

| | |
|-----------------------------------|---|
| Document Title | Specification of Platform Types for Adaptive Platform |
| Document Owner | AUTOSAR |
| Document Responsibility | AUTOSAR |
| Document Identification No | 875 |

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

| Document Change History | | | |
|--------------------------------|----------------|----------------------------|---|
| Date | Release | Changed by | Description |
| 2019-03-29 | 19-03 | AUTOSAR Release Management | <ul style="list-style-type: none"> minor corrections / clarifications / editorial changes; |
| 2018-10-31 | 18-10 | AUTOSAR Release Management | <ul style="list-style-type: none"> Rework to CppImplementationDataTypes |
| 2018-03-29 | 18-03 | AUTOSAR Release Management | <ul style="list-style-type: none"> Editorial changes |
| 2017-10-27 | 17-10 | AUTOSAR Release Management | <ul style="list-style-type: none"> Initial release |

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

| | | |
|-------|---|----|
| 1 | Introduction and functional overview | 5 |
| 2 | Acronyms and Abbreviations | 6 |
| 3 | Related documentation | 7 |
| 3.1 | Input documents & related standards and norms | 7 |
| 3.2 | Related specification | 7 |
| 4 | Constraints and assumptions | 8 |
| 4.1 | Limitations | 8 |
| 4.2 | Applicability to car domains | 8 |
| 5 | Dependencies to other modules | 9 |
| 6 | Requirements Tracing | 10 |
| 7 | Primitive ImplementationDataTypes and their mapping to C++ datatypes | 11 |
| 7.1 | Bool | 11 |
| 7.2 | Signed Integer | 12 |
| 7.2.1 | int8_t | 12 |
| 7.2.2 | int16_t | 12 |
| 7.2.3 | int32_t | 12 |
| 7.2.4 | int64_t | 13 |
| 7.3 | Unsigned Integer | 13 |
| 7.3.1 | uint8_t | 13 |
| 7.3.2 | uint16_t | 13 |
| 7.3.3 | uint32_t | 14 |
| 7.3.4 | uint64_t | 14 |
| 7.4 | Floating point types | 15 |
| 7.4.1 | float | 15 |
| 7.4.2 | double | 15 |
| A | Mentioned Class Tables | 16 |
| B | History of Specification Items | 20 |
| B.1 | Constraint and Specification Item History of this document according to AUTOSAR Release 17-10 | 20 |
| B.1.1 | Added Traceables in 17-10 | 20 |
| B.1.2 | Changed Traceables in 17-10 | 21 |
| B.1.3 | Deleted Traceables in 17-10 | 21 |
| B.2 | Constraint and Specification Item History of this document according to AUTOSAR Release 18-03 | 22 |
| B.2.1 | Added Traceables in 18-03 | 22 |
| B.2.2 | Changed Traceables in 18-03 | 22 |
| B.2.3 | Deleted Traceables in 18-03 | 22 |

| | | |
|-------|---|----|
| B.3 | Constraint and Specification Item History of this document according to AUTOSAR Release 18-10 | 22 |
| B.3.1 | Added Traceables in 18-10 | 22 |
| B.3.2 | Changed Traceables in 18-10 | 22 |
| B.3.3 | Deleted Traceables in 18-10 | 23 |
| B.4 | Constraint and Specification Item History of this document according to AUTOSAR Release 19-03 | 24 |
| B.4.1 | Added Traceables in 19-03 | 24 |
| B.4.2 | Changed Traceables in 19-03 | 24 |
| B.4.3 | Deleted Traceables in 19-03 | 24 |

1 Introduction and functional overview

This document defines primitive `CppImplementationDataTypes` that can be used in `ServiceInterface` descriptions provided in ARXML as defined in TPS_ManifestSpecification [1].

The definition of common used `CppImplementationDataTypes` increases the portability of applications and prevents from re-defining the same types for each application.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations used in this document that are not included in the [2, AUTOSAR glossary].

| Terms: | Description: |
|----------------|---|
| 2's complement | method of signed number representation. |

3 Related documentation

3.1 Input documents & related standards and norms

- [1] Specification of Manifest
AUTOSAR_TPS_ManifestSpecification
- [2] Glossary
AUTOSAR_TR_Glossary
- [3] Specification of Communication Management
AUTOSAR_SWS_CommunicationManagement
- [4] General Requirements specific to Adaptive Platform
AUTOSAR_RS_General
- [5] ISO/IEC 14882:2011, Information technology – Programming languages – C++
<http://www.iso.org>
- [6] Guidelines for the use of the C++14 language in critical and safety-related systems
AUTOSAR_RS_CPP14Guidelines
- [7] Specification of Core Types for Adaptive Platform
AUTOSAR_SWS_CoreTypes

3.2 Related specification

The TPS Manifest specification [1] defines the meta-model that is used for the description of primitive datatypes that are presented in this document.

The specification SWS CommunicationManagement [3] defines the language binding rules for model artifacts.

4 Constraints and assumptions

4.1 Limitations

No limitations known.

4.2 Applicability to car domains

No restrictions to applicability.

5 Dependencies to other modules

This document is dependent on the language binding rules defined in SWS CommunicationManagement [3].

6 Requirements Tracing

Requirements against this document are exclusively stated in the corresponding requirements document [4].

The following table 6.1 references the requirements specified in the corresponding requirements document and provides information about individual specification items that fulfill a given requirement.

| Requirement | Description | Satisfied by |
|---------------|--|--|
| [RS_AP_00111] | The AUTOSAR Adaptive Platform shall support source code portability for AUTOSAR Adaptive applications. | [SWS_APT_00001] [SWS_APT_00004] [SWS_APT_00007] [SWS_APT_00010] [SWS_APT_00022] [SWS_APT_00025] [SWS_APT_00028] [SWS_APT_00031] [SWS_APT_00043] [SWS_APT_00046] [SWS_APT_00049] |

Table 6.1: RequirementsTracing

7 Primitive ImplementationDataTypes and their mapping to C++ datatypes

This chapter describes diverse primitive `StdCppImplementationDataTypes` that are predefined by AUTOSAR for the usage in the Adaptive Platform and defines their mapping to C++ datatypes.

The mapping of a primitive `StdCppImplementationDataType` that is used in a `ServiceInterface` to a C++ datatype is defined in `SWS_CommunicationManagement` [3].

Please note that [RS_AP_00114] in [4] defines that interfaces of AUTOSAR Adaptive platform are designed to be compatible with C++11 [5] but at the same time it is allowed to use newer C++ versions like C++14. Guidelines for the use of the C++14 language in critical and safety-related systems are defined in [6]. In addition the Adaptive Core Types document [7] defines common classes and functionality that is used by multiple AUTOSAR functional clusters as part of their public interfaces.

7.1 Bool

[SWS_APT_00049]{DRAFT} primitive Implementation Data Type *bool* [The primitive Implementation Data Type *bool* is defined by the `StdCppImplementationDataType` with the `category` VALUE and the `shortName` `bool`.](RS_AP_00111)

Listing 7.1: Boolean ImplementationDataType

```
<STD-CPP-IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>bool</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
</STD-CPP-IMPLEMENTATION-DATA-TYPE>
```

The *bool* `StdCppImplementationDataType` will be mapped to the `bool`-type in C++, that is capable of holding one of the two values: true or false. Please note that in C++ `sizeof(bool)` is implementation-defined.

Please note that according to [TPS_MANI_01176] the `typeEmitter` is not defined for fundamental data types like *bool* which means that the ARA generator shall not generate the corresponding data type definition for a `StdCppImplementationDataType` with the `category` VALUE and the `shortName` `bool`.

7.2 Signed Integer

7.2.1 int8_t

[SWS_APT_00001]{DRAFT} **primitive Implementation Data Type *int8_t*** [The signed integer type of 8 bits is defined by the [StdCppImplementationDataType](#) with the [category](#) VALUE and the [shortName](#) `int8_t`.]([RS_AP_00111](#))

Listing 7.2: int8_t StdCppImplementationDataType

```
<STD-CPP-IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>int8_t</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
  <TYPE-EMITTER>cstdint</TYPE-EMITTER>
</STD-CPP-IMPLEMENTATION-DATA-TYPE>
```

The *int8_t* [StdCppImplementationDataType](#) will be mapped to `int8_t` of the C++ standard library with width of exactly 8 bit.

7.2.2 int16_t

[SWS_APT_00004]{DRAFT} **primitive Implementation Data Type *int16_t*** [The signed integer type of 16 bits is defined by the [StdCppImplementationDataType](#) with the [category](#) VALUE and the [shortName](#) `int16_t`.]([RS_AP_00111](#))

Listing 7.3: int16_t StdCppImplementationDataType

```
<STD-CPP-IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>int16_t</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
  <TYPE-EMITTER>cstdint</TYPE-EMITTER>
</STD-CPP-IMPLEMENTATION-DATA-TYPE>
```

The *int16_t* [StdCppImplementationDataType](#) will be mapped to `int16_t` of the C++ standard library with width of exactly 16 bit.

7.2.3 int32_t

[SWS_APT_00007]{DRAFT} **primitive Implementation Data Type *int32_t*** [The signed integer type of 32 bits is defined by the [StdCppImplementationDataType](#) with the [category](#) VALUE and the [shortName](#) `int32_t`.]([RS_AP_00111](#))

Listing 7.4: int32_t StdCppImplementationDataType

```
<STD-CPP-IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>int32_t</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
  <TYPE-EMITTER>cstdint</TYPE-EMITTER>
</STD-CPP-IMPLEMENTATION-DATA-TYPE>
```

The `int32_t StdCppImplementationDataType` will be mapped to `int32_t` of the C++ standard library with width of exactly 32 bit.

7.2.4 int64_t

[SWS_APT_00010]{DRAFT} **primitive Implementation Data Type `int64_t`** [The signed integer type of 64 bits is defined by the `StdCppImplementationDataType` with the `category` VALUE and the `shortName` `int64_t`.]([RS_AP_00111](#))

Listing 7.5: int64_t StdCppImplementationDataType

```
<STD-CPP-IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>int64_t</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
  <TYPE-EMITTER>cstdint</TYPE-EMITTER>
</STD-CPP-IMPLEMENTATION-DATA-TYPE>
```

The `int64_t StdCppImplementationDataType` will be mapped to `int64_t` of the C++ standard library with width of exactly 64 bit.

7.3 Unsigned Integer

7.3.1 uint8_t

[SWS_APT_00022]{DRAFT} **primitive Implementation Data Type `uint8_t`** [The unsigned integer type of 8 bits is defined by the `StdCppImplementationDataType` with the `category` VALUE and the `shortName` `uint8_t`.]([RS_AP_00111](#))

Listing 7.6: uint8_t StdCppImplementationDataType

```
<STD-CPP-IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>uint8_t</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
  <TYPE-EMITTER>cstdint</TYPE-EMITTER>
</STD-CPP-IMPLEMENTATION-DATA-TYPE>
```

The `uint8_t StdCppImplementationDataType` will be mapped to `uint8_t` of the C++ standard library with width of exactly 8 bit.

7.3.2 uint16_t

[SWS_APT_00025]{DRAFT} **primitive Implementation Data Type `uint16_t`** [The unsigned integer type of 16 bits is defined by the `StdCppImplementationDataType` with the `category` VALUE and the `shortName` `uint16_t`.]([RS_AP_00111](#))

Listing 7.7: uint16_t StdCppImplementationDataType

```
<STD-CPP-IMPLEMENTATION-DATA-TYPE>
```

```
<SHORT-NAME>uint16_t</SHORT-NAME>
<CATEGORY>VALUE</CATEGORY>
<TYPE-EMITTER>cstdint</TYPE-EMITTER>
</STD-CPP-IMPLEMENTATION-DATA-TYPE>
```

The `uint16_t StdCppImplementationDataType` will be mapped to `uint16_t` of the C++ standard library with width of exactly 16 bit.

7.3.3 uint32_t

[SWS_APT_00028]{DRAFT} primitive Implementation Data Type `uint32_t` [The unsigned integer type of 32 bits is defined by the `StdCppImplementationDataType` with the `category` VALUE and the `shortName` `uint32_t`.]([RS_AP_00111](#))

Listing 7.8: `uint32_t StdCppImplementationDataType`

```
<STD-CPP-IMPLEMENTATION-DATA-TYPE>
<SHORT-NAME>uint32_t</SHORT-NAME>
<CATEGORY>VALUE</CATEGORY>
<TYPE-EMITTER>cstdint</TYPE-EMITTER>
</STD-CPP-IMPLEMENTATION-DATA-TYPE>
```

The `uint32_t StdCppImplementationDataType` will be mapped to `uint32_t` of the C++ standard library with width of exactly 32 bit.

7.3.4 uint64_t

[SWS_APT_00031]{DRAFT} primitive Implementation Data Type `uint64_t` [The unsigned integer type of 64 bits is defined by the `StdCppImplementationDataType` with the `category` VALUE and the `shortName` `uint64_t`.]([RS_AP_00111](#))

Listing 7.9: `uint64_t StdCppImplementationDataType`

```
<STD-CPP-IMPLEMENTATION-DATA-TYPE>
<SHORT-NAME>uint64_t</SHORT-NAME>
<CATEGORY>VALUE</CATEGORY>
<TYPE-EMITTER>cstdint</TYPE-EMITTER>
</STD-CPP-IMPLEMENTATION-DATA-TYPE>
```

The `uint64_t StdCppImplementationDataType` will be mapped to `uint64_t` of the C++ standard library with width of exactly 64 bit.

7.4 Floating point types

7.4.1 float

[SWS_APT_00043]{DRAFT} **primitive Implementation Data Type *float*** [The single precision floating point type is defined by the `StdCppImplementationDataType` with the `category` VALUE and the `shortName` `float`.] ([RS_AP_00111](#))

Listing 7.10: float StdCppImplementationDataType

```
<STD-CPP-IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>float</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
</STD-CPP-IMPLEMENTATION-DATA-TYPE>
```

The `float` `StdCppImplementationDataType` will be mapped in C++ to `float` that is the 32 bit floating point type.

Please note that according to [TPS_MANI_01176] the `typeEmitter` is not defined for fundamental data types like `float` which means that the ARA generator shall not generate the corresponding data type definition for a `StdCppImplementationDataType` with the `category` VALUE and the `shortName` `float`.

7.4.2 double

[SWS_APT_00046]{DRAFT} **primitive Implementation Data Type *double*** [The double precision floating point type is defined by the `StdCppImplementationDataType` with the `category` VALUE and the `shortName` `double`.] ([RS_AP_00111](#))

Listing 7.11: double StdCppImplementationDataType

```
<STD-CPP-IMPLEMENTATION-DATA-TYPE>
  <SHORT-NAME>double</SHORT-NAME>
  <CATEGORY>VALUE</CATEGORY>
</STD-CPP-IMPLEMENTATION-DATA-TYPE>
```

The `double` `StdCppImplementationDataType` will be mapped in C++ to `double` that is the 64 bit floating point type.

Please note that according to [TPS_MANI_01176] the `typeEmitter` is not defined for fundamental data types like `double` which means that the ARA generator shall not generate the corresponding data type definition for a `StdCppImplementationDataType` with the `category` VALUE and the `shortName` `double`.

A Mentioned Class Tables

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

| | | | | |
|----------------------------|--|-------------|-------------|--|
| Class | <i>CppImplementationDataType</i> (abstract) | | | |
| Package | M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType | | | |
| Note | This meta-class represents the way to specify a reusable data type definition taken as a the basis for a C++ language binding Tags: atp.Status=draft | | | |
| Base | <i>ARElement</i> , <i>ARObject</i> , <i>AbstractImplementationDataType</i> , <i>AtpBlueprint</i> , <i>AtpBlueprintable</i> , <i>AtpClassifier</i> , <i>AtpType</i> , <i>AutosarDataType</i> , <i>CollectableElement</i> , <i>CppImplementationDataTypeContextTarget</i> , <i>Identifiable</i> , <i>MultilanguageReferrable</i> , <i>PackageableElement</i> , <i>Referrable</i> | | | |
| Subclasses | CustomCppImplementationDataType, StdCppImplementationDataType | | | |
| Attribute | Type | Mul. | Kind | Note |
| arraySize | PositiveInteger | 0..1 | attr | This attribute can be used to specify the array size if the enclosing CppImplementationDataType has array semantics. Stereotypes: atpVariation Tags: vh.latestBindingTime=preCompileTime |
| namespace (ordered) | SymbolProps | * | aggr | This aggregation allows for the definition an own namespace for the enclosing CppImplementationDataType. Tags: atp.Status=draft |
| subElement (ordered) | CppImplementationDataTypeElement | * | aggr | This represents the collection of sub-elements of the enclosing CppImplementationDataType Tags: atp.Status=draft |
| templateArgument (ordered) | CppTemplateArgument | * | aggr | This aggregation allows for the specification of properties of template arguments Tags: atp.Status=draft |
| typeEmitter | NameToken | 0..1 | attr | This attribute can be taken to control how the respective CppImplementationDataType is contributed to the language binding. |
| typeReference | CppImplementationDataType | 0..1 | ref | This reference shall be defined to define a type reference (a.k.a. typedef). Tags: atp.Status=draft |

Table A.1: CppImplementationDataType

| | |
|----------------|--|
| Class | <i>Identifiable</i> (abstract) |
| Package | M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable |
| Note | Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables. |
| Base | <i>ARObject</i> , <i>MultilanguageReferrable</i> , <i>Referrable</i> |





| | | | | |
|-------------------|--|-------------|-------------|---|
| Class | Identifiable (abstract) | | | |
| Subclasses | <p>ARPackage, <i>AbstractEvent</i>, <i>AbstractImplementationDataTypeElement</i>, <i>AbstractServiceInstance</i>, <i>AdaptiveModuleInstantiation</i>, AdaptiveSwcInternalBehavior, ApplicationEndpoint, ApplicationError, ApplicationPartitionToEcuPartitionMapping, AsynchronousServerCallResultPoint, <i>AtpBlueprint</i>, <i>AtpBlueprintable</i>, <i>AtpClassifier</i>, <i>AtpFeature</i>, AutosarOperationArgumentInstance, AutosarVariableInstance, BswInternalTriggeringPoint, BswModuleDependency, <i>BuildActionEntity</i>, BuildActionEnvironment, CanTpAddress, CanTpChannel, CanTpNode, Chapter, CheckpointTransition, ClassContentConditional, ClientIdDefinition, ClientServerOperation, Code, <i>CollectableElement</i>, ComManagementMapping, <i>CommConnectorPort</i>, <i>CommunicationConnector</i>, <i>CommunicationController</i>, Compiler, ConsistencyNeeds, ConsumedEventGroup, CouplingPort, <i>CouplingPortStructuralElement</i>, <i>CryptoServiceMapping</i>, DataPrototypeGroup, DataTransformation, DdsRpcServiceDeployment, DependencyOnArtifact, DeterministicClientResourceNeeds, <i>DiagEventDebounceAlgorithm</i>, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticFunctionInhibitSource, DiagnosticMasterToSlaveEventMapping, <i>DiagnosticRoutineSubfunction</i>, DolpLogicAddress, E2EProfileConfiguration, ECUMapping, <i>EOCExecutableEntityRefAbstract</i>, EcuPartition, EcucContainerValue, <i>EcucDefinitionElement</i>, EcucDestinationUriDef, EcucEnumerationLiteralDef, EcucQuery, EcucValidationCondition, End2EndEventProtectionProps, EndToEndProtection, EventMapping, ExclusiveArea, <i>ExecutableEntity</i>, <i>ExecutionTime</i>, FMAAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeatureMapElement, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FieldMapping, FireAndForgetMapping, FlatInstanceDescriptor, FlexrayArTpNode, FlexrayTpConnectionControl, FlexrayTpNode, FlexrayTpPduPool, <i>FrameTriggering</i>, GeneralParameter, GlobalTimeGateway, <i>GlobalTimeMaster</i>, <i>GlobalTimeSlave</i>, <i>HealthChannel</i>, <i>HeapUsage</i>, HwAttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, IPSecRule, IPv6ExtHeaderFilterList, ISignalToLPduMapping, ISignalTriggering, <i>IdentCaption</i>, InterfaceMapping, InternalTriggeringPoint, J1939SharedAddressCluster, J1939TpNode, Keyword, LifeCycleState, LinScheduleTable, LinTpNode, Linker, MacMulticastGroup, McDataInstance, MemorySection, MethodMapping, ModeDeclaration, ModeDeclarationMapping, ModeSwitchPoint, NetworkEndpoint, <i>NmCluster</i>, <i>NmNode</i>, NvBlockDescriptor, <i>PackageableElement</i>, ParameterAccess, PduToFrameMapping, PduTriggering, PerInstanceMemory, PersistencyFileProxy, PersistencyKeyValuePair, PhmAction, <i>PhmActionItem</i>, PhmActionList, PhmArbitration, PhmLogicalExpression, PhmRule, <i>PhmSupervision</i>, <i>PhysicalChannel</i>, PortGroup, <i>PortInterfaceMapping</i>, PossibleErrorReaction, ProcessDesignToMachineDesignMapping, ProcessToMachineMapping, Processor, ProcessorCore, PskIdentityToKeySlotMapping, ResourceConsumption, ResourceGroup, <i>RestAbstractEndpoint</i>, RestElementDef, RestResourceDef, RootSwComponentPrototype, RootSwCompositionPrototype, RptComponent, RptContainer, RptExecutableEntity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptServicePoint, RunnableEntityGroup, <i>SdgAttribute</i>, SdgClass, SecOcJobMapping, SecOcJobRequirement, <i>SecureComProps</i>, SecureCommunicationAuthenticationProps, <i>SecureCommunicationDeployment</i>, SecureCommunicationFreshnessProps, <i>ServerCallPoint</i>, <i>ServiceEventDeployment</i>, <i>ServiceFieldDeployment</i>, ServiceInstanceToSignalMapping, <i>ServiceInterfaceElementMapping</i>, ServiceInterfaceElementSecureComConfig, ServiceInterfaceMapping, <i>ServiceMethodDeployment</i>, <i>ServiceNeeds</i>, SignalBasedFieldToSignalTriggeringMapping, SocketAddress, SomeipEventGroup, SomeipProvidedEventGroup, SomeipTpChannel, <i>SpecElementReference</i>, <i>StackUsage</i>, StartupConfig, StructuredReq, SupervisionCheckpoint, SwGenericAxisParamType, SwServiceArg, SwServiceDependency, SwcToApplicationPartitionMapping, SwcToEcuMapping, SwcToImplMapping, SystemMapping, TcpOptionFilterList, <i>TimeBaseResource</i>, TimingCondition, <i>TimingConstraint</i>, <i>TimingDescription</i>, TimingExtensionResource, TimingModelInstance, TlsCryptoCipherSuite, TlsJobMapping, Topic1, TpAddress, TraceableText, <i>TracedFailure</i>, <i>TransformationProps</i>, TransformationPropsToServiceInterfaceElementMapping, TransformationTechnology, Trigger, VariableAccess, VariationPointProxy, ViewMap, VlanConfig, WaitPoint</p> | | | |
| Attribute | Type | Mul. | Kind | Note |
| desc | MultiLanguageOverviewParagraph | 0..1 | aggr | <p>This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question.</p> <p>More elaborate documentation, (in particular how the object is built or used) should go to "introduction".</p> <p>Tags: xml.sequenceOffset=-60</p> |
| category | CategoryString | 0..1 | attr | <p>The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints.</p> <p>Tags: xml.sequenceOffset=-50</p> |





| Class | Identifiable (abstract) | | | |
|--------------|--------------------------------|------|------|--|
| adminData | AdminData | 0..1 | aggr | This represents the administrative data for the identifiable object. Tags: xml.sequenceOffset=-40 |
| annotation | Annotation | * | aggr | Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. Tags: xml.sequenceOffset=-25 |
| introduction | DocumentationBlock | 0..1 | aggr | This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. Tags: xml.sequenceOffset=-30 |
| uuid | String | 0..1 | attr | The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp. Tags: xml.attribute=true |

Table A.2: Identifiable

| | | | | |
|-------------------|---|-------------|-------------|-------------|
| Class | Referrable (abstract) | | | |
| Package | M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable | | | |
| Note | Instances of this class can be referred to by their identifier (while adhering to namespace borders). | | | |
| Base | ARObject | | | |
| Subclasses | AtpDefinition, BswDistinguishedPartition, BswModuleCallPoint, BswModuleClientServerEntry, BswVariableAccess, CouplingPortTrafficClassAssignment, CppImplementationData TypeContextTarget, DiagnosticDebounceAlgorithmProps, DiagnosticEnvModeElement, EthernetPriorityRegeneration, Event Handler, ExclusiveAreaNestingOrder, HwDescriptionEntity, ImplementationProps, LinSlaveConfigIdent, ModeTransition, MultilanguageReferrable, NetworkConfiguration, NmNetworkHandle, PncMappingIdent, SingleLanguageReferrable, SocketConnectionBundle, SomeipRequiredEventGroup, TimeSyncServer Configuration, TpConnectionIdent | | | |
| Attribute | Type | Mul. | Kind | Note |





| Class | Referrable (abstract) | | | |
|--------------------|-----------------------|---|------|---|
| shortName | Identifier | 1 | attr | This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference. Tags: xml.enforceMinMultiplicity=true xml.sequenceOffset=-100 |
| shortName Fragment | ShortNameFragment | * | aggr | This specifies how the Referrable.shortName is composed of several shortNameFragments. Tags: xml.sequenceOffset=-90 |

Table A.3: Referrable

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

Table A.4: ServiceInterface

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





| | | | | |
|--------------|---|---|---|---|
| Class | StdCplusplusImplementationDataType | | | |
| - | - | - | - | - |

Table A.5: StdCplusplusImplementationDataType

B History of Specification Items

B.1 Constraint and Specification Item History of this document according to AUTOSAR Release 17-10

B.1.1 Added Traceables in 17-10

| Number | Heading |
|-----------------|--|
| [SWS_APT_00001] | primitive Implementation Data Type <i>sint8</i> |
| [SWS_APT_00002] | SwBaseType <i>sint8</i> |
| [SWS_APT_00003] | Platform specific settings in SwBaseType <i>sint8</i> |
| [SWS_APT_00004] | primitive Implementation Data Type <i>sint16</i> |
| [SWS_APT_00005] | SwBaseType <i>sint16</i> |
| [SWS_APT_00006] | Platform specific settings in SwBaseType <i>sint16</i> |
| [SWS_APT_00007] | primitive Implementation Data Type <i>sint32</i> |
| [SWS_APT_00008] | SwBaseType <i>sint32</i> |
| [SWS_APT_00009] | Platform specific settings in SwBaseType <i>sint32</i> |
| [SWS_APT_00010] | primitive Implementation Data Type <i>sint64</i> |
| [SWS_APT_00011] | SwBaseType <i>sint64</i> |
| [SWS_APT_00012] | Platform specific settings in SwBaseType <i>sint64</i> |
| [SWS_APT_00013] | primitive Implementation Data Type <i>sint8_least</i> |
| [SWS_APT_00014] | SwBaseType <i>sint8_least</i> |
| [SWS_APT_00015] | Platform specific settings in SwBaseType <i>sint8_least</i> |
| [SWS_APT_00016] | primitive Implementation Data Type <i>sint16_least</i> |
| [SWS_APT_00017] | SwBaseType <i>sint16_least</i> |
| [SWS_APT_00018] | Platform specific settings in SwBaseType <i>sint16_least</i> |
| [SWS_APT_00019] | primitive Implementation Data Type <i>sint32_least</i> |
| [SWS_APT_00020] | SwBaseType <i>sint32_least</i> |
| [SWS_APT_00021] | Platform specific settings in SwBaseType <i>sint32_least</i> |
| [SWS_APT_00022] | primitive Implementation Data Type <i>uint8</i> |
| [SWS_APT_00023] | SwBaseType <i>uint8</i> |
| [SWS_APT_00024] | Platform specific settings in SwBaseType <i>uint8</i> |





| Number | Heading |
|-----------------|--|
| [SWS_APT_00025] | primitive Implementation Data Type <i>uint16</i> |
| [SWS_APT_00026] | SwBaseType <i>uint16</i> |
| [SWS_APT_00027] | Platform specific settings in SwBaseType <i>uint16</i> |
| [SWS_APT_00028] | primitive Implementation Data Type <i>uint32</i> |
| [SWS_APT_00029] | SwBaseType <i>uint32</i> |
| [SWS_APT_00030] | Platform specific settings in SwBaseType <i>uint32</i> |
| [SWS_APT_00031] | primitive Implementation Data Type <i>uint64</i> |
| [SWS_APT_00032] | SwBaseType <i>uint64</i> |
| [SWS_APT_00033] | Platform specific settings in SwBaseType <i>uint64</i> |
| [SWS_APT_00034] | primitive Implementation Data Type <i>uint8_least</i> |
| [SWS_APT_00035] | SwBaseType <i>uint8_least</i> |
| [SWS_APT_00036] | Platform specific settings in SwBaseType <i>uint8_least</i> |
| [SWS_APT_00037] | primitive Implementation Data Type <i>uint16_least</i> |
| [SWS_APT_00038] | SwBaseType <i>uint16_least</i> |
| [SWS_APT_00039] | Platform specific settings in SwBaseType <i>uint16_least</i> |
| [SWS_APT_00040] | primitive Implementation Data Type <i>uint32_least</i> |
| [SWS_APT_00041] | SwBaseType <i>uint32_least</i> |
| [SWS_APT_00042] | Platform specific settings in SwBaseType <i>uint32_least</i> |
| [SWS_APT_00043] | primitive Implementation Data Type <i>float32</i> |
| [SWS_APT_00044] | SwBaseType <i>float32</i> |
| [SWS_APT_00045] | Platform specific settings in SwBaseType <i>float32</i> |
| [SWS_APT_00046] | primitive Implementation Data Type <i>float64</i> |
| [SWS_APT_00047] | SwBaseType <i>float64</i> |
| [SWS_APT_00048] | Platform specific settings in SwBaseType <i>float64</i> |
| [SWS_APT_00049] | primitive Implementation Data Type <i>boolean</i> |
| [SWS_APT_00050] | SwBaseType <i>boolean</i> |
| [SWS_APT_00051] | Platform specific settings in SwBaseType <i>boolean</i> |

Table B.1: Added Traceables in 17-10

B.1.2 Changed Traceables in 17-10

none

B.1.3 Deleted Traceables in 17-10

none

B.2 Constraint and Specification Item History of this document according to AUTOSAR Release 18-03

B.2.1 Added Traceables in 18-03

none

B.2.2 Changed Traceables in 18-03

| Number | Heading |
|-----------------|---|
| [SWS_APT_00003] | Platform specific settings in SwBaseType <i>sint8</i> |
| [SWS_APT_00015] | Platform specific settings in SwBaseType <i>sint8_least</i> |
| [SWS_APT_00024] | Platform specific settings in SwBaseType <i>uint8</i> |
| [SWS_APT_00036] | Platform specific settings in SwBaseType <i>uint8_least</i> |

Table B.2: Changed Traceables in 18-03

B.2.3 Deleted Traceables in 18-03

none

B.3 Constraint and Specification Item History of this document according to AUTOSAR Release 18-10

B.3.1 Added Traceables in 18-10

none

B.3.2 Changed Traceables in 18-10

| Number | Heading |
|-----------------|---|
| [SWS_APT_00001] | primitive Implementation Data Type <i>int8_t</i> |
| [SWS_APT_00004] | primitive Implementation Data Type <i>int16_t</i> |
| [SWS_APT_00007] | primitive Implementation Data Type <i>int32_t</i> |
| [SWS_APT_00010] | primitive Implementation Data Type <i>int64_t</i> |
| [SWS_APT_00022] | primitive Implementation Data Type <i>uint8_t</i> |





| Number | Heading |
|-----------------|--|
| [SWS_APT_00025] | primitive Implementation Data Type <i>uint16_t</i> |
| [SWS_APT_00028] | primitive Implementation Data Type <i>uint32_t</i> |
| [SWS_APT_00031] | primitive Implementation Data Type <i>uint64_t</i> |
| [SWS_APT_00043] | primitive Implementation Data Type <i>float</i> |
| [SWS_APT_00046] | primitive Implementation Data Type <i>double</i> |
| [SWS_APT_00049] | primitive Implementation Data Type <i>bool</i> |

Table B.3: Changed Traceables in 18-10

B.3.3 Deleted Traceables in 18-10

| Number | Heading |
|-----------------|--|
| [SWS_APT_00002] | SwBaseType <i>sint8</i> |
| [SWS_APT_00003] | Platform specific settings in SwBaseType <i>sint8</i> |
| [SWS_APT_00005] | SwBaseType <i>sint16</i> |
| [SWS_APT_00006] | Platform specific settings in SwBaseType <i>sint16</i> |
| [SWS_APT_00008] | SwBaseType <i>sint32</i> |
| [SWS_APT_00009] | Platform specific settings in SwBaseType <i>sint32</i> |
| [SWS_APT_00011] | SwBaseType <i>sint64</i> |
| [SWS_APT_00012] | Platform specific settings in SwBaseType <i>sint64</i> |
| [SWS_APT_00013] | primitive Implementation Data Type <i>sint8_least</i> |
| [SWS_APT_00014] | SwBaseType <i>sint8_least</i> |
| [SWS_APT_00015] | Platform specific settings in SwBaseType <i>sint8_least</i> |
| [SWS_APT_00016] | primitive Implementation Data Type <i>sint16_least</i> |
| [SWS_APT_00017] | SwBaseType <i>sint16_least</i> |
| [SWS_APT_00018] | Platform specific settings in SwBaseType <i>sint16_least</i> |
| [SWS_APT_00019] | primitive Implementation Data Type <i>sint32_least</i> |
| [SWS_APT_00020] | SwBaseType <i>sint32_least</i> |
| [SWS_APT_00021] | Platform specific settings in SwBaseType <i>sint32_least</i> |
| [SWS_APT_00023] | SwBaseType <i>uint8</i> |
| [SWS_APT_00024] | Platform specific settings in SwBaseType <i>uint8</i> |
| [SWS_APT_00026] | SwBaseType <i>uint16</i> |
| [SWS_APT_00027] | Platform specific settings in SwBaseType <i>uint16</i> |
| [SWS_APT_00029] | SwBaseType <i>uint32</i> |
| [SWS_APT_00030] | Platform specific settings in SwBaseType <i>uint32</i> |
| [SWS_APT_00032] | SwBaseType <i>uint64</i> |
| [SWS_APT_00033] | Platform specific settings in SwBaseType <i>uint64</i> |



△

| Number | Heading |
|-----------------|--|
| [SWS_APT_00034] | primitive Implementation Data Type <i>uint8_least</i> |
| [SWS_APT_00035] | SwBaseType <i>uint8_least</i> |
| [SWS_APT_00036] | Platform specific settings in SwBaseType <i>uint8_least</i> |
| [SWS_APT_00037] | primitive Implementation Data Type <i>uint16_least</i> |
| [SWS_APT_00038] | SwBaseType <i>uint16_least</i> |
| [SWS_APT_00039] | Platform specific settings in SwBaseType <i>uint16_least</i> |
| [SWS_APT_00040] | primitive Implementation Data Type <i>uint32_least</i> |
| [SWS_APT_00041] | SwBaseType <i>uint32_least</i> |
| [SWS_APT_00042] | Platform specific settings in SwBaseType <i>uint32_least</i> |
| [SWS_APT_00044] | SwBaseType <i>float32</i> |
| [SWS_APT_00045] | Platform specific settings in SwBaseType <i>float32</i> |
| [SWS_APT_00047] | SwBaseType <i>float64</i> |
| [SWS_APT_00048] | Platform specific settings in SwBaseType <i>float64</i> |
| [SWS_APT_00050] | SwBaseType <i>boolean</i> |
| [SWS_APT_00051] | Platform specific settings in SwBaseType <i>boolean</i> |

Table B.4: Deleted Traceables in 18-10

B.4 Constraint and Specification Item History of this document according to AUTOSAR Release 19-03

B.4.1 Added Traceables in 19-03

none

B.4.2 Changed Traceables in 19-03

none

B.4.3 Deleted Traceables in 19-03

none