the macro with inputs provided in the resolution, physical lower limit, higher limit and offset value columns. Units column utilizes the physical units defined in the work sheet “13_Units” referenced by “Unit Display Name”.

In addition this worksheet contains columns marked with grey color headings and these are used for consistency check results.

Figure 42 and Figure 43 present above show the data information flow between interfaces to DataTypes_ContinuousValue.

5.1.7 Sheet 08_DataTypes_Enumeration

This worksheet contains enumeration DataTypes with value to be utilized in any of the port interfaces defined in any of worksheets 06xx and complex DataTypes.

In this sheet, Enumeration (enum) Data type short name, Enum Data type long name and enum DataType description of each of the enum DataType definition entry are present in columns with headings “Data Type Name”, “Long Name” and “Description” respectively. Following the administrative data columns information on minimum bits required for enum DataType, value and name of enum elements and comments for each enum are defined in separate columns with headings “Minimal Number of Bits”, “value”, “name” and “comment” respectively. The Minimal Bits Size required for each enum DataType are calculated from the macro and this value is placed in the first

enum element row. Every enum data element value is defined in a separate row, hence more than one row belong to each enum DataType definition.

If the first line of an Enumeration data type definition contains an “X” in column “is boolean”, the generator will assign category “BOOLEAN” to the data type. Otherwise, category “VALUE” will be used. Any data type marked “is boolean” must consist of exactly two lines of definition, containing literal definitions for values 0 and 1.

In addition this worksheet contains columns marked with grey color headings and these are used for consistency check results.

The diagrams below show the data information flow between interfaces worksheet to the enum DataType sheets marked with red rectangular boxes.

SenderReceiverInterface ShortName	Long Name	Initiator WP	Milestone	1st data element				
				Name	Type	Description	Queueing	Signal Qualifier
EngTracCtrlActv1	EngineTractionControlActive1	10.3	S3MS4	EngTracCtrlActv	Boolean			
EntryReq1		ID/10.1	S3MS4	Req	EntryEasy1			

Figure 45: Sheet 06_Interface_DataElements – AI Table SenderReceiver Interfaces specification

Data Type Name	Long Name	Description	Initiator WP	Milestone	Enumeration values			
					value	name	comment	is boolean
MemBtn1			10.1	S1MS4	5	Btn5		
EntryEasy1		Indicates the easy entry status of the front/rear seats	10.1	S3MS4	3	0	Idle	no user request
EntryEasy1			10.1	S3MS4	1	EntryActvEasyFnt	allow easy entry from the front of the seat	
EntryEasy1			10.1	S3MS4	2	EntryRelsEasyFnt		
EntryEasy1			10.1	S3MS4	3	EntryActvEasyRr	allow easy entry behind the seat	
EntryEasy1			10.1	S3MS4	4	EntryRelsEasyRr		
EntryEasy1			10.1	S3MS4	0	EveStsPassd		
EveSts1	EventStatus	Event status	10.1	S1MS1	2			

Figure 46: Sheet 08_DataTypes_Enumeration – AI Table non-Boolean enumeration DataType definitions

SenderReceiverInterface ShortName	Long Name	Initiator WP	Milestone	Description	1st data element	
					Name	Type
EgyMngtSts1	EnergyManagementStatus	ID/10.1	S5MS2	This interface is used to optimize the energy available in the vehicle. This interface provides the information whether all, basic or only keep-alive components/functionalties are allowed to run.	Sts	EgyMngtSts1
EngHoodSts1	EngineHoodStatus	10.1	S5MS2	Provides information whether the engine hood is open or closed. As long as the status is valid the corresponding value should be provided.	Sts	OpenClsd1
EngN1	Actual Engine Speed	ID/10.2	S3MS4	The actual rotational speed of the engine crankshaft.	EngN	N1

Figure 47: 06_Interface_DataElements – AI Table SenderReceiver Interface with Boolean type Enumeration DataType

Data Type Name	Long Name	Description	Initiator WP	Milestone	Minimal	Enumeration values			
						value	name	comment	is boolean
EveSts			10.1	S1MS1		3	EveStsPreFaild		
OpndClsd1		Indicates open / closed status	10.1	S5MS1		0	Opnd		
OpndClsd1		Indicates the type of the generic request for the Central locking system without specifying the actuator and the source of the request	10.1	S5MS2		1	Clsd		
LockgCenReq1			10.1	S5MS2		0	Idle		

Figure 48: Sheet 08_DataTypes_Enumeration – AI Table Boolean enumeration DataType definitions

5.1.8 Sheet 09_DataTypes_Array

This worksheet contains array DataTypes defined to be utilized in any of the port interfaces defined in any of work book sheets 06xx and complex DataTypes.

In this sheet, Array DataType short name, Array DataType long name and Array DataType description of each of the Array DataType definition entry are present in columns with headings “Data Type Name”, “Long Name” and “Description” respectively. Following the administrative data columns information on array elements DataType short name and size of the array are defined in separate columns with headings “Type Name” and ” Number of Elements” respectively. Type name of an array can be found in one of the work sheets 07_DataTypes_ContinuousValue, 08_DataTypes_Enumeration, 09_DataTypes_Array and 11_DataTypes_Record.

In addition this worksheet contains columns marked with grey color headings and these are used for consistency check results.

The diagrams below show the data information flow between interfaces worksheet 06_Interfaces_DataElements to the array DataType sheets marked with red rectangular boxes.

SenderReceiverInterface ShortName	Long Name	Initiator WP	Milestone	Description	Reference	1st data element	
						Name	Type
VehBodyAVert1	VehicleBodyAccelerationVertical			Generic interface for vehicle vertical body accelerations. Refer to ISO 8855. The convention is Index 0 = Front left Index 1 = Front right Index 2 = Rear left Index 3 = Rear right		VehBodyAVert	VehBodyAVert1
WhlIAVert1	WheelAccelerationVertical	10.3	S3MS4	Vehicle vertical wheel acceleration. Refer to ISO 8855. The convention is Index 0 = Front left Index 1 = Front right Index 2 = Rear left Index 3 = Rear right		WhlIAVert	WhlIAVert1

Figure 49: Sheet 06_Interface_DataElements – AI Table Sender-Receiver-interfaces specification

Data Type Name	Long Name	Description	Initiator WP	Milestone	Type Name	Number of Element
VehBodyAVert1	VehicleBodyAccelerationVertical1	Generic datatype for vehicle vertical body accelerations. Refer to ISO 8855. The convention is Index 0 = Front left Index 1 = Front right Index 2 = Rear left Index 3 = Rear right	10.3	S3MS4	A2	4
WHLAVert1	WheelAccelerationVertical1	Vehicle vertical wheel acceleration. Refer to ISO 8855. The convention is Index 0 = Front left Index 1 = Front right Index 2 = Rear left Index 3 = Rear right	10.3	S3MS4	A5	4

Figure 50: Sheet 09_DataTypes_Array – AI Table array DataType definitions

Short name	Long name	Description	Initiator WP	Minimal Bits Size recommended	Resolution	Physical Lower Limit	Physical Upper Limit	Offset	Unit	Reference
A4	Acceleration4	Generic acceleration data type 4	10.3	SInt6	0.00098	-18.00000	18.00000	0.00000	m/s^2	Esc
A2	Acceleration2	Generic acceleration data type 2	10.3	SInt4	0.00391	-30.00000	30.00000	0.00000	m/s^2	Suspension
A5	Acceleration5	Generic acceleration data type 5	10.3	SInt7	0.00391	-200.00000	200.00000	0.00000	m/s^2	Suspension

Figure 51: Sheet 07_DataTypes_ContinuousValue – AI Table ContinuousValue DataType definitions

Unit Name (short name)	Long Name	Unit Display Name	Description	Physical Dimension						
				electrical current	luminous intensity	time	mass	amount of substance	thermodynamic temperature	length
MtrPerSec		m/s	velocity (derived from SI units)							
MtrPerSecSq		m/s^2	acceleration (derived from SI units)							

Figure 52: Sheet 13_Units - Unit Definition

5.1.9 Sheet 11_DataTypes_Record

This worksheet contains record DataTypes in which each of the record elements / entries may have different (sub) DataTypes. These are similar DataType definitions like the “C” language structure types. These record DataTypes are defined to be utilized in any of the port interfaces defined in any of work book sheets 06xx and complex DataTypes.

In this sheet, record DataType short name, record DataType long name and record DataType description of each of the record DataType definition entry are present in columns with headings “Record Type Name”, “Long Name” and “Description” respectively. Following the administrative data columns information on number of record data elements in a defined DataType, name of the record element, (sub) DataType of the record element and comments for each element are defined with headings “Number of element”, “Name”, “Type Name” and ” Comment” respectively. Record DataTypes are utilized to store the multiple values of different DataTypes. The record DataTypes can have one or more record elements, each record element

will have a different shortname and may have different (sub) DataTypes. For every record DataType first row will have the number of the elements defined in the column with heading “Number of element”. Each of the record elements is defined in separate rows. Record elements (sub) DataType definition short name can be found in one of the following work sheets 07_DataTypes_ContinuousValue, 08_DataTypes_Enumeration, 09_DataTypes_Array and 11_DataTypes_Record.

In addition this worksheet contains columns marked with grey color headings and these are used for consistency check results.

The diagram below shows the data information flow between interfaces worksheet 06_Interfaces_DataElements to the record DataType worksheet marked with red rectangular boxes.

SenderReceiverInterface ShortName	Long Name	Initiator WP	Milestone	1st data element				
				Name	Type	Description	Queueing	Signal Qualifier
CmdForSnsr1	CommandForSensor	10.1	S3MS4	Cmd	SnsrAlrmCmd1			
CogPosnEstimd1	CogPositionEstimated1	10.3	S2MS4	CogPosnEstimd	Len5			

Figure 53: Sheet 06_Interface_DataElements – AI Table Sender-Receiver-interfaces specification

Record Type Name	Long Name	Description	Initiator WP	Milestone	Number of element	1st data element		
						Name	Type Name	Comment
SnsrAlrmCmd1	SensorAlarmCommand	Separate commands for each sensor: arm, disarm, arm immediate, exclude	10.1	S3MS4	4	ArmCmd	ReqSts1	0-no arming command 1-arming command
SnsrAlrmCmd1			10.1	S3MS4		DisarmCmd	ReqSts1	0-no disarming command 1-disarming command
SnsrAlrmCmd1			10.1	S3MS4		SnsrExclsn	OnOff	0-exclude 1-not exclude
SnsrAlrmCmd1			10.1	S3MS4		ArmlmdtCmd	ReqSts1	0-no command to immediate arming 1-command for immediate arm

Figure 54: Sheet 11_DataTypes_Record – AI Table Record DataType definition

Data Type Name	Long Name	Description	Initiator WP	Milestone	Minimal Number of Bits	Enumeration values			
						value	name	comment	is boolean
OnOff1	On Off 1	Generic On/Off information	10.1	S3MS4	1	15	Prog15		
OnOff1			10.1	S3MS4		0	Off		x
OnOff1			10.1	S3MS4		1	On		
ReqSts1	Request Status 1	Generic request status 1	ID/10.3	S4MS4	1	0	NotReqd	Not requested.	
ReqSts1			ID/10.3	S4MS4		1	Reqd	Requested.	
TirePMonWarnSts1	TPMS Warning Status 1	Tire pressure monitoring warning stat	10.3	S4MS4	1	0	NoPLoss	No Pressure lost, normal operation.	
TirePMonWarnSts1			10.3	S4MS4		1	PLoss	Pressure lost, able to warn.	

Figure 55: Sheet 08_DataTypes_Enumeration – AI Table enumeration DataType definition

5.1.10 Sheet 13_Units

In this sheet, the units used for specification of the continuous DataTypes are defined. Units are referenced by the unit display names.

In this sheet, physical unit short name, physical unit long name, physical unit description and physical unit display name of each of the physical unit entry are present in columns with headings “Unit Name (short name)”, “Long Name”, “Unit Display Name” and “Description” respectively. After these columns, Physical Dimension list columns are present. The seven base quantities of International System of Units are represented between columns E and K. The factor and offset used for the units are mentioned in columns L & M. Following this, administrative data columns and columns marked with grey color headings used for consistency check results are present.

SenderReceiverInterface ShortName	Long Name	Description	1st data element		
			Name	Type	Description
DrvReqTqCluFast1	Driver request of clutch torque (fast torque path)	Torque at clutch requested by the driver to	DrvReqTqCluFast	Tq5	
DrvReqTqCluSlow1	Driver request of clutch torque (slow torque path)	Torque at clutch requested by the driver to	DrvReqTqCluSlow	Tq5	
DrvReqTqWhlFast1	Driver request of wheel	Wheel torque (to all the driven wheels) requested	DrvReqTqWhlFast	Tq7	

Figure 56: Sheet 06_Interface_DataElements – AI Table SenderReceiverInterfaces specification

Short name	Long name	Description	Initiator WP	Minimal Bits Size recommended	Resolution	Physical Lower Limit	Physical Upper Limit	Offset	Unit
SprgCoeff1	Spring coefficient	Generic spring coefficient	10.3	SInt19	1.00000	#####	150000.0000	0.00000	N/m
Tq1	Torque1	Generic torque data type 1	10.3	SInt14	0.00391	-30.00000	30.00000	0.00000	Nm
Tq15	Torque15	Generic torque data type 15	10.3	SInt13	0.00391	-10.00000	10.00000	0.00000	Nm
Tq5	Torque5	examples for usage: torque at clutch and torque at crankshaft signals	10.2	SInt16	0.06250	#####	2047.9375	0.00000	Nm
Tq6	Torque6	Generic torque data type 6	10.2	SInt16	0.06250	#####	0.00000	0.00000	Nm
Tq7	Torque7	examples for usage: torque at	10.2	SInt16	0.06250	#####	131071.0375	0.00000	Nm

Figure 57: Sheet 07_DataTypes_ContinuousValue – ContinuousValue DataType with a unit

Unit Name (short name)	Long Name	Unit Display Name	Description	Physical Dimension							FACTOR-SI-TO-UNIT	OFFSET-SI-TO-UNIT	
				electrical current	luminous intensity	time	mass	amount of substance	thermodynamic temperature	length			
MtrPerSecSqd	Meter Per Second Squared	m/s ²	SI derived unit of acceleration			-2					1	1	0
Nwt	Newton	N	SI derived unit of force			-2	1				1	1	0
NwtMtr	Newton Meter	Nm	SI derived unit of moment of force (torque = kg m ² /s ²)			-2	1				2	1	0
NwtMtrPerSec	Newton Meter Per Second	Nm/s	SI derived unit of time-derivative of torque			-3	1				2	1	0
Rad	Radian	rad	SI derived unit of plane angle (dimensionless)									1	0
RadPerSec	Radian Per Second	rad/s	SI derived unit of angular velocity			-1						1	0

Figure 58: Sheet 13_Units – Unit definition

5.1.11 Sheet 15_Redirected_Ports

This sheet is used for providing PortPrototypeBlueprint definitions for ports that have been renamed/redirected in the connection matrix specified in one of the 05 sheets.

As described above, it is possible for any given component prototype to locally rename or redirect the port name given at the beginning of a line by specifying a new port short name in the connection matrix instead of using an "X" in the "P" or "R" fields. In such cases, often the long name and description specified within the first columns in the row will not be correct for the PortPrototypeBlueprint generated for the renamed port.

In order to come to a full definition for these ports, they can either be defined in more detail within the context of another 05 composition sheet; or they can be defined in a generic (i.e. composition type independent way) by adding an entry in worksheet 15_Redirected_Ports.

When the macro "Update and Check" detects that one such redirected/renamed port does not have a proper definition, it will create a new entry in Sheet 15. However, it will leave the entries for "long name" and "description" blank; these need to be filled in by a human user. Until the entries are completed, the generator will signal error messages in consecutive runs.

If a port is defined more than once, i.e. either one of the 05 sheets contains a usable port definition or the same port is defined more than once in sheet 15, the generator will flag the error message "redundant port def. for redirected port" or "unused def. of redirected port". The user should then remove the duplicate entry from Sheet 15 in order to remove the error.

PortInterface Shortname	Shortname of Port	Longname of Port	Description of Port	Initiator WP	Milestone	PF (Unique)		multiple
						Initiator WP -->		
						transmissionAcknowledgmentTimeout	canInvalidate	
DoorSts1	DoorSts	DoorStatus	Information on a door, indicates whether this door is opened or closed	ID/10.1	S5MS2			
LockgCenReq1	KeyMech	FrontDoorMechanicalKey	Locking request with the mechanical key from the front	10.1	S5MS2			

Figure 59: Sheet 15_Redirected_Ports

5.2 Complete List of all Sheets of the AI Table

Title	Content
1. 01_History	History of changes to the table
2. 02_General Purposes	Contains a list of column titles in relation to the sheets; explanations are given in order to add or change data sets within the table
3. 04_Keywords	List of agreed keywords and their abbreviations along with their usage context description
4. 0500_TopLevel	Top level composition contains the information related to inter-domain port prototypes of major domain compositions (e.g. Body, Powertrain)
5. 0501_Body	(1) Body domain composition
6. 050101_CentralLocking	Central locking component
7. 050102_InteriorLight	Interior light component
8. 050103_MirrorAdjustment	Mirror adjustment component
9. 050104_MirrorTinting	Mirror tinting component
10. 050105_SeatAdjustment	Seat adjustment component
11. 05010501_Seat	Seat component
12. 0501050101_SeatAxis	SeatAxis component
13. 050106_ExteriorLight	Exterior light component
14. 050107_WindowControl	Window control component
15. 050108_WiperWasher	Wiper washer component
16. 05010801_NozzleHeater	Nozzle heater component
17. 05010802_Wiper	Wiper component
18. 05010803_Washer	Washer component
19. 05010804_WasherFluidTank	Washer fluid tank component
20. 05010805_RainSensing	Rain sensing component
21. 050109_AntiTheftSystem	Anti theft system component
22. 050110_HornControl	Horn control component
23. 050111_ConvertibleControl	Convertible control component
24. 050112_DefrostControl	Defrost control component
25. 050113_ParkDistanceControl	Park distance control component

	Title	Content
26.	050114_Immobilizer	Immobilizer component
27.	050115_BodySensors	Body sensors component
28.	050117_RemoteKeylessEntry	Keyless access component
29.	050118_KeyPad	Key pad component
30.	050119_PassiveEntry	Passive entry component
31.	050120_TerminalClampControl	Terminal clamp control component
32.	050121_SeatClimatization	Seat climatization component
33.	0502_Powertrain	(2) Powertrain composition
34.	050201_CombustionEngine	Combustion engine component
35.	050299_VehicleMotionforPt	Vehicle motion for Powertrain component
36.	0503_Chassis	(3) Chassis composition
37.	050301_CrsCtrlAndAcc	Cruise control and adaptive cruise control component
38.	0504_OPsafety	(4) Occupant safety composition
39.	0504001_OcctPedSftySnrsPool	Occupant and pedestrian safety sensor pool component
40.	0504002_I_OcctPedSftyActrPool	Occupant and pedestrian safety actuator pool component I
41.	0504002_II_OcctPedSftyActrPool	Occupant and pedestrian safety actuator pool component II
42.	0504002_III_OcctPedSftyActrPool	Occupant and pedestrian safety actuator pool component III
43.	0504102_SeatBltRmn	Seat belt reminder component
44.	0505_MM_T_HMI	(5) Multi media, telematics, human machine interface component
45.	06_Interface_DataElements	List of sender-receiver interface definitions
46.	06_Interface_ClientServer	List of client server interface definitions
47.	07_DataTypes_ContinuousValue	List of continuous value DataTypes
48.	08_DataTypes_Enumeration	List of enumeration DataTypes
49.	09_DataTypes_Array	List of array DataTypes
50.	11_DataTypes_Record	List of record DataTypes
51.	13_Units	List of units
52.	15_Redirected_Ports	List of definition of redirected ports
53.	101_Description	Explanation of results of consistency checks presented in summary dialogue box
54.	102_User_Documentation	Contains a list of available Visual Basic macros and their functionality.
The sheets below are administrative sheets and are filled automatically by the macros.		
55.	Compositions	Overview of compositions / components available in the AI Table
56.	Compositions_Err	Failed consistency check results of compositions and their decomposition.
57.	Instances	Overview of composition prototype

Title		Content
		(instances) available in the AI Table
58.	Instances_Err	Failed consistency check results of composition prototypes (instances)
59.	90_ReportMSDiagram	Diagram representing history of the distribution of table entries with model elements in relation to milestones. This data is generated by macros.
60.	90_ReportMSTable	Pivot table history of the distribution of table entries in relation to milestones and steps. This data is generated by macros.
61.	90_ReportMSTableNoSteps	Pivot table history of the distribution of table entries in relation to milestones. The step information will be excluded. This data is generated by macros.
62.	91_ReportErrDiagram	Diagram representing an overview on detected errors. This data is generated by macros.
63.	91_ReportErrTable	Pivot table of detected errors. This data is generated by macros.

6 Relationship between AI Table data and XML Output

The data from the AI Table, which reflects the structure defined in the AUTOSAR Meta-Model, is used to generate XML descriptions of the AUTOSAR Application Interfaces. The XML descriptions shall adhere to the AUTOSAR Schema [3] which is generated from the AUTOSAR Meta-Model [7].

6.1 Overview

6.1.1 Dependencies of XML Generation

Figure 60 illustrates the dependencies in the XML generation process. Currently, the AI Table reflects the structure for multiple releases in one database, i.e. for R3.0 and R4.0. This common database is then used by the AI XML Generation, to generate XML descriptions for each supported release. This approach implies that not all data from the AI Table will be reflected in all generated XML files as only the data for R4.0 is taken into account.

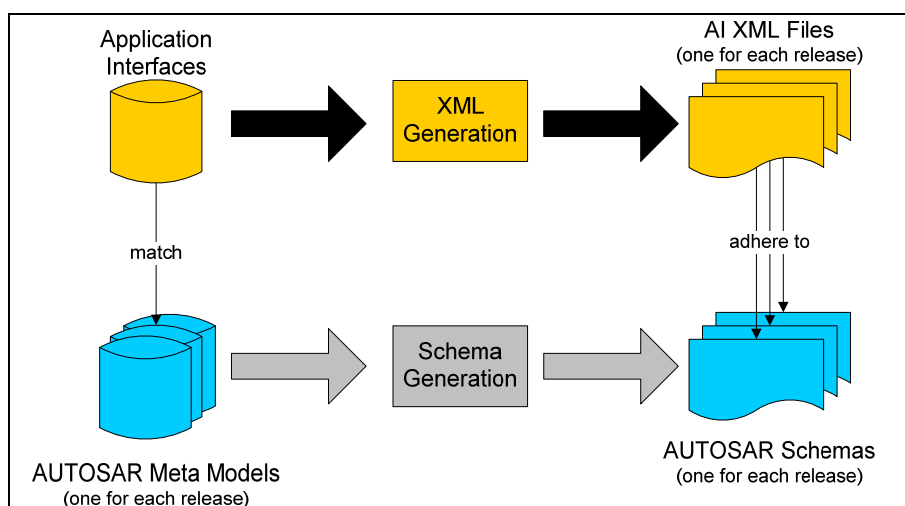


Figure 60: Dependencies in the XML generation process for application interfaces

6.1.2 Contents of Generated XML

The XML files contain descriptions of the following elements:

- Common elements
 - Package Structure and Categories
 - References
 - Instance References
 - Type References
 - Descriptions
- Composition Types with
 - Ports
 - Component Prototypes and
 - Connectors
- PortPrototypeBlueprints
 - BlueprintMappings for PortPrototypeBlueprints

- Interfaces
 - Sender-Receiver-Interfaces
 - Client-Server-Interfaces
 - BlueprintMappings for PortInterfaces
- Application Data Types, being
 - Primitive Types with constraints
 - Array Types
 - Record Types
 - BlueprintMappings for ApplicationDataTypes
 - Data Constraints
 - Computation Methods
 - BlueprintMappings for CompuMethods
- Units
 - Physical Dimension
- Keywords

6.1.2.1 File Distribution

The XML generated from the AI Table is divided into four .arxml files. Each of the categories STANDARD, BLUEPRINT and EXAMPLE constitute one .arxml file and the fourth .arxml file contains the Keywords.

6.1.3 Schema Structure

In order to understand the XML generation, it is necessary to understand the relation between meta-model and schema. Generally, the schema contains an `xsd:group` for each class of the meta-model. The group contains all attributes of the class, including aggregations and references as sequence of `xsd:element`. Concrete classes (in contrast to abstract classes) also have a corresponding `xsd:complexType`. These are sequences of all inherited groups from parent elements.

The general concept behind the structure of the schema will be described according to the example depicted in the following diagram, which shows the structure of the Unit element in the meta-model, including its inheritance hierarchy.

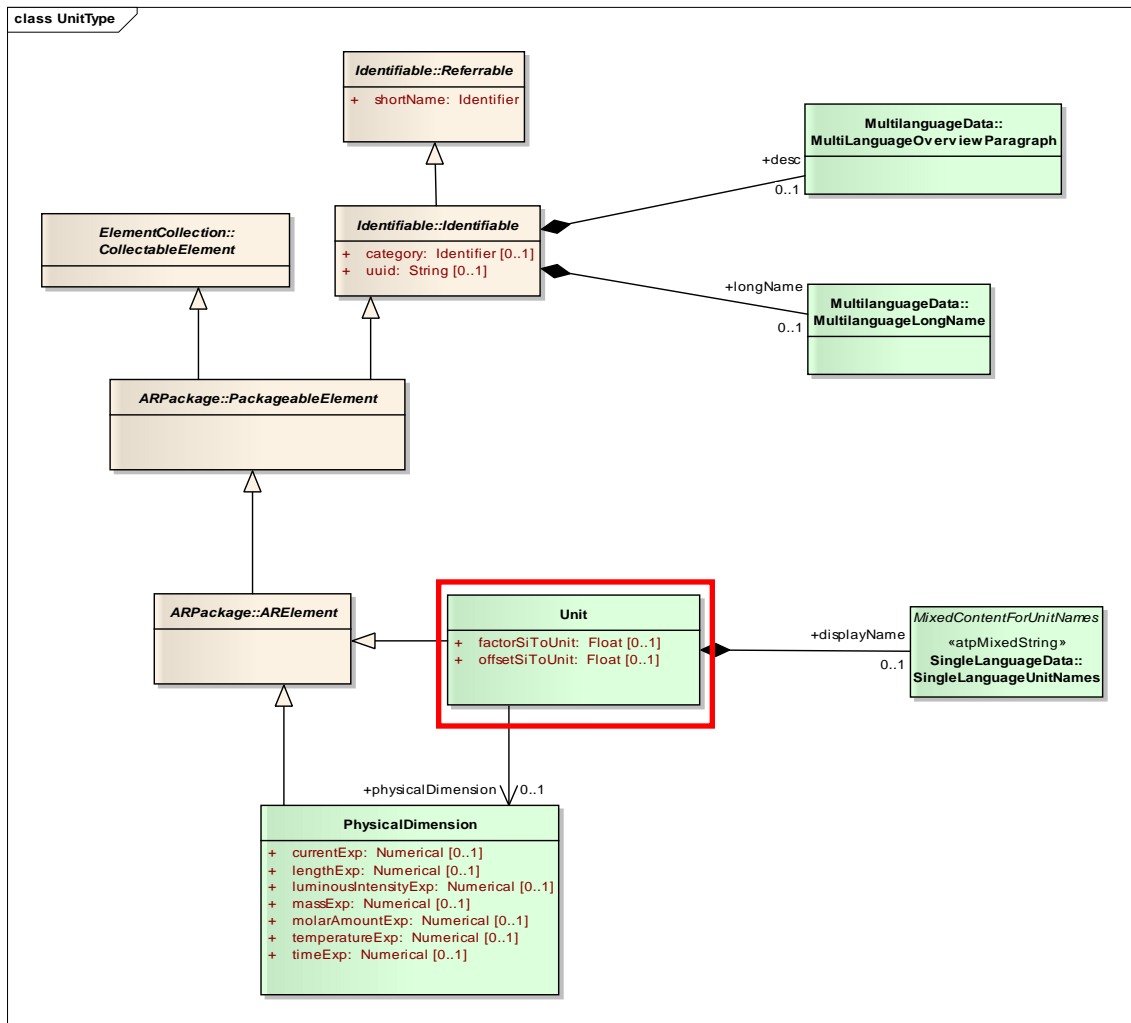


Figure 61: Cut-out from the meta-model defining the structure of the element Unit

This structure can be found in the following elements in the AUTOSAR schema:

```
<xsd:group name="UNIT">
  <xsd:annotation>...</xsd:annotation>
  <xsd:sequence>
    <xsd:element
      name="DISPLAY-NAME"
      type="AR:SINGLE-LANGUAGE-UNIT-NAMES"
      minOccurs="0"/>
    <xsd:element
      name="FACTOR-SI-TO-UNIT"
      type="xsd:double" minOccurs="0"/>
    <xsd:element
      name="OFFSET-SI-TO-UNIT"
      type="xsd:double" minOccurs="0"/>
    <xsd:element
      name="PHYSICAL-DIMENSION-REF"
      minOccurs="0">
      ...
    </xsd:element>
  </xsd:sequence>
</xsd:group>

<!-- complex type for class Units::Unit -->
<xsd:complexType name="UNIT" abstract="false" mixed="false">
  <xsd:sequence>
    <xsd:group ref="AR:AR-OBJECT"/>
  </xsd:sequence>
</xsd:complexType>
```

```

<xsd:group ref="AR:REFERRABLE" />
<xsd:group ref="AR:IDENTIFIABLE" />
<xsd:group ref="AR:COLLECTABLE-ELEMENT" />
<xsd:group ref="AR:PACKAGEABLE-ELEMENT" />
<xsd:group ref="AR:AR-ELEMENT" />
<xsd:group ref="AR:UNIT" />
</xsd:sequence>
...
</xsd:complexType>

```

This structuring into groups (for all classes including abstract ones) and complex types (for concrete classes only) leads to the particularity, that only concrete classes can be used on XML instance level, and that the inheritance hierarchy is not visible on instance level. The following example shows a unit from table “13_Units” on instance level, this table only contains attributes from Identifiable:

```

<UNIT>
  <SHORT-NAME>DegCgrd</SHORT-NAME>
  <LONG-NAME><L-4 L="EN">Degree Centigrade</L-4></LONG-NAME>
  <DESC><L-2 L="EN">temperature, no SI unit, (degC = Kelvin - 273.15)</L-2></DESC>
  <DISPLAY-NAME>degC</DISPLAY-NAME>
  <FACTOR-SI-TO-UNIT>1</FACTOR-SI-TO-UNIT>
  <OFFSET-SI-TO-UNIT>-273.15</OFFSET-SI-TO-UNIT>
  <PHYSICAL-DIMENSION-REF DEST="PHYSICAL-DIMENSION"
  BASE="PhysicalDimensions">T1</PHYSICAL-DIMENSION-REF>
</UNIT>

```

Details on the relation of meta-model and AUTOSAR Schema can be found in the Model Persistence Rules for XML [5]. See also Figure 60.

The following sections describe in detail, how the AI Table is related to the elements on XML instance level. Details about the relation of AI Table and meta-model are described in Chapter 4.

All following descriptions refer to the AUTOSAR R4.0 schema.

6.2 Common Elements

6.2.1 Package Structure

The XML content is structured into hierarchical packages. The top-level package is named **AUTOSAR** and contains one package named **AI Specification**. Beneath this package the output is structured into 13 different packages.

The different packages under AI Specification are

- DataConstrs: Package of the category STANDARD, contains all the data constraints
- PhysicalDimensions: Package of the category STANDARD, contains all the physical dimensions
- Units: Package of the category STANDARD, contains all the standardized units
- KeywordSets: Package of the category STANDARD, contains all the standardized keywords
- ApplicationDataTypes_Blueprint: Package of the category BLUEPRINT, contains all ApplicationDataTypes

- CompuMethods_Blueprint: Package of the category BLUEPRINT, contains all computation methods
- PortInterfaces_Blueprint: Package of the category BLUEPRINT, contains all PortInterface Blueprints
- PortPrototypeBlueprints_Blueprint: Package of the category BLUEPRINT, contains all PortPrototypeBlueprints
- SwComponentTypes_Example: Package of the category EXAMPLE, contains all SwComponentTypes
- BlueprintMappingSets_Example: Package of the category EXAMPLE, contains BlueprintMappingSets for all Blueprint elements
- ApplicationDataTypes_Example: Package of the category EXAMPLE, exists in this package only as a copy of ApplicationDataTypes_Blueprint elements
- PortInterfaces_Example: Package of the category EXAMPLE, exists in this package only as a copy of PortInterfaces_Blueprint
- CompuMethods_Example: Package of the category EXAMPLE, exists in this package only as a copy of CompuMethods_Blueprint elements

The XML extract below shows the package structure of each of these categories.

AUTOSAR_MOD_AISpecification_Blueprint.arxml

```
<?xml version="1.0" encoding="UTF-8"?>
  <AUTOSAR
    xmlns="http://autosar.org/schema/r4.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://autosar.org/schema/r4.0 AUTOSAR_4-0-3.xsd"
  >
    <ADMIN-DATA>
      <LANGUAGE>EN</LANGUAGE>
      <USED-LANGUAGES>
        <L-10 L="EN" xml:space="default">English</L-10>
      </USED-LANGUAGES>
    </ADMIN-DATA>
    <AR-PACKAGES>
      <AR-PACKAGE>
        <SHORT-NAME>AUTOSAR</SHORT-NAME>
        <LONG-NAME><L-4 L="EN">AUTOSAR</L-4></LONG-NAME>
      </AR-PACKAGES>
      <AR-PACKAGE>
        <SHORT-NAME>AISpecification</SHORT-NAME>
      </AR-PACKAGES>
      <AR-PACKAGE>
        <SHORT-NAME>
          ApplicationDataTypes_Blueprint
        </SHORT-NAME>
        <CATEGORY>BLUEPRINT</CATEGORY>
        <REFERENCE-BASES> ... </REFERENCE-BASES>
        <ELEMENTS> ... </ELEMENTS>
      </AR-PACKAGE>
      <AR-PACKAGE>
        <SHORT-NAME>
          CompuMethods_Blueprint
        </SHORT-NAME>
        <CATEGORY>BLUEPRINT</CATEGORY>
        <REFERENCE-BASES> ... </REFERENCE-BASES>
        <ELEMENTS> ... </ELEMENTS>
      </AR-PACKAGE>
      <AR-PACKAGE>
        <SHORT-NAME>
          PortInterfaces_Blueprint
```

```

        </SHORT-NAME>
        <CATEGORY>BLUEPRINT</CATEGORY>
        <REFERENCE-BASES> ... </REFERENCE-BASES>
        <ELEMENTS> ... </ELEMENTS>
    </AR-PACKAGE>
    <AR-PACKAGE>
        <SHORT-NAME>
            PortPrototypeBlueprints_Blueprint
        </SHORT-NAME>
        <CATEGORY>BLUEPRINT</CATEGORY>
        <REFERENCE-BASES> ... </REFERENCE-BASES>
        <ELEMENTS> ... </ELEMENTS>
    </AR-PACKAGE>
</AR-PACKAGES>
</AR-PACKAGE>
</AR-PACKAGES>
</AR-PACKAGE>
</AR-PACKAGES>
</AUTOSAR>

```

AUTOSAR_MOD_AISpecification_Example.arxml

```

<?xml version="1.0" encoding="UTF-8"?>
<AUTOSAR
  xmlns="http://autosar.org/schema/r4.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://autosar.org/schema/r4.0 AUTOSAR_4-0-3.xsd"
>
  <ADMIN-DATA>
    <LANGUAGE>EN</LANGUAGE>
    <USED-LANGUAGES>
      <L-10 L="EN" xml:space="default">English</L-10>
    </USED-LANGUAGES>
  </ADMIN-DATA>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>AUTOSAR</SHORT-NAME>
      <LONG-NAME><L-4 L="EN">AUTOSAR</L-4></LONG-NAME>
      <AR-PACKAGES>
        <AR-PACKAGE>
          <SHORT-NAME>AISpecification</SHORT-NAME>
          <AR-PACKAGES>
            <AR-PACKAGE>
              <SHORT-NAME>
                ApplicationDataTypes_Example
              </SHORT-NAME>
              <CATEGORY>EXAMPLE</CATEGORY>
              <REFERENCE-BASES> ... </REFERENCE-BASES>
              <ELEMENTS> ... </ELEMENTS>
            </AR-PACKAGE>
            <AR-PACKAGE>
              <SHORT-NAME>
                BlueprintMappingSets_Example
              </SHORT-NAME>
              <CATEGORY>EXAMPLE</CATEGORY>
              <REFERENCE-BASES> ... </REFERENCE-BASES>
              <ELEMENTS> ... </ELEMENTS>
            </AR-PACKAGE>
            <AR-PACKAGE>
              <SHORT-NAME>CompuMethods_Example</SHORT-NAME>
              <CATEGORY>EXAMPLE</CATEGORY>
              <REFERENCE-BASES> ... </REFERENCE-BASES>
              <ELEMENTS> ... </ELEMENTS>
            </AR-PACKAGE>
            <AR-PACKAGE>
              <SHORT-NAME>PortInterfaces_Example</SHORT-NAME>
              <CATEGORY>EXAMPLE</CATEGORY>

```

```

        <REFERENCE-BASES> ... </REFERENCE-BASES>
        <ELEMENTS> ... </ELEMENTS>
    </AR-PACKAGE>
    <AR-PACKAGE>
        <SHORT-NAME>SwComponentTypes_Example</SHORT-NAME>
        <CATEGORY>EXAMPLE</CATEGORY>
        <REFERENCE-BASES> ... </REFERENCE-BASES>
        <ELEMENTS> ... </ELEMENTS>
    </AR-PACKAGE>
</AR-PACKAGES>
</AR-PACKAGE>
    </AR-PACKAGES>
</AR-PACKAGE>
</AR-PACKAGES>
</AUTOSAR>

```

AUTOSAR_MOD_AISpecification_Keyword.arxml

```

<?xml version="1.0" encoding="UTF-8"?>
<AUTOSAR
  xmlns="http://autosar.org/schema/r4.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://autosar.org/schema/r4.0 AUTOSAR_4-0-3.xsd"
>
  <ADMIN-DATA>
    <LANGUAGE>EN</LANGUAGE>
    <USED-LANGUAGES>
      <L-10 L="EN" xml:space="default">English</L-10>
    </USED-LANGUAGES>
  </ADMIN-DATA>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>AUTOSAR</SHORT-NAME>
      <LONG-NAME><L-4 L="EN">AUTOSAR</L-4></LONG-NAME>
    </AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>AISpecification</SHORT-NAME>
    </AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>KeywordSets</SHORT-NAME>
      <CATEGORY>STANDARD</CATEGORY>
      <ELEMENTS> ... </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AR-PACKAGE>
</AR-PACKAGES>
</AUTOSAR>

```

AUTOSAR_MOD_AISpecification_Standard.arxml

```

<?xml version="1.0" encoding="UTF-8"?>
<AUTOSAR
  xmlns="http://autosar.org/schema/r4.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://autosar.org/schema/r4.0 AUTOSAR_4-0-3.xsd"
>
  <ADMIN-DATA>
    <LANGUAGE>EN</LANGUAGE>
    <USED-LANGUAGES>
      <L-10 L="EN" xml:space="default">English</L-10>
    </USED-LANGUAGES>
  </ADMIN-DATA>
  <AR-PACKAGES>

```

```

<AR-PACKAGE>
  <SHORT-NAME>AUTOSAR</SHORT-NAME>
  <LONG-NAME><L-4 L="EN">AUTOSAR</L-4></LONG-NAME>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>AISpecification</SHORT-NAME>
      <AR-PACKAGES>
        <AR-PACKAGE>
          <SHORT-NAME>DataConstrs</SHORT-NAME>
          <CATEGORY>STANDARD</CATEGORY>
          <REFERENCE-BASES> ... </REFERENCE-BASES>
          <ELEMENTS> ... </ELEMENTS>
        </AR-PACKAGE>
        <AR-PACKAGE>
          <SHORT-NAME>PhysicalDimensions</SHORT-NAME>
          <CATEGORY>STANDARD</CATEGORY>
          <ELEMENTS> ... </ELEMENTS>
        </AR-PACKAGE>
        <AR-PACKAGE>
          <SHORT-NAME>Units</SHORT-NAME>
          <CATEGORY>STANDARD</CATEGORY>
          <REFERENCE-BASES> ... </REFERENCE-BASES>
          <ELEMENTS> ... </ELEMENTS>
        </AR-PACKAGE>
      </AR-PACKAGES>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AUTOSAR>
    
```

6.2.2 References

The generated XML for the Application Interfaces consistently makes use of reference-bases and relative paths for all referencing. A relative path is identified by not starting with a slash (“/”). The following XML snippet shows the reference to a port of a composition type. The **DEST**-attribute defines the type of the reference XML element, the **BASE**-Attribute references the nearest reference-base defined in any parent package, and the content defines the reference target, in this case the port **WipgSpdIntlFromHmi** in the composition type **WiprWshr** from the package **/AUTOSAR/AISpecification/SwComponentTypes_Example**.

Example for the definition of reference base:

```

<REFERENCE-BASE>
  <SHORT-LABEL>SwComponentTypes</SHORT-LABEL>
  <IS-DEFAULT>>false</IS-DEFAULT>
  <IS-GLOBAL>>false</IS-GLOBAL>
  <BASE-IS-THIS-PACKAGE>>false</BASE-IS-THIS-PACKAGE>
  <PACKAGE-REF DEST="AR-PACKAGE">
    /AUTOSAR/AISpecification/SwComponentTypes_Example
  </PACKAGE-REF>
</REFERENCE-BASE>
    
```

Example for the usage of the reference base:

```

<DELEGATION-SW-CONNECTOR>
  <SHORT-NAME>WipgSpdIntlFromHmiToWipgSpdIntlFromHmiOfWiprWshrMgr</SHORT-NAME>
  <INNER-PORT-IREF>
    <R-PORT-IN-COMPOSITION-INSTANCE-REF>
      <CONTEXT-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE"
        BASE="SwComponentTypes">WiprWshr/WiprWshrMgr</CONTEXT-COMPONENT-REF>
    </R-PORT-IN-COMPOSITION-INSTANCE-REF>
  </INNER-PORT-IREF>
</DELEGATION-SW-CONNECTOR>
    
```

```

        <TARGET-R-PORT-REF DEST="R-PORT-PROTOTYPE"
BASE="SwComponentTypes">WiprWshrMgr/WipgSpdIntlFromHmi</TARGET-R-PORT-REF>
        </R-PORT-IN-COMPOSITION-INSTANCE-REF>
    </INNER-PORT-IREF>
    <OUTER-PORT-REF DEST="R-PORT-PROTOTYPE"
BASE="SwComponentTypes">WiprWshr/WipgSpdIntlFromHmi</OUTER-PORT-REF>
</DELEGATION-SW-CONNECTOR>
    
```

6.2.3 Instance References

AUTOSAR XML allows to reference elements from the type definition for a particular instance of the type using instance references. E.g. component prototypes do not define ports, but only reference their composition type, which defines ports. If it is required to reference this port, a reference to the context element is needed. The reference contains the instance and the target element. For a port, the instance is the component prototype, and the target element is the port definition in the composition type. The following example defines a reference to the port **WipgSpdIntlFromHmi** at the component prototype

/AUTOSAR/AISpecification/SwComponentTypes_Example/WiprWshr/WiprWshrMgr, which is defined in the composition type

/AUTOSAR/AISpecification/SwComponentTypes_Example/WiprWshrMgr.

```

<DELEGATION-SW-CONNECTOR>
    <SHORT-NAME>WipgSpdIntlFromHmiToWipgSpdIntlFromHmiOfWiprWshrMgr</SHORT-
NAME>
    <INNER-PORT-IREF>
        <R-PORT-IN-COMPOSITION-INSTANCE-REF>
            <CONTEXT-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE"
BASE="SwComponentTypes">WiprWshr/WiprWshrMgr</CONTEXT-COMPONENT-REF>
            <TARGET-R-PORT-REF DEST="R-PORT-PROTOTYPE"
BASE="SwComponentTypes">WiprWshrMgr/WipgSpdIntlFromHmi</TARGET-R-PORT-REF>
            </R-PORT-IN-COMPOSITION-INSTANCE-REF>
        </INNER-PORT-IREF>
        <OUTER-PORT-REF DEST="R-PORT-PROTOTYPE"
BASE="SwComponentTypes">WiprWshr/WipgSpdIntlFromHmi</OUTER-PORT-REF>
    </DELEGATION-SW-CONNECTOR>
    
```

6.2.4 Type References

In case the target element is referenced as type of the source element, AUTOSAR XML uses type references, i.e. a ***-TREF**-element, e.g. the following snippet references the element

/AUTOSAR/AISpecification/ApplicationDataTypes_Blueprint/WipgSpdIntl1 as the type of the data prototype **Req**.

```

<VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>Req</SHORT-NAME>
    <TYPE-TREF
        DEST="APPLICATION-PRIMITIVE-DATA-TYPE"
        BASE="ApplicationDataTypes">
        WipgSpdIntl1
    </TYPE-TREF>
</VARIABLE-DATA-PROTOTYPE>
    
```

6.2.5 Descriptions

Descriptions are not simply put into one description element, but parsed and split into multiple different elements, where indicated. The following rules apply to the parsing of description fields:

- Blank lines separate sections of documentation/description (Hint: line-breaks shall be introduced by Alt+Enter = Chr(10))

In XML, these description fields are mapped into the following two different elements:

- The first section goes to the **DESC** element, which is supposed to contain a brief description.
- The subsequent sections go to the **INTRODUCTION** element as separate sub-elements
 - Sections starting with a line that ends with a colon (:) and that is completely capitalized (e.g. REMARK:) will become **NOTE** elements with the first line being the **LABEL**, the rest a **P** element
 - Sections without label will become simple **P** elements within the **INTRODUCTION**. Additional information for the specific PortPrototype can be added here.
 - Sections starting with a star ("*") or a hyphen ("-") become list items. If the previous section is not a list item, a list element will be started
 - Sections starting with a blank will become part of a verbatim environment. If the previous section is not part of a verbatim environment, the verbatim environment will be started

These verbatim environments from a cell would be translated to the following structure in XML.

Text from Cell:

```
Returns the gear ratio for a given gear

Theoretical transmission ratio
i = ntransmission_in / ntransmission_out

transmission_in = after converter
transmission_out = gearbox out

The gear ratio means the theoretical/physical ratio belonging to each gear and
not any actual measured value (proposal for Continuously Variable
Transmission(CVT): if there is a wide range for gear states, this value could
deliver a theoretical value).

Negative values: Reverse driving direction.

Without considering the:
* axle ratio
* converter ratio
* High/Low-Range ratio

REMARK:
Default value after reset is 1.0
```

XML Structure:

```
<DESC><L-2 L="EN">Returns the gear ratio for a given gear</L-2></DESC>
<INTRODUCTION>
  <P><L-1 L="EN">
    Theoretical transmission ratio i = ntransmission_in /
    ntransmission_out
  </L-1></P>
  <P><L-1 L="EN">
    transmission_in = after converter transmission_out = gearbox out
  </L-1></P>
  <P><L-1 L="EN">
```

The gear ratio means the theoretical/physical ratio belonging to each gear and not any actual measured value (proposal for Continuously Variable Transmission(CVT): if there is a wide range for gear states, this value could deliver a theoretical value).

```

</L-1></P>
<P><L-1 L="EN">
    Negative values: Reverse driving direction.
</L-1></P> <P><L-1 L="EN">
    Without considering the:
</L-1></P>
<LIST TYPE="UNNUMBER">
    <ITEM><P><L-1 L="EN"> axle ratio </L-1></P></ITEM>
    <ITEM><P><L-1 L="EN"> converter ratio </L-1></P></ITEM>
    <ITEM><P><L-1 L="EN"> High/Low-Range ratio </L-1></P></ITEM>
</LIST>
<NOTE>
    <LABEL><L-4 L="EN">REMARK</L-4></LABEL>
    <P><L-1 L="EN">Default value after reset is 1.0 </L-1></P>
</NOTE>
</INTRODUCTION>
    
```

6.3 Component Types

Data for composition types are collected in the “05”-sheets. The rows at the top define composition types, while the rows below define the ports and connectors of composition types.

Each “05”-sheet defines one outer composition type (yellow columns) and multiple inner components, called component prototypes (blue columns). Each component prototype must reference a component (composition) type. If this type is not declared on another “05”-sheet as outer composition type (referenced by the hyperlink) it is defined locally. In the latter case the composition type is defined the same as the component prototype and shall not be reused in other sheets.

1	PortInterface ShortName	Shortname of Port	Initiator WP	WiprWshr		WiprWshrMgr	NozHeatr	WiprWshrEnaDi	Wipr	Wshr	Wshr											
				P	R	WiprWshrMgr	NozHeatrRe, Noz	WiprWshrEnaDi	WiprFrnt, WiprR	WshrFrnt	WshrRe											
2	3	4	5	S3MS4		S3MS4	SIMS4	S3MS4	SIMS4	SIMS4	SIMS4											
6				Wipr Washer Manager	Nozale Heater	Wipr Washer Enable Disable	Wipr Front, Wiper Rear	Washer Front	Washer Rear													
7	10.1		10.1		10.1		10.1		10.1		10.1											
8	P		R		P	R	core cond opt	IV	P	R	core cond opt	IV	P	R	core cond opt	IV	P	R	core cond opt	IV		
34	WipgSpdIntReq1	WipgSpdIntFromHmi	ID/10.1		X	X																
35	WshngReq1	WshngReFromHmi	ID/10.1		X	X																
36	WipgReq1	WipgReFromHmi	ID/10.1		X	X																
37	WiprPosnCmnd1	WiprPosnReqFromHmi	ID/10.1		X	X																
38	BattU1	BattU	ID/10.1		X	X																
39	OperModSts1	OperMod	ID/10.1		X	X																
40	WiprSts1	WiprStsFrnt	ID/10.1	X		X																
41	NozHeatrCmnd1	NozHeatrActonOf[NozHeatr]	10.1			X																
42	NozHeatrSts1	NozHeatrStsOf[NozHeatr]	10.1			X																
43	WshngCmnd1	ActvnOfWshngCmndOfWshrFrnt	10.1	X		X																
44	WshngCmnd1	ActvnOfWshngCmndOfWshrRe	10.1			X																
45	WshngCmnd1	ActvnOfWshngCmndOfWshrHdlamp	10.1			X																
46	WshrSts1	WshrStsOfWshrFrnt	10.1			X																
47	WshrSts1	WshrStsOfWshrRe	10.1			X																
48	WshrSts1	WshrStsOfWshrHdlamp	10.1			X																
49	WiprWipgCmnd1	WipgCmndFor[Wipr]	10.1			X																
50	WiprPosnCmnd1	WiprPosngFor[Wipr]	ID/10.1			X																

Figure 62: Sheet 050108_WiperWasher – Example specification from the AI Table for the composition type WiprWshr

6.3.1 Composition Types

The XML generator creates per sheet one composition type for the yellow column and a composition type for each blue column without hyperlink (the composition types for the blue columns *with* hyperlink are created later, when iterating over the linked "05"-sheets). The definition is written to the package **/AUTOSAR/AISpecification/SwComponentTypes_Example**. The composition type is defined by its ports (green columns), components (blue columns) and connectors (connector matrix with X's). The composition short-name is taken from the first row in the yellow columns as shown in Figure 62 (Cell Z1).

The following XML snippet shows the XML generated for the **WiprWshr** component:

```
<COMPOSITION-SW-COMPONENT-TYPE>
  <SHORT-NAME>WiprWshr</SHORT-NAME>
  <PORTS>
    ... (See Section 6.3.2)
  </PORTS>
  <COMPONENTS>
    ... (See Section 6.3.3)
  </COMPONENTS>
  <CONNECTORS>
    ... (See Section 6.3.4)
  </CONNECTORS>
</COMPOSITION-SW-COMPONENT-TYPE>
```

The following sections describe the elements from the three collections Ports, Components and Connectors.

6.3.2 Ports

The lower part of the “05”-sheets defines ports and connectors. The green columns define the ports, the right part (below the components) define existence and connections of ports. E.g. row 35 in the screenshot from Figure 62 defines a required port **WipgSpdIntlFromHmi** for composite component **WiprWshr** (marked by an X in Cell AA35). The mark in column AA from Figure 62 results in an **R-Port-Prototype** item in the ports collection of the composite component type **WiprWshr** as follows:

```
<R-PORT-PROTOTYPE>
  <SHORT-NAME>WipgSpdIntlFromHmi</SHORT-NAME>
  <LONG-NAME><L-4 L="EN">Wiping Speed Interval From Hmi</L-4></LONG-NAME>
  <REQUIRED-INTERFACE-TREF
    DEST="SENDER-RECEIVER-INTERFACE"
    BASE="PortInterfaces">
    WipgSpdIntlReql
  </REQUIRED-INTERFACE-TREF>
</R-PORT-PROTOTYPE>
```

The referenced interface must be a valid interface from the 06*-sheets (Refer 5.1.4 and 5.1.5). The interface is referenced via a type-reference.

A similar port with the same name is generated for composition type **WiprWshrMgr** because of the mark in Cell AC35 in Figure 62. The (blue) port columns “core/cond/opt” and IV are currently not relevant for XML generation.

6.3.3 Components

The internal component prototypes (instances) are taken from the blue columns. Each component prototype has a short-name, taken from row 2 (e.g. AB2), and references a composition type in row 1 (e.g. AB1) as shown in Figure 62. The following XML snippet is generated for the **WiprWshrMgr** component prototype into the **<COMPONENTS>** collection of the composition type **WiprWshr**:

```
<SW-COMPONENT-PROTOTYPE>
  <SHORT-NAME>WiprWshrMgr</SHORT-NAME>
  <LONG-NAME><L-4 L="EN">Wiper Washer Manager</L-4></LONG-NAME>
  <DESC><L-2 L="EN">
    Wiper Washer Manager commands Wiper and Washers of the vehicle
  </L-2></DESC>
```

```

<TYPE-TREF
  DEST="COMPOSITION-SW-COMPONENT-TYPE"
  BASE="SwComponentTypes">
  WiprWshrMgr
</TYPE-TREF>
</SW-COMPONENT-PROTOTYPE>
    
```

Via a **TYPE-TREF**, the component prototype references a composition type, which is generated according to the prototype definition, because the cell AB1 shown in Figure 62 does not contain a hyperlink.

In case of multiple instances, such a description will be generated for each instance from the comma-separated list in row 2 (e.g. AG2).

Another possibility to declare multiple instances can be achieved while defining the same component more than once with different prototype names like in the following example:

AV	AW	AX	AY	AZ	BABB	BC	BD	BE	BF	BG	BH	BI	BJ
		Wshr				Wshr					Wshr		
		WshrFrnt				WshrRe					WshrHdlamp		
		S1MS4				S1MS4					S1MS4		
		Washer Front				Washer Rear					Washer Head Lamp		
		Washer module commands the washers of the vehicle				Washer module commands the washers of the vehicle					Washer module commands the washers of the vehicle		

Figure 63: Multiple instance with one instance per column

6.3.4 Connectors

Information about connectors is taken from the connector matrix beginning in Cell Z8 (not visible in Figure 62 as the rows 8-34 are hidden). A connection can be declared with the value X or a specific short name itself. Special values like empty cells or the literal "N/A" won't establish a connector. Connectors are defined as follows:

Delegation Connectors are created for each X in the blue columns that has the same direction as the X in the yellow column, e.g. the X's in cells AA35 and AC35 in Figure 62 will result in the following delegation connector:

```

<DELEGATION-SW-CONNECTOR>
  <SHORT-NAME>WipgSpdIntlFromHmiToWipgSpdIntlFromHmiOfWiprWshrMgr</SHORT-NAME>
  <INNER-PORT-IREF>
    <R-PORT-IN-COMPOSITION-INSTANCE-REF>
      <CONTEXT-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE"
      BASE="SwComponentTypes">WiprWshr/WiprWshrMgr</CONTEXT-COMPONENT-REF>
      <TARGET-R-PORT-REF DEST="R-PORT-PROTOTYPE"
      BASE="SwComponentTypes">WiprWshrMgr/WipgSpdIntlFromHmi</TARGET-R-PORT-REF>
    </R-PORT-IN-COMPOSITION-INSTANCE-REF>
    </INNER-PORT-IREF>
    <OUTER-PORT-REF DEST="R-PORT-PROTOTYPE"
    BASE="SwComponentTypes">WiprWshr/WipgSpdIntlFromHmi</OUTER-PORT-REF>
  </DELEGATION-SW-CONNECTOR>
    
```

The generator creates the short name of the delegation connector according to the rule defined below;

```

<outerPortName>To<innerPortName>Of<InnerComponentPrototypeName>
    
```

In the above example `<outerPortName>` is `WipgSpdIntlFromHmi`, `<innerPortName>` is `WipgSpdIntlFromHmi`, and `<InnerComponentPrototypeName>` is `WiprWshrMgr`. Therefore, the short name of the delegation connector shown above is `WipgSpdIntlFromHmiToWipgSpdIntlFromHmiOfWiprWshrMgr`.

The inner port references the port from the component prototype found in the blue column using an instance reference (see Section 6.2.3). Note, that the context of the IREF is the component prototype **WiprWshrMgr** inside the composition **WiprWshr**, while the target port references the component type **WiprWshrMgr** in the general package for all composition types `SwComponentTypes`, thus the different path prefixes. The outer port references the port from the yellow column, belonging to the composition type **WiprWshr**.

Assembly Connectors are created for each required port (mark in the R-column) and the corresponding P-Port of internal component prototypes from the connector matrix, e.g. for cells AW43 and AB43 from Figure 62:

```
<ASSEMBLY-SW-CONNECTOR>
  <!-- sheet=050108_WiperWasher, row=43, p_col=28, r_col=48-->
  <SHORT-
NAME>ActvnOfWshngCmdOfWshrFrntOfWiprWshrMgrToActvnOfWshngCmdOfWshrFrnt</SHORT-
NAME>
  <PROVIDER-IREF>
    <CONTEXT-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE"
BASE="SwComponentTypes">WiprWshr/WiprWshrMgr</CONTEXT-COMPONENT-REF>
    <TARGET-P-PORT-REF DEST="P-PORT-PROTOTYPE"
BASE="SwComponentTypes">WiprWshrMgr/ActvnOfWshngCmdOfWshrFrnt</TARGET-P-PORT-
REF>
  </PROVIDER-IREF>
  <REQUESTER-IREF>
    <CONTEXT-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE"
BASE="SwComponentTypes">WiprWshr/WshrFrnt</CONTEXT-COMPONENT-REF>
    <TARGET-R-PORT-REF DEST="R-PORT-PROTOTYPE"
BASE="SwComponentTypes">Wshr/ActvnOfWshngCmd</TARGET-R-PORT-REF>
  </REQUESTER-IREF>
</ASSEMBLY-SW-CONNECTOR>
```

As can be seen from the snippet, both ports are referenced via instance references.

The generator creates the name of the assembly connector according to the rule below;

```
<P-Port-shortName>Of<P-Port-SW-C-Prototype-shortName>To<R-Port-
shortName>Of<R-Port-SW-C-Prototype-shortName>
```

In the context of the above example `<P-Port-shortName>` is `ActvnOfWshngCmdOfWshrFrnt`, `<P-Port-SW-C-Prototype-shortName>`, is `WiprWshrMgr`, `<R-Port-shortName>` is `ActvnOfWshngCmd`, and `<R-Port-SW-C-Prototype-shortName>` is `WshrFrnt`. Therefore, name of the assembly connector is `ActvnOfWshngCmdOfWshrFrntOfWiprWshrMgrToActvnOfWshngCmdOfWshrFrnt`.

Multiple Instantiation is a particularity in this case. The name of the port `WipgCmdFor[Wipr]` (cell B49 in Figure 62) is expanded according to the instance names from the column referenced by **Wipr**. All components with this port that are not defined in the referenced column, have multiple ports according to all instance names, e.g. `WipgCmdForWiprFrnt` and `WipgCmdForWiprRe` for **WiprWshrMgr**, while

the name is contracted to **WipgCmd** for all components from the column providing the instance iterator, i.e. the first column with **Wipr** in the first row.

Semantic Constraints: To guarantee a semantically correct generation of the AUTOSAR XML, there may be at most one X in a P-column of the blue components. This means that the Application Interfaces will support the generation of the AUTOSAR XML even though the constraints are not met. The following snapshot shows what is considered a semantic error:

PortInterface ShortName	Shortname of Port	Initiat or WP	Milestone	PPortComSpec (UnqueuedSenderComSpec)	Consistency Check	WiprWshr	WiprWshrMgr	NozHeatr	WiprWshrEnaDi										
						S3MS4	WiprWshrMgr	t, NozHeatrRe, Noz	WiprWshrEnaDi										
						WiperWasherManager	NozzleHeater	WiperWasherEnableDis	able										
						Wiper Washer Manager commands Wiper and Washers of the vehicle	Nozzle Heater is able to heat the washing liquid	Wiper Washer Enable Disable module takes decision to inhibit or not											
Initiator WP -->						1	0	0											
transmissionAck knowledge Timeout	canInvalidate	multiple Provider	no Provider	no Receiver		P	R	P	R	core cond opt	IV	P	R	core cond opt	IV	P	R	core cond opt	IV
WiprSts1	WiprStsFmt	ID/10.1	S4MS2	FALSE	X			X									X		

Figure 64: Sheet 050108_WiperWasher – Erroneously connected P-Port

As shown in Figure 64 there are two delegations specified (two P-Columns on blue background marked with X). As this represents an ambiguous definition of the port prototype at composition it is considered a violation of the semantic constraint.

Therefore, the existence of multiple X marks in P-Columns is considered a semantic error. Please note, that a generation of XML is nevertheless possible.

6.4 PortPrototypeBlueprints

It is not in the scope of the AUTOSAR Application Interfaces to define complete system compositions. All software component composition types are defined in a package with category EXAMPLE and meant only as illustration of usage of the standardized elements. However, it is in the scope of the Application Interfaces to describe the roles that interfaces can play in compositions. This can be done using PortPrototypeBlueprints, which define potential ports of a component type and can carry more attributes to pre-define values for usages of the blueprint, e.g. an initial value. For details on PortPrototypeBlueprints see the Standardization Template [2].

The PortPrototypeBlueprints are collected within a single package /AUTOSAR/AISpecification/PortPrototypeBlueprints_Blueprint:

```
<AR-PACKAGE>
  <SHORT-NAME>PortPrototypeBlueprints_Blueprint</SHORT-NAME>
  <CATEGORY>BLUEPRINT</CATEGORY>
  <REFERENCE-BASES>
    . . .
  </REFERENCE-BASES>
  <ELEMENTS>
    <PORT-PROTOTYPE-BLUEPRINT>
```

```

        <SHORT-NAME NAME-PATTERN="{anyName}"> AbsCtrlIntvg</SHORT-NAME>
        <LONG-NAME><L-4 L="EN">ABS Control Intervening</L-4></LONG-NAME>
        <DESC><L-2 L="EN"> Antilock Braking System's (ABS) control is
active (at least one wheel)</DESC>
        <INTERFACE-REF DEST="SENDER-RECEIVER-INTERFACE"
BASE="PortInterfaces">AbsCtrlIntvg1</INTERFACE-REF>
    </PORT-PROTOTYPE-BLUEPRINT>
    <PORT-PROTOTYPE-BLUEPRINT>
        <SHORT-NAME NAME-PATTERN="{anyName}">AbsFlgActv</SHORT-NAME>
        <LONG-NAME><L-4 L="EN">AbsControlActive</L-4></LONG-NAME>
        <DESC><L-2 L="EN">Anti Blocking Systems (ABS) control is active (at
least one wheel)</DESC>
        <INTERFACE-REF DEST="SENDER-RECEIVER-INTERFACE"
BASE="PortInterfaces">AbsCtrlIntvg1</INTERFACE-REF>
    </PORT-PROTOTYPE-BLUEPRINT>
    . . .
</ELEMENTS>
</AR-PACKAGE>
    
```

Since the blueprint mechanism is meant as an aid for the creation of PortPrototypes the AUTOSAR XML also provides a mapping mechanism which describes the relation between a blueprint and a prototype. This mechanism allows decoupling of the PortPrototype from the blueprint without interfering with architectural requirements. The mapping is specified as a sequence of pairs as part of the package `/AUTOSAR/AISpecification/SwComponentTypes_Example` like in the following example:

```

<BLUEPRINT-MAPPING-SET>
    <SHORT-NAME>PortPrototypeBlueprintMappings</SHORT-NAME>
    <BLUEPRINT-MAPS>
        <BLUEPRINT-MAPPING>
            <BLUEPRINT-REF DEST="PORT-PROTOTYPE-BLUEPRINT"
BASE="PortPrototypeBlueprints_Blueprint">AbsCtrlIntvg</BLUEPRINT-REF>
            <DERIVED-OBJECT-REF DEST="P-PORT-PROTOTYPE"
BASE="SwComponentTypes">Chassis/AbsCtrlIntvg</DERIVED-OBJECT-REF>
        </BLUEPRINT-MAPPING>
        <BLUEPRINT-MAPPING>
            <BLUEPRINT-REF DEST="PORT-PROTOTYPE-BLUEPRINT"
BASE="PortPrototypeBlueprints_Blueprint">AbsCtrlIntvg</BLUEPRINT-REF>
            <DERIVED-PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE"
BASE="SwComponentTypes">Body/AbsCtrlIntvg</DERIVED-OBJECT-REF>
        </BLUEPRINT-MAPPING>
        <BLUEPRINT-MAPPING>
            . . .
    </BLUEPRINT-MAPS>
</BLUEPRINT-MAPPING-SET>
    
```

6.5 PortInterfaces

The provided or required PortPrototype references a PortInterface defined in the AI Table's "06"-sheets, separated for Sender-Receiver- and Client-Server-Interfaces. Both sender-receiver-interfaces and client-server-interfaces are kept in the package `/AUTOSAR/AISpecification/PortInterfaces_Blueprint`. They are also part of `/AUTOSAR/AISpecification/PortInterfaces_Example`, but only as a copy of the blueprint so that they could be used for PortPrototypes.

6.5.1 Sender-Receiver-Interface

Figure 65 shows one interface from the sender-receiver-interface table, which defines short-name, long-name, description and the contained data elements (the

table is capable of capturing up to six data elements per sender receiver interface). The direction of the dataflow is defined by the port.

SenderReceiverInterface ShortName	Long Name	Initiator WP	Milestone	Description	1st data element				
					Name	Type	Description	Queueing	Signal Qualifier
WipgReq1	Wiping Request	10.1	S5MS4	User request for wiping to switch on one strike, interval, low speed or high speed wiping. As long as a wipe sequence is requested, the corresponding value in the table below should be set.	Req	UsrReqForWipg1			
WipgSpdIntlReq1	Wiping Speed Interval Request	10.1	S5MS4	Requests the interval speed. As long as a interval wipe sequence is requested the provided value of interval speed has to be used.	Req	WipgSpdIntl1			
WiprPosnCmd1	Wiper Position Command	10.1	S5MS4	Provides information to run in a predefined position. As long as the information is provided the actuator should run/ stay in this position. If there is a valid Window Wiper Command, this supersedes commands given via this interface.	PosnReqd	WiprPosnReqd1			

Figure 65: Structure of the sheet 06_Interface_DataElements

The XML generator generates the following output for the above table row:

```
<SENDER-RECEIVER-INTERFACE>
  <SHORT-NAME NAME-PATTERN="{anyName}">WipgSpdIntlReq1</SHORT-NAME>
  <LONG-NAME><L-4 L="EN">Wiping Speed Interval Request</L-4></LONG-NAME>
  <DESC><L-2 L="EN">Requests the interval speed. As long as a interval wipe
sequence is requested the provided value of interval speed has to be used.</L-
2></DESC>
  <IS-SERVICE>>false</IS-SERVICE>
  <DATA-ELEMENTS>
    <VARIABLE-DATA-PROTOTYPE>
      <SHORT-NAME>Req</SHORT-NAME>
      <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE"
BASE="ApplicationDataTypes">WipgSpdIntl1</TYPE-TREF>
    </VARIABLE-DATA-PROTOTYPE>
  </DATA-ELEMENTS>
</SENDER-RECEIVER-INTERFACE>
```

The information on queuing and signal qualifiers is currently not used for XML generation. The referenced type for each data element must be defined in the DataType sheets.

6.5.2 Client-Server-Interface

Figure 66 shows one interface from the client-server-interface table, which defines short-name, long-name, description and the contained operations (one operation per row, the table is capable of capturing up to three arguments per operation) and will merge all operations with the same interface short name into one interface.

	A	B	E	G	H	I	J	K	L	M	N
1				OperationName			Argument1				
2	ClientServer Interface ShortName	Long Name	Description	ShortName	Long Name	Description	Argument Name Shortname	Long Name	Description	Data Type	IN/OUT /INOUT
3	TrsmRatGear1	Gear ratio for a given gear	Returns the gear ratio for a given gear Theoretical transmission ratio $i = n_{\text{transmission_in}} / n_{\text{transmission_out}}$ transmission_in = after converter transmission_out = gearbox out The gear ratio means the theoretical/physical ratio belonging to each gear and not any actual measured value (proposal for Continuously Variable Transmission(CVT): if there is a wide range for gear states, this value could deliver a theoretical value). Negative values: Reverse driving direction. Without considering the: * axle ratio * converter ratio * High/Low-Range ratio REMARK: Default value after reset is 1.0	GetTrsmRatGear	Gear ratio for a given gear	Returns the gear ratio for a given gear	Gear	Gear	Gear for which the ratio should be returned	Nr4	IN
4											

Figure 66: Structure of the sheet 06_Interface_ClientServer

The XML generator generates the following output for the above example:

```

<CLIENT-SERVER-INTERFACE>
  <SHORT-NAME NAME-PATTERN="{anyName}">TrsmRatGear1</SHORT-NAME>
  <LONG-NAME><L-4 L="EN">Transmission: Gear ratio for a given gear</L-
4></LONG-NAME>
  <DESC><L-2 L="EN">Returns the gear ratio for a given gear</L-2></DESC>
  <INTRODUCTION> ... </INTRODUCTION>
  <IS-SERVICE>>false</IS-SERVICE>
  <OPERATIONS>
    <CLIENT-SERVER-OPERATION>
      <SHORT-NAME>GetTrsmRatGear</SHORT-NAME>
      <LONG-NAME><L-4 L="EN">Returns the gear ratio for a given gear</L-
4></LONG-NAME>
      <DESC><L-2 L="EN">Returns the gear ratio for a given gear</L-
2></DESC>
      <ARGUMENTS>
        <ARGUMENT-DATA-PROTOTYPE>
          <SHORT-NAME>Gear</SHORT-NAME>
          <LONG-NAME><L-4 L="EN">Gear for which the ratio should be
returned</L-4></LONG-NAME>
          <DESC><L-2 L="EN">Gear for which the ratio should be
returned</L-2></DESC>
          <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE"
BASE="ApplicationDataTypes">Nr4</TYPE-TREF>
          <DIRECTION>IN</DIRECTION>
        </ARGUMENT-DATA-PROTOTYPE>
        <ARGUMENT-DATA-PROTOTYPE>
          <SHORT-NAME>Rat</SHORT-NAME>
          <LONG-NAME><L-4 L="EN">Gear ratio of given gear</L-
4></LONG-NAME>
          <DESC><L-2 L="EN">Gear ratio of given gear</L-2></DESC>
          <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE"
BASE="ApplicationDataTypes">Fac1</TYPE-TREF>
          <DIRECTION>OUT</DIRECTION>
        </ARGUMENT-DATA-PROTOTYPE>
      </ARGUMENTS>
    </CLIENT-SERVER-OPERATION>
  </OPERATIONS>
</CLIENT-SERVER-INTERFACE>
    
```

For details on the generation of description and introduction elements see Section 6.2.5.

6.6 PortInterface Blueprints

Similar to PortPrototypeBlueprints, PortInterface blueprints are provided with the respective mapping as well. The mapping set “PortInterfaceBlueprintMappings” contains interface blueprint mappings to the corresponding interfaces. They’re defined in package **/AUTOSAR/AISpecification/BlueprintMappingSets_Example**:

```
<BLUEPRINT-MAPPING-SET>
  <SHORT-NAME>PortInterfaceBlueprintMappings</SHORT-NAME>
  <BLUEPRINT-MAPS>
    <BLUEPRINT-MAPPING>
      <BLUEPRINT-REF DEST="CLIENT-SERVER-INTERFACE"
BASE="PortInterfaces_Blueprint">
        TrsmRatGear1
      </BLUEPRINT-REF>
      <DERIVED-OBJECT-REF DEST="CLIENT-SERVER-INTERFACE"
BASE="PortInterfaces">
        TrsmRatGear1
      </DERIVED-PORT-INTERFACE-REF>
    </BLUEPRINT-MAPPING>
    <BLUEPRINT-MAPPING>
      <BLUEPRINT-REF DEST="SENDER-RECEIVER-INTERFACE"
BASE="PortInterfaces_Blueprint">
        ALgt2
      </BLUEPRINT-REF>
      <DERIVED-OBJECT-REF DEST="SENDER-RECEIVER-INTERFACE"
BASE="PortInterfaces">
        ALgt2
      </DERIVED-PORT-INTERFACE-REF>
    </BLUEPRINT-MAPPING>
    . . .
  </BLUEPRINT-MAPS>
</BLUEPRINT-MAPPING-SET>
```

This mapping shows that the package **/AUTOSAR/AISpecification/PortInterfaces_Blueprint** provides the blueprint interfaces which are derived within the EXAMPLE categorized package **/AUTOSAR/AISpecification/PortInterfaces_Example**.

6.7 Data Types

The interfaces reference DataTypes in data elements for sender-receiver interfaces and arguments for client-server interfaces. The DataTypes are defined in the sheets “Sheet 07_DataTypes_ContinuousValue”, “Sheet 08_DataTypes_Enumeration”, “Sheet 09_DataTypes_Array”, “Sheet 11_DataTypes_Record”.

All DataTypes are kept in the package **/AUTOSAR/AISpecification/ApplicationDataTypes_Blueprint**. They are also part of **/AUTOSAR/AISpecification/ApplicationDataTypes_Example**, but as a copy of the blueprint so that they could be used for PortInterfaces.

Application DataTypes define the data attributes which are needed from the application point of view, in order to exchange data between software components or between a software component and a measurement and calibration tool.

The AI Table does not standardize implementation DataTypes. For more details please refer [1].

6.7.1 Continuous Value Types

Short name	Long name	Description	Initiator WP	Minimal Bits Size recommended	Resolution	Physical Lower Limit	Physical Upper Limit	Offset	Unit
Perc8	Percent8	Generic percent data type 8	10.4	UInt16	0.00031	-5.00000	15.00000	5.00000	%
Perc7	Percent7	Generic percent data type 7	10.1	UInt14	0.10000	0.00000	1001.00000	0.00000	%
Perc1	Percent1	Generic percent data type 1	ID/10.2	UInt15	0.00391	0.00000	100.00000	0.00000	%
Perc2	Percent2	Generic percent data type 2	10.3	UInt8	0.50000	0.00000	100.00000	0.00000	%
Perc4	Percent4	Generic percent data type 4	ID	UInt7	1.00000	0.00000	100.00000	0.00000	%
Perc6	Percent6	Generic percent data type 6	10.3	SInt8	1.00000	-128.00000	127.00000	0.00000	%
Permil1	Permille1	Generic permille data type 1	10.1	UInt10	1.00000	0.00000	1000.00000	0.00000	%
PercGrdt1	PercentGradient1	Generic percent gradient data type 1	10.3	SInt15	0.06250	-1000.00000	1000.00000	0.00000	%/s
Crv1	Curvature1	Generic curvature data type 1	10.3	SInt16	3.0519E-06	-0.10000	0.10000	0.00000	1/m
P1	Pressure1	Generic pressure data type 1	10.3	UInt11	0.00781	0.00000	10.00000	0.00000	bar
T2	Temperature2	Generic temperature data type 2	10.2	SInt13	0.10000	-256.00000	255.90000	0.00000	degC
T1	Temperature1	Generic temperature data type 1	10.1	SInt9	0.50000	-50.00000	75.00000	0.00000	degC
T3	Temperature3	Generic temperature data type 3	10.3	UInt8	1.00000	-50.00000	200.00000	50.00000	degC

Figure 67: Structure of the sheet 07_DataTypes_ContinuousValue

The continuous values need to be scaled to integer numbers; the XML generator creates three elements for a continuous type, the type element itself, which references a SwDataDefProps defining the scale and a data constraint defining the limits of the integral values. The following snippets define the type **Perc8**.

The type element gives the name (short and long), description and uses the field “Minimal Bits Size recommended” to provide a recommended implementation type, as well as the Unit field for the unit reference. The references to computation method and data constraint are generated:

```

<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME NAME-PATTERN="{anyName}">Perc8</SHORT-NAME>
  <LONG-NAME><L-4 L="EN">Percent 8</L-4></LONG-NAME>
  <DESC><L-2 L="EN">Generic data type for percent</L-2></DESC>
  <CATEGORY>VALUE</CATEGORY>
  <SW-DATA-DEF-PROPS>
    <SW-DATA-DEF-PROPS-VARIANTS>
      <SW-DATA-DEF-PROPS-CONDITIONAL>
        <SW-CALIBRATION-ACCESS>READ-ONLY</SW-CALIBRATION-ACCESS>
        <COMPU-METHOD-REF DEST="COMPU-METHOD">
          BASE="CompuMethods">Perc8</COMPU-METHOD-REF>
        <DATA-CONSTR-REF DEST="DATA-CONSTR">
          BASE="DataConstrs">Perc8</DATA-CONSTR-REF>
        <SW-INTENDED-RESOLUTION>0.00031</SW-INTENDED-RESOLUTION>
        <UNIT-REF DEST="UNIT" BASE="Units">Perc</UNIT-REF>
      </SW-DATA-DEF-PROPS-CONDITIONAL>
    </SW-DATA-DEF-PROPS-VARIANTS>
  </SW-DATA-DEF-PROPS>
</APPLICATION-PRIMITIVE-DATA-TYPE>

```

Computation methods are also defined as blueprints. BlueprintMapping for computation methods are also provided to help projects create actual Computation methods from the blueprints. The BlueprintMappings for Computation methods are grouped in the set CompuMethodBlueprintMappings in the package BlueprintMappingSets_Example. The XML fragment below shows the BlueprintMapping for the CompuMethod Perc8.

```
<BLUEPRINT-MAPPING>
  <BLUEPRINT-REF DEST="APPLICATION-PRIMITIVE-DATA-TYPE"
    BASE="ApplicationDataTypes_Blueprint"
  >
    Perc8
  </BLUEPRINT-REF>
  <DERIVED-OBJECT-REF DEST="APPLICATION-PRIMITIVE-DATA-TYPE"
    BASE="ApplicationDataTypes"
  >
    Perc8
  </DERIVED-OBJECT-REF>
</BLUEPRINT-MAPPING>
```

The data constraints belong to the package /AUTOSAR/AISpecification/DataConstrs.

```
<DATA-CONSTR>
  <SHORT-NAME>Perc8</SHORT-NAME>
  <DATA-CONSTR-RULES>
    <DATA-CONSTR-RULE>
      <PHYS-CONSTRS>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">-5</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">15</UPPER-LIMIT>
        <UNIT-REF DEST="UNIT" BASE="Units">Perc</UNIT-REF>
      </PHYS-CONSTRS>
    </DATA-CONSTR-RULE>
  </DATA-CONSTR-RULES>
</DATA-CONSTR>
```

The data constraints for the lower and upper limit for signed ranges are calculated using to the below mentioned formulae:

$$\text{lowerLimit} = \text{Round}((\text{phys_lower_limit} - \text{offset}) / \text{factor})$$

$$\text{upperLimit} = \text{Round}((\text{phys_upper_limit} - \text{offset}) / \text{factor})$$

For unsigned ranges the below mentioned formulae are used:

$$\text{lowerLimit} = 0$$

$$\text{upperLimit} = \text{Round}((\text{phys_lower_limit} - \text{offset}) / \text{factor}) + \text{Round}((\text{phys_upper_limit} - \text{offset}) / \text{factor}) + 1$$

These lowerLimit and upperLimit are used for calculation of minimum required bits to represent the entire signal range.

The Computation methods belong to the package /AUTOSAR/AISpecification/CompuMethods_Blueprint.

```
<COMPU-METHOD>
  <SHORT-NAME NAME-PATTERN="{anyName}">Perc8</SHORT-NAME>
  <LONG-NAME><L-4 L="EN">Percent 8</L-4></LONG-NAME>
  <DESC><L-2 L="EN">Generic data type for percent</L-2></DESC>
  <CATEGORY>LINEAR</CATEGORY>
  <UNIT-REF DEST="UNIT" BASE="Units">Perc</UNIT-REF>
  <COMPU-PHYS-TO-INTERNAL>
```

```

<COMPU-SCALES>
  <COMPU-SCALE>
    <LOWER-LIMIT INTERVAL-TYPE="CLOSED">-5</LOWER-LIMIT>
    <UPPER-LIMIT INTERVAL-TYPE="CLOSED">15</UPPER-LIMIT>
    <COMPU-RATIONAL-COEFFS>
      <COMPU-NUMERATOR>
        <V>5</V>
        <V>1</V>
      </COMPU-NUMERATOR>
      <COMPU-DENOMINATOR>
        <V>0.00031</V>
      </COMPU-DENOMINATOR>
    </COMPU-RATIONAL-COEFFS>
  </COMPU-SCALE>
</COMPU-SCALES>
</COMPU-PHYS-TO-INTERNAL>
</COMPU-METHOD>

```

6.7.2 Enumeration Types

Data Type Name	Long Name	Description	Initiator WP	Milestone	Minimal Number of Bits	Enumeration values			
						value	name	comment	is boolean
TrsmTyp1		Information on standard transmission. Other transmission types on value 5 - 15	10.2	S1MS4	3	0 Mt		manual transmission	
TrsmTyp1			10.2	S1MS4		1 At		automatic transmission	
TrsmTyp1			10.2	S1MS4		2 /Amt		direct shift/ automated Mt	
TrsmTyp1			10.2	S1MS4		3 Cvt		continuously variable transmission	
TrsmTyp1		Information about the state of the shift	10.2	S1MS4		4 Det		twin-clutch gearbox or dual or double clutch transmission (D	

Figure 68: Structure of the sheet 08_DataTypes_Enumeration

All (consecutive) lines from the table with the same Data Type Name are contained by one enumeration type. The type references a computation method defining the literals, whose size is computed from the count of literals, and a data constraint for the base type, also reflecting the count of literals.

Note: The tag `<CATEGORY>` `</CATEGORY>` will contain either “BOOLEAN” or “VALUE” depending on whether the column “is boolean” is marked with an ‘x’ or not.

```

<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME NAME-PATTERN="{anyName}">TrsmTyp1</SHORT-NAME>
  <LONG-NAME><L-4 L="EN">Transmission Type</L-4></LONG-NAME>
  <DESC><L-2 L="EN">Information on standard transmission. Other transmission
types on value 5 - 15</L-2></DESC>
  <CATEGORY>VALUE</CATEGORY>
  <SW-DATA-DEF-PROPS>
    <SW-DATA-DEF-PROPS-VARIANTS>
      <SW-DATA-DEF-PROPS-CONDITIONAL>
        <SW-CALIBRATION-ACCESS>READ-ONLY</SW-CALIBRATION-ACCESS>
        <COMPU-METHOD-REF DEST="COMPU-METHOD">
          BASE="CompuMethods">TrsmTyp1</COMPU-METHOD-REF>
        <DATA-CONSTR-REF DEST="DATA-CONSTR">
          BASE="DataConstrs">TrsmTyp1</DATA-CONSTR-REF>
        </SW-DATA-DEF-PROPS-CONDITIONAL>
      </SW-DATA-DEF-PROPS-VARIANTS>
    </SW-DATA-DEF-PROPS>
  </APPLICATION-PRIMITIVE-DATA-TYPE>

```

The XML fragment below shows the BlueprintMapping for the Enumeration DataType.

```
<BLUEPRINT-MAPPING>
```

```

<BLUEPRINT-REF DEST="APPLICATION-PRIMITIVE-DATA-TYPE"
                BASE="ApplicationDataTypes_Blueprint"
>
    TrsmTyp1
</BLUEPRINT-REF>
<DERIVED-OBJECT-REF DEST="APPLICATION-PRIMITIVE-DATA-TYPE"
                    BASE="ApplicationDataTypes"
>
    TrsmTyp1
</DERIVED-OBJECT-REF>
</BLUEPRINT-MAPPING>
    
```

The literals of the enumeration type are encoded in a computation method, with one `COMPU-SCALE` for each literal. Descriptions of the literals are not parsed into multiple elements as described in Section 6.2.5. The long name of the computation method is copied from its respective data type's long name.

The Computation methods belong to the package `/AUTOSAR/AISpecification/CompuMethods_Blueprint`.

```

<COMPU-METHOD>
  <SHORT-NAME NAME-PATTERN="{anyName}">TrsmTyp1</SHORT-NAME>
  <LONG-NAME><L-4 L="EN">Transmission Type</L-4></LONG-NAME>
  <CATEGORY>TEXTTABLE</CATEGORY>
  <COMPU-INTERNAL-TO-PHYS>
    <COMPU-SCALES>
      <COMPU-SCALE>
        <DESC><L-2 L="EN">0 = Mt (manual transmission)</L-2></DESC>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">0</UPPER-LIMIT>
        <COMPU-CONST><VT>Mt</VT></COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <DESC><L-2 L="EN">1 = At (automatic transmission)</L-2></DESC>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">1</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">1</UPPER-LIMIT>
        <COMPU-CONST><VT>At</VT></COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <DESC><L-2 L="EN">2 = Amt (direct shift/ automated Mt)</L-
2></DESC>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">2</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">2</UPPER-LIMIT>
        <COMPU-CONST><VT>Amt</VT></COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <DESC><L-2 L="EN">3 = Cvt (continuously variable
transmission)</L-2></DESC>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">3</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">3</UPPER-LIMIT>
        <COMPU-CONST><VT>Cvt</VT></COMPU-CONST>
      </COMPU-SCALE>
      <COMPU-SCALE>
        <DESC><L-2 L="EN">4 = Dct (twin-clutch gearbox or dual or
double clutch transmission (Dct))</L-2></DESC>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">4</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">4</UPPER-LIMIT>
        <COMPU-CONST><VT>Dct</VT></COMPU-CONST>
      </COMPU-SCALE>
    </COMPU-SCALES>
  </COMPU-INTERNAL-TO-PHYS>
</COMPU-METHOD>
    
```

The data constraints limit the use of the base type to the really needed values and they belong to the package `/AUTOSAR/AISpecification/DataConstrs`.

```

<DATA-CONSTR>
  <SHORT-NAME>TrsmTyp1</SHORT-NAME>
  <DATA-CONSTR-RULES>
    <DATA-CONSTR-RULE>
      <PHYS-CONSTRS>
        <LOWER-LIMIT INTERVAL-TYPE="CLOSED">0</LOWER-LIMIT>
        <UPPER-LIMIT INTERVAL-TYPE="CLOSED">4</UPPER-LIMIT>
      </PHYS-CONSTRS>
    </DATA-CONSTR-RULE>
  </DATA-CONSTR-RULES>
</DATA-CONSTR>
    
```

6.7.3 Array Types

Data Type Name	Long Name	Description	Initiator WP	Milestone	Type Name	Number of Element
PrioOfObj1	Prioritisation Of Objects	Priority of target object for follow control is given by array position (higher priority with lower array index). If no particular object is selected, then ID=0 will be sent. Tire pressures at wheels.	10.3	S3MS4	Nr2	16
TirePPerWhl1	Tire Pressure Per Wheel 1	Convention is: - Index 0 = Front Left - Index 1 = Front Right - Index 2 = Rear Left - Index 3 = Rear Right - Index 4 = Spare Wheel Tire temperatures.	10.3	S5MS4	P1	5
TireTPerWhl1	Tire Temperature Per Wheel 1	Convention is: - Index 0 = Front Left - Index 1 = Front Right - Index 2 = Rear Left - Index 3 = Rear Right - Index 4 = Spare Wheel	10.3	S5MS4	T3	5

Figure 69: Structure of the sheet 09_DataTypes_Array

The XML generation is straightforward from the table entries, e.g. the XML for row 5 from Figure 69 would be:

```

<APPLICATION-ARRAY-DATA-TYPE>
  <SHORT-NAME NAME-PATTERN="{anyName}">TirePPerWhl1</SHORT-NAME>
  <LONG-NAME><L-4 L="EN">Tire Pressure Per Wheel 1</L-4></LONG-NAME>
  <DESC><L-2 L="EN">Tire pressures at wheels.</L-2></DESC>
  <CATEGORY>ARRAY</CATEGORY>
  <INTRODUCTION>
    <P><L-1 L="EN">Convention is: </L-1></P>
    <LIST TYPE="UNNUMBER">
      <ITEM><P><L-1 L="EN"> Index 0 = Front Left </L-1></P></ITEM>
      <ITEM><P><L-1 L="EN"> Index 1 = Front Right </L-1></P></ITEM>
      <ITEM><P><L-1 L="EN"> Index 2 = Rear Left </L-1></P></ITEM>
      <ITEM><P><L-1 L="EN"> Index 3 = Rear Right </L-1></P></ITEM>
      <ITEM><P><L-1 L="EN"> Index 4 = Spare Wheel </L-1></P></ITEM>
    </LIST>
  </INTRODUCTION>
  <SW-DATA-DEF-PROPS>
    <SW-DATA-DEF-PROPS-VARIANTS>
      <SW-DATA-DEF-PROPS-CONDITIONAL>
        <SW-CALIBRATION-ACCESS>READ-ONLY</SW-CALIBRATION-ACCESS>
      </SW-DATA-DEF-PROPS-CONDITIONAL>
    </SW-DATA-DEF-PROPS-VARIANTS>
  </SW-DATA-DEF-PROPS>
  <ELEMENT>
    <SHORT-NAME>TirePPerWhl1</SHORT-NAME>
    
```



```
<TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE"
BASE="ApplicationDataTypes">P1</TYPE-TREF>
<MAX-NUMBER-OF-ELEMENTS>5</MAX-NUMBER-OF-ELEMENTS>
</ELEMENT>
</APPLICATION-ARRAY-DATA-TYPE>
```

The BlueprintMapping for the array DataType is as shown in the XML fragment below.

```
<BLUEPRINT-MAPPING>
<BLUEPRINT-REF DEST="APPLICATION-ARRAY-DATA-TYPE"
BASE="ApplicationDataTypes_Blueprint"
>
TirePPERWhl1
</BLUEPRINT-REF>
<DERIVED-OBJECT-REF DEST="APPLICATION-ARRAY-DATA-TYPE"
BASE="ApplicationDataTypes"
>
TirePPERWhl1
</DERIVED-OBJECT-REF>
</BLUEPRINT-MAPPING>
```

Descriptions are parsed according to Section 6.2.5.

6.7.4 Record Types

	A	B	C	D	E	F	G	H	I
1	Record Type Name	Long Name	Description	Initiator WP	Milestone	Number of element			
2									
3							Name	Type Name	Comment
4	LockgCenSts1		Status of the Central Locking. The status undefined is valid if there are no feedback switches in the latch and the ECU lost the information of actual status	10.1	S1MS4				Length (bit) = 8 (see data type) Recommendations: Init value= {Undefd, NoTrigSrc}, can invalidate=No
5	LockgCenSts1			10.1	S1MS4		2 LockgSt TrigSrc	LockSt2 LockTrigSrc1	
6	DiagcLock1	DiagnosticLock	Diagnostic of the status of the locking. States if the lock is working or not.	10.1	S3MS4		2 Lock	LockActv1	Length (bit) = 3 Recommendations: Init value= {Off,Off}, can invalidate =No
7	DiagcLock1			10.1	S3MS4		Diagc	OnOff	
8	DiagcChdProtn1	DiagnosticChildProtection	Diagnostic of the status of the child protection. States if the child protection is working or not.	10.1	S3MS4		2 ChdProtn	ChdProtnCmd1	Length (bit) = 3 Recommendations: Init value= {Off,Off}, can invalidate =No
9	DiagcChdProtn1			10.1	S3MS4		Diagc	OnOff	
10	DiagcPullAid1	DiagnosticPullAid	Diagnostic of the status of the pull aid of the door locking mechanism. States if the pull aid is working or not.	10.1	S3MS4		2 PullAid	PullAidCmd1	Length (bit) = 3 Recommendations: Init value= {Off,Off}, can invalidate =No
11	DiagcPullAid1			10.1	S3MS4		Diagc	OnOff	

Figure 70: Structure of the sheet 11_DataTypes_Record

As with enumeration types, all consecutive lines with the same record type name are included in one record type, with one line for each record element. E.g. rows 6 and 7 from Figure 70 will result in the following XML.

```
<APPLICATION-RECORD-DATA-TYPE>
<SHORT-NAME NAME-PATTERN="{anyName}">DiagcLock1</SHORT-NAME>
<LONG-NAME><L-4 L="EN">Diagnostic Lock</L-4></LONG-NAME>
<DESC><L-2 L="EN">Diagnostic of the status of the locking. States if the
lock is working or not.</L-2></DESC>
<CATEGORY>STRUCTURE</CATEGORY>
<SW-DATA-DEF-PROPS>
<SW-DATA-DEF-PROPS-VARIANTS>
<SW-DATA-DEF-PROPS-CONDITIONAL>
<SW-CALIBRATION-ACCESS>READ-ONLY</SW-CALIBRATION-ACCESS>
```

```

        </SW-DATA-DEF-PROPS-CONDITIONAL>
    </SW-DATA-DEF-PROPS-VARIANTS>
</SW-DATA-DEF-PROPS>
<ELEMENTS>
    <APPLICATION-RECORD-ELEMENT>
        <SHORT-NAME>Lock</SHORT-NAME>
        <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE "
BASE="ApplicationDataTypes">LockActvnl</TYPE-TREF>
    </APPLICATION-RECORD-ELEMENT>
    <APPLICATION-RECORD-ELEMENT>
        <SHORT-NAME>Diagc</SHORT-NAME>
        <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE "
BASE="ApplicationDataTypes">OnOffl</TYPE-TREF>
    </APPLICATION-RECORD-ELEMENT>
</ELEMENTS>
</APPLICATION-RECORD-DATA-TYPE>

```

BlueprintMapping for Record Data Type is as shown below.

```

<BLUEPRINT-MAPPING>
    <BLUEPRINT-REF DEST="APPLICATION-RECORD-DATA-TYPE "
        BASE="ApplicationDataTypes_Blueprint "
    >
        DiagcLockl
    </BLUEPRINT-REF>
    <DERIVED-OBJECT-REF DEST="APPLICATION-RECORD-DATA-TYPE "
        BASE="ApplicationDataTypes "
    >
        DiagcLockl
    </DERIVED-OBJECT-REF>
</BLUEPRINT-MAPPING>

```

Descriptions are parsed according to Section 6.2.5.

6.8 Units

Unit Name (short name)	Long Name	Unit Display Name	Description	Physical Dimension							FACTOR-SI-TO-UNIT	OFFSET-SI-TO-UNIT	
				electric current	luminous intensity	time	mass	amount of substance	thermodynamic temperature	length			
Hz	Hertz	Hz	SI derived unit of frequency			-1						1	0
KiloGr	Kilo Gram	kg	SI base unit of mass				1					1	0
KiloHz	Kilo Hertz	kHz	SI derived unit of frequency with prefix			-1						0.001	0
KiloMtr	Kilo Meter	km	SI base unit of length with prefix							1		1	0
Mtr	Meter	m	SI base unit of length							1		1	0
MtrPerSec	Meter Per Second	m/s	SI derived unit of velocity			-1				1		1	0
MtrPerSecSqd	Meter Per Second Squared	m/s ²	SI derived unit of acceleration			-2				1		1	0
Newt	Newton	N	SI derived unit of force			-2	1			1		1	0

Figure 71: Structure of the sheet 13_Units

The Units belong to the package /AUTOSAR/AISpecification/Units. The generated XML is derived from the table entries straight-forward according to the following example. The Physical Dimensions are also generated in the XML and referenced to the relevant Units. The short name of the Physical Dimensions are derived according to the rules below

- Usage of already existing keyword abbreviations
- I for electrical current
- Cd for luminous intensity
- Ti for time
- M for mass
- Mol for amount of substance
- T for thermodynamic temperature
- Len for length
- Neg for negative values

Short names are created as concatenation of the dimensions.

Long names are constructed similar to short names only that the full words are used (Ex: Length, Mass, Time, Amount of Substance etc). The long names for the negative units shall use '-' instead of Negative

For dimensionless Units a PhysicalDimension "NoDimension" will be used.

```
<UNIT>
  <SHORT-NAME>KiloGr</SHORT-NAME>
  <LONG-NAME><L-4 L="EN">Kilo Gram</L-4></LONG-NAME>
  <DESC><L-2 L="EN">SI base unit of mass</L-2></DESC>
  <DISPLAY-NAME>kg</DISPLAY-NAME>
  <FACTOR-SI-TO-UNIT>1</FACTOR-SI-TO-UNIT>
  <PHYSICAL-DIMENSION-REF DEST="PHYSICAL-DIMENSION"
BASE="PhysicalDimensions" >
    M1
  </PHYSICAL-DIMENSION-REF>
</UNIT>
```

The Unit KiloGr refers to the physical dimension of M1, which is represented in XML as below.

```
<PHYSICAL-DIMENSION>
  <SHORT-NAME>M1</SHORT-NAME>
  <LONG-NAME><L-4 L="EN">Mass 1</L-4></LONG-NAME>
  <MASS-EXP>1</MASS-EXP>
</PHYSICAL-DIMENSION>
```

7 References

In this section, the references used in this document are listed.

7.1 Standard documents

- [1] Software Component Template
- [2] Standardization Template
- [3] AUTOSAR XML schema
- [4] Generic Structure Template
- [5] Model Persistence Rules for XML
- [6] AI Specification

7.2 Auxiliary documents

- [7] AUTOSAR Metamodel
- [8] Application Interface table (AI Table)
- [9] SW-C and System Modeling Guide
- [10] AUTOSAR Methodology
- [11] AUTOSAR domain explanation Body and Comfort
- [12] AUTOSAR domain explanation Powertrain
- [13] AUTOSAR domain explanation Chassis
- [14] AUTOSAR domain explanation Occupant and Pedestrian Safety
- [15] AUTOSAR domain explanation Multimedia, Telematics, Human Machine Interface.
- [16] Unique Names for Documentation, Measurement and Calibration