Document Title	Specification of Persistency
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	858

Document Status	Final
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	18-03

Document Change History

Date	Release	Changed by	Description
2018-03-29	18-03	AUTOSAR Release Management	• Installation/update of persistent data • Data types supported by KeyValueStorage API
2017-10-27	17-10	AUTOSAR Release Management	• Introduction of AUTOSAR model • Security added • Redundancy added • Rework of FileProxy/Stream API
2017-03-31	17-03	AUTOSAR Release Management	• Initial release

Table of Contents

1 Introduction and functional overview

This document is the software specification of the `Persistency` functional cluster within the `Adaptive Platform`.

`Persistency` offers mechanisms to `Adaptive Application`s to store information in the non-volatile memory of a machine. The data is available over boot and ignition cycles.

The `Persistency` functional cluster will typically be implemented as a library that runs within an `Adaptive Application`, with the rights of that `Adaptive Application`.

2 Acronyms and Abbreviations

There are no acronyms and abbreviations relevant within this document that are not included in the [1, AUTOSAR glossary].

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_TR_Glossary

[2] Requirements on Persistency
AUTOSAR_RS_Persistency

[3] General Requirements specific to Adaptive Platform
AUTOSAR_RS_General

[4] Requirements on Update and Configuration Management
AUTOSAR_RS_UpdateAndConfigManagement

[5] Specification of Update and Configuration Management
AUTOSAR_SWS_UpdateAndConfigManagement

[6] Specification of Manifest
AUTOSAR_TPS_ManifestSpecification

[7] Specification of Platform Types for Adaptive Platform
AUTOSAR_SWS_AdaptivePlatformTypes

4 Constraints and assumptions

4.1 Limitations

- The interpretation of deployment related information in the AUTOSAR model is not yet covered in detail in this specification. Remarks are added to the relevant chapters (e.g. 6.1.1).

4.2 Constraints on Configuration

[SWS_PER_00200] ⌈The `Persistency` cluster shall reject any configuration in which multiple `PersistencyPortPrototypeToKeyValueDatabaseMapping`s map to the same `PersistencyKeyValueDatabase`. ⌋*(RS_PER_00010)*

[SWS_PER_00201] ⌈The `Persistency` cluster shall reject any configuration in which multiple `PersistencyPortPrototypeToFileArrayMapping`s map to the same `PersistencyFileArray`. ⌋*(RS_PER_00010)*

5 Requirements Tracing

The following table references the features specified in [2], [3], [4] and links to the fulfillments of these.

Feature	Description	Satisfied by
[RS_AP_00111]	The AUTOSAR Adaptive platform shall support source code portability for AUTOSAR Adaptive applications.	[SWS_PER_UNUSED]
[RS_AP_00113]	API specification shall comply with selected coding guidelines.	[SWS_PER_UNUSED]
[RS_AP_00114]	C++ binding shall be compatible with C++11.	[SWS_PER_UNUSED]
[RS_AP_00115]	Standardized scope/namespace definition	[SWS_PER_00002]
[RS_AP_00116]	Header file name	[SWS_PER_00146] [SWS_PER_00147] [SWS_PER_00172]
[RS_AP_00117]	Class and structure names	[SWS_PER_00146] [SWS_PER_00147]
[RS_AP_00118]	Exceptions	[SWS_PER_00132] [SWS_PER_00133] [SWS_PER_00134] [SWS_PER_00172] [SWS_PER_00173] [SWS_PER_00174] [SWS_PER_00175]
[RS_AP_00119]	Return values / application errors	[SWS_PER_UNUSED]
[RS_AP_00120]	Method and Function names	[SWS_PER_UNUSED]
[RS_AP_00121]	Parameter names	[SWS_PER_UNUSED]
[RS_AP_00122]	Type names	[SWS_PER_UNUSED]

[RS_AP_00124]	Variable names	[SWS_PER_UNUSED]
[RS_AP_00125]	Enumerator and constant names	[SWS_PER_UNUSED]
[RS_PER_00001]	Persistency shall support storage of persistent data	[SWS_PER_00106] [SWS_PER_00107] [SWS_PER_00108] [SWS_PER_00110] [SWS_PER_00111] [SWS_PER_00112] [SWS_PER_00113] [SWS_PER_00114] [SWS_PER_00115] [SWS_PER_00116] [SWS_PER_00119] [SWS_PER_00122] [SWS_PER_00124] [SWS_PER_00125] [SWS_PER_00126] [SWS_PER_00127] [SWS_PER_00128] [SWS_PER_00140] [SWS_PER_00141] [SWS_PER_00142] [SWS_PER_00143] [SWS_PER_00144] [SWS_PER_00145] [SWS_PER_00148] [SWS_PER_00160] [SWS_PER_00161] [SWS_PER_00162] [SWS_PER_00163] [SWS_PER_00164] [SWS_PER_00165] [SWS_PER_00166] [SWS_PER_00167] [SWS_PER_00168] [SWS_PER_00169] [SWS_PER_00170] [SWS_PER_00171] [SWS_PER_00176] [SWS_PER_00302] [SWS_PER_00303] [SWS_PER_00304]
[RS_PER_00002]	Persistency shall support to retrieve data that has been persistently stored on a platform instance	[SWS_PER_00042] [SWS_PER_00043] [SWS_PER_00044] [SWS_PER_00046] [SWS_PER_00047] [SWS_PER_00048] [SWS_PER_00049] [SWS_PER_00052] [SWS_PER_00500]

[RS_PER_00003]	Persistency shall support identification of data using a unique identifier	[SWS_PER_00004] [SWS_PER_00041] [SWS_PER_00050] [SWS_PER_00080] [SWS_PER_00129] [SWS_PER_00130] [SWS_PER_00131] [SWS_PER_00146] [SWS_PER_00147] [SWS_PER_00180] [SWS_PER_00181] [SWS_PER_00182]
[RS_PER_00004]	Persistency shall support access to file-like structures	[SWS_PER_00106] [SWS_PER_00107] [SWS_PER_00108] [SWS_PER_00110] [SWS_PER_00111] [SWS_PER_00112] [SWS_PER_00113] [SWS_PER_00114] [SWS_PER_00115] [SWS_PER_00116] [SWS_PER_00119] [SWS_PER_00122] [SWS_PER_00124] [SWS_PER_00125] [SWS_PER_00126] [SWS_PER_00127] [SWS_PER_00128] [SWS_PER_00140] [SWS_PER_00141] [SWS_PER_00142] [SWS_PER_00143] [SWS_PER_00144] [SWS_PER_00145] [SWS_PER_00148] [SWS_PER_00160] [SWS_PER_00161] [SWS_PER_00162] [SWS_PER_00163] [SWS_PER_00164] [SWS_PER_00165] [SWS_PER_00166] [SWS_PER_00167] [SWS_PER_00168] [SWS_PER_00169] [SWS_PER_00170] [SWS_PER_00171] [SWS_PER_00176]
[RS_PER_00005]	Persistency shall support encryption/decryption of persistent data	[SWS_PER_00210] [SWS_PER_00211]
[RS_PER_00008]	Persistency shall support detection of data corruption in persistent memory	[SWS_PER_00220] [SWS_PER_00221]
[RS_PER_00009]	Persistency shall support data recovery mechanisms if persistent data was corrupted	[SWS_PER_00220] [SWS_PER_00222]

[RS_PER_00010]	The layout of persistent data shall be configurable	[SWS_PER_00200] [SWS_PER_00201]
[RS_UCM_00001]	UCM shall support installing new software on Adaptive Platform	[SWS_PER_00250] [SWS_PER_00251] [SWS_PER_00252] [SWS_PER_00253] [SWS_PER_00254] [SWS_PER_00255] [SWS_PER_00256] [SWS_PER_00257] [SWS_PER_00258] [SWS_PER_00259] [SWS_PER_00260] [SWS_PER_00261] [SWS_PER_00262] [SWS_PER_00264] [SWS_PER_00265] [SWS_PER_00266] [SWS_PER_00267] [SWS_PER_00268] [SWS_PER_00269] [SWS_PER_00270] [SWS_PER_00271] [SWS_PER_00272] [SWS_PER_00273]
[RS_UCM_00002]	UCM shall support reporting version information for an Adaptive Platform	[SWS_PER_00250] [SWS_PER_00251] [SWS_PER_00274] [SWS_PER_00275] [SWS_PER_00276] [SWS_PER_00277] [SWS_PER_00278] [SWS_PER_00279] [SWS_PER_00280] [SWS_PER_00281] [SWS_PER_00282] [SWS_PER_00283] [SWS_PER_00284] [SWS_PER_00285]
[RS_UCM_00003]	UCM shall support updating installed software on Adaptive Platform	[SWS_PER_00250] [SWS_PER_00300] [SWS_PER_00301]
[RS_UCM_00004]	UCM shall support uninstalling software on Adaptive Platform	[SWS_PER_UNUSED]
[RS_UCM_00005]	UCM shall make sure that persistent data owned by uninstalled software is deleted	[SWS_PER_UNUSED]
[RS_UCM_00006]	UCM shall check Software Package authentication during processing using signature verification	[SWS_PER_UNUSED]
[RS_UCM_00007]	UCM shall check that software dependencies are fulfilled	[SWS_PER_UNUSED]
[RS_UCM_00008]	UCM shall support a recovery mechanism in case of failed update process	[SWS_PER_UNUSED]
[RS_UCM_00010]	UCM shall support reporting of Software Packages downloaded for Adaptive Platform	[SWS_PER_UNUSED]

[RS_UCM_00011]	UCM shall support reporting software versions which have been installed and will be activated when new versions are activated	[SWS_PER_UNUSED]
[RS_UCM_00012]	UCM shall check the consistency of Software Package during processing	[SWS_PER_UNUSED]
[RS_UCM_00013]	UCM shall check that it has enough resources to receive, process and store the Software Package and associated data	[SWS_PER_UNUSED]
[RS_UCM_00014]	UCM shall check that correct amount of data has been transferred for the Software Package	[SWS_PER_UNUSED]
[RS_UCM_00015]	UCM shall remove all unneeded data after Software Package processing has finished	[SWS_PER_UNUSED]
[RS_UCM_00016]	UCM shall check installed software is consistent with provided Software Cluster Manifest	[SWS_PER_UNUSED]
[RS_UCM_00017]	UCM shall support installing and updating the persistent data storage for an Adaptive Application	[SWS_PER_UNUSED]
[RS_UCM_00018]	UCM shall announce when an application has been installed, updated or uninstalled	[SWS_PER_UNUSED]
[RS_UCM_00019]	UCM shall support simultaneous transfer from multiple clients	[SWS_PER_UNUSED]
[RS_UCM_00020]	UCM shall support cancel of an update or install operation	[SWS_PER_UNUSED]
[RS_UCM_00021]	UCM shall support atomic activation of installed or updated packages	[SWS_PER_UNUSED]
[RS_UCM_00022]	UCM shall support logging of the update or installation process	[SWS_PER_UNUSED]
[RS_UCM_00023]	UCM shall provide an interface to read progress of the update	[SWS_PER_UNUSED]
[RS_UCM_00024]	UCM shall provide an interface to read internal status of UCM	[SWS_PER_UNUSED]
[RS_UCM_00025]	UCM shall support efficient streaming of Software Package data	[SWS_PER_UNUSED]

6 Functional specification

6.1 General description

The functional cluster `Persistency` offers two different mechanisms to access persistent memory.

Key-Value-Storage offers access to one or multiple Key-Value-Databases for every `AdaptiveApplicationSwComponentType`. Every Key-Value-Database is represented by a `PortPrototype` typed by a `PersistencyKeyValueDatabaseInterface` in Application Design for the respective `AdaptiveApplicationSwComponentType`. Every Key-Value-Database can hold multiple Key-Value-Pairs.

A Key-Value database with predefined Key-Value pairs can be deployed with default data during installation or update of an Adaptive Application. This operation is triggered by the `UCM` module (see [5]) during installation or update using the deployment information and data provided by the installation package of the Adaptive Application. See section 6.4.

File-Proxies are represented by a PortPrototype typed by a Persistency-FileProxyInterface in Application Design for the respective AdaptiveApplica-tionSwComponentType. Every File-Proxy can hold multiple files as described in [6]. Similar to the Key-Value Pairs mentioned above, additional files can be created by the implementation of the Adaptive Application using the Persistency API (see 7.2.2.1.4 and 7.2.2.1.6).

A File-Proxy with predefined files with initial content can be deployed during installation or update. This operation is triggered by the UCM module too. All needed deployment information and files come with the installation package of the Adaptive Application. See section 6.4.

6.1.1 Architectural concepts

Persistent data is always private to one application instance. This design decision was taken to prevent an additional communication path for applications besides the mechanisms provided by the functional cluster Communication Management. If persistent data needs to be shared between multiple applications it is the duty of an application developer to provide Service Interfaces for communication.

6.1.2 Design decisions

The API specification holds classes for Key-Value-Storage and File-Proxy access, taking the shortName of PortPrototype typed by a PersistencyKeyValue-DatabaseInterface or a PersistencyFileProxyInterface as an std::string input parameter (see 7.2.1.1 and 7.2.2.1.7).

6.2 Security concepts

Security requirements of the Key-Value-Storage and File-Proxy are modeled with the CryptoNeed element from [6]. For details which security algorithms etc. are supported refer to the CryptoNeed element.

[SWS_PER_00210] ⌈ The Persistency cluster shall encrypt data before storing it to the persistent memory according to the CryptoNeedToCryptoJobMapping. ⌋ *(RS_PER_00005)*

[SWS_PER_00211] ⌈ The Persistency cluster shall decrypt data after reading it from persistent memory according to the CryptoNeedToCryptoJobMapping. ⌋ *(RS_PER_00005)*

6.3 Redundancy concepts

The implementer of the `Persistency` functional cluster shall take care of the error detection and error correction mechanisms which are implementation specific. The need of a Key-Value-Storage for redundancy can be modeled with element `PersistencyRedundancyEnum` from [6].

Since this specification makes no assumptions about the used memory technology in an Adaptive Platform and available error detection and correction mechanisms the requirements on the implementation of the Redundancy concept is rather unspecific.

[SWS_PER_00220] ⌈ The `Persistency` cluster shall introduce redundant information for every Key-Value-Pair stored in a Key-Value-Database represented by a `PortPrototype` typed by a `PersistencyKeyValueDatabaseInterface` with a `PersistencyDataProvidedComSpec` where `PersistencyDataProvidedComSpec.redundancy` equals `redundant`. ⌋*(RS_PER_00008, RS_PER_00009)*

[SWS_PER_00221] ⌈ The `Persistency` cluster shall use the redundant information to detect data corruption in the persistent memory. ⌋*(RS_PER_00008)*

[SWS_PER_00222] ⌈ The `Persistency` cluster shall use the redundant information to correct corrupted data if possible. ⌋*(RS_PER_00009)*

6.4 Persistent data in Update and Configuration Management

There are three main use cases in `Update and Configuration Management` for handling of `Adaptive Application`s over the lifecycle of a car, ECU or Adaptive Machine:

- Installation of new software
- Update of already installed software
- Uninstallation of installed software

It is obvious that for all three use cases data in `Persistency` needs to be handled which includes the scenarios:

- Deployment of persistent data that was defined by an application designer
- Deployment of persistent data that was defined by an application designer and changed by an integrator
- Deployment of persistent data that was defined by an integrator
- Definition of update strategies for persistent data when a new version of an application is installed
- Removing persistent data when an application is uninstalled

Based on the fact that persistent data can and will be changed by Adaptive Applications during their execution, a flexible and fine-grained configuration approach is needed to define the actions taken for persistent data in the Update and Configuration Management use cases.

[SWS_PER_00250] ⌈ All actions concerning handling of persistent data during installation, update and uninstallation of Adaptive Applications described in this section (6.4) shall be executed in the ProcessSwPackage method described in [5]. ⌋ *(RS_UCM_00001, RS_UCM_00002, RS_UCM_00003)*

[SWS_PER_00251] ⌈ All specification items on the updateStrategy in this chapter (6.4) shall always refer to the final updateStrategy in a given configuration (e.g. PersistencyKeyValueDatabase.updateStrategy overrides PersistencyKeyValueDatabaseInterface.updateStrategy). ⌋*(RS_UCM_00001, RS_UCM_00002)*

6.4.1 Installation of Key-Value-Databases

[SWS_PER_00252] ⌈ When installing a new Adaptive Application, for every PersistencyDataElement in a PersistencyKeyValueDatabaseInterface used in a PortPrototype, Persistency shall create an entry in the Key-Value-Database addressed by this PortPrototype. ⌋*(RS_UCM_00001)*

[SWS_PER_00253] ⌈ The created entry in the Key-Value-Database shall have the shortName of the PersistencyDataElement as key. ⌋*(RS_UCM_00001)*

[SWS_PER_00254] ⌈ The created entry in the Key-Value-Database shall be of the datatype defined in PersistencyDataElement. ⌋*(RS_UCM_00001)*

[SWS_PER_00255] ⌈ The value of the created entry in the Key-Value-Database shall be taken from the PersistencyDataRequiredComSpec.initValue referring to this PersistencyDataElement in dataElement, if no PersistencyKeyValuePair with the same shortName as the PersistencyDataElement exists in the PersistencyKeyValueDatabase that is mapped to the aggregating PortPrototype typed by the PersistencyKeyValueDatabaseInterface with a PersistencyPortPrototypeToKeyValueDatabaseMapping in the context of the Process of this Adaptive Application. ⌋*(RS_UCM_00001)*

[SWS_PER_00256] ⌈ The value of the created entry in the Key-Value-Database shall be taken from the PersistencyKeyValuePair.initValue, if a PersistencyKeyValuePair with the same shortName as the PersistencyDataElement exists in the PersistencyKeyValueDatabase that is mapped to the aggregating PortPrototype typed by the PersistencyKeyValueDatabaseInterface with a PersistencyPortPrototypeToKeyValueDatabaseMapping in the context of the Process of this Adaptive Application. ⌋*(RS_UCM_00001)*

[SWS_PER_00257] ⌈ If a PersistencyDataElement exists that is neither referenced by a PersistencyDataRequiredComSpec with a PersistencyDataRequiredComSpec.initValue nor a PersistencyKeyValuePair with the same shortName as the PersistencyDataElement in the PersistencyKeyValue-Database that is mapped to the aggregating PortPrototype typed by the PersistencyKeyValueDatabaseInterface with a PersistencyPortPrototype-ToKeyValueDatabaseMapping in the context of the Process of this Adaptive Application exists, no entry in the Key-Value-Database shall be created. ⌋ *(RS_UCM_00001)*

[SWS_PER_00258] ⌈ Persistency shall reject any configuration in which incompatible AutosarDataTypes are given in the PersistencyDataElement and the mapped PersistencyKeyValuePair.valueDataType. ⌋*(RS_UCM_00001)*

[SWS_PER_00259] ⌈ When installing a new Adaptive Application, for every PersistencyKeyValuePair in a PersistencyKeyValueDatabase that is mapped to a PortPrototype typed by the PersistencyKeyValueDatabaseInterface using a PersistencyPortPrototypeToKeyValueDatabaseMapping, Persistency shall create an entry in this Key-Value-Database if no Persistency-DataElement in the PersistencyKeyValueDatabaseInterface exists with the same shortName as the PersistencyKeyValuePair. ⌋*(RS_UCM_00001)*

[SWS_PER_00260] ⌈ The created entry in the Key-Value-Database shall have the shortName of the PersistencyKeyValuePair as key. ⌋*(RS_UCM_00001)*

[SWS_PER_00261] ⌈ The created entry in the Key-Value-Database shall be of the datatype defined in PersistencyKeyValuePair.valueDataType. ⌋ *(RS_UCM_00001)*

[SWS_PER_00262] ⌈ The created entry in the Key-Value-Database shall have the PersistencyKeyValuePair.initValue as value. ⌋*(RS_UCM_00001)*

[SWS_PER_00264] ⌈ If the final updateStrategy of an entry to be created is delete, no entry in the Key-Value-Database shall be created. ⌋*(RS_UCM_00001)*

6.4.2 Installation of File-Proxies

[SWS_PER_00265] ⌈ When installing a new Adaptive Application, for every PersistencyFileProxy in a PersistencyFileProxyInterface used in a PortPrototype, Persistency shall create an entry in the File-Proxy addressed by this PortPrototype. ⌋*(RS_UCM_00001)*

[SWS_PER_00266] ⌈ The created entry in the File-Proxy shall have the PersistencyFileProxy.fileName as key. ⌋*(RS_UCM_00001)*

[SWS_PER_00267] ⌈ The content of the created entry in the File-Proxy shall be taken from a file in the Software Package addressed by PersistencyFileProxy.contentUri if no PersistencyFile with the same shortName as the Persistency-FileProxy exists in the PersistencyFileArray that is mapped to the aggregating PortPrototype typed by the PersistencyFileProxyInterface with a PersistencyPortPrototypeToFileArrayMapping in the context of the Process of this Adaptive Application. ⌋*(RS_UCM_00001)*

[SWS_PER_00268] ⌈ The content of the created entry in the File-Proxy shall be taken from a file in the Software Package addressed by PersistencyFile.contentUri, if a PersistencyFile with the same shortName as the PersistencyFileProxy exists in the PersistencyFileArray that is mapped to the aggregating PortPrototype typed by the PersistencyFileProxyInterface with a PersistencyPortPrototypeToFileArrayMapping in the context of the Process of this Adaptive Application. ⌋*(RS_UCM_00001)*

[SWS_PER_00269] ⌈ When installing a new Adaptive Application, for every PersistencyFile in a PersistencyFileArray that is mapped to a PersistencyFileProxyInterface using a PersistencyPortPrototypeToFileArrayMapping, Persistency shall create an entry in this File-Proxy if no PersistencyFileProxy in the PortPrototype typed by the PersistencyFileProxyInterface exists with the same shortName as the PersistencyFile. ⌋*(RS_UCM_00001)*

[SWS_PER_00270] ⌈ The created entry in the File-Proxy shall have the PersistencyFile.fileName as key. ⌋*(RS_UCM_00001)*

[SWS_PER_00271] ⌈ The content of the created entry in the File-Proxy shall be taken from a file in the Software Package addressed by the PersistencyFile.contentUri. ⌋*(RS_UCM_00001)*

[SWS_PER_00272] ⌈ Persistency shall reject any configuration in which a PersistencyFileProxy and a PersistencyFile with the same fileName but different shortNames exist that are mapped by a PersistencyPortPrototypeToFileArrayMapping referring to the PortPrototype typed by the PersistencyFileProxyInterface and the PersistencyFileArray. ⌋*(RS_UCM_00001)*

[SWS_PER_00273] ⌈ If the final updateStrategy of an entry to be created is delete, no entry in the File-Proxy shall be created. ⌋*(RS_UCM_00001)*

6.4.3 Update of Key-Value-Databases

[SWS_PER_00274] ⌈ When updating an Adaptive Application, for Key-Value-Databases the same requirements as described in subsection 6.4.1 shall apply but the final updateStrategy also needs to be respected. ⌋*(RS_UCM_00002)*

[SWS_PER_00275] ⌈ If the final updateStrategy (enumeration PersistencyElementLevelUpdateStrategyEnum) of a PersistencyDataElement or a PersistencyKeyValuePair is overwrite, the entry to the Key-Value-Database shall be created and even overwrite an existing Key-Value-Pair with the same key. ⌋*(RS_UCM_00002)*

[SWS_PER_00276] ⌈ If the final updateStrategy (enumeration PersistencyElementLevelUpdateStrategyEnum) of a PersistencyDataElement or a PersistencyKeyValuePair is keepExisting, an existing Key-Value-Pair with the same key shall be kept in the Key-Value-Database. If no Key-Value-Pair with the same key exists, the entry to the Key-Value-Database shall be created. ⌋*(RS_UCM_00002)*

[SWS_PER_00277] ⌈ If the final updateStrategy (enumeration PersistencyElementLevelUpdateStrategyEnum) of a PersistencyDataElement or a PersistencyKeyValuePair is delete, an existing Key-Value-Pair with the same key shall be deleted and no entry to the Key-Value-Database shall be created. ⌋*(RS_UCM_00002)*

[SWS_PER_00278] ⌈ If the final updateStrategy (enumeration PersistencyCollectionLevelUpdateStrategyEnum) of a PersistencyKeyValueDatabaseInterface or a PersistencyKeyValueDatabase is keepExisting, all Key-Value-Pairs in the Key-Value-Database that are not explicitly modeled as PersistencyDataElement or PersistencyKeyValuePair shall be kept. ⌋*(RS_UCM_00002)*

[SWS_PER_00279] ⌈ If the final updateStrategy (enumeration PersistencyCollectionLevelUpdateStrategyEnum) of a PersistencyKeyValueDatabaseInterface or a PersistencyKeyValueDatabase is delete, all Key-Value-Pairs in the Key-Value-Database that are not explicitly modeled as PersistencyDataElement or PersistencyKeyValuePair shall be deleted. ⌋*(RS_UCM_00002)*

6.4.4 Update of File-Proxies

[SWS_PER_00280] ⌈ When updating an Adaptive Application, for File-Proxies the same requirements as described in subsection 6.4.2 shall apply but the final updateStrategy also needs to be respected. ⌋*(RS_UCM_00002)*

[SWS_PER_00281] ⌈ If the final updateStrategy (enumeration PersistencyElementLevelUpdateStrategyEnum) of a PersistencyFileProxy or a PersistencyFile is overwrite, the entry to the File-Proxy shall be created and even overwrite an existing entry with the same key. ⌋*(RS_UCM_00002)*

[SWS_PER_00282] ⌈ If the final updateStrategy (enumeration PersistencyElementLevelUpdateStrategyEnum) of a PersistencyFileProxy or a PersistencyFile is keepExisting, an existing entry with the same key shall be kept in the File-Proxy. If no entry with the same key exists, the entry to the File-Proxy shall be created. ⌋*(RS_UCM_00002)*

[SWS_PER_00283] ⌈ If the final updateStrategy (enumeration PersistencyElementLevelUpdateStrategyEnum) of a PersistencyFileProxy or a PersistencyFile is delete, an existing entry with the same key shall be deleted and no entry to the File-Proxy shall be created. ⌋*(RS_UCM_00002)*

[SWS_PER_00284] ⌈ If the final updateStrategy (enumeration PersistencyCollectionLevelUpdateStrategyEnum) of a PersistencyFileProxyInterface or a PersistencyFileArray is keepExisting, all entries in the File-Proxy that are not explicitly modeled as PersistencyFileProxy or PersistencyFile shall be kept. ⌋*(RS_UCM_00002)*

[SWS_PER_00285] ⌈ If the final updateStrategy (enumeration PersistencyCollectionLevelUpdateStrategyEnum) of a PersistencyFileProxyInterface or a PersistencyFileArray is delete, all entries in the File-Proxy that are not explicitly modeled as PersistencyFileProxy or PersistencyFile shall be deleted. ⌋*(RS_UCM_00002)*

6.4.5 Uninstallation of Key-Value-Databases

[SWS_PER_00300] ⌈ When deleting an Adaptive Application, Persistency shall delete all Key-Value-Databases used by this Adaptive Application from the Adaptive Machine. ⌋*(RS_UCM_00003)*

6.4.6 Uninstallation of File-Proxies

[SWS_PER_00301] ⌈ When deleting an Adaptive Application, Persistency shall delete all File-Proxies used by this Adaptive Application from the Adaptive Machine. ⌋*(RS_UCM_00003)*

6.5 Supported data types in KeyValueStorage

The Persistency cluster supports several data types for PersistencyKeyValueDatabases, which can be used in templated functions for getting and setting the values of that database. See sections 7.2.1.5 and 7.2.1.6. The following classes of data types are supported.

[SWS_PER_00302] ⌈ The Persistency cluster shall support all datatypes described in [7] in templated functions for access to the PersistencyKeyValueDatabase. ⌋ *(RS_PER_00001)*

[SWS_PER_00303] ⌈ The Persistency cluster shall support byte arrays which contain streamed data types in templated functions for access to the PersistencyKeyValueDatabase. ⌋*(RS_PER_00001)*

[SWS_PER_00304] ⌈ The Persistency cluster shall support all ImplementationDataTypes referred via PersistencyKeyValueDatabaseInterface.dataTypeForSerialization or via PersistencyKeyValueDatabaseInterface.dataElement in the application design in templated functions for access to the PersistencyKeyValueDatabase. See [6]. ⌋*(RS_PER_00001)*

7 API specification

7.1 Type definitions

7.1.1 KeyValueStorage::Status

Name:	**[SWS_PER_00004]** ⌈`KeyValueStorage::Status`⌋*(RS_PER_00003)*		
Type:	`Scoped Enumeration of uint8_t`		

Range:	`kSuccess`
	`kNotFound`
	`kCheckSumError`
	`kGeneralError`
		0
		1
		2
		3
			--
			--
			--
			--

Syntax:	`enum class Type : uint8_t {`
	`kSuccess,`
	`kNotFound,`
	`kCheckSumError,`
	`kGeneralError,`
	`};`		

Header file:	key_value_storage.h		
Description:	Defines the states of a `KeyValueStorage::Status` (see subsection 7.2.1). `kSuccess` - indicates that the value was successfully restored from the KVS-storage `kNotFound` - requested key was not found `kCheckSumError` - the key-value pair was found, but the checksum of it is incorrect `kGeneralError` - any other failure		

Table 7.1: KeyValueStorage::Status

7.1.2 BasicOperations::SeekDirection

Name:	**[SWS_PER_00146]** ⌈`BasicOperations::SeekDirection`⌋ *(RS_PER_00003, RS_AP_00117, RS_AP_00116)*		
Type:	`Scoped enumeration`		

Range:	`kBeg`
	`kEnd`
	`kCur`
		0
		1
		2
			--
			--
			--

Syntax:	`enum class SeekDirection {`
	`kBeg,`
	`kEnd,`
	`kCur`
	`};`		

Header file:	basic_operations.h		
Description:	Specification of seek direction. `BasicOperations::SeekDirection` (see subsubsection 7.2.2.2).		

Table 7.2: BasicOperations::Type

7.1.3 BasicOperations::OpenMode

Name:	**[SWS_PER_00147]** ⌈BasicOperations::OpenMode⌋*(RS_PER_00003, RS_AP_00117, RS_AP_00116)*		
Type:	`Scoped enumeration uint8_t`		
Range:	`kApp`	`1 << 0`	`--`
	`kBinary`	`1 << 1`	`--`
	`kIn`	`1 << 2`	`--`
	`kOut`	`1 << 3`	`--`
	`kTrunc`	`1 << 4`	`--`
	`kAte`	`1 << 5`	`--`
Syntax:	`class OpenMode : uint8_t {` `kApp,` `kBinary,` `kIn,` `kOut,` `kTrunc,` `kAte` `};`		
Header file:	basic_operations.h		
Description:	How the file shall be opened. `BasicOperations::OpenMode` (see subsubsection 7.2.2.2).		

Table 7.3: BasicOperations::OpenMode

7.1.4 int_type

Name:	**[SWS_PER_00180]** ⌈int_type⌋*(RS_PER_00003)*
Type:	`integer`
Description:	Indicate an output for certain file proxy functions.

Table 7.4: int_type

7.1.5 pos_type

Name:	**[SWS_PER_00181]** ⌈pos_type⌋*(RS_PER_00003)*
Type:	`integer`
Description:	Specify position for certain file proxy functions.

Table 7.5: pos_type

7.1.6 off_type

Name:	**[SWS_PER_00182]** ⌈off_type⌋*(RS_PER_00003)*
Type:	`signed integer`
Description:	Indicate an offset (for certain file proxy functions)

Table 7.6: off_type

7.2 Class definitions

[SWS_PER_00002] ⌈ All specified classes within the `Persistency` specification shall reside within the C++ namespace `ara::per`. ⌋*(RS_AP_00115)*

7.2.1 KeyValueStorage class

[SWS_PER_00080] ⌈ The `KeyValueStorage` class defined in the `ara/per/key_value_storage.h` header file shall define an interface to the common key-value store functions. ⌋*(RS_PER_00003)*

[SWS_PER_00041] ⌈ The `KeyValueStorage` class shall delete the following member functions as the class shall not be copyable or assignable.
`KeyValueStorage(const KeyValueStorage&) = delete;`
`KeyValueStorage& operator=(const KeyValueStorage&) = delete;` ⌋
(RS_PER_00003)

7.2.1.1 KeyValueStorage::KeyValueStorage

Service name:	**[SWS_PER_00052]** ⌈`KeyValueStorage Constructor`⌋*(RS_PER_00002)*	
Syntax:	`KeyValueStorage(` `const std::string& database)`	
Parameters (in):	`const std::string&`	`shortName` of `PortPrototype` typed by a `PersistencyKeyValueDatabaseInterface`
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	yes	Check exceptions chapter (subsection 7.2.3).
Return value:	None	
Description:	Creates an instance of `KeyValueStorage` which configures the storage location.	

Table 7.7: KeyValueStorage Constructor

7.2.1.2 KeyValueStorage::~KeyValueStorage

Service name:	**[SWS_PER_00050]** ⌈`KeyValueStorage Destructor`⌋*(RS_PER_00003)*
Syntax:	`~KeyValueStorage()`
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Exceptions:	None
Return value:	None
Description:	~KeyValueStorage deletes the `KeyValueStorage` instance.

Table 7.8: KeyValueStorage Destructor

7.2.1.3 KeyValueStorage::GetAllKeys

Service name:	**[SWS_PER_00042]** ⌈`KeyValueStorage::GetAllKeys`⌋*(RS_PER_00002)*	
Syntax:	`std::vector<std::string> GetAllKeys()`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	yes	Check exceptions chapter (subsection 7.2.3).
Return value:	std::vector<std::string>	Vector of type `std::string`.
Description:	The `KeyValueStorage` class shall provide a method to get a list of all keys explicitly set in the dataset. It shall return the list of available keys.	

Table 7.9: KeyValueStorage::GetAllKeys

7.2.1.4 KeyValueStorage::HasKey

Service name:	**[SWS_PER_00043]** ⌈`KeyValueStorage::HasKey`⌋*(RS_PER_00002)*	
Syntax:	`bool HasKey(const std::string& key)`	
Parameters (in):	const std::string&	Value of the key that shall be checked.
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	`bool`	Returns if the `key` could be located.
Description:	The `KeyValueStorage` class shall provide a method to determine whether the key exists in the dataset. It shall return true if the key exists in the dataset otherwise false.	

Table 7.10: KeyValueStorage::HasKey

7.2.1.5 KeyValueStorage::GetValue

Service name:	**[SWS_PER_00044]** ⌈ `KeyValueStorage::GetValue`⌋ *(RS_PER_00002)*	
Syntax:	`template <class T> Status GetValue( const std::string& key, T& value)`	
Parameters (in):	const std::string&	Value of the key associated with the value.
Parameters (in):	T&	value of associated type.
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	yes	Check exceptions chapter (subsection 7.2.3).
Return value:	`Status`	Returns an enum of type `Status`. KeyValueStorage::Status.
Description:	The `KeyValueStorage` class shall provide a method to get the value assigned to the key.	

Table 7.11: KeyValueStorage::GetValue

7.2.1.6 KeyValueStorage::SetValue

Service name:	**[SWS_PER_00046]** ⌈`KeyValueStorage::SetValue`⌋*(RS_PER_00002)*

Syntax:	`template <class T> Status SetValue( const std::string& key, const T& value)`	
Parameters (in):	const std::string&	
Parameters (in):	T&	value of associated type.
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	yes	Check exceptions chapter (subsection 7.2.3).
Return value:	`Status`	Returns an enum of type `Status`. KeyValueStorage::Status.
Description:	The KeyValueStorage class shall provide a method to assign the value to the key. It will be explicitly stored it in the dataset.	

Table 7.12: KeyValueStorage::SetValue

7.2.1.7 KeyValueStorage::RemoveKey

Service name:	**[SWS_PER_00047]** ⌈`KeyValueStorage::RemoveKey`⌋*(RS_PER_00002)*	
Syntax:	`void RemoveKey(const std::string& key)`	
Parameters (in):	const std::string&	Value of the key associated with the value.
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	None	
Description:	The `KeyValueStorage` class shall provide a method that removes the key and associated value.	

Table 7.13: KeyValueStorage::RemoveKey

7.2.1.8 KeyValueStorage::RemoveAllKeys

Service name:	**[SWS_PER_00048]** ⌈`KeyValueStorage::RemoveAllKeys`⌋ *(RS_PER_00002)*
Syntax:	`void RemoveAllKeys(void)`
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Exceptions:	None
Return value:	None
Description:	The `KeyValueStorage` class shall provide a method that removes all keys and associated values.

Table 7.14: KeyValueStorage::RemoveAllKeys

7.2.1.9 KeyValueStorage::SyncToStorage

Service name:	**[SWS_PER_00049]** ⌈`KeyValueStorage::SyncToStorage`⌋ *(RS_PER_00002)*
Syntax:	`void SyncToStorage(void)`
Parameters (in):	None
Parameters (inout):	None

Parameters (out):	None	
Exceptions:	yes	Check exceptions chapter (subsection 7.2.3).
Return value:	None	
Description:	The KeyValueStorage class shall provide a method to trigger flushing of key-value pairs to the physical storage.	

Table 7.15: KeyValueStorage::SyncToStorage

7.2.2 FileProxy

7.2.2.1 FileProxyAccessorFactory class

[SWS_PER_00176] ⌈ The `Persistency` cluster shall provide an abstract class which creates objects to read or write memory blocks. This class shall be called `FileProxyAccessorFactory` and shall be defined in the `ara/per/file_proxy_accessor_factory.h` header. ⌋*(RS_PER_00001, RS_PER_00004)*

7.2.2.1.1 FileProxyAccessorFactory::GetAllKeys

Service name:	**[SWS_PER_00110]** ⌈`FileProxyAccessorFactory::GetAllKeys`⌋ *(RS_PER_00001, RS_PER_00004)*	
Syntax:	`std::vector<std::string> GetAllKeys()`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	std::vector <std::string>	Returns a vector of identifiers.
Description:	The function returns a vector of available (file) identifiers within this proxy class.	

Table 7.16: FileProxyAccessorFactory GetAllKeys

7.2.2.1.2 FileProxyAccessorFactory::DeleteKey

Service name:	**[SWS_PER_00111]** ⌈`FileProxyAccessorFactory::DeleteKey`⌋ *(RS_PER_00001, RS_PER_00004)*	
Syntax:	`void DeleteKey (const std::string& key)`	
Parameters (in):	const std::string& key	Identifier of the file.
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	None	
Description:	The function deletes a file with the given identifier from this accessor.	

Table 7.17: FileProxyAccessorFactory DeleteKey

7.2.2.1.3 FileProxyAccessorFactory::HasKey

Service name:	**[SWS_PER_00112]** ⌈`FileProxyAccessorFactory::HasKey`⌋ *(RS_PER_00001, RS_PER_00004)*	
Syntax:	`bool HasKey (const std::string& key)`	
Parameters (in):	const std::string& key	Identifier of the file.
Parameters (inout):	None	

Parameters (out):	None	
Exceptions:	None	
Return value:	bool	Returns true if the file exists, false otherwise.
Description:	Query if a file with the given identifier is available in the proxy.	

Table 7.18: FileProxyAccessorFactory HasKey

7.2.2.1.4 FileProxyAccessorFactory::CreateRWAccess

Service name:	**[SWS_PER_00113]** ⌈`FileProxyAccessorFactory::CreateRWAccess`⌋ (*RS_PER_00001*, *RS_PER_00004*)	
Syntax:	`std::unique_ptr<ReadWriteAccessor> CreateRWAccess (std::string const& key, BasicOperations::OpenMode const mode = BasicOperations::OpenMode::kOut | BasicOperations::OpenMode::kIn)`	
Parameters (in):	std::string const& key	Identifier of the file.
Parameters (in):	BasicOperations:: OpenMode const mode	Mode with which the file shall be opened.
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	std::unique_ptr <ReadWrite Accessor>	ReadWriteAccessor associated to the file.
Description:	The function creates an accessor for reading and writing. For keys that are not yet defined in the context of the File-Proxy the same error handling (setting of failbit) shall apply as defined in the C++ specification for std::fstream. If an accessor with a new key would be created the method shall check the attribute `PersistencyFileProxyInterface`.`maxNumberOfFiles` and set the failbit for the returned accessor if the defined number is already reached.	

Table 7.19: FileProxyAccessorFactory CreateRWAccess

7.2.2.1.5 FileProxyAccessorFactory::CreateReadAccess

Service name:	**[SWS_PER_00114]** ⌈`FileProxyAccessorFactory::CreateReadAccess`⌋ (*RS_PER_00001*, *RS_PER_00004*)	
Syntax:	`std::unique_ptr<ReadAccessor> CreateReadAccess (std::string const& key, BasicOperations::OpenMode const mode = BasicOperations::OpenMode::kIn)`	
Parameters (in):	std::string const& key	Identifier of the file.
Parameters (in):	BasicOperations:: OpenMode const mode	Mode with which the file shall be opened.
Parameters (inout):	None	
Parameters (out):	None	

Exceptions:	None	
Return value:	std::unique_ptr <ReadAccessor>	ReadAccessor associated to the file.
Description:	The function creates an accessor for reading. For keys that are not yet defined in the context of the File-Proxy the same error handling (setting of failbit) shall apply as defined in the C++ specification for std::fstream.	

Table 7.20: FileProxyAccessorFactory CreateReadAccess

7.2.2.1.6 FileProxyAccessorFactory::CreateWriteAccess

Service name:	**[SWS_PER_00115]** ⌈`FileProxyAccessorFactory::CreateWriteAccess`⌋*(RS_PER_00001, RS_PER_00004)*	
Syntax:	`std::unique_ptr<WriteAccessor> CreateWriteAccess (std::string const& key, BasicOperations::OpenMode const mode = BasicOperations::OpenMode::kOut)`	
Parameters (in):	std::string const& key	Identifier of the file.
Parameters (in):	BasicOperations:: OpenMode const mode	Mode with which the file shall be opened.
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	std::unique_ptr <WriteAccessor>	WriteAccessor associated to the file.
Description:	The function creates an accessor for writing. For keys that are not yet defined in the context of the File-Proxy the same error handling (setting of failbit) shall apply as defined in the C++ specification for std::fstream. If an accessor with a new key would be created the method shall check the attribute `PersistencyFileProxyInterface.maxNumberOfFiles` and set the failbit for the returned accessor if the defined number is already reached.	

Table 7.21: FileProxyAccessorFactory CreateWriteAccess

7.2.2.1.7 CreateFileAccessorFactory

Service name:	**[SWS_PER_00116]** ⌈`CreateFileAccessorFactory`⌋ *(RS_PER_00001, RS_PER_00004)*	
Syntax:	`std::unique_ptr<FileProxyAccessorFactory> CreateFileAccessorFactory(std::string proxy)`	
Parameters (in):	std::string proxy	`shortName` of `PortPrototype` typed by a `PersistencyFileProxyInterface`
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	ara::per:: StorageLocation NotFound	In case the proxy is not modelled.

Return value:	std::unique_ptr <FileProxy AccessorFactory>	Creates an instance of FileProxyAccessorFactory.
Description:	The `FileProxyAccessorFactory` function shall be used as factory to create objects to read and write persistent memory. Please note: This operator is not a member of the FileProxyAccessor-Factory class.	

Table 7.22: CreateFileAccessorFactory

7.2.2.2 BasicOperations class

[SWS_PER_00148] ⌈ The `Persistency` cluster shall provide an abstract class for basic file accessor operations. This class shall be called `BasicOperations` and shall be defined in the `ara/per/basic_operations.h` header. ⌋*(RS_PER_00001, RS_PER_00004)*

7.2.2.2.1 BasicOperations::tell

Service name:	**[SWS_PER_00162]** ⌈ `BasicOperations::tell`⌋*(RS_PER_00001, RS_PER_00004)*	
Syntax:	`pos_type tell()`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	pos_type	Get the current position in bytes in the file from start.
Description:	The function returns the current position in bytes in the file from start.	

Table 7.23: BasicOperations tell

7.2.2.2.2 BasicOperations::seek

Service name:	**[SWS_PER_00163]** ⌈ `BasicOperations::seek`⌋*(RS_PER_00001, RS_PER_00004)*	
Syntax:	`void seek(pos_type const pos)`	
Parameters (in):	pos_type const pos	Sets position in bytes for seek from beginning of file.
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	None	
Description:	The function sets the position in bytes in the file from beginning.	

Table 7.24: BasicOperations seek

Service name:	**[SWS_PER_00164]** ⌈ `BasicOperations::seek`⌋*(RS_PER_00001, RS_PER_00004)*
Syntax:	`void seek(off_type const off, SeekDirection const dir)`

Parameters (in):	off_type const pos	Sets the relative position for seek to start.
Parameters (in):	SeekDirection const dir	Sets the direction for (from Begin, End or Current position).
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	None	
Description:	The function sets the position in bytes in the file in relation to the SeekDirection.	

Table 7.25: BasicOperations seek

7.2.2.2.3 BasicOperations::good

Service name:	**[SWS_PER_00106]** ⌈ `BasicOperations::good`⌋*(RS_PER_00001, RS_PER_00004)*	
Syntax:	`bool good()`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	bool	Returns true if no error occurred.
Description:	Returns true if no error occurred.	

Table 7.26: BasicOperations good

7.2.2.2.4 BasicOperations::eof

Service name:	**[SWS_PER_00107]** ⌈ `BasicOperations::eof`⌋*(RS_PER_00001, RS_PER_00004)*	
Syntax:	`bool eof()`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	bool	Returns true if end of file was reached.
Description:	Returns true if end of file was reached.	

Table 7.27: BasicOperations eof

7.2.2.2.5 BasicOperations::fail

Service name:	**[SWS_PER_00108]** ⌈ `BasicOperations::fail`⌋*(RS_PER_00001, RS_PER_00004)*	
Syntax:	`bool fail()`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	bool	Returns true if an error occurred during operation.

Description:	Returns true if an error occurred during operation.

Table 7.28: BasicOperations fail

7.2.2.2.6 BasicOperations::bad

Service name:	**[SWS_PER_00140]** ⌈ `BasicOperations::bad`⌋*(RS_PER_00001, RS_PER_00004)*	
Syntax:	`bool bad()`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	bool	Returns true if an error occurred during operation with a loss of integrity of the stream.
Description:	Returns true if an error occurred during operation with a loss of integrity of the stream.	

Table 7.29: BasicOperations bad

7.2.2.2.7 BasicOperations::operator!

Service name:	**[SWS_PER_00142]** ⌈ `BasicOperations::operator!`⌋ *(RS_PER_00001, RS_PER_00004)*	
Syntax:	`bool operator!()`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	bool	Returns true if an error occurred during operation.
Description:	Synonym of fail(). Returns true if an error occurred during operation.	

Table 7.30: BasicOperations operator!

7.2.2.2.8 BasicOperations::operator bool

Service name:	**[SWS_PER_00143]** ⌈ `BasicOperations::operator bool`⌋ *(RS_PER_00001, RS_PER_00004)*	
Syntax:	`explicit operator bool()`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	bool	Returns false if an error occurred during operation.
Description:	Synonym of !fail(). Returns false if an error occurred during operation.	

Table 7.31: BasicOperations operator bool

7.2.2.2.9 BasicOperations::clear

Service name:	**[SWS_PER_00141]** ⌈ `BasicOperations::clear`⌋(*RS_PER_00001*, *RS_PER_00004*)	
Syntax:	`void clear()`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	None	
Description:	Clears error flags.	

Table 7.32: BasicOperations clear

7.2.2.2.10 operator| for BasicOperations::OpenMode

Service name:	**[SWS_PER_00144]** ⌈`operator|`⌋(*RS_PER_00001*, *RS_PER_00004*)	
Syntax:	`BasicOperations::OpenMode operator| (BasicOperations::OpenMode const& left, BasicOperations::OpenMode const& right)`	
Parameters (in):	BasicOperations:: OpenMode const& left	Left OpenMode modifiers.
Parameters (in):	BasicOperations:: OpenMode const& right	Right OpenMode modifiers.
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	BasicOperations:: OpenMode	Returns merged OpenMode modifiers.
Description:	Merge all OpenMode modifiers into one OpenMode object. Please note: This operator is not a member of the BasicOperations class.	

Table 7.33: operator| for BasicOperations::OpenMode

7.2.2.2.11 operator& for BasicOperations::OpenMode

Service name:	**[SWS_PER_00145]** ⌈`operator&`⌋(*RS_PER_00001*, *RS_PER_00004*)	
Syntax:	`BasicOperations::OpenMode operator& (BasicOperations::OpenMode const& left, BasicOperations::OpenMode const& right)`	
Parameters (in):	BasicOperations:: OpenMode const& left	Left OpenMode modifiers.
Parameters (in):	BasicOperations:: OpenMode const& right	Right OpenMode modifiers.
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	BasicOperations:: OpenMode	Returns intersection of OpenMode modifiers.

Description:	Intersect all OpenMode modifiers into one OpenMode object. Please note: This operator is not a member of the BasicOperations class.

Table 7.34: operator& for BasicOperations::OpenMode

7.2.2.3 ReadAccessor class

[SWS_PER_00169] ⌈ The `Persistency` cluster shall provide an abstract class for read accessor operations. This class shall be called `ReadAccessor` and shall be defined in the `ara/per/read_accessor.h` header.

⌋*(RS_PER_00001, RS_PER_00004)*

[SWS_PER_00131] ⌈ The `ReadAccessor` class shall inherit from class ara::per::BasicOperations. ⌋*(RS_PER_00003)*

7.2.2.3.1 ReadAccessor::peek

Service name:	**[SWS_PER_00167]** ⌈ `ReadAccessor::peek`⌋*(RS_PER_00001, RS_PER_00004)*	
Syntax:	`int_type peek()`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	int_type	Returns the first char.
Description:	The function peeks the first char from the buffer.	

Table 7.35: ReadAccessor peek

7.2.2.3.2 ReadAccessor::get

Service name:	**[SWS_PER_00168]** ⌈ `ReadAccessor::get`⌋*(RS_PER_00001, RS_PER_00004)*	
Syntax:	`int_type get()`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	int_type	Returns the first char.
Description:	The function gets the first char from the buffer.	

Table 7.36: ReadAccessor get

7.2.2.3.3 ReadAccessor::read

Service name:	**[SWS_PER_00165]** ⌈ `ReadAccessor::read`⌋*(RS_PER_00001, RS_PER_00004)*

Syntax:	`pos_type read(char* const s, std::size_t const n)`	
Parameters (in):	char* const s	Destination pointer for read bytes.
Parameters (in):	std::size_t const n	Number of bytes that should be read.
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	pos_type	Returns the actual number of read bytes.
Description:	The function reads n bytes into char pointer and returns actual number that were read.	

Table 7.37: ReadAccessor read

7.2.2.3.4 ReadAccessor::getline

Service name:	**[SWS_PER_00119]** ⌈ `ReadAccessor::getline`⌋*(RS_PER_00001, RS_PER_00004)*	
Syntax:	`ReadAccessor& getline(std::string& string, char const delim)`	
Parameters (in):	char const delim	Char that is used as delimiter.
Parameters (inout):	None	
Parameters (out):	std::string& string	Destination for next string that will be read.
Exceptions:	None	
Return value:	ReadAccessor&	Returns the ReadAccessor object.
Description:	The function reads the next string and stores it in string.	

Table 7.38: ReadAccessor getline

7.2.2.3.5 ReadAccessor::operator»

Service name:	**[SWS_PER_00160]** ⌈ `ReadAccessor::operator»`⌋ *(RS_PER_00001, RS_PER_00004)*	
Syntax:	`ReadAccessor& operator»(std::string& string)`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	std::string& string	Destination for next string that will be read.
Exceptions:	None	
Return value:	ReadAccessor&	Returns the ReadAccessor object.
Description:	The operator reads the next string and stores it in string.	

Table 7.39: ReadAccessor operator»

7.2.2.3.6 getline

Service name:	**[SWS_PER_00161]** ⌈`getline`⌋*(RS_PER_00001, RS_PER_00004)*	
Syntax:	`ReadAccessor& getline(ReadAccessor& ifstream, std::string& string, char const delim)`	
Parameters (in):	char const delim	Char that is used as delimiter.
Parameters (inout):	ReadAccessor& ifstream	ReadAccessor to be read from.
Parameters (out):	std::string& string	Destination for next string that will be read.

Exceptions:	None	
Return value:	ReadAccessor&	Returns the ReadAccessor object.
Description:	The operator reads the next string from ifstream and stores it in string. Please note: This function is not a member of the ReadAccessor class.	

Table 7.40: getline

7.2.2.4 ReadWriteAccessor class

[SWS_PER_00170] ⌈ The `Persistency` cluster shall provide an abstract class for read/write accessor operations. This class shall be called `ReadWriteAccessor` and shall be defined in the `ara/per/read_write_accessor.h` header.

[SWS_PER_00129] ⌈ The `ReadWriteAccessor` class shall inherit from class ara::per::ReadAccessor and ara::per::WriteAccessor. ⌋*(RS_PER_00003)*

⌋*(RS_PER_00003)*

7.2.2.5 WriteAccessor class

[SWS_PER_00171] ⌈ The `Persistency` cluster shall provide an abstract class for write accessor operations. This class shall be called `WriteAccessor` and shall be defined in the `ara/per/write_accessor.h` header. ⌋*(RS_PER_00001, RS_PER_00004)*

[SWS_PER_00130] ⌈ The `WriteAccessor` class shall inherit from class ara::per::BasicOperations. ⌋*(RS_PER_00003)*

7.2.2.5.1 WriteAccessor::fsync

Service name:	**[SWS_PER_00122]** ⌈ `WriteAccessor::fsync`⌋*(RS_PER_00001, RS_PER_00004)*
Syntax:	`void fsync()`
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Exceptions:	None
Return value:	None
Description:	The function flushes and force the OS to write its data to persistent storage.

Table 7.41: WriteAccessor fsync

7.2.2.5.2 WriteAccessor::write

Service name:	**[SWS_PER_00166]** ⌈ `WriteAccessor::write`⌋*(RS_PER_00001, RS_PER_00004)*

Syntax:	`pos_type write(char const* const s, std::size_t const n)`	
Parameters (in):	char const* const s	Source of bytes to write.
Parameters (in):	std::size_t const n	Number of bytes to write.
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	pos_type	Number of actual written bytes.
Description:	The function writes n bytes from s and returns actual number of written bytes.	

Table 7.42: WriteAccessor write

7.2.2.5.3 WriteAccessor::flush

Service name:	**[SWS_PER_00124]** ⌈ `WriteAccessor::flush`⌋*(RS_PER_00001, RS_PER_00004)*	
Syntax:	`void flush()`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	None	
Description:	The function passes write buffer to OS.	

Table 7.43: WriteAccessor flush

7.2.2.5.4 WriteAccessor::operator«

Service name:	**[SWS_PER_00125]** ⌈ `WriteAccessor::operator«`⌋ *(RS_PER_00001, RS_PER_00004)*	
Syntax:	`WriteAccessor& operator«(std::string const& string)`	
Parameters (in):	std::string const& string	The string to be put into the WriteAccessor.
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	WriteAccessor&	Returns the WriteAccessor object.
Description:	The operator writes string to the WriteAccessor.	

Table 7.44: WriteAccessor operator«

Service name:	**[SWS_PER_00126]** ⌈ `WriteAccessor::operator«`⌋ *(RS_PER_00001, RS_PER_00004)*	
Syntax:	`WriteAccessor& operator«(WriteAccessor& (*op)(WriteAccessor&))`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	WriteAccessor&	Returns the WriteAccessor object.

Description:	Operator overload for endl and flush (see below).

Table 7.45: WriteAccessor operator«

7.2.2.5.5 endl

Service name:	**[SWS_PER_00127]** ⌈ `endl`⌋*(RS_PER_00001, RS_PER_00004)*	
Syntax:	`WriteAccessor& endl(WriteAccessor&)`	
Parameters (in):	None	
Parameters (inout):	WriteAccessor&	WriteAccessor object to be modified.
Parameters (out):	None	
Exceptions:	None	
Return value:	WriteAccessor&	Returns the WriteAccessor object.
Description:	The function writes newline to the file and calls flush(). Please note: This function is not a member of the WriteAccessor class.	

Table 7.46: endl

7.2.2.5.6 flush

Service name:	**[SWS_PER_00128]** ⌈ `flush`⌋*(RS_PER_00001, RS_PER_00004)*	
Syntax:	`WriteAccessor& flush(WriteAccessor&)`	
Parameters (in):	None	
Parameters (inout):	WriteAccessor&	WriteAccessor object to be modified.
Parameters (out):	None	
Exceptions:	None	
Return value:	WriteAccessor&	Returns the WriteAccessor object.
Description:	The functions flush calls flush(). Please note: This function is not a member of the WriteAccessor class.	

Table 7.47: flush

7.2.3 Exceptions

[SWS_PER_00172] ⌈ The `Persistency` cluster shall provide exception classes as defined in this chapter. These classes shall be defined in the `ara/per/per_exceptions.h` header. ⌋*(RS_AP_00116, RS_AP_00118)*

[SWS_PER_00500] Exceptions mapping table ⌈

Exceptions	`Logic Error Exception`	`Storage Location NotFound Exception`	`Physical Storage Error Exception`
KeyValueStorage::KeyValueStorage	no	yes	yes
KeyValueStorage::~KeyValueStorage	no	no	no
KeyValueStorage::GetAllKeys	yes	no	yes
KeyValueStorage::HasKey	no	no	no
KeyValueStorage::GetValue	yes	no	no
KeyValueStorage::SetValue	yes	no	yes
KeyValueStorage::RemoveKey	no	no	no
KeyValueStorage::RemoveAllKeys	no	no	no
KeyValueStorage::SyncToStorage	yes	no	yes

⌋*(RS_PER_00002)*

7.2.3.1 LogicErrorException class

To check which API can throw the `LogicErrorException` exception check the `Exceptions mapping table` chapter (subsection 7.2.3).

7.2.3.1.1 LogicErrorException Constructor

Service name:	**[SWS_PER_00173]** ⌈`LogicErrorException Constructor`⌋ *(RS_AP_00118)*	
Syntax:	`LogicErrorException (const std::string& message);`	
Parameters (in):	const std::string&	Error message.
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	None	
Description:	The `LogicErrorException` class shall provide an exception which can be thrown by functions defined within the `ara::per` package. This exception is raised when logical errors occur during runtime.	

Table 7.48: LogicErrorException Constructor

7.2.3.1.2 LogicErrorException::what

Service name:	**[SWS_PER_00132]** ⌈`LogicErrorException::what`⌋*(RS_AP_00118)*	
Syntax:	`const char* what();`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	`const char*`	Reason of exception.
Description:	This method gives access to the error message contained in the exception.	

Table 7.49: LogicErrorException::what

7.2.3.2 StorageLocationNotFoundException class

To check which API can throw the `StorageLocationNotFoundException` exception check the `Exceptions mapping table` chapter (subsection 7.2.3).

7.2.3.2.1 StorageLocationNotFoundException Constructor

Service name:	**[SWS_PER_00174]** ⌈`StorageLocationNotFoundException Constructor`⌋*(RS_AP_00118)*	
Syntax:	`StorageLocationNotFoundException (const std::string& location);`	
Parameters (in):	const std::string&	Error message.
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	None	
Description:	The `StorageLocationNotFoundException` class shall provide an exception which will be thrown if the requested storage location is not found or not configured in the AUTOSAR model.	

Table 7.50: StorageLocationNotFoundException Constructor

7.2.3.2.2 StorageLocationNotFoundException::what

Service name:	**[SWS_PER_00133]** ⌈`StorageLocationNotFoundException::what`⌋*(RS_AP_00118)*	
Syntax:	`const char* what();`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	`const char*`	Reason of exception.
Description:	This method gives access to the error message contained in the exception.	

Table 7.51: StorageLocationNotFoundException::what

7.2.3.3 PhysicalStorageErrorException class

To check which API can throw the `PhysicalStorageErrorException` exception check the `Exceptions mapping table` chapter (subsection 7.2.3).

7.2.3.3.1 PhysicalStorageErrorException Constructor

Service name:	**[SWS_PER_00175]** ⌈`PhysicalStorageErrorException Constructor`⌋*(RS_AP_00118)*	
Syntax:	`PhysicalStorageErrorException (const std::string& message);`	
Parameters (in):	const std::string&	Error message.
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	None	
Description:	The `PhysicalStorageErrorException` class shall provide an exception which is thrown if a severe error which might happen during the operation, such as out of memory or writing/reading to the storage return an error.	

Table 7.52: PhysicalStorageErrorException Constructor

7.2.3.3.2 PhysicalStorageErrorException::what

Service name:	**[SWS_PER_00134]** ⌈`PhysicalStorageErrorException::what`⌋*(RS_AP_00118)*	
Syntax:	`const char* what();`	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Exceptions:	None	
Return value:	`const char*`	Reason of exception.
Description:	This method gives access to the error message contained in the exception.	

Table 7.53: PhysicalStorageErrorException::what

A Not applicable requirements

[SWS_PER_UNUSED] ⌈ These requirements are not applicable to this specification. ⌋*(RS_AP_00111, RS_AP_00113, RS_AP_00114, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00122, RS_AP_00124, RS_AP_00125, RS_UCM_00004,* *RS_UCM_00005*, *RS_UCM_00006*, *RS_UCM_00007*, *RS_UCM_00008*, *RS_UCM_00010*, *RS_UCM_00011*, *RS_UCM_00012*, *RS_UCM_00013*, *RS_UCM_00014*, *RS_UCM_00015*, *RS_UCM_00016*, *RS_UCM_00017*, *RS_UCM_00018*, *RS_UCM_00019*, *RS_UCM_00020*, *RS_UCM_00021*, *RS_UCM_00022*, *RS_UCM_00023*, *RS_UCM_00024*, *RS_UCM_00025*)

B 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	**AdaptiveApplicationSwComponentType**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This meta-class represents the ability to support the formal modeling of application software on the AUTOSAR adaptive platform. Consequently, it shall only be used on the AUTOSAR adaptive platform. **Tags:** atp.Status=draft; atp.recommendedPackage=AdaptiveApplicationSw ComponentTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
internalBeh avior	AdaptiveSwcIn ternalBehavior	0..1	aggr	This aggregation represents the internal behavior of the AdaptiveApplicationSwComponentType for the AUTOSAR adaptive platform. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalBehavior, variation Point.shortLabel; atp.Status=draft vh.latestBindingTime=preCompileTime

Table B.1: AdaptiveApplicationSwComponentType

Class	***AutosarDataType* (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	Abstract base class for user defined AUTOSAR data types for ECU software.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractImplementationDataType*, *ApplicationDataType*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
swDataDef Props	SwDataDefPro ps	0..1	aggr	The properties of this AutosarDataType.

Table B.2: AutosarDataType

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

Table B.3: CryptoNeed

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

Table B.4: CryptoNeedToCryptoJobMapping

Class	**ImplementationDataType**			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. **Tags:** atp.recommendedPackage=ImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dynamicArraySizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.

Attribute	***Type***	***Mul.***	***Kind***	***Note***
subElement (ordered)	ImplementationDataTypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a ImplementationDataType representing a structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the ImplementationDataType. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName
typeEmitter	NameToken	0..1	attr	This attribute is used to control which part of the AUTOSAR toolchain is supposed to trigger data type definitions.

Table B.5: ImplementationDataType

Enumeration	**PersistencyCollectionLevelUpdateStrategyEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface
Note	This enumeration provides possible values for the update strategy on interface/database level. **Tags:** atp.Status=draft
Literal	***Description***
delete	The update strategy is to delete all values on the level of the respective collection. **Tags:** atp.EnumerationValue=1
keepExisting	The update strategy is to keep the existing values on the level of the respective collection. **Tags:** atp.EnumerationValue=0

Table B.6: PersistencyCollectionLevelUpdateStrategyEnum

Class	**PersistencyDataElement**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class represents the ability to formally specify a piece of data that is subject to persistency in the context of the enclosing PersistencyKeyValueDatabaseInterface. PersistencyDataElement represents also a key of the deployed PersistencyKeyValueDatabase and provides an initial value. **Tags:** atp.Status=draft			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
updateStrategy	PersistencyElementLevelUpdateStrategyEnum	1	attr	This attribute shall be used to specify the update strategy of the respective PersistencyDataElement.

Table B.7: PersistencyDataElement

Class	**PersistencyDataProvidedComSpec**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec			
Note	This meta-class represents the ability to define port-specific attributes for supporting use cases of data persistency on the provided side. **Tags:** atp.Status=draft			
Base	*ARObject*, *PPortComSpec*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataElement	PersistencyDataElement	1	ref	This refrence represents the PersistencyDataElement for which the PersistencyDataProvidedComSpec applies. **Tags:** atp.Status=draft
initValue	ValueSpecification	0..1	aggr	This aggregation represents the definition of an initial value for the PersistencyDataElement referenced by the enclosing PersistencyDataProvidedComSpec **Tags:** atp.Status=draft
redundancy	PersistencyRedundancyEnum	0..1	attr	This attribute represents a requirement towards the redundancy of storage.

Table B.8: PersistencyDataProvidedComSpec

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

Table B.9: PersistencyDataRequiredComSpec

Enumeration	**PersistencyElementLevelUpdateStrategyEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface
Note	This enumeration provides possible values for the update strategy on element level. **Tags:** atp.Status=draft
Literal	***Description***
delete	The update strategy is to delete the value of the respective data item. **Tags:** atp.EnumerationValue=2
keepExisting	The update strategy is to keep the existing value of the respective data item. **Tags:** atp.EnumerationValue=1
overwrite	The update strategy is to overwrite the respective data item. **Tags:** atp.EnumerationValue=0

Table B.10: PersistencyElementLevelUpdateStrategyEnum

Class	**PersistencyFile**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Persistency			
Note	This meta-class represents the model of a file as part of the persistency on deployment level. **Tags:** atp.ManifestKind=ApplicationManifest; atp.Status=draft; atp.recommended Package=PersistencyFiles			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

Attribute	***Type***	***Mul.***	***Kind***	***Note***
contentUri	UriString	0..1	attr	This attribute represents the URI that identifies the initial content of the PersistencyFile.
fileName	String	1	attr	This attribute holds filename part of the storage location for the PersistencyFile, e.g. file on the file system. **Tags:** atp.Status=draft
updateStrategy	PersistencyElementLevelUpdateStrategyEnum	1	attr	This attribute can be used to specify the update strategy of the respective PersistencyFile.

Table B.11: PersistencyFile

Class	**PersistencyFileArray**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Persistency			
Note	This meta-class comes with the ability to define an array of single files that creates the deployment-side counterpart to a PortPrototype typed by a PersistencyFileProxyInterface. **Tags:** atp.ManifestKind=ApplicationManifest; atp.Status=draft; atp.recommendedPackage=PersistencyFileArrays			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
file	PersistencyFile	*	aggr	This aggregation represents the collection of files aggregated by the PersistencyFileArray. **Tags:** atp.Status=draft
updateStrategy	PersistencyCollectionLevelUpdateStrategyEnum	1	attr	This attribute can be used to specify the update strategy of the respective PersistencyFileArray as a whole.
uri	UriString	1	attr	This attribute holds the storage location for the PersistencyFileArray, e.g. a directory on the file system.

Table B.12: PersistencyFileArray

Class	**PersistencyFileProxy**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class has the ability to represent a file at design time such that it is possible to configure the behavior for accessing the represented file at run-time. **Tags:** atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
contentUri	UriString	1	attr	This attribute represents the URI that identifies the initial content of the PersistencyFile.

Attribute	***Type***	***Mul.***	***Kind***	***Note***
fileName	String	1	attr	This attribute holds filename part of the storage location for the PersistencyFileProxy, e.g. file on the file system.
updateStrategy	PersistencyElementLevelUpdateStrategyEnum	1	attr	This attribute can be used to specify the update strategy of the respective PersistencyFileProxy.

Table B.13: PersistencyFileProxy

Class	**PersistencyFileProxyInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a PortInterface for supporting persistency use cases for files. **Tags:** atp.Status=draft; atp.recommendedPackage=PersistencyFileProxyInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyInterface*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
encoding	BaseTypeEncodingString	0..1	attr	This attribute supports the definition of an encoding of the corresponding physical files. The possible values of this attribute may be partially standardized by AUTOSAR. But it is also possible to extend the set of values in a custom way (provided that the custom values use a notation that ensures the absence of clashes with further extensions of the standardized values, e.g. by using a company-specific prefix).
fileProxy	PersistencyFileProxy	*	aggr	This aggregation represents the collection of PersistencyFileProxys in the context of the enclosing PersistencyFileProxyInterface. **Tags:** atp.Status=draft
maxNumberOfFiles	PositiveInteger	0..1	attr	This attribute represents the definition of an upper bound for the handling of files at run-time in the context of the enclosing PersistencyFileProxyInterface.
updateStrategy	PersistencyCollectionLevelUpdateStrategyEnum	1	attr	This attribute can be used to specify the update strategy of the respective PersistencyFileProxyInterface as a whole.

Table B.14: PersistencyFileProxyInterface

Class	**PersistencyKeyValueDatabase**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Persistency			
Note	This meta-class represents the ability to model a key/value data base on deployment level. **Tags:** atp.ManifestKind=ApplicationManifest; atp.Status=draft; atp.recommended Package=PersistencyKeyValueDatabases			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
keyValuePair	PersistencyKeyValuePair	*	aggr	This aggregation represents the key-value-pairs owned by the enclosing PersistencyKeyValueDatabase **Tags:** atp.Status=draft
updateStrategy	PersistencyCollectionLevelUpdateStrategyEnum	1	attr	This attribute can be used to specify the update strategy of the respective PersistencyKeyValueDatabase as a whole.
uri	UriString	0..1	attr	This attribute holds the storage location for the PersistencyKeyValueDatabase / PersistencyFile, e.g. file on the file system.

Table B.15: PersistencyKeyValueDatabase

Class	**PersistencyKeyValueDatabaseInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a PortInterface for supporting persistency use cases for data. **Tags:** atp.Status=draft; atp.recommendedPackage=PersistencyKeyValueDatabase Interfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyInterface*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
dataElement	PersistencyDataElement	*	aggr	This aggregation represents the collection of PersistencyDataElements in the context of the enclosing PersistencyKeyValueDatabaseInterface. **Tags:** atp.Status=draft
dataTypeForSerialization	ImplementationDataType	*	ref	This reference identifies ImplementationDataTypes that shall be supported for storing in a key-value data base in addition to the types already referenced as PersistencyDataElement. **Tags:** atp.Status=draft
updateStrategy	PersistencyCollectionLevelUpdateStrategyEnum	1	attr	This attribute shall be used to specify the update strategy of the respective PersistencyKeyValueDatabaseInterface as a whole.

Attribute	***Type***	***Mul.***	***Kind***	***Note***

Table B.16: PersistencyKeyValueDatabaseInterface

Class	**PersistencyKeyValuePair**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Persistency			
Note	This meta-class represents the ability to formally model a key-value pair in the context of the deployment of persistency. **Tags:** atp.ManifestKind=ApplicationManifest; atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
initValue	ValueSpecification	1	aggr	This aggregation represents the ability to define an initial value for the value side of the key-value pair. **Tags:** atp.Status=draft
updateStrategy	PersistencyElementLevelUpdateStrategyEnum	1	attr	This attribute can be used to specify the update strategy of the respective PersistencyKeyValuePair.
valueDataType	ImplementationDataType	1	ref	This reference represents the data type applicable for the value of the key-value pair. **Tags:** atp.Status=draft

Table B.17: PersistencyKeyValuePair

Class	**PersistencyPortPrototypeToFileArrayMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Persistency			
Note	This meta-class represents the ability to define a mapping between an array of files on deployment level to a given PortPrototype. **Tags:** atp.ManifestKind=ApplicationManifest; atp.Status=draft; atp.recommended Package=PersistentFileProxyToFileMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
persistencyFileArray	PersistencyFileArray	1	ref	This reference represents the mapped array of files. **Tags:** atp.Status=draft
portPrototype	PortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **Tags:** atp.Status=draft
process	Process	1	ref	This reference represents the process required as context for the mapping. **Tags:** atp.Status=draft

Table B.18: PersistencyPortPrototypeToFileArrayMapping

Class	**PersistencyPortPrototypeToKeyValueDatabaseMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Persistency			
Note	This meta-class represents the ability to define a mapping between a PortPrototype and a key value database used in a persistent storage. **Tags:** atp.ManifestKind=ApplicationManifest; atp.Status=draft; atp.recommended Package=PersistentPortPrototypeToKeyValueDatabaseMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
keyValueSt orage	PersistencyKe yValueDatabas e	1	ref	This reference represents the mapped key-value storage. **Tags:** atp.Status=draft
portPrototyp e	PortPrototype	0..1	iref	This reference represents the affected Persistency PortPrototype **Tags:** atp.Status=draft
process	Process	1	ref	This reference represents the process required for context of the mapping. **Tags:** atp.Status=draft

Table B.19: PersistencyPortPrototypeToKeyValueDatabaseMapping

Enumeration	**PersistencyRedundancyEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec
Note	This meta-class provides a way to specify the behavior of a given persistent data element with respect to redundancy. **Tags:** atp.Status=draft
Literal	*Description*
none	This value represents the requirement that a piece of data to be stored persistently shall not end up in a redundant persistent storage facility. **Tags:** atp.EnumerationValue=1
redundant	This value represents the requirement that a piece of data to be stored persistently shall end up in a redundant persistent storage facility. The nature of the redundant persistent storage is not further qualified and subject to integrator decisions. **Tags:** atp.EnumerationValue=0

Table B.20: PersistencyRedundancyEnum

Class	***PortPrototype* (abstract)**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
clientServerAnnotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/server communication.
delegatedPortAnnotation	DelegatedPortAnnotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstractionServerAnnotation	IoHwAbstractionServerAnnotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePortAnnotation	ModePortAnnotation	*	aggr	Annotations on this mode port.
nvDataPortAnnotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port.
parameterPortAnnotation	ParameterPortAnnotation	*	aggr	Annotations on this parameter port.
portPrototypeProps	PortPrototypeProps	0..1	aggr	This attribute allows for the definition of further qualification of the semantics of a PortPrototype. **Tags:** atp.Status=draft
senderReceiverAnnotation	SenderReceiverAnnotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPortAnnotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table B.21: PortPrototype

Class	**Process**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::Deployment::Process			
Note	This meta-class provides information required to execute the referenced executable. **Tags:** atp.ManifestKind=ApplicationManifest; atp.Status=draft; atp.recommended Package=Processes			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mul.**	**Kind**	**Note**
applicationModeMachine	ModeDeclarationGroupPrototype	0..1	aggr	Set of ApplicationStates (Modes) that are defined for the process. **Tags:** atp.Status=draft

Attribute	***Type***	***Mul.***	***Kind***	***Note***
design	ProcessDesign	0..1	ref	This reference represents the identification of the design-time representation for the Process that owns the reference. **Tags:** atp.Status=draft
executable	Executable	0..1	ref	Reference to executable that is executed in the process. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
logTraceDefaultLogLevel	LogTraceDefaultLogLevelEnum	0..1	attr	This attribute allows to set the initial log reporting level for a logTraceProcessId (ApplicationId).
logTraceFilePath	UriString	0..1	attr	This attribute defines the destination file to which the logging information is passed.
logTraceLogMode	LogTraceLogModeEnum	0..1	attr	This attribute defines the destination of log messages provided by the process.
logTraceProcessDesc	String	0..1	attr	This attribute can be used to describe the logTraceProcessId that is used in the log and trace message in more detail.
logTraceProcessId	String	0..1	attr	This attribute identifies the process in the log and trace message (ApplicationId).
modeDependentStartupConfig	ModeDependentStartupConfig	*	aggr	Applicable startup configurations. **Tags:** atp.Status=draft

Table B.22: Process

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, DiagnosticDebounceAlgorithmProps, *DiagnosticEnvModeElement*, EthernetPriorityRegeneration, EventHandler, ExclusiveAreaNestingOrder, *HwDescriptionEntity*, *ImplementationProps*, LinSlaveConfigIdent, ModeTransition, *MultilanguageReferrable*, PncMappingIdent, *SingleLanguageReferrable*, SocketConnectionBundle, SomeipRequiredEventGroup, TimeSyncServerConfiguration, TpConnectionIdent			
Attribute	***Type***	***Mul.***	***Kind***	***Note***

shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference.
				
				Tags: xml.enforceMinMultiplicity=true;
				xml.sequenceOffset=-100

Attribute	***Type***	***Mul.***	***Kind***	***Note***
shortName Fragment	ShortNameFra gment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:** xml.sequenceOffset=-90

Table B.23: Referrable