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	18-10

Document Change History

Date	Release	Changed by	Description
2018-10-31	18-10	AUTOSAR Release Management	• Introduction of ara::core types and switch to exceptionless API • Rework of redundancy approach • Support for resource limitation • Improvements and harmonization of KeyValueStorage and FileProxy API
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

Disclaimer

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 Applications 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 a Process of an Adaptive Application, with the rights of that Process.

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] Specification of Manifest
AUTOSAR_TPS_ManifestSpecification

[3] Requirements on Persistency
AUTOSAR_RS_Persistency

[4] General Requirements specific to Adaptive Platform
AUTOSAR_RS_General

[5] Requirements on Update and Configuration Management
AUTOSAR_RS_UpdateAndConfigManagement

[6] Specification of Update and Configuration Management
AUTOSAR_SWS_UpdateAndConfigManagement

[7] Specification of Platform Types for Adaptive Platform
AUTOSAR_SWS_AdaptivePlatformTypes

[8] Specification of Core Types for Adaptive Platform
AUTOSAR_SWS_CoreTypes

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. In addition, the concept of a roll-back after an update is not yet supported.
- The configuration of encryption for `Persistency` is not defined in [2].

4.2 Constraints on Configuration

There are several constraints on the `Persistency` configuration that need to be observed by the tooling which creates/processes this part of the `Execution Manifest`. These constraints are defined in [2].

5 Dependencies to other modules

The `Persistency` is (at least partially) compiled as part of an `Executable` of an `Adaptive Application`, and therefore also executed as part of a `Process`, which creates an implicit dependency on the `ExecutionManagement`.

For the implementation of redundancy and security purposes, the `Persistency` accesses services of the `AdaptiveCryptoInterface`.

For the installation, update, and deletion of persisted data, the `Persistency` interacts with the `UpdateAndConfigManagement` (`UCM`).

6 Requirements Tracing

The following table references the features specified in [3], [4], [5] 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_NA]
[RS_AP_00113]	API specification shall comply with selected coding guidelines.	[SWS_PER_NA]
[RS_AP_00114]	C++ interface shall be compatible with C++11.	[SWS_PER_NA]
[RS_AP_00115]	Namespaces.	[SWS_PER_00002]
[RS_AP_00116]	Header file name.	[SWS_PER_NA]

[RS_AP_00119]	Return values / application errors.	[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_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_00125] [SWS_PER_00126] [SWS_PER_00127] [SWS_PER_00128] [SWS_PER_00140] [SWS_PER_00142] [SWS_PER_00143] [SWS_PER_00144] [SWS_PER_00145] [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_00313] [SWS_PER_00314] [SWS_PER_00315] [SWS_PER_00323] [SWS_PER_00325] [SWS_PER_00327] [SWS_PER_00329] [SWS_PER_00332] [SWS_PER_00333] [SWS_PER_00334] [SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00337] [SWS_PER_00338] [SWS_PER_00345] [SWS_PER_00347]

[RS_AP_00120]	Method and Function names.	[SWS_PER_00042] [SWS_PER_00043] [SWS_PER_00044] [SWS_PER_00046] [SWS_PER_00047] [SWS_PER_00048] [SWS_PER_00049] [SWS_PER_00050] [SWS_PER_00052] [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_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_00313] [SWS_PER_00314] [SWS_PER_00315] [SWS_PER_00322] [SWS_PER_00323] [SWS_PER_00324] [SWS_PER_00325]

		[SWS_PER_00326] [SWS_PER_00327] [SWS_PER_00328] [SWS_PER_00329] [SWS_PER_00330] [SWS_PER_00332] [SWS_PER_00333] [SWS_PER_00334] [SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00337] [SWS_PER_00338] [SWS_PER_00344] [SWS_PER_00345] [SWS_PER_00346] [SWS_PER_00347] [SWS_PER_00348]
[RS_AP_00121]	Parameter names.	[SWS_PER_00043] [SWS_PER_00044] [SWS_PER_00046] [SWS_PER_00047] [SWS_PER_00052] [SWS_PER_00111] [SWS_PER_00112] [SWS_PER_00113] [SWS_PER_00114] [SWS_PER_00115] [SWS_PER_00116] [SWS_PER_00119] [SWS_PER_00125] [SWS_PER_00126] [SWS_PER_00127] [SWS_PER_00128] [SWS_PER_00144] [SWS_PER_00145] [SWS_PER_00160] [SWS_PER_00161] [SWS_PER_00163] [SWS_PER_00164] [SWS_PER_00165] [SWS_PER_00166] [SWS_PER_00315] [SWS_PER_00322] [SWS_PER_00323] [SWS_PER_00326] [SWS_PER_00327] [SWS_PER_00332] [SWS_PER_00333] [SWS_PER_00334] [SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00337] [SWS_PER_00338] [SWS_PER_00344] [SWS_PER_00345]

[RS_AP_00122]	Type names.	[SWS_PER_00146] [SWS_PER_00147] [SWS_PER_00180] [SWS_PER_00181] [SWS_PER_00182] [SWS_PER_00311] [SWS_PER_00312] [SWS_PER_00339] [SWS_PER_00340] [SWS_PER_00341] [SWS_PER_00342] [SWS_PER_00343]
[RS_AP_00124]	Variable names.	[SWS_PER_NA]
[RS_AP_00127]	Usage of ara::core types.	[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_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_00125] [SWS_PER_00160] [SWS_PER_00161] [SWS_PER_00165] [SWS_PER_00166] [SWS_PER_00311] [SWS_PER_00312] [SWS_PER_00332] [SWS_PER_00333] [SWS_PER_00334] [SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00337] [SWS_PER_00338]

[RS_AP_00128]	Use of exceptions in API.	[SWS_PER_00044] [SWS_PER_00046] [SWS_PER_00047] [SWS_PER_00048] [SWS_PER_00049] [SWS_PER_00052] [SWS_PER_00111] [SWS_PER_00113] [SWS_PER_00114] [SWS_PER_00115] [SWS_PER_00116] [SWS_PER_00122] [SWS_PER_00311] [SWS_PER_00312] [SWS_PER_00313] [SWS_PER_00314] [SWS_PER_00315] [SWS_PER_00316] [SWS_PER_00332] [SWS_PER_00333] [SWS_PER_00334] [SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00337] [SWS_PER_00338]
[RS_AP_00129]	Public types defined by functional clusters shall be designed to allow implementation without dynamic memory allocation.	[SWS_PER_00042] [SWS_PER_00044] [SWS_PER_00046] [SWS_PER_00047] [SWS_PER_00048] [SWS_PER_00049] [SWS_PER_00050] [SWS_PER_00052] [SWS_PER_00110] [SWS_PER_00111] [SWS_PER_00113] [SWS_PER_00114] [SWS_PER_00115] [SWS_PER_00116] [SWS_PER_00119] [SWS_PER_00122] [SWS_PER_00160] [SWS_PER_00161] [SWS_PER_00322] [SWS_PER_00326] [SWS_PER_00330] [SWS_PER_00332] [SWS_PER_00333] [SWS_PER_00334] [SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00337] [SWS_PER_00338] [SWS_PER_00344] [SWS_PER_00348]

[RS_AP_00130]	AUTOSAR Adaptive Platform shall represent a rich and modern programming environment.	[SWS_PER_NA]
[RS_AP_00131]	Use of verbal forms to express requirement levels.	[SWS_PER_NA]
[RS_AP_00132]	Usage of noexcept keyword.	[SWS_PER_00042] [SWS_PER_00043] [SWS_PER_00044] [SWS_PER_00046] [SWS_PER_00047] [SWS_PER_00048] [SWS_PER_00049] [SWS_PER_00050] [SWS_PER_00052] [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_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_00313] [SWS_PER_00314] [SWS_PER_00315] [SWS_PER_00322] [SWS_PER_00323] [SWS_PER_00326] [SWS_PER_00327] [SWS_PER_00330] [SWS_PER_00332]

		[SWS_PER_00333] [SWS_PER_00334] [SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00337] [SWS_PER_00338] [SWS_PER_00344] [SWS_PER_00345] [SWS_PER_00348]
[RS_AP_00134]	Library destructors shall be tagged with noexcept.	[SWS_PER_00050] [SWS_PER_00330] [SWS_PER_00348]
[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_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_00302] [SWS_PER_00303] [SWS_PER_00304] [SWS_PER_00309] [SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00337] [SWS_PER_00338]

[RS_PER_00002]	Persistency shall support to retrieve data that has been persistently stored on a platform instance	[SWS_PER_00049] [SWS_PER_00050] [SWS_PER_00322] [SWS_PER_00323] [SWS_PER_00324] [SWS_PER_00325] [SWS_PER_00339] [SWS_PER_00344] [SWS_PER_00345] [SWS_PER_00346] [SWS_PER_00347] [SWS_PER_00348]
[RS_PER_00003]	Persistency shall support identification of data using a unique identifier	[SWS_PER_00042] [SWS_PER_00043] [SWS_PER_00044] [SWS_PER_00046] [SWS_PER_00047] [SWS_PER_00048] [SWS_PER_00052] [SWS_PER_00146] [SWS_PER_00147] [SWS_PER_00180] [SWS_PER_00181] [SWS_PER_00182] [SWS_PER_00331] [SWS_PER_00332] [SWS_PER_00333] [SWS_PER_00334] [SWS_PER_00341]
[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_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_00326] [SWS_PER_00327] [SWS_PER_00328] [SWS_PER_00329] [SWS_PER_00330] [SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00337] [SWS_PER_00338] [SWS_PER_00340] [SWS_PER_00342] [SWS_PER_00343]
[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_00221] [SWS_PER_00317] [SWS_PER_00318] [SWS_PER_00319]
[RS_PER_00009]	Persistency shall support data recovery mechanisms if persistent data was corrupted	[SWS_PER_00222] [SWS_PER_00317] [SWS_PER_00318] [SWS_PER_00319]
[RS_PER_00010]	The layout of persistent data shall be configurable	[SWS_PER_00044] [SWS_PER_00046] [SWS_PER_00047] [SWS_PER_00048] [SWS_PER_00052] [SWS_PER_00113] [SWS_PER_00114] [SWS_PER_00115] [SWS_PER_00116] [SWS_PER_00210] [SWS_PER_00211] [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] [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] [SWS_PER_00304] [SWS_PER_00317] [SWS_PER_00318] [SWS_PER_00319] [SWS_PER_00320] [SWS_PER_00321] [SWS_PER_00332] [SWS_PER_00333] [SWS_PER_00334] [SWS_PER_00335] [SWS_PER_00336]
[RS_PER_00011]	Persistency shall be able to ensure and limit the amount of storage used by persisted data	[SWS_PER_00320] [SWS_PER_00321]

[RS_PER_00012]	Persistency shall support installation of persistent data	[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_PER_00013]	Persistency shall support update of persistent data	[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_PER_00014]	Persistency shall support roll-back of persistent data	[SWS_PER_NA]
[RS_PER_00015]	Persistency shall support removal of persistent data	[SWS_PER_00300] [SWS_PER_00301]
[RS_PER_00016]	Persistency shall support finalization of an update of persistent data	[SWS_PER_NA]

7 Functional specification

7.1 Architecture

The typical usage of the `Persistency` within an `Adaptive Application` is depicted in Figure 7.1. As shown there, an `Adaptive Application` can use a combination of multiple key-value databases and multiple file proxies.

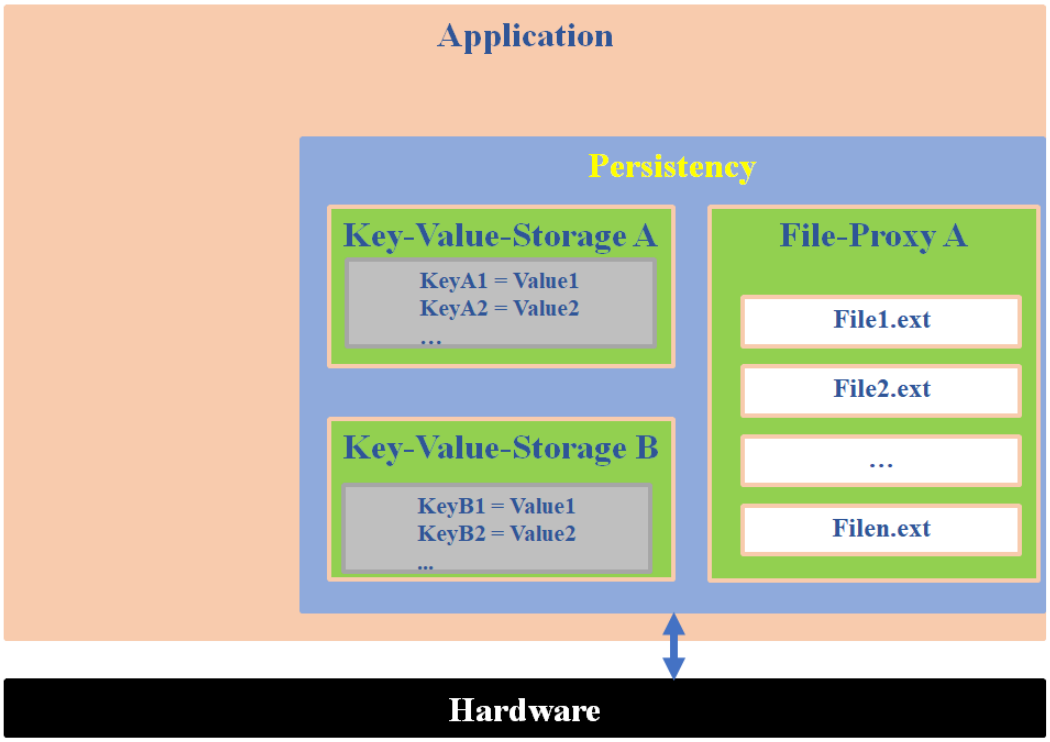

Figure 7.1: Typical usage of Persistency within an Adaptive Application

The functional cluster `Persistency` offers two different mechanisms to access persistent memory as shown in Figure 7.1.

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 the 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 [6]) during installation or update using the deployment information and data provided by the installation package of the `Adaptive Application`. See section 7.4.

File-Proxies offer access to a set of files, they are similar to a directory of a file system. Every File-Proxy is represented by a `PortPrototype` typed by a `PersistencyFileProxyInterface` in the application design for the respective `AdaptiveApplicationSwComponentType`. Every File-Proxy can hold multiple files as described in [2]. Similar to the Key-Value Pairs mentioned above, additional files can be created by the `Adaptive Application` using the `Persistency` API (see 8.2.7.9 and 8.2.7.11).

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 7.4.

The API specification holds classes for Key-Value-Storage and File-Proxy access, taking the shortName of PortPrototype typed by a PersistencyKeyValueDatabaseInterface or a PersistencyFileProxyInterface as an ara::core::StringView input parameter (see 8.1.1 and 8.2.1). Depending on the nature of the PortPrototype, the Key-Value-Storage or File-Proxy can be only read (when the PortPrototype is instantiated as RPortPrototype) or read and written (when the PortPrototype is instantiated as PRPortPrototype) or only be written (when the PortPrototype is instantiated as PPortPrototype).

The Persistency shall not provide an additional communication path for applications besides the mechanisms provided by the functional cluster Communication Management (e.g. using ara::com). Therefore, persistent data shall never be shared between two (or more) processes.

[SWS_PER_00309] ⌈ Persistent data shall always be local to one Process. ⌋ *(RS_PER_00001)*

If persistent data needs to be accessed by multiple processes (of the same or different applications), it is the duty of the application designer to provide Service Interfaces for communication.

7.2 Security concepts

Security requirements of the Key-Value-Storage and File-Proxy are currently not modeled in [2].

[SWS_PER_00210] ⌈ The Persistency cluster shall encrypt data before storing it to the persistent memory. ⌋*(RS_PER_00005, RS_PER_00010)*

[SWS_PER_00211] ⌈ The Persistency cluster shall decrypt data after reading it from persistent memory. ⌋*(RS_PER_00005, RS_PER_00010)*

7.3 Redundancy concepts

The Persistency functional cluster shall take care of the integrity of the stored data. The measures taken to ensure integrity are configurable. The application designer can use PersistencyInterface.redundancy to request redundancy. During deployment, the integrator can define the actual measures taken to ensure integrity using PersistencyDeployment.redundancyHandling.

[SWS_PER_00317] ⌈ The Persistency cluster shall store redundant information for every Key-Value-Database and every File-Proxy represented by a PortPrototype

typed by a PersistencyInterface where PersistencyInterface.redundancy is set to redundant. ⌋(*RS_PER_00008*, *RS_PER_00009*, *RS_PER_00010*)

[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 recover corrupted data if possible. ⌋(*RS_PER_00009*)

The type of redundancy that is applied by the Persistency functional cluster is defined by the set of PersistencyRedundancyHandling classes aggregated as PersistencyDeployment.redundancyHandling.

[SWS_PER_00318] ⌈ In case a PersistencyRedundancyHandling aggregated as PersistencyDeployment.redundancyHandling is derived as PersistencyRedundancyCrc, the Persistency cluster shall calculate a CRC value with the bit width defined by length when persisting the Key-Value-Database or a file in the File-Proxy, and shall use this CRC to check the Key-Value-Database or the file in the File-Proxy when it is read back. ⌋(*RS_PER_00008*, *RS_PER_00009*, *RS_PER_00010*)

[SWS_PER_00319] ⌈ In case a PersistencyRedundancyHandling aggregated as PersistencyDeployment.redundancyHandling is derived as PersistencyRedundancyMOutOfN, the Persistency cluster shall store N copies when persisting the Key-Value-Database or a file in the File-Proxy, and shall check that at least M of the N copies of the Key-Value-Database or the file in the File-Proxy are identical when it is read back. N is defined by n, and M is defined by m. ⌋(*RS_PER_00008*, *RS_PER_00009*, *RS_PER_00010*)

7.4 Persistent data in Update and Configuration Management

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

- Installation of new software
- Update of already installed software
- Roll-back of updated software (not yet described here)
- 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_00251] ⌈ All specification items on the updateStrategy in this section shall always refer to the final updateStrategy in a given configuration (e.g. PersistencyKeyValueDatabase.updateStrategy overrides PersistencyKeyValueDatabaseInterface.updateStrategy). ⌋(*RS_PER_00012, RS_PER_00013, RS_PER_00010*)

7.4.1 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_PER_00012, RS_PER_00010*)

[SWS_PER_00253] ⌈ The created entry in the Key-Value-Database shall have the shortName of the PersistencyDataElement as key. ⌋(*RS_PER_00012, RS_PER_00010*)

[SWS_PER_00254] ⌈ The created entry in the Key-Value-Database shall be of the datatype defined in PersistencyDataElement. ⌋(*RS_PER_00012, RS_PER_00010*)

[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_PER_00012, RS_PER_00010*)

[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_PER_00012, RS_PER_00010*)

[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 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 exists, no entry in the Key-Value-Database shall be created. ⌋ *(RS_PER_00012, RS_PER_00010)*

[SWS_PER_00258] ⌈ Persistency shall reject any configuration in which incompatible AutosarDataTypes are given in the PersistencyDataElement and the mapped PersistencyKeyValuePair.valueDataType. ⌋*(RS_PER_00012, RS_PER_00010)*

[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 PersistencyDataElement in the PersistencyKeyValueDatabaseInterface exists with the same shortName as the PersistencyKeyValuePair. ⌋*(RS_PER_00012, RS_PER_00010)*

[SWS_PER_00260] ⌈ The created entry in the Key-Value-Database shall have the shortName of the PersistencyKeyValuePair as key. ⌋*(RS_PER_00012, RS_PER_00010)*

[SWS_PER_00261] ⌈ The created entry in the Key-Value-Database shall be of the datatype defined in PersistencyKeyValuePair.valueDataType. ⌋ *(RS_PER_00012, RS_PER_00010)*

[SWS_PER_00262] ⌈ The created entry in the Key-Value-Database shall have the PersistencyKeyValuePair.initValue as value. ⌋*(RS_PER_00012, RS_PER_00010)*

[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_PER_00012, RS_PER_00010)*

7.4.2 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_PER_00012, RS_PER_00010)*

[SWS_PER_00266] ⌈ The created entry in the File-Proxy shall have the PersistencyFileProxy.fileName as key. ⌋*(RS_PER_00012, RS_PER_00010)*

[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 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_PER_00012, RS_PER_00010)*

[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_PER_00012, RS_PER_00010)*

[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_PER_00012, RS_PER_00010)*

[SWS_PER_00270] ⌈ The created entry in the File-Proxy shall have the PersistencyFile.fileName as key. ⌋*(RS_PER_00012, RS_PER_00010)*

[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_PER_00012, RS_PER_00010)*

[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_PER_00012, RS_PER_00010)*

[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_PER_00012, RS_PER_00010)*

7.4.3 Update of Key-Value-Databases

[SWS_PER_00274] ⌈ When updating an `Adaptive Application`, the requirements for installation of Key-Value-Databases shall apply but the final `updateStrategy` also needs to be respected. ⌋*(RS_PER_00013, RS_PER_00010)*

[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_PER_00013, RS_PER_00010)*

[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_PER_00013, RS_PER_00010)*

[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_PER_00013, RS_PER_00010)*

[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_PER_00013, RS_PER_00010)*

[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_PER_00013, RS_PER_00010)*

7.4.4 Update of File-Proxies

[SWS_PER_00280] ⌈ When updating an `Adaptive Application`, the requirements for installation of File-Proxies shall apply but the final `updateStrategy` also needs to be respected. ⌋*(RS_PER_00013, RS_PER_00010)*

[SWS_PER_00281] ⌈ If the final `updateStrategy` (enumeration `PersistencyElementLevelUpdateStrategyEnum`) of a `PersistencyFileProxy` or a `Persis-`

tencyFile is overwrite, the entry to the File-Proxy shall be created and even overwrite an existing entry with the same key. ⌋(*RS_PER_00013, RS_PER_00010*)

[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_PER_00013, RS_PER_00010*)

[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_PER_00013, RS_PER_00010*)

[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_PER_00013, RS_PER_00010*)

[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_PER_00013, RS_PER_00010*)

7.4.5 Uninstallation of Key-Value-Databases

[SWS_PER_00300] ⌈ When uninstalling an Adaptive Application, Persistency shall remove all Key-Value-Databases used by this Adaptive Application from the Adaptive Machine. ⌋(*RS_PER_00015*)

7.4.6 Uninstallation of File-Proxies

[SWS_PER_00301] ⌈ When uninstalling an Adaptive Application, Persistency shall remove all File-Proxies and all files contained in these File-Proxies used by this Adaptive Application from the Adaptive Machine. ⌋(*RS_PER_00015*)

7.5 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 8.1.4.6 and 8.1.4.7. 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 [2]. ⌋*(RS_PER_00001, RS_PER_00010)*

7.6 Resource management concepts

The Persistency cluster supports configuration of both an upper and a lower limit for the resources used by a Key-Value-Database or a File-Proxy.

The lower limit may already be defined by the application developer using PersistencyInterface.minimumSustainedSize.

During deployment, the integrator may update the lower limit using PersistencyDeployment.minimumSustainedSize and add an upper limit using PersistencyDeployment.maximumAllowedSize.

[SWS_PER_00320] ⌈ The Persistency cluster shall ensure that the space configured by PersistencyDeployment.minimumSustainedSize is always available for the Key-Value-Database or File-Proxy. ⌋*(RS_PER_00010, RS_PER_00011)*

One possibility to achieve this would be to initially allocate the minimum size during deployment, and never reduce the size below this value when persistent data is removed. But the implementation of the Persistency cluster is free to chose other appropriate measures.

[SWS_PER_00321] ⌈ The Persistency cluster shall ensure that the space actually allocated by a Key-Value-Database or File-Proxy never surpasses the amount configured by PersistencyDeployment.maximumAllowedSize. ⌋*(RS_PER_00010, RS_PER_00011)*

This could be ensured by supervising all write accesses to persistent data. But again, the implementation of the Persistency cluster is free to chose other appropriate measures.

8 API specification

The API of the `Persistency` cluster was designed with the following paradigms in the mind:

- The API of the key value storage and the file access should be as similar as possible. This leads to the decision that files are called "'keys"' on the top level.
- The API to access files is modeled relatively close to the POSIX API for accessing files. This applies especially to the BasicOperations class.

Still, the APIs for accessing files and databases are completely separate, and therefore divided into separate sections.

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

The `ara::per` API is based heavily on the `ara::core` types defined in [8]. `ara::core::Result` is used wherever possible, and because of this, most methods are defined as `noexcept`.

8.1 Key-Value Storage

This section lists all functions and classes that are required to operate a Key-Value Storage.

The following functions are used to get access to a Key-Value Storage database, to recover as much as possible after it was corrupted, and to reset it to the deployed defaults.

8.1.1 CreateKeyValueStorage

[SWS_PER_00052] ⌈ The function ara::per::CreateKeyValueStorage is defined in Table 8.1. ⌋*(RS_PER_00003, RS_PER_00010, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00132)*

Symbol:	ara::per::CreateKeyValueStorage(ara::core::StringView database)	
Kind:	function	
Scope:	namespace ara::per	
Syntax:	`ara::core::Result<std::unique_ptr<KeyValueStorage> > CreateKeyValueStorage (ara::core::StringView database) noexcept;`	
Parameters (in):	database	The shortName of a PortPrototype typed by a PersistencyKeyValueDatabaseInterface.
Return value:	ara::core::Result< std::unique_ptr< KeyValueStorage > >	A Result, containing an instance of KeyValue Storage, or one of the errors defined for Persistency in PerErrc.

▽

△

Exception Safety:	noexcept
Thread Safety:	no
Header file:	#include "ara/per/key_value_storage.h"
Description:	Creates an instance of KeyValueStorage which configures the storage location.

Table 8.1: function ara::per::CreateKeyValueStorage

8.1.2 RecoverKeyValueStorage

[SWS_PER_00333] ⌈ The function ara::per::RecoverKeyValueStorage is defined in Table 8.2. ⌋(*RS_PER_00003*, *RS_PER_00010*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00128*, *RS_AP_00129*, *RS_AP_00132*)

Symbol:	ara::per::RecoverKeyValueStorage(ara::core::StringView database)	
Kind:	function	
Scope:	namespace ara::per	
Syntax:	`ara::core::Result<void> RecoverKeyValueStorage (ara::core::StringView database) noexcept;`	
Parameters (in):	database	The shortName of a PortPrototype typed by a PersistencyKeyValueDatabaseInterface.
Return value:	ara::core::Result< void >	A Result, being either empty or containing one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Recover an instance of KeyValueStorage. This method allows to recover a database when the redundancy checks fail. It will fail with a k ResourceBusyError when the database is currently open. This method does a best-effort recovery of all keys. After recovery, keys might show outdated or initial value, or might be lost.	

Table 8.2: function ara::per::RecoverKeyValueStorage

8.1.3 ResetKeyValueStorage

[SWS_PER_00334] ⌈ The function ara::per::ResetKeyValueStorage is defined in Table 8.3. ⌋(*RS_PER_00003*, *RS_PER_00010*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00128*, *RS_AP_00129*, *RS_AP_00132*)

Symbol:	ara::per::ResetKeyValueStorage(ara::core::StringView database)	
Kind:	function	
Scope:	namespace ara::per	
Syntax:	`ara::core::Result<void> ResetKeyValueStorage (ara::core::StringView database) noexcept;`	
Parameters (in):	database	The shortName of a PortPrototype typed by a PersistencyKeyValueDatabaseInterface.
Return value:	ara::core::Result< void >	A Result, being either empty or containing one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Reset an instance of KeyValueStorage to the initial state. This method allows to reset a database to the initial state, containing only keys which were deployed from the manifest, with their initial values. It will fail with a kResourceBusyError when the database is currently open.	

Table 8.3: function ara::per::ResetKeyValueStorage

8.1.4 KeyValueStorage class

This section shows the methods available for a `KeyValueStorage` object obtained from a call to 8.1.1.

[SWS_PER_00331] ⌈ Operations that modify a `KeyValueStorage` shall only be executed temporarily, such that following operations are aware of the change. The actual storage shall only be updated when `SyncToStorage` is called. ⌋*(RS_PER_00003)*

Therefore, if the `KeyValueStorage` is just destructed (also implicitly when the `Process` terminates), the storage is not updated, and the next time the `KeyValueStorage` is accessed, the application will see the last saved state.

[SWS_PER_00339] ⌈ The class ara::per::KeyValueStorage is defined in Table 8.4. ⌋ *(RS_PER_00002, RS_AP_00122)*

Kind:	class
Base class:	None
Syntax:	`class KeyValueStorage`
Header file:	#include "ara/per/key_value_storage.h"
Description:	Interface to the common key-value storage functions. .

Table 8.4: class ara::per::KeyValueStorage

8.1.4.1 KeyValueStorage::KeyValueStorage

[SWS_PER_00322] ⌈ The function ara::per::KeyValueStorage::KeyValueStorage is defined in Table 8.5. ⌋*(RS_PER_00002, RS_AP_00120, RS_AP_00121, RS_AP_00129, RS_AP_00132)*

Symbol:	ara::per::KeyValueStorage::KeyValueStorage(KeyValueStorage &&kvs)	
Kind:	function	
Scope:	class ara::per::KeyValueStorage	
Syntax:	`KeyValueStorage (KeyValueStorage &&kvs) noexcept;`	
Parameters (in):	kvs	The KeyValueStorage object to be moved.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Move constructor for KeyValueStorage.	

Table 8.5: function ara::per::KeyValueStorage::KeyValueStorage

[SWS_PER_00324] ⌈ The function ara::per::KeyValueStorage::KeyValueStorage is defined in Table 8.6. ⌋*(RS_PER_00002, RS_AP_00120)*

Symbol:	ara::per::KeyValueStorage::KeyValueStorage(const KeyValueStorage &)
Kind:	function
Scope:	class ara::per::KeyValueStorage
Syntax:	`KeyValueStorage (const KeyValueStorage &)=delete;`
Thread Safety:	no
Header file:	#include "ara/per/key_value_storage.h"
Description:	The copy constructor for KeyValueStorage shall not be used.

Table 8.6: function ara::per::KeyValueStorage::KeyValueStorage

8.1.4.2 KeyValueStorage::operator=

[SWS_PER_00323] ⌈ The function ara::per::KeyValueStorage::operator= is defined in Table 8.7. ⌋*(RS_PER_00002, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00132)*

Symbol:	ara::per::KeyValueStorage::operator=(KeyValueStorage &&kvs)	
Kind:	function	
Scope:	class ara::per::KeyValueStorage	
Syntax:	`KeyValueStorage& operator= (KeyValueStorage &&kvs) noexcept;`	
Parameters (in):	kvs	The KeyValueStorage object to be moved.
Return value:	KeyValueStorage &	The moved KeyValueStorage object.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Move assignment operator for KeyValueStorage.	

Table 8.7: function ara::per::KeyValueStorage::operator=

[SWS_PER_00325] ⌈ The function ara::per::KeyValueStorage::operator= is defined in Table 8.8. ⌋*(RS_PER_00002, RS_AP_00119, RS_AP_00120)*

Symbol:	ara::per::KeyValueStorage::operator=(const KeyValueStorage &)
Kind:	function
Scope:	class ara::per::KeyValueStorage
Syntax:	`KeyValueStorage& operator= (const KeyValueStorage &)=delete;`
Thread Safety:	no
Header file:	#include "ara/per/key_value_storage.h"
Description:	The copy assignment operator for KeyValueStorage shall not be used.

Table 8.8: function ara::per::KeyValueStorage::operator=

8.1.4.3 KeyValueStorage::~KeyValueStorage

[SWS_PER_00050] ⌈ The function ara::per::KeyValueStorage::~KeyValueStorage is defined in Table 8.9. ⌋*(RS_PER_00002, RS_AP_00120, RS_AP_00129, RS_AP_00132, RS_AP_00134)*

Symbol:	ara::per::KeyValueStorage::~KeyValueStorage()
Kind:	function
Scope:	class ara::per::KeyValueStorage
Syntax:	`~KeyValueStorage () noexcept;`
Exception Safety:	noexcept
Thread Safety:	no
Header file:	#include "ara/per/key_value_storage.h"
Description:	Destructor for KeyValueStorage.

Table 8.9: function ara::per::KeyValueStorage::~KeyValueStorage

8.1.4.4 KeyValueStorage::GetAllKeys

[SWS_PER_00042] ⌈ The function ara::per::KeyValueStorage::GetAllKeys is defined in Table 8.10. ⌋*(RS_PER_00003, RS_AP_00119, RS_AP_00120, RS_AP_00127, RS_AP_00129, RS_AP_00132)*

Symbol:	ara::per::KeyValueStorage::GetAllKeys()	
Kind:	function	
Scope:	class ara::per::KeyValueStorage	
Syntax:	`ara::core::Vector<ara::core::String> GetAllKeys () const noexcept;`	
Return value:	ara::core::Vector< ara::core::String >	A list of available keys.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/key_value_storage.h"	

▽

△

Description:	Returns a list of all currently available keys of the KeyValueStorage. A list of available keys.

Table 8.10: function ara::per::KeyValueStorage::GetAllKeys

8.1.4.5 KeyValueStorage::HasKey

[SWS_PER_00043] ⌈ The function ara::per::KeyValueStorage::HasKey is defined in Table 8.11. ⌋(*RS_PER_00003*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00132*)

Symbol:	ara::per::KeyValueStorage::HasKey(ara::core::StringView key)	
Kind:	function	
Scope:	class ara::per::KeyValueStorage	
Syntax:	`bool HasKey (ara::core::StringView key) const noexcept;`	
Parameters (in):	key	The key that shall be checked.
Return value:	bool	True if the key could be located, false otherwise.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Checks if a key exists in the KeyValueStorage.	

Table 8.11: function ara::per::KeyValueStorage::HasKey

8.1.4.6 KeyValueStorage::GetValue

[SWS_PER_00044] ⌈ The function ara::per::KeyValueStorage::GetValue is defined in Table 8.12. ⌋(*RS_PER_00003*, *RS_PER_00010*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00128*, *RS_AP_00129*, *RS_AP_00132*)

Symbol:	ara::per::KeyValueStorage::GetValue(ara::core::StringView key, T &value)	
Kind:	function	
Scope:	class ara::per::KeyValueStorage	
Syntax:	`template <class T>` `ara::core::Result<void> GetValue (ara::core::StringView key, T &value) const noexcept;`	
Template param:	T	The type of the value that shall be retrieved.
Parameters (in):	key	The key to look up.
Parameters (out):	value	The retrieved value.
Return value:	ara::core::Result< void >	A Result, being empty or containing one of the errors defined for Persistency in PerErrc.

▽

△

Exception Safety:	noexcept
Thread Safety:	no
Header file:	#include "ara/per/key_value_storage.h"
Description:	Returns the value assigned to a key of the KeyValueStorage. This method may be useful to avoid superfluous instantiation of complex types.

Table 8.12: function ara::per::KeyValueStorage::GetValue

[SWS_PER_00332] ⌈ The function ara::per::KeyValueStorage::GetValue is defined in Table 8.13. ⌋(*RS_PER_00003*, *RS_PER_00010*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00128*, *RS_AP_00129*, *RS_AP_00132*)

Symbol:	ara::per::KeyValueStorage::GetValue(ara::core::StringView key)	
Kind:	function	
Scope:	class ara::per::KeyValueStorage	
Syntax:	`template <class T>` `ara::core::Result<T> GetValue (ara::core::StringView key) const noexcept;`	
Template param:	T	The type of the value that shall be retrieved.
Parameters (in):	key	The key to look up.
Return value:	ara::core::Result< T >	A Result, being either the retrieved value or containing one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Returns the value assigned to a key of the KeyValueStorage. This method is mainly useful for primitive types.	

Table 8.13: function ara::per::KeyValueStorage::GetValue

8.1.4.7 KeyValueStorage::SetValue

[SWS_PER_00046] ⌈ The function ara::per::KeyValueStorage::SetValue is defined in Table 8.14. ⌋(*RS_PER_00003*, *RS_PER_00010*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00128*, *RS_AP_00129*, *RS_AP_00132*)

Symbol:	ara::per::KeyValueStorage::SetValue(ara::core::StringView key, const T &value)	
Kind:	function	
Scope:	class ara::per::KeyValueStorage	
Syntax:	`template <class T>` `ara::core::Result<void> SetValue (ara::core::StringView key, const T &value) noexcept;`	
Template param:	T	The type of the value that shall be set.

▽

△

Parameters (in):	key	The key to assign the value to.
	value	The value to store.
Return value:	ara::core::Result< void >	A Result, being empty or containing one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Stores a key in the KeyValueStorage.	

Table 8.14: function ara::per::KeyValueStorage::SetValue

8.1.4.8 KeyValueStorage::RemoveKey

[SWS_PER_00047] ⌈ The function ara::per::KeyValueStorage::RemoveKey is defined in Table 8.15. ⌋*(RS_PER_00003, RS_PER_00010, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00132)*

Symbol:	ara::per::KeyValueStorage::RemoveKey(ara::core::StringView key)	
Kind:	function	
Scope:	class ara::per::KeyValueStorage	
Syntax:	`ara::core::Result<void> RemoveKey (ara::core::StringView key) noexcept;`	
Parameters (in):	key	The key to be removed.
Return value:	ara::core::Result< void >	A Result, being empty or containing one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Removes a key and the associated value from the KeyValueStorage.	

Table 8.15: function ara::per::KeyValueStorage::RemoveKey

8.1.4.9 KeyValueStorage::RemoveAllKeys

[SWS_PER_00048] ⌈ The function ara::per::KeyValueStorage::RemoveAllKeys is defined in Table 8.16. ⌋*(RS_PER_00003, RS_PER_00010, RS_AP_00119, RS_AP_00120, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00132)*

Symbol:	ara::per::KeyValueStorage::RemoveAllKeys()	
Kind:	function	
Scope:	class ara::per::KeyValueStorage	
Syntax:	`ara::core::Result<void> RemoveAllKeys () noexcept;`	
Return value:	ara::core::Result< void >	A Result, being empty or containing one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Removes all keys and associated values from the KeyValueStorage. A Result, being empty or containing one of the errors defined for Persistency in PerErrc.	

Table 8.16: function ara::per::KeyValueStorage::RemoveAllKeys

8.1.4.10 KeyValueStorage::SyncToStorage

[SWS_PER_00049] ⌈ The function ara::per::KeyValueStorage::SyncToStorage is defined in Table 8.17. ⌋(*RS_PER_00002*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00127*, *RS_AP_00128*, *RS_AP_00129*, *RS_AP_00132*)

Symbol:	ara::per::KeyValueStorage::SyncToStorage()	
Kind:	function	
Scope:	class ara::per::KeyValueStorage	
Syntax:	`ara::core::Result<void> SyncToStorage () const noexcept;`	
Return value:	ara::core::Result< void >	A Result, being either empty or containing one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Triggers flushing of key-value pairs to the physical storage of the KeyValueStorage. A Result, being either empty or containing one of the errors defined for Persistency in PerErrc.	

Table 8.17: function ara::per::KeyValueStorage::SyncToStorage

8.2 FileProxy

This section lists all functions and classes that are required to operate a File Proxy.

The following functions are used to get access to a File Proxy, to recover as much as possible after it was corrupted, and to reset it to the deployed defaults.

8.2.1 CreateFileAccessorFactory

[SWS_PER_00116] ⌈ The function ara::per::CreateFileAccessorFactory is defined in Table 8.18. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_PER_00010*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00128*, *RS_AP_00129*, *RS_AP_00132*)

Symbol:	ara::per::CreateFileAccessorFactory(ara::core::StringView proxy)	
Kind:	function	
Scope:	namespace ara::per	
Syntax:	`ara::core::Result<std::unique_ptr<FileProxyAccessorFactory> > CreateFileAccessorFactory (ara::core::StringView proxy) noexcept;`	
Parameters (in):	proxy	The shortName of a PortPrototype typed by a PersistencyFileProxyInterface.
Return value:	ara::core::Result< std::unique_ptr< FileProxyAccessorFactory > >	A Result, containing an instance of FileProxyAccessorFactory, or one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/file_proxy_accessor_factory.h"	
Description:	Creates a factory to create objects to read and write persistent memory.	

Table 8.18: function ara::per::CreateFileAccessorFactory

8.2.2 RecoverFileProxy

[SWS_PER_00335] ⌈ The function ara::per::RecoverFileProxy is defined in Table 8.19. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_PER_00010*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00128*, *RS_AP_00129*, *RS_AP_00132*)

Symbol:	ara::per::RecoverFileProxy(ara::core::StringView proxy)
Kind:	function
Scope:	namespace ara::per
Syntax:	`ara::core::Result<void> RecoverFileProxy (ara::core::StringView proxy) noexcept;`

▽

△

Parameters (in):	proxy	The shortName of a PortPrototype typed by a PersistencyFileProxyInterface.
Return value:	ara::core::Result< void >	A Result, being either empty or containing one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/file_proxy_accessor_factory.h"	
Description:	Recover the whole file proxy, including all files. This method allows to recover a file-proxy when the redundancy checks fail. It will fail with a k ResourceBusyError when the file-proxy is currently open. This method does a best-effort recovery of all files. After recovery, files might show outdated or initial content, or might be lost.	

Table 8.19: function ara::per::RecoverFileProxy

8.2.3 ResetFileProxy

[SWS_PER_00336] ⌈ The function ara::per::ResetFileProxy is defined in Table 8.20. ⌋ *(RS_PER_00001, RS_PER_00004, RS_PER_00010, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00132)*

Symbol:	ara::per::ResetFileProxy(ara::core::StringView proxy)	
Kind:	function	
Scope:	namespace ara::per	
Syntax:	`ara::core::Result<void> ResetFileProxy (ara::core::StringView proxy) noexcept;`	
Parameters (in):	proxy	The shortName of a PortPrototype typed by a PersistencyFileProxyInterface.
Return value:	ara::core::Result< void >	A Result, being either empty or containing one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/file_proxy_accessor_factory.h"	
Description:	Reset the whole file proxy, including all files. This method allows to reset a file-proxy to the initial state, containing only the files which were deployed from the manifest, with their initial content. It will fail with a kResourceBusyError when the file-proxy is currently open.	

Table 8.20: function ara::per::ResetFileProxy

8.2.4 Helper functions for BasicOperations class

The following functions can be used by the application when accessing 8.2.7.10, 8.2.7.11, and 8.2.7.9 to combine the values of `BasicOperations::OpenMode`.

8.2.4.1 operator| for BasicOperations::OpenMode

[SWS_PER_00144] ⌈ The function ara::per::operator| is defined in Table 8.21. ⌋ *(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121)*

Symbol:	ara::per::operator|(BasicOperations::OpenMode const &left, BasicOperations::OpenMode const &right)	
Kind:	function	
Scope:	namespace ara::per	
Syntax:	`BasicOperations::OpenMode operator| (BasicOperations::OpenMode const &left, BasicOperations::OpenMode const &right);`	
Parameters (in):	left	First OpenMode modifiers.
	right	Second OpenMode modifiers.
Return value:	BasicOperations::OpenMode	returns Merged OpenMode modifiers.
Thread Safety:	no	
Header file:	#include "ara/per/basic_operations.h"	
Description:	Merges two OpenMode modifiers into one. BasicOperations class.	

Table 8.21: function ara::per::operator|

8.2.4.2 operator& for BasicOperations::OpenMode

[SWS_PER_00145] ⌈ The function ara::per::operator& is defined in Table 8.22. ⌋ *(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121)*

Symbol:	ara::per::operator&(BasicOperations::OpenMode const &left, BasicOperations::OpenMode const &right)	
Kind:	function	
Scope:	namespace ara::per	
Syntax:	`BasicOperations::OpenMode operator& (BasicOperations::OpenMode const &left, BasicOperations::OpenMode const &right);`	
Parameters (in):	left	First OpenMode modifiers.
	right	Second OpenMode modifiers,
Return value:	BasicOperations::OpenMode	returns Intersected OpenMode modifiers.
Thread Safety:	no	
Header file:	#include "ara/per/basic_operations.h"	
Description:	Intersect two OpenMode modifiers into one.	

Table 8.22: function ara::per::operator&

8.2.5 Helper functions for ReadAccessor class

The following functions can be used by the application to work with a `ReadAccessor` object.

8.2.5.1 getline

[SWS_PER_00161] ⌈ The function ara::per::getline is defined in Table 8.23. ⌋ *(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00129, RS_AP_00132)*

Symbol:	ara::per::getline(ReadAccessor &ra, ara::core::String &string, char const delim= '\n')	
Kind:	function	
Scope:	namespace ara::per	
Syntax:	`ReadAccessor& getline (ReadAccessor &ra, ara::core::String &string, char const delim= '\n') noexcept;`	
Parameters (in):	ra	The ReadAccessor object to read from.
	delim	The character that is used as delimiter.
Parameters (out):	string	A string where the line will be stored.
Return value:	ReadAccessor &	The ReadAccessor object.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/read_accessor.h"	
Description:	Reads a complete line from a ReadAccessor into a sting, advancing the current position.	

Table 8.23: function ara::per::getline

8.2.6 Helper functions for WriteAccessor class

The following functions can be used by the application within a `WriteAccessor` stream.

8.2.6.1 endl

[SWS_PER_00127] ⌈ The function ara::per::endl is defined in Table 8.24. ⌋ *(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00132)*

Symbol:	ara::per::endl(ReadWriteAccessor &rwa)	
Kind:	function	
Scope:	namespace ara::per	
Syntax:	`ReadWriteAccessor& endl (ReadWriteAccessor &rwa) noexcept;`	
Parameters (in):	rwa	The ReadWriteAccessor object.
Return value:	ReadWriteAccessor &	The ReadWriteAccessor object.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/read_write_accessor.h"	
Description:	Writes a newline to the file and calls flush().	

Table 8.24: function ara::per::endl

8.2.6.2 flush

[SWS_PER_00128] ⌈ The function ara::per::flush is defined in Table 8.25. ⌋ *(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00132)*

Symbol:	ara::per::flush(ReadWriteAccessor &rwa)	
Kind:	function	
Scope:	namespace ara::per	
Syntax:	`ReadWriteAccessor& flush (ReadWriteAccessor &rwa) noexcept;`	
Parameters (in):	rwa	The ReadWriteAccessor object.
Return value:	ReadWriteAccessor &	The ReadWriteAccessor object.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/read_write_accessor.h"	
Description:	Calls flush() on the file.	

Table 8.25: function ara::per::flush

8.2.7 FileProxyAccessorFactory class

This section shows the methods available for a `FileProxyAccessorFactory` object obtained from a call to 8.2.1.

[SWS_PER_00340] ⌈ The class ara::per::FileProxyAccessorFactory is defined in Table 8.26. ⌋ *(RS_PER_00004, RS_AP_00122)*

Kind:	class
Base class:	None
Syntax:	`class FileProxyAccessorFactory`
Header file:	#include "ara/per/file_proxy_accessor_factory.h"
Description:	The FileProxyAccessorFactory creates objects to read or write memory blocks. Which block is going to be read or written is specified by a key.

Table 8.26: class ara::per::FileProxyAccessorFactory

8.2.7.1 FileProxyAccessorFactory::FileProxyAccessorFactory

[SWS_PER_00326] ⌈ The function ara::per::FileProxyAccessorFactory::FileProxy AccessorFactory is defined in Table 8.27. ⌋ *(RS_PER_00004, RS_AP_00120, RS_AP_00121, RS_AP_00129, RS_AP_00132)*

Symbol:	ara::per::FileProxyAccessorFactory::FileProxyAccessorFactory(FileProxyAccessorFactory &&fp)	
Kind:	function	
Scope:	class ara::per::FileProxyAccessorFactory	
Syntax:	`FileProxyAccessorFactory (FileProxyAccessorFactory &&fp) noexcept;`	
Parameters (in):	fp	The FileProxyAccessorFactory object to be moved.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/file_proxy_accessor_factory.h"	
Description:	Move constructor for FileProxyAccessorFactory.	

Table 8.27: function ara::per::FileProxyAccessorFactory::FileProxyAccessorFactory

[SWS_PER_00328] ⌈ The function ara::per::FileProxyAccessorFactory::FileProxyAccessorFactory is defined in Table 8.28. ⌋*(RS_PER_00004, RS_AP_00120)*

Symbol:	ara::per::FileProxyAccessorFactory::FileProxyAccessorFactory(const FileProxyAccessor Factory &)
Kind:	function
Scope:	class ara::per::FileProxyAccessorFactory
Syntax:	`FileProxyAccessorFactory (const FileProxyAccessorFactory &)=delete;`
Thread Safety:	no
Header file:	#include "ara/per/file_proxy_accessor_factory.h"
Description:	The copy constructor for FileProxyAccessorFactory shall not be used.

Table 8.28: function ara::per::FileProxyAccessorFactory::FileProxyAccessorFactory

8.2.7.2 FileProxyAccessorFactory::operator=

[SWS_PER_00327] ⌈ The function ara::per::FileProxyAccessorFactory::operator= is defined in Table 8.29. ⌋*(RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00132)*

Symbol:	ara::per::FileProxyAccessorFactory::operator=(FileProxyAccessorFactory &&fp)	
Kind:	function	
Scope:	class ara::per::FileProxyAccessorFactory	
Syntax:	`FileProxyAccessorFactory& operator= (FileProxyAccessorFactory &&fp) noexcept;`	
Parameters (in):	fp	The FileProxyAccessorFactory object to be moved.
Return value:	FileProxyAccessorFactory &	The moved FileProxyAccessorFactory object.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/file_proxy_accessor_factory.h"	
Description:	Move assignment operator for FileProxyAccessorFactory.	

Table 8.29: function ara::per::FileProxyAccessorFactory::operator=

[SWS_PER_00329] ⌈ The function ara::per::FileProxyAccessorFactory::operator= is defined in Table 8.30. ⌋(*RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*)

Symbol:	ara::per::FileProxyAccessorFactory::operator=(const FileProxyAccessorFactory &)
Kind:	function
Scope:	class ara::per::FileProxyAccessorFactory
Syntax:	`FileProxyAccessorFactory& operator= (const FileProxyAccessorFactory &)=delete;`
Thread Safety:	no
Header file:	#include "ara/per/file_proxy_accessor_factory.h"
Description:	The copy assignment operator for FileProxyAccessorFactory shall not be used.

Table 8.30: function ara::per::FileProxyAccessorFactory::operator=

8.2.7.3 FileProxyAccessorFactory::~FileProxyAccessorFactory

[SWS_PER_00330] ⌈ The function ara::per::FileProxyAccessorFactory::~FileProxyAccessorFactory is defined in Table 8.31. ⌋(*RS_PER_00004*, *RS_AP_00120*, *RS_AP_00129*, *RS_AP_00132*, *RS_AP_00134*)

Symbol:	ara::per::FileProxyAccessorFactory::~FileProxyAccessorFactory()
Kind:	function
Scope:	class ara::per::FileProxyAccessorFactory
Syntax:	`~FileProxyAccessorFactory () noexcept;`
Exception Safety:	noexcept
Thread Safety:	no
Header file:	#include "ara/per/file_proxy_accessor_factory.h"
Description:	Destructor for FileProxyAccessorFactory.

Table 8.31: function ara::per::FileProxyAccessorFactory::~FileProxyAccessorFactory

8.2.7.4 FileProxyAccessorFactory::GetAllKeys

[SWS_PER_00110] ⌈ The function ara::per::FileProxyAccessorFactory::GetAllKeys is defined in Table 8.32. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00127*, *RS_AP_00129*, *RS_AP_00132*)

Symbol:	ara::per::FileProxyAccessorFactory::GetAllKeys()	
Kind:	function	
Scope:	class ara::per::FileProxyAccessorFactory	
Syntax:	`ara::core::Vector<ara::core::String> GetAllKeys () const noexcept;`	
Return value:	ara::core::Vector< ara::core::String >	A list of availables files.

▽

△

Exception Safety:	noexcept
Thread Safety:	no
Header file:	#include "ara/per/file_proxy_accessor_factory.h"
Description:	Returns a list of available files within this proxy. A list of availables files.

Table 8.32: function ara::per::FileProxyAccessorFactory::GetAllKeys

8.2.7.5 FileProxyAccessorFactory::DeleteKey

[SWS_PER_00111] ⌈ The function ara::per::FileProxyAccessorFactory::DeleteKey is defined in Table 8.33. ⌋*(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00132)*

Symbol:	ara::per::FileProxyAccessorFactory::DeleteKey(ara::core::StringView key)	
Kind:	function	
Scope:	class ara::per::FileProxyAccessorFactory	
Syntax:	`ara::core::Result<void> DeleteKey (ara::core::StringView key) noexcept;`	
Parameters (in):	key	The identifier of the file.
Return value:	ara::core::Result< void >	A Result, being empty or containing one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/file_proxy_accessor_factory.h"	
Description:	Deletes a file from this proxy. This operation will fail with a kResourceBusyError when the file is currently open.	

Table 8.33: function ara::per::FileProxyAccessorFactory::DeleteKey

8.2.7.6 FileProxyAccessorFactory::HasKey

[SWS_PER_00112] ⌈ The function ara::per::FileProxyAccessorFactory::HasKey is defined in Table 8.34. ⌋*(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00132)*

Symbol:	ara::per::FileProxyAccessorFactory::HasKey(ara::core::StringView key)
Kind:	function
Scope:	class ara::per::FileProxyAccessorFactory
Syntax:	`bool HasKey (ara::core::StringView key) noexcept;`

▽

△

Parameters (in):	key	Identifier of the file.
Return value:	bool	True if the file exists, false otherwise
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/file_proxy_accessor_factory.h"	
Description:	Queries if a file is available in this proxy.	

Table 8.34: function ara::per::FileProxyAccessorFactory::HasKey

8.2.7.7 FileProxyAccessorFactory::RecoverKey

[SWS_PER_00337] ⌈ The function ara::per::FileProxyAccessorFactory::RecoverKey is defined in Table 8.35. ⌋*(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00132)*

Symbol:	ara::per::FileProxyAccessorFactory::RecoverKey(ara::core::StringView key)	
Kind:	function	
Scope:	class ara::per::FileProxyAccessorFactory	
Syntax:	`ara::core::Result<void> RecoverKey (ara::core::StringView key) noexcept;`	
Parameters (in):	key	The identifier of the file.
Return value:	ara::core::Result< void >	A Result, being empty or containing one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/file_proxy_accessor_factory.h"	
Description:	Recovers a file of this proxy. This method allows to recover a single file when the redundancy checks fail. It will fail with a k ResourceBusyError when the file is currently open. This method does a best-effort recovery of the file. After recovery, the file might show outdated or initial content, or might be lost.	

Table 8.35: function ara::per::FileProxyAccessorFactory::RecoverKey

8.2.7.8 FileProxyAccessorFactory::ResetKey

[SWS_PER_00338] ⌈ The function ara::per::FileProxyAccessorFactory::ResetKey is defined in Table 8.36. ⌋*(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00132)*

Symbol:	ara::per::FileProxyAccessorFactory::ResetKey(ara::core::StringView key)	
Kind:	function	
Scope:	class ara::per::FileProxyAccessorFactory	
Syntax:	`ara::core::Result<void> ResetKey (ara::core::StringView key) noexcept;`	
Parameters (in):	key	The identifier of the file.
Return value:	ara::core::Result< void >	A Result, being empty or containing one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/file_proxy_accessor_factory.h"	
Description:	Resets a file of this proxy to its initial content. This method allows to reset a single file to its initial content. It will fail with a kResourceBusy Error when the file is currently open, and with a kInitValueNotAvailableError when deployment does not define an initial content for the file.	

Table 8.36: function ara::per::FileProxyAccessorFactory::ResetKey

8.2.7.9 FileProxyAccessorFactory::CreateRWAccess

[SWS_PER_00113] ⌈ The function ara::per::FileProxyAccessorFactory::CreateRWAccess is defined in Table 8.37. ⌋*(RS_PER_00001, RS_PER_00004, RS_PER_00010, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00132)*

Symbol:	ara::per::FileProxyAccessorFactory::CreateRWAccess(ara::core::StringView key, BasicOperations::OpenMode const mode=BasicOperations::OpenMode::kOut|BasicOperations::OpenMode::kIn)	
Kind:	function	
Scope:	class ara::per::FileProxyAccessorFactory	
Syntax:	`ara::core::Result<std::unique_ptr<ReadWriteAccessor> > CreateRWAccess (ara::core::StringView key, BasicOperations::OpenMode const mode=BasicOperations::OpenMode::kOut|BasicOperations::OpenMode::kIn) noexcept;`	
Parameters (in):	key	Identifier of the file. May correspond to the PersistencyFile.fileName of a configured file.
	mode	Mode with which the file shall be opened.
Return value:	ara::core::Result< std::unique_ptr< ReadWriteAccessor > >	A Result, containing an instance of ReadWrite Accessor, or one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/file_proxy_accessor_factory.h"	
Description:	Creates an accessor for reading and writing a file of the proxy. An error that occurs when a new file is created in the proxy shall be reported using a failbit similarly to std::fstream. Such an error could occur when the number of files would afterwards exceed PersistencyFileProxy Interface.maxNumberOfFiles.	

Table 8.37: function ara::per::FileProxyAccessorFactory::CreateRWAccess

8.2.7.10 FileProxyAccessorFactory::CreateReadAccess

[SWS_PER_00114] ⌈ The function ara::per::FileProxyAccessorFactory::CreateReadAccess is defined in Table 8.38. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_PER_00010*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00128*, *RS_AP_00129*, *RS_AP_00132*)

Symbol:	ara::per::FileProxyAccessorFactory::CreateReadAccess(ara::core::StringView key, BasicOperations::OpenMode const mode=BasicOperations::OpenMode::kIn)	
Kind:	function	
Scope:	class ara::per::FileProxyAccessorFactory	
Syntax:	`ara::core::Result<std::unique_ptr<ReadAccessor> > CreateReadAccess (ara::core::StringView key, BasicOperations::OpenMode const mode=BasicOperations::OpenMode::kIn) noexcept;`	
Parameters (in):	key	Identifier of the file. May correspond to the PersistencyFile.fileName of a configured file.
	mode	Mode with which the file shall be opened.
Return value:	ara::core::Result< std::unique_ptr< ReadAccessor > >	A Result, containing an instance of ReadAccessor, or one of the errors defined for Persistency in Per Errc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/file_proxy_accessor_factory.h"	
Description:	Creates an accessor for reading a file of the proxy.	

Table 8.38: function ara::per::FileProxyAccessorFactory::CreateReadAccess

8.2.7.11 FileProxyAccessorFactory::CreateWriteAccess

[SWS_PER_00115] ⌈ The function ara::per::FileProxyAccessorFactory::CreateWriteAccess is defined in Table 8.39. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_PER_00010*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00128*, *RS_AP_00129*, *RS_AP_00132*)

Symbol:	ara::per::FileProxyAccessorFactory::CreateWriteAccess(ara::core::StringView key, BasicOperations::OpenMode const mode=BasicOperations::OpenMode::kOut)	
Kind:	function	
Scope:	class ara::per::FileProxyAccessorFactory	
Syntax:	`ara::core::Result<std::unique_ptr<ReadWriteAccessor> > CreateWrite Access (ara::core::StringView key, BasicOperations::OpenMode const mode=BasicOperations::OpenMode::kOut) noexcept;`	
Parameters (in):	key	Identifier of the file. May correspond to the PersistencyFile.fileName of a configured file.
	mode	Mode with which the file shall be opened.
Return value:	ara::core::Result< std::unique_ptr< ReadWriteAccessor > >	A Result, containing an instance of ReadWrite Accessor, or one of the errors defined for Persistency in PerErrc.

▽

△

Exception Safety:	noexcept
Thread Safety:	no
Header file:	#include "ara/per/file_proxy_accessor_factory.h"
Description:	Creates an accessor for writing a file of the proxy. An error that occurs when a new file is created in the proxy shall be reported using a failbit similarly to std::fstream. Such an error could occur when the number of files would afterwards exceed PersistencyFileProxy Interface.maxNumberOfFiles.

Table 8.39: function ara::per::FileProxyAccessorFactory::CreateWriteAccess

8.2.8 Char Traits Wrapper

This section shows the types that are used by the classes 8.2.9, 8.2.10, and 8.2.11. They correspond to the `std::char_traits` types of the same name.

8.2.8.1 int_type

[SWS_PER_00180] ⌈ The type alias ara::per::int_type is defined in Table 8.40. ⌋ *(RS_PER_00003, RS_AP_00122)*

Kind:	type alias
Scope:	namespace ara::per
Derived from:	typedef __implementation_specific__
Syntax:	using ara::per::int_type = __implementation_specific__;
Header file:	#include "ara/per/char_traits_wrapper.h"
Description:	Value read from a file, used in file-proxy operations similarly to std::char_traits::int_type.

Table 8.40: type alias ara::per::int_type

8.2.8.2 pos_type

[SWS_PER_00181] ⌈ The type alias ara::per::pos_type is defined in Table 8.41. ⌋ *(RS_PER_00003, RS_AP_00122)*

Kind:	type alias
Scope:	namespace ara::per
Derived from:	typedef __implementation_specific__
Syntax:	using ara::per::pos_type = __implementation_specific__;
Header file:	#include "ara/per/char_traits_wrapper.h"
Description:	Position in a file, used in file-proxy operations similarly to std::char_traits::pos_type.

Table 8.41: type alias ara::per::pos_type

8.2.8.3 off_type

[SWS_PER_00182] ⌈ The type alias ara::per::off_type is defined in Table 8.42. ⌋ *(RS_PER_00003, RS_AP_00122)*

Kind:	type alias
Scope:	namespace ara::per
Derived from:	typedef __implementation_specific__
Syntax:	using ara::per::off_type = __implementation_specific__;
Header file:	#include "ara/per/char_traits_wrapper.h"
Description:	Offset in a file, used in file-proxy operations similarly to std::char_traits::off_type.

Table 8.42: type alias ara::per::off_type

8.2.9 BasicOperations class

This section shows the types and methods defined by the `BasicOperations` class that are used by the classes 8.2.10 and 8.2.11. They correspond roughly to the types and methods provided by `std::iostream`.

[SWS_PER_00341] ⌈ The class ara::per::BasicOperations is defined in Table 8.43. ⌋ *(RS_PER_00003, RS_AP_00122)*

Kind:	class
Base class:	None
Syntax:	`class BasicOperations`
Header file:	#include "ara/per/basic_operations.h"
Description:	The basic operations have to be supported by all accessor interfaces. It contains seeking and error checking.

Table 8.43: class ara::per::BasicOperations

8.2.9.1 BasicOperations::BasicOperations

[SWS_PER_00344] ⌈ The function ara::per::BasicOperations::BasicOperations is defined in Table 8.44. ⌋ *(RS_PER_00002, RS_AP_00120, RS_AP_00121, RS_AP_00129, RS_AP_00132)*

Symbol:	ara::per::BasicOperations::BasicOperations(BasicOperations &&kvs)	
Kind:	function	
Scope:	class ara::per::BasicOperations	
Syntax:	`BasicOperations (BasicOperations &&kvs) noexcept;`	
Parameters (in):	kvs	The BasicOperations object to be moved.

▽

△

Exception Safety:	noexcept
Thread Safety:	no
Header file:	#include "ara/per/basic_operations.h"
Description:	Move constructor for BasicOperations.

Table 8.44: function ara::per::BasicOperations::BasicOperations

[SWS_PER_00346] ⌈ The function ara::per::BasicOperations::BasicOperations is defined in Table 8.45. ⌋*(RS_PER_00002, RS_AP_00120)*

Symbol:	ara::per::BasicOperations::BasicOperations(const BasicOperations &)
Kind:	function
Scope:	class ara::per::BasicOperations
Syntax:	`BasicOperations (const BasicOperations &)=delete;`
Thread Safety:	no
Header file:	#include "ara/per/basic_operations.h"
Description:	The copy constructor for BasicOperations shall not be used.

Table 8.45: function ara::per::BasicOperations::BasicOperations

8.2.9.2 BasicOperations::operator=

[SWS_PER_00345] ⌈ The function ara::per::BasicOperations::operator= is defined in Table 8.46. ⌋*(RS_PER_00002, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00132)*

Symbol:	ara::per::BasicOperations::operator=(BasicOperations &&kvs)	
Kind:	function	
Scope:	class ara::per::BasicOperations	
Syntax:	`BasicOperations& operator= (BasicOperations &&kvs) noexcept;`	
Parameters (in):	kvs	The BasicOperations object to be moved.
Return value:	BasicOperations &	The moved BasicOperations object.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/basic_operations.h"	
Description:	Move assignment operator for BasicOperations.	

Table 8.46: function ara::per::BasicOperations::operator=

[SWS_PER_00347] ⌈ The function ara::per::BasicOperations::operator= is defined in Table 8.47. ⌋*(RS_PER_00002, RS_AP_00119, RS_AP_00120)*

Symbol:	ara::per::BasicOperations::operator=(const BasicOperations &)
Kind:	function
Scope:	class ara::per::BasicOperations
Syntax:	`BasicOperations& operator= (const BasicOperations &)=delete;`
Thread Safety:	no
Header file:	#include "ara/per/basic_operations.h"
Description:	The copy assignment operator for BasicOperations shall not be used.

Table 8.47: function ara::per::BasicOperations::operator=

8.2.9.3 BasicOperations::~BasicOperations

[SWS_PER_00348] ⌈ The function ara::per::BasicOperations::~BasicOperations is defined in Table 8.48. ⌋*(RS_PER_00002, RS_AP_00120, RS_AP_00129, RS_AP_00132, RS_AP_00134)*

Symbol:	ara::per::BasicOperations::~BasicOperations()
Kind:	function
Scope:	class ara::per::BasicOperations
Syntax:	`~BasicOperations () noexcept;`
Exception Safety:	noexcept
Thread Safety:	no
Header file:	#include "ara/per/basic_operations.h"
Description:	Destructor for BasicOperations.

Table 8.48: function ara::per::BasicOperations::~BasicOperations

8.2.9.4 BasicOperations::SeekDirection

[SWS_PER_00146] ⌈ The enum ara::per::BasicOperations::SeekDirection is defined in Table 8.49. ⌋*(RS_PER_00003, RS_AP_00122)*

Kind:	enum	
Values:	kBeg= 0	Seek from the beginning.
	kEnd= 1	Seek from the end.
	kCur= 2	Seek from the current position.
Header file:	#include "ara/per/basic_operations.h"	
Description:	Specification of seek direction.	

Table 8.49: enum ara::per::BasicOperations::SeekDirection

8.2.9.5 BasicOperations::OpenMode

[SWS_PER_00147] ⌈ The enum ara::per::BasicOperations::OpenMode is defined in Table 8.50. ⌋(*RS_PER_00003*, *RS_AP_00122*)

Kind:	enum	
Values:	kApp= 1 « 0	Append to the end. Seeks to the end of the file before writing.
	kBinary= 1 « 1	Opens the file as binary. Otherwise (if not specified), the file will be opened as text.
	kIn= 1 « 2	Opens the file for reading.
	kOut= 1 « 3	Opens the file for writing.
	kTrunc= 1 « 4	Deletes existing content when the file is opened.
	kAte= 1 « 5	Sets the seek pointer to the end of the file when the file is opened.
Header file:	#include "ara/per/basic_operations.h"	
Description:	This enumeration defines how a file shall be opened. The values can be combined (using & and |) as long as they do not contradict each other.	

Table 8.50: enum ara::per::BasicOperations::OpenMode

8.2.9.6 BasicOperations::tell

[SWS_PER_00162] ⌈ The function ara::per::BasicOperations::tell is defined in Table 8.51. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

Symbol:	ara::per::BasicOperations::tell()	
Kind:	function	
Scope:	class ara::per::BasicOperations	
Syntax:	`pos_type tell () noexcept;`	
Return value:	pos_type	Current position in the file in bytes from the beginning.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/basic_operations.h"	
Description:	Returns the current position relative to the beginning of the file. Current position in the file in bytes from the beginning.	

Table 8.51: function ara::per::BasicOperations::tell

8.2.9.7 BasicOperations::seek

[SWS_PER_00163] ⌈ The function ara::per::BasicOperations::seek is defined in Table 8.52. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

Symbol:	ara::per::BasicOperations::seek(pos_type const pos)	
Kind:	function	
Scope:	class ara::per::BasicOperations	
Syntax:	`ara::per::BasicOperations& seek (pos_type const pos) noexcept;`	
Parameters (in):	pos	Current position in the file in bytes from the beginning.
Return value:	ara::per::BasicOperations &	BasicOperations object for chaining.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/basic_operations.h"	
Description:	Sets the current position relative to the beginning of the file.	

Table 8.52: function ara::per::BasicOperations::seek

[SWS_PER_00164] ⌈ The function ara::per::BasicOperations::seek is defined in Table 8.53. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

Symbol:	ara::per::BasicOperations::seek(off_type const off, SeekDirection const dir)	
Kind:	function	
Scope:	class ara::per::BasicOperations	
Syntax:	`ara::per::BasicOperations& seek (off_type const off, SeekDirection const dir) noexcept;`	
Parameters (in):	off	Current offset in bytes relative to dir.
	dir	Direction into which to move off bytes.
Return value:	ara::per::BasicOperations &	BasicOperations object for chaining.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/basic_operations.h"	
Description:	Sets the current position in the file according to the SeekDirection.	

Table 8.53: function ara::per::BasicOperations::seek

8.2.9.8 BasicOperations::good

[SWS_PER_00106] ⌈ The function ara::per::BasicOperations::good is defined in Table 8.54. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

Symbol:	ara::per::BasicOperations::good()
Kind:	function
Scope:	class ara::per::BasicOperations
Syntax:	`bool good () const noexcept;`

▽

△

Return value:	bool	True if no error occurred, false otherwise.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/basic_operations.h"	
Description:	Checks if no error occurred during an operation. True if no error occurred, false otherwise.	

Table 8.54: function ara::per::BasicOperations::good

8.2.9.9 BasicOperations::eof

[SWS_PER_00107] ⌈ The function ara::per::BasicOperations::eof is defined in Table 8.55. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

Symbol:	ara::per::BasicOperations::eof()	
Kind:	function	
Scope:	class ara::per::BasicOperations	
Syntax:	`bool eof () const noexcept;`	
Return value:	bool	True if the end of the file was reached, false otherwise.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/basic_operations.h"	
Description:	Checks if end of file was reached during an operation. True if the end of the file was reached, false otherwise.	

Table 8.55: function ara::per::BasicOperations::eof

8.2.9.10 BasicOperations::fail

[SWS_PER_00108] ⌈ The function ara::per::BasicOperations::fail is defined in Table 8.56. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

Symbol:	ara::per::BasicOperations::fail()	
Kind:	function	
Scope:	class ara::per::BasicOperations	
Syntax:	`bool fail () const noexcept;`	
Return value:	bool	True if an error occurred, false otherwise.
Exception Safety:	noexcept	

▽

△

Thread Safety:	no
Header file:	#include "ara/per/basic_operations.h"
Description:	Checks if an error occurred during an operation. True if an error occurred, false otherwise.

Table 8.56: function ara::per::BasicOperations::fail

8.2.9.11 BasicOperations::bad

[SWS_PER_00140] ⌈ The function ara::per::BasicOperations::bad is defined in Table 8.57. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

Symbol:	ara::per::BasicOperations::bad()	
Kind:	function	
Scope:	class ara::per::BasicOperations	
Syntax:	`bool bad () const noexcept;`	
Return value:	bool	True if an error occurred and the integrity of the stream was lost, false otherwise.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/basic_operations.h"	
Description:	Checks if an error occurred during an operation which destroyed the integrity of the stream. True if an error occurred and the integrity of the stream was lost, false otherwise.	

Table 8.57: function ara::per::BasicOperations::bad

8.2.9.12 BasicOperations::operator!

[SWS_PER_00142] ⌈ The function ara::per::BasicOperations::operator! is defined in Table 8.58. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

Symbol:	ara::per::BasicOperations::operator!()	
Kind:	function	
Scope:	class ara::per::BasicOperations	
Syntax:	`bool operator! () const noexcept;`	
Return value:	bool	True if an error occurred, false otherwise.
Exception Safety:	noexcept	
Thread Safety:	no	

▽

△

Header file:	#include "ara/per/basic_operations.h"
Description:	Checks if an error occurred during operation, functionally equivalent to ara::per::Basic Operations::fail(). True if an error occurred, false otherwise.

Table 8.58: function ara::per::BasicOperations::operator!

8.2.9.13 BasicOperations::operator bool

[SWS_PER_00143] ⌈ The function ara::per::BasicOperations::operator bool is defined in Table 8.59. ⌋*(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00132)*

Symbol:	ara::per::BasicOperations::operator bool()
Kind:	function
Scope:	class ara::per::BasicOperations
Syntax:	`explicit operator bool () const noexcept;`
Exception Safety:	noexcept
Thread Safety:	no
Header file:	#include "ara/per/basic_operations.h"
Description:	Checks if no error occurred during operation, functionally equivalent to ara::per::Basic Operations::good(). True if no error occurred, false otherwise.

Table 8.59: function ara::per::BasicOperations::operator bool

8.2.9.14 BasicOperations::clear

[SWS_PER_00141] ⌈ The function ara::per::BasicOperations::clear is defined in Table 8.60. ⌋*(RS_PER_00001, RS_PER_00004, RS_AP_00120, RS_AP_00132)*

Symbol:	ara::per::BasicOperations::clear()
Kind:	function
Scope:	class ara::per::BasicOperations
Syntax:	`void clear () noexcept;`
Return value:	None
Exception Safety:	noexcept
Thread Safety:	no
Header file:	#include "ara/per/basic_operations.h"
Description:	Clears all error flags.

Table 8.60: function ara::per::BasicOperations::clear

8.2.10 ReadAccessor class

This section shows the methods available for a ReadAccessor object obtained from a call to 8.2.7.10, and for the inheriting ReadWriteAccessor object obtained from a call to 8.2.7.11 or 8.2.7.9.

[SWS_PER_00342] ⌈ The class ara::per::ReadAccessor is defined in Table 8.61. ⌋ *(RS_PER_00004, RS_AP_00122)*

Kind:	class
Base class:	ara::per::BasicOperations
Syntax:	`class ReadAccessor : public BasicOperations`
Header file:	#include "ara/per/read_accessor.h"
Description:	ReadAccessor is used to read file data. For unformatted reading it provides the read() method and for formatted reading it provides the operator»

Table 8.61: class ara::per::ReadAccessor

8.2.10.1 ReadAccessor::peek

[SWS_PER_00167] ⌈ The function ara::per::ReadAccessor::peek is defined in Table 8.62. ⌋ *(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00132)*

Symbol:	ara::per::ReadAccessor::peek()	
Kind:	function	
Scope:	class ara::per::ReadAccessor	
Syntax:	`int_type peek () noexcept;`	
Return value:	int_type	The character at the current position.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/read_accessor.h"	
Description:	Returns the character at the current position in the file. The character at the current position.	

Table 8.62: function ara::per::ReadAccessor::peek

8.2.10.2 ReadAccessor::get

[SWS_PER_00168] ⌈ The function ara::per::ReadAccessor::get is defined in Table 8.63. ⌋ *(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00132)*

Symbol:	ara::per::ReadAccessor::get()	
Kind:	function	
Scope:	class ara::per::ReadAccessor	
Syntax:	`int_type get () noexcept;`	
Return value:	int_type	The character at the current position.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/read_accessor.h"	
Description:	Returns the character at the current position in the file, advancing the current position. The character at the current position.	

Table 8.63: function ara::per::ReadAccessor::get

8.2.10.3 ReadAccessor::read

[SWS_PER_00165] ⌈ The function ara::per::ReadAccessor::read is defined in Table 8.64. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00132*)

Symbol:	ara::per::ReadAccessor::read(ara::core::Span< char > s)	
Kind:	function	
Scope:	class ara::per::ReadAccessor	
Syntax:	`pos_type read (ara::core::Span< char > s) noexcept;`	
Parameters (out):	s	A span of char where the read characters shall be stored.
Return value:	pos_type	Actual number of charactes that have been read.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/read_accessor.h"	
Description:	Reads a number of characters into a char pointer, advancing the current position. Returns the actual number of characters that were read.	

Table 8.64: function ara::per::ReadAccessor::read

8.2.10.4 ReadAccessor::getline

[SWS_PER_00119] ⌈ The function ara::per::ReadAccessor::getline is defined in Table 8.65. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00129*, *RS_AP_00132*)

Symbol:	ara::per::ReadAccessor::getline(ara::core::String &string, char const delim= '\n')	
Kind:	function	
Scope:	class ara::per::ReadAccessor	
Syntax:	`ReadAccessor& getline (ara::core::String &string, char const delim= '\n') noexcept;`	
Parameters (in):	delim	The character that is used as delimiter.
Parameters (out):	string	A string where the line will be stored.
Return value:	ReadAccessor &	The ReadAccessor object.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/read_accessor.h"	
Description:	Reads a complete line into a sting, advancing the current position.	

Table 8.65: function ara::per::ReadAccessor::getline

8.2.10.5 ReadAccessor::operator»

[SWS_PER_00160] ⌈ The function ara::per::ReadAccessor::operator» is defined in Table 8.66. ⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00129*, *RS_AP_00132*)

Symbol:	ara::per::ReadAccessor::operator»(ara::core::String &string)	
Kind:	function	
Scope:	class ara::per::ReadAccessor	
Syntax:	`ReadAccessor& operator» (ara::core::String &string) noexcept;`	
Parameters (out):	string	A string where the read characters will be stored.
Return value:	ReadAccessor &	The ReadAccessor object.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/read_accessor.h"	
Description:	Reads all available characters into a string, advancing the current position.	

Table 8.66: function ara::per::ReadAccessor::operator»

8.2.11 ReadWriteAccessor class

This section shows the methods available for a ReadWriteAccessor object obtained from a call to 8.2.7.11 or 8.2.7.9.

[SWS_PER_00343] ⌈ The class ara::per::ReadWriteAccessor is defined in Table 8.67. ⌋(*RS_PER_00004*, *RS_AP_00122*)

Kind:	class
Base class:	ara::per::ReadAccessor
Syntax:	`class ReadWriteAccessor : public ReadAccessor`
Header file:	#include "ara/per/read_write_accessor.h"
Description:	ReadWriteAccessor is used to read and write file data. For unformatted reading it provides the read() method and for formatted reading it provides the operator» For unformatted writing it provides the write() method and for formatted writing it provides the operator«. It also provides the ability to force an fsync to flush the buffer of the operating system to the storage.

Table 8.67: class ara::per::ReadWriteAccessor

8.2.11.1 ReadWriteAccessor::fsync

[SWS_PER_00122] ⌈ The function ara::per::ReadWriteAccessor::fsync is defined in Table 8.68. ⌋*(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00128, RS_AP_00127, RS_AP_00129, RS_AP_00132)*

Symbol:	ara::per::ReadWriteAccessor::fsync()	
Kind:	function	
Scope:	class ara::per::ReadWriteAccessor	
Syntax:	`ara::core::Result<void> fsync () noexcept;`	
Return value:	ara::core::Result< void >	A Result, being either empty or containing one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/read_write_accessor.h"	
Description:	Flushes and forces the write buffer to the persistent storage of the file. A Result, being either empty or containing one of the errors defined for Persistency in PerErrc.	

Table 8.68: function ara::per::ReadWriteAccessor::fsync

8.2.11.2 ReadWriteAccessor::write

[SWS_PER_00166] ⌈ The function ara::per::ReadWriteAccessor::write is defined in Table 8.69. ⌋*(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00132)*

Symbol:	ara::per::ReadWriteAccessor::write(ara::core::Span< char > s)	
Kind:	function	
Scope:	class ara::per::ReadWriteAccessor	
Syntax:	`pos_type write (ara::core::Span< char > s) noexcept;`	
Parameters (in):	s	A span of char from where the characters shall be taken.
Return value:	pos_type	Actual number of characters that have been written.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/read_write_accessor.h"	
Description:	Writes a number of characters from a char pointer. Returns the actual number of characters that were written.	

Table 8.69: function ara::per::ReadWriteAccessor::write

8.2.11.3 ReadWriteAccessor::flush

[SWS_PER_00124] ⌈ The function ara::per::ReadWriteAccessor::flush is defined in Table 8.70. ⌋*(RS_PER_00001, RS_PER_00004, RS_AP_00120, RS_AP_00132)*

Symbol:	ara::per::ReadWriteAccessor::flush()
Kind:	function
Scope:	class ara::per::ReadWriteAccessor
Syntax:	`void flush () noexcept;`
Return value:	None
Exception Safety:	noexcept
Thread Safety:	no
Header file:	#include "ara/per/read_write_accessor.h"
Description:	Flushes the write buffer to the file.

Table 8.70: function ara::per::ReadWriteAccessor::flush

8.2.11.4 ReadWriteAccessor::operator«

[SWS_PER_00125] ⌈ The function ara::per::ReadWriteAccessor::operator« is defined in Table 8.71. ⌋*(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00132)*

Symbol:	ara::per::ReadWriteAccessor::operator«(ara::core::StringView s)
Kind:	function
Scope:	class ara::per::ReadWriteAccessor
Syntax:	`ReadWriteAccessor& operator« (ara::core::StringView s) noexcept;`

▽

△

Parameters (in):	s	The string to be written.
Return value:	ReadWriteAccessor &	The ReadWriteAccessor object.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/read_write_accessor.h"	
Description:	Writes a string to the file.	

Table 8.71: function ara::per::ReadWriteAccessor::operator«

[SWS_PER_00126] ⌈ The function ara::per::ReadWriteAccessor::operator« is defined in Table 8.72. ⌋*(RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00132)*

Symbol:	ara::per::ReadWriteAccessor::operator«(ReadWriteAccessor &(*op)	
Kind:	function	
Scope:	class ara::per::ReadWriteAccessor	
Syntax:	`ReadWriteAccessor& operator« (ReadWriteAccessor &(*op)(ReadWrite Accessor &)) noexcept;`	
Parameters (in):	op	The operation to be executed on the file.
Return value:	ReadWriteAccessor &	The ReadWriteAccessor object.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/read_write_accessor.h"	
Description:	Executes endl or flush operations on the file.	

Table 8.72: function ara::per::ReadWriteAccessor::operator«

8.3 Errors

The `Persistency` cluster implements an error handling based on `ara::core::Result`. The errors supported by the `Persistency` cluster are listed in section 8.3.1.

8.3.1 PerErrc

[SWS_PER_00311] ⌈ The enum ara::per::PerErrc is defined in Table 8.73. ⌋ *(RS_AP_00122, RS_AP_00127, RS_AP_00128)*

Kind:	enum	
Values:	kStorageLocationNotFoundError= 1	Requested storage location is not found or not configured in the AUTOSAR model.
	kKeyNotFoundError= 2	The key was not found.
	kIllegalWriteAccessError= 3	Opening the resource for writing failed because it is configured read-only.
	kPhysicalStorageError= 4	A severe error which might happen during the operation, such as out of memory or writing/reading to the storage return an error.
	kIntegrityError= 5	The integrity of the storage could not be established. This can happen when the structure of a key value database is corrupted, or a read-only file has no content.
	kValidationError= 6	The validation of redundancy measures failed for a single key, for the whole key value data base, or for a file.
	kEncryptionError= 7	The encryption or decryption failed for a single key, for the whole key value data base, or for a file.
	kDataTypeMismatchError= 8	The provided data type does not match the stored data type.
	kInitValueNotAvailableError= 9	The operation could not be performed because no initial value is available.
	kResourceBusyError= 10	The operation could not be performed because the resource is currently busy.
	kInternalError= 11	Undefined error, implementation specific.
Header file:	#include "ara/per/error.h"	
Description:	Defines the errors for Persistency. The enumeration values 0 - 255 are reserved for AUTOSAR assigned errors, the stack provider is free to define additional errors starting from 256.	

Table 8.73: enum ara::per::PerErrc

8.3.2 PerErrorDomain

The error handling requires an `ara::core::ErrorDomain`, which can be used to check the errors returned via `ara::core::Result`.

[SWS_PER_00312] ⌈ The class ara::per::PerErrorDomain is defined in Table 8.74. ⌋ *(RS_AP_00122, RS_AP_00127, RS_AP_00128)*

Kind:	class
Base class:	ara::core::ErrorDomain
Syntax:	`class PerErrorDomain : public ErrorDomain`
Header file:	#include "ara/per/error.h"
Description:	Defines the error domain for Persistency.

Table 8.74: class ara::per::PerErrorDomain

8.3.2.1 PerErrorDomain::kId

[SWS_PER_00316] ⌈ The variable ara::per::PerErrorDomain::kId is defined in Table 8.75. ⌋*(RS_AP_00128)*

Kind:	variable
Type:	const ErrorDomain::IdType
Scope:	class ara::per::PerErrorDomain
Syntax:	const ErrorDomain::IdType ara::per::PerErrorDomain::kId;
Header file:	#include "ara/per/error.h"
Description:	Key ID for persistency error domain.

Table 8.75: variable ara::per::PerErrorDomain::kId

8.3.2.2 PerErrorDomain::PerErrorDomain

[SWS_PER_00313] ⌈ The function ara::per::PerErrorDomain::PerErrorDomain is defined in Table 8.76. ⌋*(RS_AP_00119, RS_AP_00120, RS_AP_00128, RS_AP_00132)*

Symbol:	ara::per::PerErrorDomain::PerErrorDomain()
Kind:	function
Scope:	class ara::per::PerErrorDomain
Syntax:	`PerErrorDomain () noexcept;`
Exception Safety:	noexcept
Thread Safety:	no
Header file:	#include "ara/per/error.h"
Description:	Creates a PerErrorDomain instance.

Table 8.76: function ara::per::PerErrorDomain::PerErrorDomain

8.3.2.3 PerErrorDomain::Name

[SWS_PER_00314] ⌈ The function ara::per::PerErrorDomain::Name is defined in Table 8.77. ⌋*(RS_AP_00119, RS_AP_00120, RS_AP_00128, RS_AP_00132)*

Symbol:	ara::per::PerErrorDomain::Name()	
Kind:	function	
Scope:	class ara::per::PerErrorDomain	
Syntax:	`char const* Name () const noexcept override;`	
Return value:	char const *	The name of the error domain
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/error.h"	
Description:	Returns the name of the error domain. The name of the error domain	

Table 8.77: function ara::per::PerErrorDomain::Name

8.3.2.4 PerErrorDomain::Message

[SWS_PER_00315] ⌈ The function ara::per::PerErrorDomain::Message is defined in Table 8.78. ⌋*(RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00128, RS_AP_00132)*

Symbol:	ara::per::PerErrorDomain::Message(CodeType errorCode)	
Kind:	function	
Scope:	class ara::per::PerErrorDomain	
Syntax:	`char const* Message (CodeType errorCode) const noexcept override;`	
Parameters (in):	errorCode	The error code number
Return value:	char const *	The message associated with the error code.
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/error.h"	
Description:	Returns the message associated with the error code.	

Table 8.78: function ara::per::PerErrorDomain::Message

A Not applicable requirements

[SWS_PER_NA] ⌈ These requirements are not applicable to this specification. ⌋*(RS_PER_00014, RS_PER_00016, RS_AP_00111, RS_AP_00113, RS_AP_00114, RS_AP_00116, RS_AP_00124, RS_AP_00130, RS_AP_00131)*

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=AdaptiveApplicationSwComponentTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
internalBehavior	AdaptiveSwcInternal Behavior	0..1	aggr	This aggregation represents the internal behavior of the AdaptiveApplicationSwComponentType for the AUTOSAR adaptive platform. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalBehavior, variationPoint.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***
swDataDefProps	SwDataDefProps	0..1	aggr	The properties of this AutosarDataType.

Table B.2: AutosarDataType

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.

△

Class	**ImplementationDataType**			
isStructWith Optional Element	Boolean	0..1	attr	This attribute is only valid if the attribute category is set to STRUCTURE. If set to True, this attribute indicates that the ImplementationDataType has been created with the intention to define at least one element of the structure as optional. **Tags:** atp.Status=draft
subElement (ordered)	ImplementationData TypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a Implementation DataType representing a structure. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the Implementation DataType. **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.3: ImplementationDataType

Class	**PPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port providing a certain port interface.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
provided Interface	PortInterface	1	tref	The interface that this port provides. **Stereotypes:** isOfType

Table B.4: PPortPrototype

Class	**PRPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This kind of PortPrototype can take the role of both a required and a provided PortPrototype.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*, *AtpBlueprintable*, *Atp Feature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
provided Required Interface	PortInterface	1	tref	This represents the PortInterface used to type the PRPort Prototype **Stereotypes:** isOfType

Table B.5: PRPortPrototype

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*, *Multilanguage Referrable*, *Referrable*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
updateStrategy	PersistencyElement LevelUpdateStrategy Enum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyDataElement.

Table B.7: PersistencyDataElement

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	PersistencyData Element	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.8: PersistencyDataRequiredComSpec

Class	***PersistencyDeployment*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This abstract meta-class serves as a base class for concrete classes representing different aspects of persistency. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Subclasses	PersistencyFileArray, PersistencyKeyValueDatabase			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
maximum AllowedSize	PositiveUnlimitedInteger	0..1	attr	The value of this attribute represents the maximum size allowed at deployment time for the enclosing Persistency Deployment.
minimum SustainedSize	PositiveInteger	0..1	attr	The value of this attribute represents the minimum size guaranteed at deployment time for the enclosing PersistencyDeployment.
redundancy Handling	PersistencyRedundancy Handling	*	aggr	This aggregation represents the chosen approaches to handle redundancy. **Tags:** atp.Status=draft
updateStrategy	PersistencyCollection LevelUpdateStrategy Enum	1	attr	This attribute shall be used to specify the update strategy of the respective PersistencyDeployment as a whole.

Table B.9: PersistencyDeployment

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::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the model of a file as part of the persistency on deployment level. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft atp.recommendedPackage=PersistencyFiles			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
contentUri	UriString	0..1	attr	This attribute represents the URI that identifies the initial content of the PersistencyFile.

△

Class	**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	0..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::PlatformModuleDeployment::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=ExecutionManifest atp.Status=draft atp.recommendedPackage=PersistencyFileArrays			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PersistencyDeployment*, *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
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.
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	0..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	BaseTypeEncoding String	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 Persistency FileProxys in the context of the enclosing PersistencyFile ProxyInterface. **Tags:** atp.Status=draft
maxNumberOf Files	PositiveInteger	0..1	attr	This attribute represents the definition of an upper bound for the handling of files at run-time in the context of the enclosing PersistencyFileProxyInterface.

Table B.14: PersistencyFileProxyInterface

Class	***PersistencyInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the abstract ability to define a PortInterface for the support of persistency use cases. **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	PersistencyFileProxyInterface, PersistencyKeyValueDatabaseInterface			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
minimum SustainedSize	PositiveInteger	0..1	attr	The value of this attribute represents the minimum size required at design time for the enclosing Persistency Interface.
redundancy	PersistencyRedundancy Enum	0..1	attr	This attribute represents a requirement towards the redundancy of storage.
updateStrategy	PersistencyCollection LevelUpdateStrategy Enum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyInterface as a whole.

Table B.15: PersistencyInterface

Class	**PersistencyKeyValueDatabase**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to model a key/value data base on deployment level. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft atp.recommendedPackage=PersistencyKeyValueDatabases			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PersistencyDeployment*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
keyValuePair	PersistencyKeyValue Pair	*	aggr	This aggregation represents the key-value-pairs owned by the enclosing PersistencyKeyValueDatabase **Tags:** atp.Status=draft
uri	UriString	0..1	attr	This attribute holds the storage location for the PersistencyKeyValueDatabase / PersistencyFile, e.g. file on the file system.

Table B.16: 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=PersistencyKeyValueDatabaseInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyInterface*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
dataElement	PersistencyData Element	*	aggr	This aggregation represents the collection of Persistency DataElements in the context of the enclosing Persistency KeyValueDatabaseInterface. **Tags:** atp.Status=draft
dataTypeFor Serialization	AbstractImplementation DataType	*	ref	This reference identifies the AbstractImplementationData Types 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

Table B.17: PersistencyKeyValueDatabaseInterface

Class	**PersistencyKeyValuePair**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to formally model a key-value pair in the context of the deployment of persistency. **Tags:** atp.ManifestKind=ExecutionManifest 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

▽

△

Class	PersistencyKeyValuePair			
updateStrategy	PersistencyElement LevelUpdateStrategy Enum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyKeyValuePair.
valueDataType	AbstractImplementation DataType	1	ref	This reference represents the data type applicable for the value of the key-value pair. **Tags:** atp.Status=draft

Table B.18: PersistencyKeyValuePair

Class	PersistencyPortPrototypeToFileArrayMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::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=ExecutionManifest atp.Status=draft atp.recommendedPackage=PersistentFileProxyToFileMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
persistencyFile Array	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.19: PersistencyPortPrototypeToFileArrayMapping

Class	PersistencyPortPrototypeToKeyValueDatabaseMapping			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::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=ExecutionManifest atp.Status=draft atp.recommendedPackage=PersistentPortPrototypeToKeyValueDatabaseMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
keyValue Storage	PersistencyKeyValue Database	1	ref	This reference represents the mapped key-value storage. **Tags:** atp.Status=draft
portPrototype	PortPrototype	0..1	iref	This reference represents the affected Persistency Port Prototype **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.20: PersistencyPortPrototypeToKeyValueDatabaseMapping

Class	**PersistencyRedundancyCrc**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class formally describes the usage of a CRC for the implementation of redundancy. **Tags:** atp.Status=draft			
Base	*ARObject*, *PersistencyRedundancyHandling*			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
algorithmFamily	String	1	attr	This attribute identifies the algorithm family that is used to execute the CRC.
length	PositiveInteger	1	attr	This attribute describes the length of the CRC in the unit bits.

Table B.21: PersistencyRedundancyCrc

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

Table B.22: PersistencyRedundancyEnum

Class	***PersistencyRedundancyHandling*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This abstract base class represents a formal description of redundancy. **Tags:** atp.Status=draft			
Base	*ARObject*			
Subclasses	PersistencyRedundancyCrc, PersistencyRedundancyMOutOfN			
Attribute	*Type*	*Mul.*	*Kind*	*Note*
–	–	–	–	–

Table B.23: PersistencyRedundancyHandling

Class	**PersistencyRedundancyMOutOfN**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency
Note	This meta-class provides the ability to describe redundancy via an "M out of N" approach. In this case N is the number of copies created and M is the minimum number of identical copies to justify a reliable read access to the data. **Tags:** atp.Status=draft

▽

△

Class	**PersistencyRedundancyMOutOfN**			
Base	*ARObject*, *PersistencyRedundancyHandling*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
m	PositiveInteger	1	attr	This attribute represents the "M" coordinate in the "M out of N" scheme.
n	PositiveInteger	1	attr	This attribute represents the "N" coordinate in the "M out of N" scheme.

Table B.24: PersistencyRedundancyMOutOfN

Class	***PortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/server communication.
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port.
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port.
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port.
portPrototype Props	PortPrototypeProps	0..1	aggr	This attribute allows for the definition of further qualification of the semantics of a PortPrototype. **Tags:** atp.Status=draft
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table B.25: PortPrototype

Class	**Process**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest
Note	This meta-class provides information required to execute the referenced executable. **Tags:** atp.ManifestKind=ExecutionManifest atp.Status=draft atp.recommendedPackage=Processes
Base	*ARElement*, *ARObject*, *AbstractExecutionContext*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadablePackageElement*

▽

△

Class	**Process**			
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
deterministic Client	DeterministicClient	0..1	ref	This reference adds further execution characteristics for deterministic clients. **Tags:** atp.Status=draft
executable	Executable	0..1	ref	Reference to executable that is executed in the process. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
logTraceDefault LogLevel	LogTraceDefaultLog LevelEnum	0..1	attr	This attribute allows to set the initial log reporting level for a logTraceProcessId (ApplicationId).
logTraceFile Path	UriString	0..1	attr	This attribute defines the destination file to which the logging information is passed.
logTraceLog Mode	LogTraceLogMode Enum	0..1	attr	This attribute defines the destination of log messages provided by the process.
logTrace ProcessDesc	String	0..1	attr	This attribute can be used to describe the logTrace ProcessId that is used in the log and trace message in more detail.
logTrace ProcessId	String	0..1	attr	This attribute identifies the process in the log and trace message (ApplicationId).
mode Dependent StartupConfig	ModeDependentStartup Config	*	aggr	Applicable startup configurations. **Tags:** atp.Status=draft
processMode Machine	ModeDeclarationGroup Prototype	0..1	aggr	Set of Process States (Modes) that are defined for the process. **Tags:** atp.Status=draft

Table B.26: Process

Class	**RPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port requiring a certain port interface.			
Base	*ARObject*, *AbstractRequiredPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
required Interface	PortInterface	1	tref	The interface that this port requires, i.e. the port depends on another port providing the specified interface. **Stereotypes:** isOfType

Table B.27: RPortPrototype

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*, *NetworkConfiguration*, 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
shortName Fragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:** xml.sequenceOffset=-90

Table B.28: Referrable