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

Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R20-11

Document Change History

Date	Release	Changed by	Description
2020-11-30	R20-11	AUTOSAR Release Management	• editorial changes;
2019-11-28	R19-11	AUTOSAR Release Management	• No content changes • Changed Document Status from Final to published
2019-03-29	19-03	AUTOSAR Release Management	• minor corrections / clarifications / editorial changes;
2018-10-31	18-10	AUTOSAR Release Management	• Rework to CppImplementationDataTypes
2018-03-29	18-03	AUTOSAR Release Management	• Editorial changes
2017-10-27	17-10	AUTOSAR Release Management	• Initial release

Disclaimer

Table of Contents

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 14882:2014, Information technology – Programming languages – C++
http://www.iso.org

[6] Specification of the Adaptive Core
AUTOSAR_SWS_AdaptiveCore

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 Functional Clusters

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 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 Functional specification

7.1 Primitive ImplementationDataTypes and their mapping to C++ datatypes

This chapter describes diverse primitive `StdCppImplementationDataType`s 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++14 [5] but at the same time it is allowed to use newer C++ versions like C++17. In addition the Adaptive Core document [6] defines common classes and functionality that is used by multiple AUTOSAR functional clusters as part of their public interfaces.

7.1.1 Bool

[SWS_APT_00049] 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>
  <TYPE-EMITTER>FundamentalType</TYPE-EMITTER>
</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_01177] if the typeEmitter is set to any value other than TYPE_EMITTER_ARA, the ARA generator shall not generate the corresponding data type definition.

7.1.2 Signed Integer

7.1.2.1 int8_t

[SWS_APT_00001] 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>
  <HEADER-FILE>cstdint</HEADER-FILE>
</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.

7.1.2.2 int16_t

[SWS_APT_00004] 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>
  <HEADER-FILE>cstdint</HEADER-FILE>
</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.

7.1.2.3 int32_t

[SWS_APT_00007] 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>
  <HEADER-FILE>cstdint</HEADER-FILE>
</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.

7.1.2.4 int64_t

[SWS_APT_00010] 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>
  <HEADER-FILE>cstdint</HEADER-FILE>
</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.

7.1.3 Unsigned Integer

7.1.3.1 uint8_t

[SWS_APT_00022] 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>
  <HEADER-FILE>cstdint</HEADER-FILE>
</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.

7.1.3.2 uint16_t

[SWS_APT_00025] 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>
  <HEADER-FILE>cstdint</HEADER-FILE>
</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.

7.1.3.3 uint32_t

[SWS_APT_00028] 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>
  <HEADER-FILE>cstdint</HEADER-FILE>
</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.

7.1.3.4 uint64_t

[SWS_APT_00031] 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>
  <HEADER-FILE>cstdint</HEADER-FILE>
</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.

7.1.4 Floating point types

7.1.4.1 float

[SWS_APT_00043] 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>
  <TYPE-EMITTER>FundamentalType</TYPE-EMITTER>
</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_01177] if the `typeEmitter` is set to any value other than TYPE_EMITTER_ARA, the ARA generator shall not generate the corresponding data type definition.

7.1.4.2 double

[SWS_APT_00046] 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>
  <TYPE-EMITTER>FundamentalType</TYPE-EMITTER>
</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_01177] if the `typeEmitter` is set to any value other than TYPE_EMITTER_ARA, the ARA generator shall not generate the corresponding data type definition.

A Mentioned Manifest Elements

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	***CppImplementationDataType*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class represents the way to specify a reusable data type definition taken as a the basis for a C++ language binding **Tags:**atp.Status=draft			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	CustomCppImplementationDataType, StdCppImplementationDataType			
Attribute	***Type***	***Mult.***	***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
headerFile	String	0..1	attr	Configuration of the Header File with the custom class declaration.
namespace (ordered)	SymbolProps	*	aggr	This aggregation allows for the definition an own namespace for the enclosing CppImplementationData Type. **Tags:**atp.Status=draft
subElement (ordered)	CppImplementation DataTypeElement	*	aggr	This represents the collection of sub-elements of the enclosing CppImplementationDataType **Tags:**atp.Status=draft
template Argument (ordered)	CppTemplateArgument	*	aggr	This aggreation allows for the specification of properties of template arguments **Tags:**atp.Status=draft
typeEmitter	NameToken	0..1	attr	This attribute can be taken to control how the respective CppImplementationDataType is contributed to the language binding.
typeReference	CppImplementation DataType	0..1	ref	This reference shall be defined to define a type reference (a.k.a. typedef). **Tags:**atp.Status=draft

Table A.1: CppImplementationDataType

Class	***Identifiable*** (abstract)
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*
Subclasses	ARPackage, *AbstractDoIpLogicAddressProps*, *AbstractEvent*, *AbstractImplementationDataTypeElement*, *AbstractSecurityEventFilter*, *AbstractSecurityIdsmInstanceFilter*, *AbstractServiceInstance*, *AbstractSignalBasedToISignalTriggeringMapping*, *AdaptiveModuleInstantiation*, AdaptiveSwcInternalBehavior, ApplicationEndpoint, ApplicationError, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, AutosarOperationArgumentInstance, AutosarVariableInstance, *BuildActionEntity*, BuildActionEnvironment, Chapter, CheckpointTransition, ClassContentConditional, ClientIdDefinition, ClientServerOperation, Code, *CollectableElement*, ComManagementMapping, *CommConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEventGroup, CouplingPort, *CouplingPortStructuralElement*, CryptoCertificate, CryptoKeySlot, CryptoProvider, *CryptoService* ▽

▽

△

Class	***Identifiable*** (abstract)			
	△ *Mapping*, DataPrototypeGroup, DataTransformation, DependencyOnArtifact, DeterministicClient ResourceNeeds, *DiagEventDebounceAlgorithm*, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticFunctionInhibitSource, *DiagnosticRoutineSubfunction*, DltArgument, DltLogChannel, Dlt Message, DoIpInterface, DoIpLogicAddress, DoIpRoutingActivation, E2EProfileConfiguration, End2End EventProtectionProps, EndToEndProtection, EthernetWakeupSleepOnDatalineConfig, EventMapping, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeatureMapAssertion, FMFeature MapCondition, FMFeatureMapElement, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FieldMapping, FireAndForgetMapping, *FrameTriggering*, GeneralParameter, GlobalSupervision, Global TimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HealthChannel*, *HeapUsage*, HwAttributeDef, Hw AttributeLiteralDef, HwPin, HwPinGroup, IPSecRule, IPv6ExtHeaderFilterList, ISignalToIPduMapping, I SignalTriggering, *IdentCaption*, InterfaceMapping, InternalTriggeringPoint, Keyword, LifeCycleState, Linker, MacMulticastGroup, McDataInstance, MemorySection, MethodMapping, ModeDeclaration, Mode DeclarationMapping, ModeSwitchPoint, NetworkEndpoint, *NmCluster*, *NmNode*, *PackageableElement*, ParameterAccess, PduToFrameMapping, PduTriggering, *PersistencyDeploymentElement*, *Persistency InterfaceElement*, *PhmSupervision*, *PhysicalChannel*, PortGroup, *PortInterfaceMapping*, PossibleError Reaction, ProcessDesignToMachineDesignMapping, ProcessToMachineMapping, Processor, Processor Core, PskIdentityToKeySlotMapping, RecoveryNotification, ResourceConsumption, ResourceGroup, *RestAbstractEndpoint*, RestElementDef, RestResourceDef, RootSwClusterDesignComponentPrototype, RootSwComponentPrototype, RootSwCompositionPrototype, RptComponent, RptContainer, Rpt ExecutableEntity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptServicePoint, *Sdg Attribute*, SdgClass, SecOcJobMapping, SecOcJobRequirement, *SecureComProps*, Secure CommunicationAuthenticationProps, *SecureCommunicationDeployment*, SecureCommunication FreshnessProps, SecurityEventContextProps, *ServiceEventDeployment*, *ServiceFieldDeployment*, ServiceInstanceToSignalMapping, *ServiceInterfaceElementMapping*, ServiceInterfaceElementSecure ComConfig, ServiceInterfaceMapping, *ServiceMethodDeployment*, *ServiceNeeds*, SignalService TranslationEventProps, SignalServiceTranslationProps, SocketAddress, SoftwarePackageStep, Someip EventGroup, SomeipProvidedEventGroup, SomeipTpChannel, *SpecElementReference*, *StackUsage*, StartupConfig, StaticSocketConnection, StructuredReq, SupervisionCheckpoint, SwGenericAxisParam Type, SwServiceArg, SwcServiceDependency, SystemMapping, SystemMemoryUsage, *TimeBase Resource*, TimingCondition, *TimingConstraint*, *TimingDescription*, TimingExtensionResource, Timing ModeInstance, TlsCryptoCipherSuite, TlsJobMapping, Topic1, TpAddress, TraceableTable, Traceable Text, *TracedFailure*, *TransformationProps*, TransformationPropsToServiceInterfaceElementMapping, TransformationTechnology, Trigger, UcmDescription, UcmStep, VariableAccess, VariationPointProxy, VehicleRolloutStep, ViewMap, VlanConfig			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
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
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. **Tags:**xml.sequenceOffset=-50
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". **Tags:**xml.sequenceOffset=-60

▽

△

Class	***Identifiable*** (abstract)			
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, *CppImplementationDataTypeContextTarget*, DiagnosticDebounceAlgorithmProps, *DiagnosticEnvModeElement*, EthernetPriorityRegeneration, EventHandler, ExclusiveAreaNestingOrder, *HwDescriptionEntity*, *ImplementationProps*, LinSlaveConfigIdent, ModeTransition, *MultilanguageReferrable*, NmNetworkHandle, PduActivationRoutingGroup, PncMappingIdent, *SingleLanguageReferrable*, SoConIPduIdentifier, SocketConnectionBundle, SomeipRequiredEventGroup, TimeSyncServerConfiguration, TpConnectionIdent			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference. **Stereotypes:** atpIdentityContributor **Tags:** xml.enforceMinMultiplicity=true xml.sequenceOffset=-100
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***	***Mult.***	***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 xml.sequenceOffset=30
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 xml.sequenceOffset=40
majorVersion	PositiveInteger	0..1	attr	Major version of the service contract. **Tags:** atp.Status=draft xml.sequenceOffset=10
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 xml.sequenceOffset=50
minorVersion	PositiveInteger	0..1	attr	Minor version of the service contract. **Tags:** atp.Status=draft xml.sequenceOffset=20

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=CppImplementationDataTypes
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataType*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*

▽

△

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

Table A.5: StdCppImplementationDataType

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

▽

△

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

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 *sint8*
[SWS_APT_00015]	Platform specific settings in SwBaseType *sint8_least*
[SWS_APT_00024]	Platform specific settings in SwBaseType *uint8*
[SWS_APT_00036]	Platform specific settings in SwBaseType *uint8_least*

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 *int8_t*
[SWS_APT_00004]	primitive Implementation Data Type *int16_t*
[SWS_APT_00007]	primitive Implementation Data Type *int32_t*
[SWS_APT_00010]	primitive Implementation Data Type *int64_t*
[SWS_APT_00022]	primitive Implementation Data Type *uint8_t*
[SWS_APT_00025]	primitive Implementation Data Type *uint16_t*
[SWS_APT_00028]	primitive Implementation Data Type *uint32_t*

▽

△

Number	Heading
[SWS_APT_00031]	primitive Implementation Data Type *uint64_t*
[SWS_APT_00043]	primitive Implementation Data Type *float*
[SWS_APT_00046]	primitive Implementation Data Type *double*
[SWS_APT_00049]	primitive Implementation Data Type *bool*

Table B.3: Changed Traceables in 18-10

B.3.3 Deleted Traceables in 18-10

Number	Heading
[SWS_APT_00002]	SwBaseType *sint8*
[SWS_APT_00003]	Platform specific settings in SwBaseType *sint8*
[SWS_APT_00005]	SwBaseType *sint16*
[SWS_APT_00006]	Platform specific settings in SwBaseType *sint16*
[SWS_APT_00008]	SwBaseType *sint32*
[SWS_APT_00009]	Platform specific settings in SwBaseType *sint32*
[SWS_APT_00011]	SwBaseType *sint64*
[SWS_APT_00012]	Platform specific settings in SwBaseType *sint64*
[SWS_APT_00013]	primitive Implementation Data Type *sint8_least*
[SWS_APT_00014]	SwBaseType *sint8_least*
[SWS_APT_00015]	Platform specific settings in SwBaseType *sint8_least*
[SWS_APT_00016]	primitive Implementation Data Type *sint16_least*
[SWS_APT_00017]	SwBaseType *sint16_least*
[SWS_APT_00018]	Platform specific settings in SwBaseType *sint16_least*
[SWS_APT_00019]	primitive Implementation Data Type *sint32_least*
[SWS_APT_00020]	SwBaseType *sint32_least*
[SWS_APT_00021]	Platform specific settings in SwBaseType *sint32_least*
[SWS_APT_00023]	SwBaseType *uint8*
[SWS_APT_00024]	Platform specific settings in SwBaseType *uint8*
[SWS_APT_00026]	SwBaseType *uint16*
[SWS_APT_00027]	Platform specific settings in SwBaseType *uint16*
[SWS_APT_00029]	SwBaseType *uint32*
[SWS_APT_00030]	Platform specific settings in SwBaseType *uint32*
[SWS_APT_00032]	SwBaseType *uint64*
[SWS_APT_00033]	Platform specific settings in SwBaseType *uint64*
[SWS_APT_00034]	primitive Implementation Data Type *uint8_least*
[SWS_APT_00035]	SwBaseType *uint8_least*
[SWS_APT_00036]	Platform specific settings in SwBaseType *uint8_least*

▽

△

Number	Heading
[SWS_APT_00037]	primitive Implementation Data Type *uint16_least*
[SWS_APT_00038]	SwBaseType *uint16_least*
[SWS_APT_00039]	Platform specific settings in SwBaseType *uint16_least*
[SWS_APT_00040]	primitive Implementation Data Type *uint32_least*
[SWS_APT_00041]	SwBaseType *uint32_least*
[SWS_APT_00042]	Platform specific settings in SwBaseType *uint32_least*
[SWS_APT_00044]	SwBaseType *float32*
[SWS_APT_00045]	Platform specific settings in SwBaseType *float32*
[SWS_APT_00047]	SwBaseType *float64*
[SWS_APT_00048]	Platform specific settings in SwBaseType *float64*
[SWS_APT_00050]	SwBaseType *boolean*
[SWS_APT_00051]	Platform specific settings in SwBaseType *boolean*

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

B.5 Constraint and Specification Item History of this document according to AUTOSAR R19-11

B.5.1 Added Traceables in R19-11

none

B.5.2 Changed Traceables in R19-11

none

B.5.3 Deleted Traceables in R19-11

none

B.6 Constraint and Specification Item History of this document according to AUTOSAR R20-11

B.6.1 Added Traceables in R20-11

none

B.6.2 Changed Traceables in R20-11

none

B.6.3 Deleted Traceables in R20-11

none