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	19-03

Document Change History

Date	Release	Changed by	Description
2019-03-29	19-03	AUTOSAR Release Management	• Improved naming of classes/methods/functions • Reworked installation/update • Support for parallel execution in multiple threads • Cleaned up usage of ara::core concepts
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

The glossary below includes acronyms and abbreviations relevant to the Persistency that are not included in the [1, AUTOSAR glossary].

Abbreviation / Acronym	Description
KVS	Key-Value Storage

Terms	Description
File Storage	A set of files that are stored persistently.
Key-Value Pair	A key with an associated value, to be stored in a Key-Value Storage together with the type of the value.
Key-Value Storage	A set of key-value pairs that are stored persistently.
Persistency	The functional cluster described in this document, which handles persistent data of AUTOSAR Adaptive Applications and other functional clusters in File Storages and Key-Value Storages.
Persistent Data	Data that is stored in the persistent memory that can be accessed by one Process. Persistency supports different mechanisms to access data in persistent memory. Concurrent access to the data by several Processes is not supported as the data is owned exclusively by one Process.

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 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].

4.3 Direct Access to Storage Hardware

Modern embedded controllers use flash memory and similar hardware to store data. These devices have the intrinsic problem that the signal that can be read from each memory cell is reduced over time, mainly influenced by the number of write accesses. In the end, the cell will produce arbitrary values on each read access.

Unfortunately, the distribution of write accesses in typical systems is very uneven. Some parameters might be updated a few times a second, while some code may stay untouched for the whole life time of the ECU. To avoid early read errors, wear leveling should be deployed, such that frequent updates of single data elements are distributed over the whole memory area.

On the other hand, most operating systems include a file system or at least a flash driver that takes care of wear leveling, such that a typical implementation of the `Per-`

sistency will not have to care about the wear leveling. This use case is therefore not described in any detail in this specification.

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 Execution Management.

For the implementation of redundancy and security purposes, the Persistency accesses services of the Adaptive Crypto Interface.

For the installation, update, and deletion of persisted data, the Persistency interacts with the Update and Configuration Management (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_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] [SWS_PER_00351] [SWS_PER_00352] [SWS_PER_00357] [SWS_PER_00358] [SWS_PER_00360] [SWS_PER_00361] [SWS_PER_00363] [SWS_PER_00364] [SWS_PER_00365] [SWS_PER_00368] [SWS_PER_00370] [SWS_PER_00372] [SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377] [SWS_PER_00400] [SWS_PER_00401]
[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_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] [SWS_PER_00350] [SWS_PER_00351] [SWS_PER_00352] [SWS_PER_00355] [SWS_PER_00356] [SWS_PER_00357] [SWS_PER_00358] [SWS_PER_00365] [SWS_PER_00367]

		[SWS_PER_00368] [SWS_PER_00369] [SWS_PER_00370] [SWS_PER_00371] [SWS_PER_00372] [SWS_PER_00373] [SWS_PER_00374] [SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377]
[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_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] [SWS_PER_00350] [SWS_PER_00351] [SWS_PER_00355] [SWS_PER_00356] [SWS_PER_00367] [SWS_PER_00368] [SWS_PER_00369] [SWS_PER_00370] [SWS_PER_00371] [SWS_PER_00372] [SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377]
[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] [SWS_PER_00354] [SWS_PER_00359] [SWS_PER_00362]
[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_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] [SWS_PER_00354] [SWS_PER_00356] [SWS_PER_00357] [SWS_PER_00358] [SWS_PER_00365] [SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377]

[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_00332] [SWS_PER_00333] [SWS_PER_00334] [SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00337] [SWS_PER_00338] [SWS_PER_00357] [SWS_PER_00358] [SWS_PER_00365] [SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377]
[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_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] [SWS_PER_00360] [SWS_PER_00361] [SWS_PER_00363] [SWS_PER_00364] [SWS_PER_00365] [SWS_PER_00367] [SWS_PER_00369] [SWS_PER_00371] [SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377] [SWS_PER_00400] [SWS_PER_00401]
[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_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] [SWS_PER_00351] [SWS_PER_00352] [SWS_PER_00355] [SWS_PER_00356] [SWS_PER_00357] [SWS_PER_00358] [SWS_PER_00360] [SWS_PER_00361] [SWS_PER_00363] [SWS_PER_00364] [SWS_PER_00365] [SWS_PER_00367] [SWS_PER_00368] [SWS_PER_00369] [SWS_PER_00370] [SWS_PER_00371] [SWS_PER_00372] [SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377] [SWS_PER_00400] [SWS_PER_00401]

[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_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] [SWS_PER_00349] [SWS_PER_00353] [SWS_PER_00360] [SWS_PER_00361] [SWS_PER_00363] [SWS_PER_00364] [SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377] [SWS_PER_00400] [SWS_PER_00401]

[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] [SWS_PER_00359] [SWS_PER_00360] [SWS_PER_00361] [SWS_PER_00362] [SWS_PER_00363] [SWS_PER_00364] [SWS_PER_00365] [SWS_PER_00371] [SWS_PER_00372] [SWS_PER_00373] [SWS_PER_00374] [SWS_PER_00400] [SWS_PER_00401]
[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] [SWS_PER_00360] [SWS_PER_00361] [SWS_PER_00363] [SWS_PER_00364] [SWS_PER_00400] [SWS_PER_00401]

[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_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] [SWS_PER_00367] [SWS_PER_00368] [SWS_PER_00369] [SWS_PER_00370] [SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377]
[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_00265] [SWS_PER_00266] [SWS_PER_00267] [SWS_PER_00275] [SWS_PER_00277] [SWS_PER_00281] [SWS_PER_00283] [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] [SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377] [SWS_PER_00378] [SWS_PER_00379] [SWS_PER_00380] [SWS_PER_00381] [SWS_PER_00382] [SWS_PER_00383] [SWS_PER_00384] [SWS_PER_00385] [SWS_PER_00386] [SWS_PER_00387] [SWS_PER_00388] [SWS_PER_00389]

		[SWS_PER_00390] [SWS_PER_00391] [SWS_PER_00392] [SWS_PER_00393] [SWS_PER_00394] [SWS_PER_00395] [SWS_PER_CONSTR_00001] [SWS_PER_CONSTR_00002] [SWS_PER_CONSTR_00003] [SWS_PER_CONSTR_00004]
[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_00265] [SWS_PER_00266] [SWS_PER_00267] [SWS_PER_00379] [SWS_PER_00380] [SWS_PER_00381] [SWS_PER_00382] [SWS_PER_00383] [SWS_PER_00384] [SWS_PER_00385] [SWS_PER_CONSTR_00001] [SWS_PER_CONSTR_00002] [SWS_PER_CONSTR_00003] [SWS_PER_CONSTR_00004]
[RS_PER_00013]	Persistency shall support update of persistent data	[SWS_PER_00251] [SWS_PER_00275] [SWS_PER_00277] [SWS_PER_00281] [SWS_PER_00283] [SWS_PER_00356] [SWS_PER_00357] [SWS_PER_00378] [SWS_PER_00379] [SWS_PER_00380] [SWS_PER_00381] [SWS_PER_00386] [SWS_PER_00387] [SWS_PER_00388] [SWS_PER_00389] [SWS_PER_00390] [SWS_PER_00391] [SWS_PER_00392] [SWS_PER_00393] [SWS_PER_00394] [SWS_PER_00395]
[RS_PER_00014]	Persistency shall support roll-back of persistent data	[SWS_PER_00378] [SWS_PER_00396]
[RS_PER_00015]	Persistency shall support removal of persistent data	[SWS_PER_00358] [SWS_PER_00397]

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 Storages and multiple File Storages.

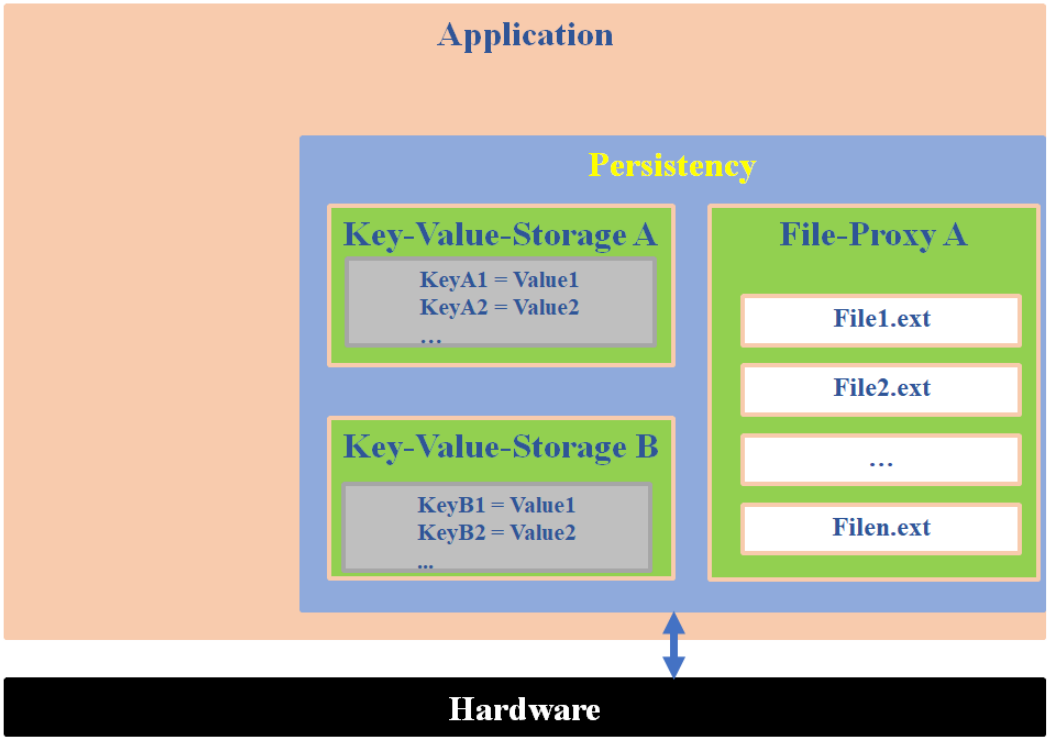

Figure 7.1: Typical usage of Persistency within an Adaptive Application

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 Storages for every AdaptiveApplicationSwComponentType. Every Key-Value Storage is represented by a PortPrototype typed by a PersistencyKeyValueDatabaseInterface in the application design for the respective AdaptiveApplicationSwComponentType. Every Key-Value Storage can hold multiple Key-Value Pairs.

A Key-Value Storage 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 software package of the Adaptive Application. See section 7.4.

File Storages offer access to a set of files, they are similar to a directory of a file system. Every File Storage is represented by a PortPrototype typed by a PersistencyFileProxyInterface in the application design for the respective AdaptiveApplicationSwComponentType. Every File Storage 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 8.2.6.9 and 8.2.6.11).

A File Storage 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 software package of the Adaptive Application. See section 7.4.

The API specification holds classes for Key-Value Storage and File Storage access with appropriate creator functions. These receive the identifier (the fully qualified shortName path) of a PortPrototype typed by a PersistencyKeyValueDatabaseInterface or a PersistencyFileProxyInterface as an ara::core::InstanceSpecifier input parameter (see 8.1.1 and 8.2.1). Depending on the nature of the PortPrototype, the Key-Value Storage or File Storage 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]{DRAFT} ⌈ 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 Storage are currently not modeled in [2].

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

[SWS_PER_00211]{DRAFT} ⌈ 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]{DRAFT} ⌈ The Persistency cluster shall store redundant information for every Key-Value Storage and every File Storage 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]{DRAFT} ⌈ The Persistency cluster shall use the redundant information to detect data corruption in the persistent memory. ⌋*(RS_PER_00008)*

[SWS_PER_00222]{DRAFT} ⌈ 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]{DRAFT} ⌈ 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 Storage or a file in the File Storage, and shall use this CRC to check the Key-Value Storage or the file in the File Storage when it is read back. ⌋*(RS_PER_00008, RS_PER_00009, RS_PER_00010)*

[SWS_PER_00319]{DRAFT} ⌈ In case a PersistencyRedundancyHandling aggregated as PersistencyDeployment.redundancyHandling is derived as PersistencyRedundancyMOutOfN, the Persistency cluster shall store N copies when persisting the Key-Value Storage or a file in the File Storage, and shall check that at least M of the N copies of the Key-Value Storage or the file in the File Storage 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 Installation and Update of Persistent Data

The Update and Configuration Management handles the life cycle of Adaptive Applications with the following phases:

- Installation of new software
- Update of already installed software

- Finalization of updated software after the update succeeded
- Roll-back of updated software after the update failed
- Removal of installed software

For all these phases, persistent data needs to be handled alongside the application. The Adaptive Application may trigger this handling explicitly by calling UpdatePersistency during the verification phase that follows the installation or update, or rely on the Persistency cluster to do this implicitly when persistent data is accessed (OpenKeyValueStorage/OpenFileStorage). In both cases, the Persistency cluster will compare the stored manifest version against the current manifest version, and perform the required action.

[SWS_PER_00378]{DRAFT} ⌈ Persistency shall store the Executable.version and the SoftwareCluster.version of the manifest persistently. ⌋*(RS_PER_00010, RS_PER_00013, RS_PER_00014)*

The Executable.version is used by Persistency to detect a change of the application (see [SWS_PER_00387]), while the SoftwareCluster.version is used to detect a change of the deployed persistent data (see [SWS_PER_00386] and [SWS_PER_00396]).

[SWS_PER_CONSTR_00001]{DRAFT} ⌈ When the Executable.version is increased, the SoftwareCluster.version needs to be increased, too. ⌋*(RS_PER_00010, RS_PER_00012)*

The SoftwareCluster.version and Executable.version are StrongRevisionLabelStrings. These strings consists of a MajorVersion, a MinorVersion, a PatchVersion, and a BuildVersion. It is assumed that the first three will be incremented when the version is changed, while the last might be arbitrary.

[SWS_PER_CONSTR_00002]{DRAFT} ⌈ When the SoftwareCluster.version or Executable.version is increased, the MajorVersion, MinorVersion, or PatchVersion have to be incremented. ⌋*(RS_PER_00010, RS_PER_00012)*

After installation of the Adaptive Application, the Persistency cluster will install pre-defined persistent data from the manifest. There are different possibilities how this persistent data can be defined in the manifest:

- Persistent data can be defined by an application designer within PersistencyKeyValueDatabaseInterface or PersistencyFileProxyInterface.
- Persistent data that was defined by an application designer can be changed by an integrator within PersistencyKeyValueDatabase or PersistencyFileArray.
- Persistent data can be directly defined by an integrator within PersistencyKeyValueDatabase or PersistencyFileArray.

[SWS_PER_00379]{DRAFT} ⌈ Elements defined in the deployment data (PersistencyKeyValueDatabase and PersistencyFileArray and associated classes) shall always be preferred over elements defined in the application design (PersistencyKeyValueDatabaseInterface and PersistencyFileProxyInterface and associated classes). The latter shall only be used if the former does not exist. ⌋ *(RS_PER_00010, RS_PER_00012, RS_PER_00013)*

Please note that the manifest contains separate deployment data for each Process that references the Executable. The Process is bound to the deployment data by a mapping class. In case of a Key-Value Storage, the PersistencyKeyValueDatabase is mapped by PersistencyPortPrototypeToKeyValueDatabaseMapping to a Process and a PortPrototype typed by a PersistencyKeyValueDatabaseInterface. In case of a File Storage, the PersistencyFileArray is mapped by a PersistencyPortPrototypeToFileArrayMapping to a Process and a PortPrototype typed by a PersistencyFileProxyInterface.

After an update of the Adaptive Application or the manifest, the Persistency cluster will create a backup of the persistent data, and then update the existing persistent data using one of the following strategies:

- Existing persistent data is kept unchanged (keepExisting).
- Existing persistent data is replaced (overwrite).
- Existing persistent data is removed (delete).
- New persistent data is added (keepExisting and overwrite).

The update strategy can be set during application design or deployment, and can be defined for the whole Key-Value Storage or File Storage (PersistencyCollectionLevelUpdateStrategyEnum – keepExisting or delete) and for a single key or file (PersistencyElementLevelUpdateStrategyEnum – keepExisting, overwrite, or delete).

[SWS_PER_00251]{DRAFT} ⌈ An update strategy defined in the deployment data (PersistencyDeployment.updateStrategy, PersistencyKeyValuePair.updateStrategy, PersistencyFile.updateStrategy) shall always be preferred over the update strategy defined in the application design (PersistencyInterface.updateStrategy, PersistencyDataElement.updateStrategy, PersistencyFileProxy.updateStrategy). The latter shall only be used if the former does not exist. ⌋*(RS_PER_00010, RS_PER_00012, RS_PER_00013)*

[SWS_PER_00380]{DRAFT} ⌈ An update strategy defined for a single key (PersistencyKeyValuePair.updateStrategy, PersistencyDataElement.updateStrategy) shall always be preferred over the update strategy defined for the enclosing Key-Value Storage (PersistencyDeployment.updateStrategy, PersistencyInterface.updateStrategy). The latter shall only be used if the former does not exist. ⌋*(RS_PER_00010, RS_PER_00012, RS_PER_00013)*

[SWS_PER_00381]{DRAFT} ⌈ An update strategy defined for a single file (PersistencyFile.updateStrategy, PersistencyFileProxy.updateStrategy) shall always be preferred over the update strategy defined for the enclosing File Storage (PersistencyDeployment.updateStrategy, PersistencyInterface.updateStrategy). The latter shall only be used if the former does not exist. ⌋*(RS_PER_00010, RS_PER_00012, RS_PER_00013)*

When the update succeeded, the Update and Configuration Management will finalize the new Adaptive Application. The Persistency cluster is not required to do anything, though it could free the resources allocated by the last backup.

When the update failed, the Update and Configuration Management will revert to the old Adaptive Application and/or manifest. The Persistency cluster will then replace the currently used persistent data by the backup created during the update.

Finally, to remove persistent data before the Adaptive Application is removed, the Adaptive Application needs to call ResetPersistency.

7.4.1 Installation of Persistent Data

[SWS_PER_00382]{DRAFT} ⌈ When a Key-Value Storage or File Storage is opened by the application using OpenKeyValueStorage or OpenFileStorage, or when UpdatePersistency is called, the Persistency shall check for the existence of stored data. If no persistent data is found, the Persistency shall initialize the persistent data. ⌋*(RS_PER_00010, RS_PER_00012)*

Initialization of persistent data is described in sections 7.4.1.1 and 7.4.1.2.

7.4.1.1 Installation of Key-Value Storage

[SWS_PER_00383]{DRAFT} ⌈ Persistency shall create a Key-Value Storage for each PortPrototype typed by a PersistencyKeyValueDatabaseInterface that is found in the manifest of a newly installed Adaptive Application. The Key-Value Storage shall be identified at run-time by the shortName path of the PortPrototype, passed as InstanceSpecifier to OpenKeyValueStorage. ⌋*(RS_PER_00010, RS_PER_00012)*

[SWS_PER_00252]{DRAFT} ⌈ Persistency shall create an entry in the Key-Value Storage for each PersistencyKeyValueDatabaseInterface.dataElement and PersistencyKeyValueDatabase.keyValuePair that is found in the manifest of a newly installed or updated Adaptive Application, and for which the update strategy is keepExisting or overwrite. ⌋ *(RS_PER_00010, RS_PER_00012)*

Key-Value Storage entries are identified by the key. An entry with identical key might be defined both in the PersistencyKeyValueDatabaseInterface

and the PersistencyKeyValueDatabase, in which case [SWS_PER_00379] applies. The update strategy is determined according to [SWS_PER_00251] and [SWS_PER_00380].

[SWS_PER_00253]{DRAFT} ⌈ Entries in the Key-Value Storage shall use the shortName of the PersistencyDataElement and/or PersistencyKeyValuePair as key. ⌋*(RS_PER_00010, RS_PER_00012)*

[SWS_PER_00254]{DRAFT} ⌈ Entries in the Key-Value Storage shall be created with the data type defined by the CppImplementationDataType which types the PersistencyDataElement and/or by the CppImplementationDataType referenced as PersistencyKeyValuePair.valueDataType. ⌋*(RS_PER_00010, RS_PER_00012)*

[SWS_PER_00384]{DRAFT} ⌈ Entries in the Key-Value Storage shall be created with the value taken from the PersistencyKeyValuePair.initValue or, if that does not exist, from the PersistencyDataRequiredComSpec.initValue. ⌋*(RS_PER_00010, RS_PER_00012)*

[SWS_PER_CONSTR_00003]{DRAFT} ⌈ A manifest is not valid if the value or data type of any PersistencyKeyValuePair or PersistencyDataElement cannot be determined, or if the determined data types are conflicting. ⌋*(RS_PER_00010, RS_PER_00012)*

Invalid manifests should be rejected by the tooling.

7.4.1.2 Installation of File Storage

[SWS_PER_00385]{DRAFT} ⌈ Persistency shall create a File Storage for each PortPrototype typed by a PersistencyFileProxyInterface that is found in the manifest of a newly installed Adaptive Application. The File Storage shall be identified at run-time by the shortName path of the PortPrototype, passed as InstanceSpecifier to OpenFileStorage. ⌋*(RS_PER_00010, RS_PER_00012)*

[SWS_PER_00265]{DRAFT} ⌈ Persistency shall create a file in the File Storage for each PersistencyFileProxyInterface.fileProxy and PersistencyFileArray.file that is found in the manifest of a newly installed or updated Adaptive Application, and for which the update strategy is keepExisting or overwrite. ⌋*(RS_PER_00010, RS_PER_00012)*

The files within a File Storage are identified by their name. A file with the same name might be defined both in the PersistencyFileProxyInterface and the PersistencyFileArray, in which case [SWS_PER_00379] applies. The update strategy is determined according to [SWS_PER_00251] and [SWS_PER_00381].

[SWS_PER_00266]{DRAFT} ⌈ Files in the File Storage shall use the name identified by PersistencyFileProxy.fileName and/or PersistencyFile.fileName. ⌋*(RS_PER_00010, RS_PER_00012)*

[SWS_PER_00267]{DRAFT} ⌈ Files in the File Storage shall be created with the content taken from the resource (within the installed SoftwarePackage) that is addressed by PersistencyFile.contentUri or, if that does not exist, by PersistencyFileProxy.contentUri. If that does not exist either, and empty file shall be created. ⌋*(RS_PER_00010, RS_PER_00012)*

[SWS_PER_CONSTR_00004]{DRAFT} ⌈ A manifest is invalid if the shortNames of a PersistencyFileProxy and a PersistencyFile with the same file name differs. ⌋*(RS_PER_00010, RS_PER_00012)*

Invalid manifests should be rejected by the tooling.

7.4.2 Update of Persistent Data

[SWS_PER_00386]{DRAFT} ⌈ When a Key-Value Storage or File Storage is opened by the application using OpenKeyValueStorage or OpenFileStorage, or when UpdatePersistency is called, the Persistency shall compare the SoftwareCluster.version in the manifest against the stored version. If the version in the manifest is higher than the stored version, the Persistency shall first create a backup of the persistent data and then update the data. ⌋*(RS_PER_00010, RS_PER_00013)*

Only one set of backup data needs to be kept at any time. When a new update is performed, old backup data could be overwritten. Update of persistent data is described in sections 7.4.2.1 and 7.4.2.2.

[SWS_PER_00387]{DRAFT} ⌈ When a Key-Value Storage or File Storage is opened by the application using OpenKeyValueStorage or OpenFileStorage, or when UpdatePersistency is called, the Persistency shall compare the Executable.version in the manifest against the stored version. If the version in the manifest is higher than the stored version, the Persistency shall call the function registered by the application using RegisterApplicationDataUpdateCallback for each Key-Value Storage and File Storage that was updated according to [SWS_PER_00386]. ⌋*(RS_PER_00010, RS_PER_00013)*

7.4.2.1 Update of Key-Value Storage

[SWS_PER_00388]{DRAFT} ⌈ When a new PortPrototype typed by a PersistencyKeyValueDatabaseInterface is detected in an updated manifest, the Persistency shall create a Key-Value Storage as specified in [SWS_PER_00383]. ⌋*(RS_PER_00010, RS_PER_00013)*

[SWS_PER_00389]{DRAFT} ⌈ When a PortPrototype typed by a PersistencyKeyValueDatabaseInterface is missing in an updated manifest, the Persistency shall remove the corresponding Key-Value Storage. ⌋*(RS_PER_00010, RS_PER_00013)*

[SWS_PER_00390]{DRAFT} ⌈ When a PersistencyKeyValueDatabaseInterface.dataElement and/or a PersistencyKeyValueDatabase.keyValuePair with a new key is detected in an updated manifest, the Persistency shall create a new entry in the Key-Value Storage as specified in [SWS_PER_00252], [SWS_PER_00253], [SWS_PER_00254], and [SWS_PER_00384]. ⌋*(RS_PER_00010, RS_PER_00013)*

[SWS_PER_00391]{DRAFT} ⌈ When an existing key of a Key-Value Storage cannot be associated with any PersistencyKeyValueDatabaseInterface.dataElement or PersistencyKeyValueDatabase.keyValuePair in an updated manifest, and the update strategy of the PersistencyKeyValueDatabase or PersistencyKeyValueDatabaseInterface corresponding to the Key-Value Storage is delete, the Persistency shall remove the entry for that key from the Key-Value Storage. ⌋*(RS_PER_00010, RS_PER_00013)*

The update strategy is determined according to [SWS_PER_00251].

[SWS_PER_00275]{DRAFT} ⌈ When an existing key of a Key-Value Storage can be associated with a PersistencyKeyValueDatabaseInterface.dataElement or PersistencyKeyValueDatabase.keyValuePair in an updated manifest, and the update strategy is overwrite, the Persistency shall replace the entry in the Key-Value Storage with the new type and value as specified in [SWS_PER_00254] and [SWS_PER_00384]. ⌋*(RS_PER_00010, RS_PER_00013)*

An entry with identical key might be defined both in the PersistencyKeyValueDatabaseInterface and the PersistencyKeyValueDatabase, in which case [SWS_PER_00379] applies. The update strategy is determined according to [SWS_PER_00251] and [SWS_PER_00380].

[SWS_PER_00277]{DRAFT} ⌈ When an existing key of a Key-Value Storage can be associated with a PersistencyKeyValueDatabaseInterface.dataElement or PersistencyKeyValueDatabase.keyValuePair in an updated manifest, and the update strategy is delete, the Persistency shall remove the entry for that key from the Key-Value Storage. ⌋*(RS_PER_00010, RS_PER_00013)*

Updated keys with the update strategy keepExisting will not be touched during an update. Persistency will neither check the value nor the type of the existing entry.

7.4.2.2 Update of File Storage

[SWS_PER_00392]{DRAFT} ⌈ When a new PortPrototype typed by a PersistencyFileProxyInterface is detected in an updated manifest, the Persistency shall create a File Storage as specified in [SWS_PER_00385]. ⌋*(RS_PER_00010, RS_PER_00013)*

[SWS_PER_00393]{DRAFT} ⌈ When a PortPrototype typed by a PersistencyFileProxyInterface is missing in an updated manifest, the Persistency shall remove the corresponding File Storage. ⌋*(RS_PER_00010, RS_PER_00013)*

[SWS_PER_00394]{DRAFT} ⌈ When a PersistencyFileProxyInterface.fileProxy and/or PersistencyFileArray.file with a new file name is detected in an updated manifest, the Persistency shall create a new file in the File Storage as specified in [SWS_PER_00265], [SWS_PER_00266], and [SWS_PER_00267]. ⌋*(RS_PER_00010, RS_PER_00013)*

[SWS_PER_00395]{DRAFT} ⌈ When an existing file of a File Storage cannot be associated with any PersistencyFileProxyInterface.fileProxy or PersistencyFileArray.file in an updated manifest, and the update strategy of the PersistencyFileArray or PersistencyFileProxyInterface corresponding to the File Storage is delete, the Persistency shall remove the file from the File Storage. ⌋*(RS_PER_00010, RS_PER_00013)*

The update strategy is determined according to [SWS_PER_00251].

[SWS_PER_00281]{DRAFT} ⌈ When an existing file of a File Storage can be associated with a PersistencyFileProxyInterface.fileProxy or PersistencyFileArray.file in an updated manifest, and the update strategy is overwrite, the Persistency shall replace the content of the file in the File Storage with the new content as specified in [SWS_PER_00267]. ⌋*(RS_PER_00010, RS_PER_00013)*

A file with the same name might be defined both in the PersistencyFileProxyInterface and the PersistencyFileArray, in which case [SWS_PER_00379] applies. The update strategy is determined according to [SWS_PER_00251] and [SWS_PER_00381].

[SWS_PER_00283]{DRAFT} ⌈ When an existing file of a File Storage can be associated with a PersistencyFileProxyInterface.fileProxy or PersistencyFileArray.file in an updated manifest, and the update strategy is delete, the Persistency shall remove the file from the File Storage. ⌋*(RS_PER_00010, RS_PER_00013)*

Updated files with the update strategy keepExisting will not be touched during an update. Persistency will not check the content of the existing file.

7.4.3 Roll-Back of Persistent Data after Failed Update

[SWS_PER_00396]{DRAFT} ⌈ When a Key-Value Storage or File Storage is opened by the application using OpenKeyValueStorage or OpenFileStorage, or when UpdatePersistency is called, the Persistency shall compare the SoftwareCluster.version in the manifest against the stored version. If the version in the manifest is lower than the stored version, the Persistency shall compare the version in the manifest against the version stored in backup data. If the versions match, the Persistency shall restore the backup. Otherwise, it shall remove all Key-Value Storages and File Storages, and re-install the lost persistent data. ⌋*(RS_PER_00014)*

Initialization of persistent data is described in section 7.4.1.

7.4.4 Removal of Persistent Data

[SWS_PER_00397]{DRAFT} ⌈ When ResetPersistency is called, the Persistency shall remove all Key-Value Storages and File Storages. ⌋ *(RS_PER_00015)*

7.5 Supported data types in Key-Value Storage

The Persistency cluster supports the following classes of data types in the functions for getting and setting the values of a Key-Value Storage. See sections 8.1.4.6 and 8.1.4.7.

[SWS_PER_00302]{DRAFT} ⌈ The Persistency cluster shall be able to store all data types described in [7] in a Key-Value Storage. ⌋*(RS_PER_00001)*

[SWS_PER_00303]{DRAFT} ⌈ The Persistency cluster shall be able to store serialized binary data in a Key-Value Storage. ⌋*(RS_PER_00001)*

This allows the application to store custom data types.

[SWS_PER_00304]{DRAFT} ⌈ The Persistency cluster shall be able to store all CppImplementationDataTypes referred via PersistencyKeyValueDatabaseInterface.dataTypeForSerialization or via PersistencyKeyValueDatabaseInterface.dataElement in the application design of a PersistencyKeyValueDatabase in the corresponding Key-Value Storage. 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 Storage or a File Storage.

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]{DRAFT} ⌈ The Persistency cluster shall ensure that the space configured by PersistencyDeployment.minimumSustainedSize is always available for the Key-Value Storage or File Storage. ⌋*(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]{DRAFT} ⌈ The Persistency cluster shall ensure that the space actually allocated by a Key-Value Storage or File Storage 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 paradigm in the mind:

- 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 File Storages and Key-Value Storage are completely separate, and therefore divided into separate sections.

[SWS_PER_00002]{DRAFT} ⌈ 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, to recover as much as possible after it was corrupted, and to reset it to the deployed defaults.

8.1.1 OpenKeyValueStorage

[SWS_PER_00052]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::OpenKeyValueStorage(ara::core::InstanceSpecifier kvs)	
Scope:	namespace ara::per	
Syntax:	`ara::core::Result<SharedHandle<KeyValueStorage> > OpenKeyValueStorage (ara::core::InstanceSpecifier kvs) noexcept;`	
Parameters (in):	kvs	The shortName path of a PortPrototype typed by a PersistencyKeyValueDatabaseInterface.
Return value:	ara::core::Result< SharedHandle< KeyValueStorage > >	A Result, containing a SharedHandle, or one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Opens a key-value storage.	

⌋*(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)*

8.1.2 RecoverKeyValueStorage

[SWS_PER_00333]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::RecoverKeyValueStorage(ara::core::InstanceSpecifier kvs)	
Scope:	namespace ara::per	
Syntax:	`ara::core::Result<void> RecoverKeyValueStorage (ara::core::InstanceSpecifier kvs) noexcept;`	
Parameters (in):	kvs	The shortName path 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:	re-entrant	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Recover an instance of KeyValueStorage. This method allows to recover a key-value storage when the redundancy checks fail. It will fail with a kResourceBusyError when the key-value storage 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.	

⌋(*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*)

8.1.3 ResetKeyValueStorage

[SWS_PER_00334]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::ResetKeyValueStorage(ara::core::InstanceSpecifier kvs)	
Scope:	namespace ara::per	
Syntax:	`ara::core::Result<void> ResetKeyValueStorage (ara::core::InstanceSpecifier kvs) noexcept;`	
Parameters (in):	kvs	The shortName path 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:	re-entrant	
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 key-value storage 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 key-value storage is currently open.	

⌋(*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*)

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]{DRAFT} ⌈ Operations that modify a `Key-Value Storage` 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 `Key-Value Storage` is just destructed (also implicitly when the `Process` terminates), the `Key-Value Storage` is not updated, and the next time the `Key-Value Storage` is accessed, the application will see the last saved state. The last saved state can also be restored using `DiscardPendingChanges`.

Please note: Threads that access a KVS in parallel need to be aware that changes done by other threads will become visible immediately, and that the effect of `SyncToStorage` and `DiscardPendingChanges` affects all threads.

[SWS_PER_00339]{DRAFT} ⌈

Kind:	class
Symbol:	ara::per::KeyValueStorage
Scope:	namespace ara::per
Syntax:	`class KeyValueStorage {...};`
Header file:	#include "ara/per/key_value_storage.h"
Description:	The key-value storage contains a set of keys with associated values. .

⌋ *(RS_PER_00002, RS_AP_00122)*

8.1.4.1 KeyValueStorage::KeyValueStorage

[SWS_PER_00322]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::KeyValueStorage::KeyValueStorage(KeyValueStorage &&kvs)	
Scope:	class ara::per::KeyValueStorage	
Syntax:	`KeyValueStorage (KeyValueStorage &&kvs) noexcept;`	
Parameters (in):	kvs	The KeyValueStorage object to be moved.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Move constructor for KeyValueStorage.	

⌋ *(RS_PER_00002, RS_AP_00120, RS_AP_00121, RS_AP_00129, RS_AP_00132)*

[SWS_PER_00324]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::KeyValueStorage::KeyValueStorage(const KeyValueStorage &)
Scope:	class ara::per::KeyValueStorage
Syntax:	`KeyValueStorage (const KeyValueStorage &)=delete;`
Header file:	#include "ara/per/key_value_storage.h"
Description:	The copy constructor for KeyValueStorage shall not be used.

⌋(*RS_PER_00002*, *RS_AP_00120*)

8.1.4.2 KeyValueStorage::operator=

[SWS_PER_00323]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::KeyValueStorage::operator=(KeyValueStorage &&kvs)	
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:	re-entrant	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Move assignment operator for KeyValueStorage.	

⌋(*RS_PER_00002*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

[SWS_PER_00325]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::KeyValueStorage::operator=(const KeyValueStorage &)
Scope:	class ara::per::KeyValueStorage
Syntax:	`KeyValueStorage& operator= (const KeyValueStorage &)=delete;`
Header file:	#include "ara/per/key_value_storage.h"
Description:	The copy assignment operator for KeyValueStorage shall not be used.

⌋(*RS_PER_00002*, *RS_AP_00119*, *RS_AP_00120*)

8.1.4.3 KeyValueStorage::~KeyValueStorage

[SWS_PER_00050]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::KeyValueStorage::~KeyValueStorage()
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.

⌋(*RS_PER_00002*, *RS_AP_00120*, *RS_AP_00129*, *RS_AP_00132*, *RS_AP_00134*)

8.1.4.4 KeyValueStorage::GetAllKeys

[SWS_PER_00042]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::KeyValueStorage::GetAllKeys()	
Scope:	class ara::per::KeyValueStorage	
Syntax:	`ara::core::Result<ara::core::Vector<ara::core::String> > GetAllKeys () const noexcept;`	
Return value:	ara::core::Result< ara::core::Vector< ara::core::String > >	A Result, containing a list of available keys, or one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Returns a list of all currently available keys of the KeyValueStorage.	

⌋(*RS_PER_00003*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00127*, *RS_AP_00129*, *RS_AP_00132*)

8.1.4.5 KeyValueStorage::HasKey

[SWS_PER_00043]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::KeyValueStorage::HasKey(ara::core::StringView key)	
Scope:	class ara::per::KeyValueStorage	
Syntax:	`ara::core::Result<bool> HasKey (ara::core::StringView key) const noexcept;`	
Parameters (in):	key	The key that shall be checked.

▽

△

Return value:	ara::core::Result< bool >	A Result, containing true if the key could be located or false if it couldn't, or one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Checks if a key exists in the KeyValueStorage.	

⌋(*RS_PER_00003*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00132*)

8.1.4.6 KeyValueStorage::GetValue

[SWS_PER_00044]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::KeyValueStorage::GetValue(ara::core::StringView key, T &value)	
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:	re-entrant	
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.	

⌋(*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*)

[SWS_PER_00332]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::KeyValueStorage::GetValue(ara::core::StringView key)
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:	re-entrant	
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.	

⌋(*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*)

8.1.4.7 KeyValueStorage::SetValue

[SWS_PER_00046]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::KeyValueStorage::SetValue(ara::core::StringView key, const T &value)	
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:	re-entrant	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Stores a key in the KeyValueStorage. If a value already exists, it is overwritten, independent of the stored data type.	

⌋(*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*)

8.1.4.8 KeyValueStorage::RemoveKey

[SWS_PER_00047]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::KeyValueStorage::RemoveKey(ara::core::StringView key)	
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:	re-entrant	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Removes a key and the associated value from the KeyValueStorage.	

⌋(*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*)

8.1.4.9 KeyValueStorage::RemoveAllKeys

[SWS_PER_00048]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::KeyValueStorage::RemoveAllKeys()	
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:	re-entrant	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Removes all keys and associated values from the KeyValueStorage.	

⌋(*RS_PER_00003*, *RS_PER_00010*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00127*, *RS_AP_00128*, *RS_AP_00129*, *RS_AP_00132*)

8.1.4.10 KeyValueStorage::SyncToStorage

[SWS_PER_00049]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::KeyValueStorage::SyncToStorage()
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:	re-entrant	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Triggers flushing of key-value pairs to the physical storage of the KeyValueStorage.	

⌋(*RS_PER_00002*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00127*, *RS_AP_00128*, *RS_AP_00129*, *RS_AP_00132*)

8.1.4.11 KeyValueStorage::DiscardPendingChanges

[SWS_PER_00365]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::KeyValueStorage::DiscardPendingChanges()	
Scope:	class ara::per::KeyValueStorage	
Syntax:	`ara::core::Result<void> DiscardPendingChanges () 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:	re-entrant	
Header file:	#include "ara/per/key_value_storage.h"	
Description:	Removes all pending changes to the KeyValueStorage since the last call to SyncToStorage() or since the KeyValueStorage was opened using OpenKeyValueStorage().	

⌋(*RS_PER_00002*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00127*, *RS_AP_00128*, *RS_AP_00129*, *RS_AP_00132*)

8.2 File Storage

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

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

8.2.1 OpenFileStorage

[SWS_PER_00116]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::OpenFileStorage(ara::core::InstanceSpecifier fs)	
Scope:	namespace ara::per	
Syntax:	`ara::core::Result<SharedHandle<FileStorage> > OpenFileStorage (ara::core::InstanceSpecifier fs) noexcept;`	
Parameters (in):	fs	The shortName path of a PortPrototype typed by a PersistencyFileProxyInterface.
Return value:	ara::core::Result< SharedHandle< File Storage > >	A Result, containing a SharedHandle, or one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/file_storage.h"	
Description:	Opens a file storage.	

⌋(*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*)

8.2.2 RecoverAllFiles

[SWS_PER_00335]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::RecoverAllFiles(ara::core::InstanceSpecifier fs)	
Scope:	namespace ara::per	
Syntax:	`ara::core::Result<void> RecoverAllFiles (ara::core::InstanceSpecifier fs) noexcept;`	
Parameters (in):	fs	The shortName path 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:	re-entrant
Header file:	#include "ara/per/file_storage.h"
Description:	Recover the whole file storage, including all files. This method allows to recover a file storage when the redundancy checks fail. It will fail with a k ResourceBusyError when the file storage 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.

⌋(*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*)

8.2.3 ResetAllFiles

[SWS_PER_00336]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::ResetAllFiles(ara::core::InstanceSpecifier fs)	
Scope:	namespace ara::per	
Syntax:	`ara::core::Result<void> ResetAllFiles (ara::core::InstanceSpecifier fs) noexcept;`	
Parameters (in):	fs	The shortName path 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:	re-entrant	
Header file:	#include "ara/per/file_storage.h"	
Description:	Reset the whole file storage, including all files. This method allows to reset a file storage 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 storage is currently open.	

⌋(*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*)

8.2.4 Helper Functions for BasicOperations Class

The following functions can be used by the application when accessing 8.2.6.10, 8.2.6.11, and 8.2.6.9 to combine the values of `BasicOperations::OpenMode`.

8.2.4.1 operator| for BasicOperations::OpenMode

[SWS_PER_00144]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::operator|(BasicOperations::OpenMode const &left, BasicOperations::OpenMode const &right)	
Scope:	namespace ara::per	
Syntax:	`constexpr 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:	re-entrant	
Header file:	#include "ara/per/basic_operations.h"	
Description:	Merges two OpenMode modifiers into one. BasicOperations class.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*)

8.2.4.2 operator& for BasicOperations::OpenMode

[SWS_PER_00145]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::operator&(BasicOperations::OpenMode const &left, BasicOperations::OpenMode const &right)	
Scope:	namespace ara::per	
Syntax:	`constexpr 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:	re-entrant	
Header file:	#include "ara/per/basic_operations.h"	
Description:	Intersects two OpenMode modifiers into one.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*)

8.2.5 Helper Functions for ReadWriteAccessor Class

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

8.2.5.1 endl

[SWS_PER_00127]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::endl(ReadWriteAccessor &rwa)	
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:	re-entrant	
Header file:	#include "ara/per/read_write_accessor.h"	
Description:	Writes a newline to the file and calls flush().	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

8.2.5.2 flush

[SWS_PER_00128]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::flush(ReadWriteAccessor &rwa)	
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:	re-entrant	
Header file:	#include "ara/per/read_write_accessor.h"	
Description:	Calls flush() on the file.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

8.2.6 FileStorage Class

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

[SWS_PER_00340]{DRAFT} ⌈

Kind:	class
Symbol:	ara::per::FileStorage
Scope:	namespace ara::per
Syntax:	`class FileStorage {...};`
Header file:	#include "ara/per/file_storage.h"
Description:	The FileStorage contains a set of files identified by their names.

⌋(*RS_PER_00004*, *RS_AP_00122*)

8.2.6.1 FileStorage::FileStorage

[SWS_PER_00326]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::FileStorage::FileStorage(FileStorage &&fs)	
Scope:	class ara::per::FileStorage	
Syntax:	`FileStorage (FileStorage &&fs) noexcept;`	
Parameters (in):	fs	The FileStorage object to be moved.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/file_storage.h"	
Description:	Move constructor for FileStorage.	

⌋(*RS_PER_00004*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00129*, *RS_AP_00132*)

[SWS_PER_00328]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::FileStorage::FileStorage(const FileStorage &)
Scope:	class ara::per::FileStorage
Syntax:	`FileStorage (const FileStorage &)=delete;`
Header file:	#include "ara/per/file_storage.h"
Description:	The copy constructor for FileStorage shall not be used.

⌋(*RS_PER_00004*, *RS_AP_00120*)

8.2.6.2 FileStorage::operator=

[SWS_PER_00327]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::FileStorage::operator=(FileStorage &&fs)	
Scope:	class ara::per::FileStorage	
Syntax:	`FileStorage& operator= (FileStorage &&fs) noexcept;`	
Parameters (in):	fs	The FileStorage object to be moved.
Return value:	FileStorage &	The moved FileStorage object.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/file_storage.h"	
Description:	Move assignment operator for FileStorage.	

⌋(*RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

[SWS_PER_00329]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::FileStorage::operator=(const FileStorage &)
Scope:	class ara::per::FileStorage
Syntax:	`FileStorage& operator= (const FileStorage &)=delete;`
Header file:	#include "ara/per/file_storage.h"
Description:	The copy assignment operator for FileStorage shall not be used.

⌋(*RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*)

8.2.6.3 FileStorage::~FileStorage

[SWS_PER_00330]{DRAFT} ⌈

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

⌋(*RS_PER_00004*, *RS_AP_00120*, *RS_AP_00129*, *RS_AP_00132*, *RS_AP_00134*)

8.2.6.4 FileStorage::GetAllFileNames

[SWS_PER_00110]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::FileStorage::GetAllFileNames()	
Scope:	class ara::per::FileStorage	
Syntax:	`ara::core::Result<ara::core::Vector<ara::core::String> > GetAllFileNames () const noexcept;`	
Return value:	ara::core::Result< ara::core::Vector< ara::core::String > >	A Result, containing a list of availables files, or one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/file_storage.h"	
Description:	Returns a list of available files within this file storage.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00127*, *RS_AP_00129*, *RS_AP_00132*)

8.2.6.5 FileStorage::DeleteFile

[SWS_PER_00111]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::FileStorage::DeleteFile(ara::core::StringView file)	
Scope:	class ara::per::FileStorage	
Syntax:	`ara::core::Result<void> DeleteFile (ara::core::StringView file) noexcept;`	
Parameters (in):	file	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:	re-entrant	
Header file:	#include "ara/per/file_storage.h"	
Description:	Deletes a file from this file storage. This operation will fail with a kResourceBusyError when the file is currently open.	

⌋(*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*)

8.2.6.6 FileStorage::FileExists

[SWS_PER_00112]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::FileStorage::FileExists(ara::core::StringView file)	
Scope:	class ara::per::FileStorage	
Syntax:	`ara::core::Result<bool> FileExists (ara::core::StringView file) const noexcept;`	
Parameters (in):	file	Identifier of the file.
Return value:	ara::core::Result< bool >	A Result, containing true if the file exists or false if it doesn't, or one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/file_storage.h"	
Description:	Queries if a file is available in this file storage.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00132*)

8.2.6.7 FileStorage::RecoverFile

[SWS_PER_00337]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::FileStorage::RecoverFile(ara::core::StringView fileName)	
Scope:	class ara::per::FileStorage	
Syntax:	`ara::core::Result<void> RecoverFile (ara::core::StringView fileName) noexcept;`	
Parameters (in):	fileName	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:	re-entrant	
Header file:	#include "ara/per/file_storage.h"	
Description:	Recovers a file of this file storage. 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.	

⌋(*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*)

8.2.6.8 FileStorage::ResetFile

[SWS_PER_00338]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::FileStorage::ResetFile(ara::core::StringView fileName)	
Scope:	class ara::per::FileStorage	
Syntax:	`ara::core::Result<void> ResetFile (ara::core::StringView fileName) noexcept;`	
Parameters (in):	fileName	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:	re-entrant	
Header file:	#include "ara/per/file_storage.h"	
Description:	Resets a file of this file storage 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.	

⌋(*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*)

8.2.6.9 FileStorage::OpenFileReadWrite

[SWS_PER_00375]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::FileStorage::OpenFileReadWrite(ara::core::StringView fileName)	
Scope:	class ara::per::FileStorage	
Syntax:	`ara::core::Result<UniqueHandle<ReadWriteAccessor> > OpenFileReadWrite (ara::core::StringView fileName) noexcept;`	
Parameters (in):	fileName	Name of the file. May correspond to the Persistency File.fileName of a configured file.
Return value:	ara::core::Result< UniqueHandle< ReadWriteAccessor > >	A Result, containing a UniqueHandle, or one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/file_storage.h"	
Description:	Opens a file of the file storage for reading and writing. An error that occurs when a new file is created in the file storage (e.g. that PersistencyFileProxyInterface.maxNumberOfFiles is surpassed) is reported using a failbit similarly to std::fstream.	

⌋(*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*)

[SWS_PER_00113]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::FileStorage::OpenFileReadWrite(ara::core::StringView fileName, Basic Operations::OpenMode const mode)	
Scope:	class ara::per::FileStorage	
Syntax:	`ara::core::Result<UniqueHandle<ReadWriteAccessor> > OpenFileReadWrite (ara::core::StringView fileName, BasicOperations::OpenMode const mode) noexcept;`	
Parameters (in):	fileName	Name of the file. May correspond to the Persistency File.fileName of a configured file.
	mode	Mode with which the file shall be opened.
Return value:	ara::core::Result< UniqueHandle< ReadWriteAccessor > >	A Result, containing a UniqueHandle, or one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/file_storage.h"	
Description:	Opens a file of the file storage for reading and writing. An error that occurs when a new file is created in the file storage (e.g. that PersistencyFileProxyInterface.maxNumberOfFiles is surpassed) is reported using a failbit similarly to std::fstream.	

⌋(*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*)

8.2.6.10 FileStorage::OpenFileReadOnly

[SWS_PER_00376]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::FileStorage::OpenFileReadOnly(ara::core::StringView fileName)	
Scope:	class ara::per::FileStorage	
Syntax:	`ara::core::Result<UniqueHandle<ReadAccessor> > OpenFileReadOnly (ara::core::StringView fileName) noexcept;`	
Parameters (in):	fileName	Name of the file. May correspond to the Persistency File.fileName of a configured file.
Return value:	ara::core::Result< UniqueHandle< ReadAccessor > >	A Result, containing a UniqueHandle, or one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/file_storage.h"	
Description:	Opens a file of the file storage for reading.	

⌋(*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*)

[SWS_PER_00114]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::FileStorage::OpenFileReadOnly(ara::core::StringView fileName, Basic Operations::OpenMode const mode)	
Scope:	class ara::per::FileStorage	
Syntax:	`ara::core::Result<UniqueHandle<ReadAccessor> > OpenFileReadOnly (ara::core::StringView fileName, BasicOperations::OpenMode const mode) noexcept;`	
Parameters (in):	fileName	Name of the file. May correspond to the Persistency File.fileName of a configured file.
	mode	Mode with which the file shall be opened.
Return value:	ara::core::Result< UniqueHandle< ReadAccessor > >	A Result, containing a UniqueHandle, or one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/file_storage.h"	
Description:	Opens a file of the file storage for reading.	

⌋(*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*)

8.2.6.11 FileStorage::OpenFileWriteOnly

[SWS_PER_00377]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::FileStorage::OpenFileWriteOnly(ara::core::StringView fileName)	
Scope:	class ara::per::FileStorage	
Syntax:	`ara::core::Result<UniqueHandle<ReadWriteAccessor> > OpenFileWriteOnly (ara::core::StringView fileName) noexcept;`	
Parameters (in):	fileName	Name of the file. May correspond to the Persistency File.fileName of a configured file.
Return value:	ara::core::Result< UniqueHandle< ReadWriteAccessor > >	A Result, containing a UniqueHandle, or one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/file_storage.h"	
Description:	Opens a file of the file storage for writing. An error that occurs when a new file is created in the file storage (e.g. that PersistencyFileProxyInterface.maxNumberOfFiles is surpassed) is reported using a failbit similarly to std::fstream.	

⌋(*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*)

[SWS_PER_00115]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::FileStorage::OpenFileWriteOnly(ara::core::StringView fileName, Basic Operations::OpenMode const mode)	
Scope:	class ara::per::FileStorage	
Syntax:	`ara::core::Result<UniqueHandle<ReadWriteAccessor> > OpenFileWriteOnly (ara::core::StringView fileName, BasicOperations::OpenMode const mode) noexcept;`	
Parameters (in):	fileName	Name of the file. May correspond to the Persistency File.fileName of a configured file.
	mode	Mode with which the file shall be opened.
Return value:	ara::core::Result< UniqueHandle< ReadWriteAccessor > >	A Result, containing a UniqueHandle, or one of the errors defined for Persistency in PerErrc.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/file_storage.h"	
Description:	Opens a file of the file storage for writing. An error that occurs when a new file is created in the file storage (e.g. that PersistencyFileProxyInterface.maxNumberOfFiles is surpassed) is reported using a failbit similarly to std::fstream.	

⌋*(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)*

8.2.7 Char Traits Wrapper

This section shows the types that are used by the classes 8.2.8, 8.2.9, and 8.2.10. They correspond to the `std::char_traits` types of the same name.

[SWS_PER_00366]{DRAFT} ⌈ The types defined in this section shall be at least 16 bits wide, i.e. shall have at least the range $0 \ldots 65535$ for unsigned ([SWS_PER_00180], [SWS_PER_00181]) and $-32768 \ldots 32767$ for signed ([SWS_PER_00182]) types. ⌋*()*

8.2.7.1 int_type

[SWS_PER_00180]{DRAFT} ⌈

Kind:	type alias
Symbol:	ara::per::int_type
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:	Character value read from a file, used in file storage operations. Signed type similar std::char_traits::int_type.

⌋(*RS_PER_00003*, *RS_AP_00122*)

8.2.7.2 pos_type

[SWS_PER_00181]{DRAFT} ⌈

Kind:	type alias
Symbol:	ara::per::pos_type
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 or number of characters, used in file storage operations. Unsigned type similar to std::char_traits::pos_type.

⌋(*RS_PER_00003*, *RS_AP_00122*)

8.2.7.3 off_type

[SWS_PER_00182]{DRAFT} ⌈

Kind:	type alias
Symbol:	ara::per::off_type
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 storage operations. Unsigned type similar to std::char_traits::off_type.

⌋(*RS_PER_00003*, *RS_AP_00122*)

8.2.8 BasicOperations class

This section shows the types and methods defined by the `BasicOperations` class that are used by the classes 8.2.9 and 8.2.10. They correspond roughly to the types and methods provided by `std::iostream`.

[SWS_PER_00341]{DRAFT} ⌈

Kind:	class
Symbol:	ara::per::BasicOperations
Scope:	namespace ara::per
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.

⌋(*RS_PER_00003*, *RS_AP_00122*)

8.2.8.1 BasicOperations::BasicOperations

[SWS_PER_00344]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::BasicOperations::BasicOperations(BasicOperations &&kvs)	
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.	

⌋(*RS_PER_00002*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00129*, *RS_AP_00132*)

[SWS_PER_00346]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::BasicOperations::BasicOperations(const BasicOperations &)
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.

⌋(*RS_PER_00002*, *RS_AP_00120*)

8.2.8.2 BasicOperations::operator=

[SWS_PER_00345]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::BasicOperations::operator=(BasicOperations &&kvs)	
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.	

⌋(*RS_PER_00002*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

[SWS_PER_00347]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::BasicOperations::operator=(const BasicOperations &)
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.

⌋(*RS_PER_00002*, *RS_AP_00119*, *RS_AP_00120*)

8.2.8.3 BasicOperations::~BasicOperations

[SWS_PER_00348]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::BasicOperations::~BasicOperations()
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.

⌋(*RS_PER_00002*, *RS_AP_00120*, *RS_AP_00129*, *RS_AP_00132*, *RS_AP_00134*)

8.2.8.4 BasicOperations::SeekDirection

[SWS_PER_00146]{DRAFT} ⌈

Kind:	enumeration	
Symbol:	ara::per::BasicOperations::SeekDirection	
Scope:	class ara::per::BasicOperations	
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.	

⌋(*RS_PER_00003*, *RS_AP_00122*)

8.2.8.5 BasicOperations::OpenMode

[SWS_PER_00147]{DRAFT} ⌈

Kind:	enumeration	
Symbol:	ara::per::BasicOperations::OpenMode	
Scope:	class ara::per::BasicOperations	
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.
	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.	

⌋(*RS_PER_00003*, *RS_AP_00122*)

8.2.8.6 BasicOperations::tell

[SWS_PER_00162]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::BasicOperations::tell()	
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.

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

8.2.8.7 BasicOperations::seek

[SWS_PER_00163]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::BasicOperations::seek(pos_type const pos)	
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.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

[SWS_PER_00164]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::BasicOperations::seek(off_type const off, SeekDirection const dir)	
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.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

8.2.8.8 BasicOperations::good

[SWS_PER_00106]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::BasicOperations::good()	
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.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

8.2.8.9 BasicOperations::eof

[SWS_PER_00107]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::BasicOperations::eof()	
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.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

8.2.8.10 BasicOperations::fail

[SWS_PER_00108]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::BasicOperations::fail()	
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.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

8.2.8.11 BasicOperations::bad

[SWS_PER_00140]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::BasicOperations::bad()	
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.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

8.2.8.12 BasicOperations::operator!

[SWS_PER_00142]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::BasicOperations::operator!()	
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().

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

8.2.8.13 BasicOperations::operator bool

[SWS_PER_00143]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::BasicOperations::operator bool()	
Scope:	class ara::per::BasicOperations	
Syntax:	`explicit operator bool () 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 operation, functionally equivalent to ara::per::Basic Operations::good().	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

8.2.8.14 BasicOperations::clear

[SWS_PER_00141]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::BasicOperations::clear()
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.

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00120*, *RS_AP_00132*)

8.2.9 ReadAccessor class

This section shows the methods available for a ReadAccessor object obtained from a call to 8.2.6.10, and for the inheriting ReadWriteAccessor object obtained from a call to 8.2.6.11 or 8.2.6.9.

[SWS_PER_00342]{DRAFT} ⌈

Kind:	class
Symbol:	ara::per::ReadAccessor
Scope:	namespace ara::per
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.

⌋(*RS_PER_00004*, *RS_AP_00122*)

8.2.9.1 ReadAccessor::peek

[SWS_PER_00167]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::ReadAccessor::peek()	
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.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

8.2.9.2 ReadAccessor::get

[SWS_PER_00168]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::ReadAccessor::get()	
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.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

8.2.9.3 ReadAccessor::read

[SWS_PER_00165]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::ReadAccessor::read(ara::core::Span< char > s)	
Scope:	class ara::per::ReadAccessor	
Syntax:	`pos_type read (ara::core::Span< char > s) noexcept;`	
Parameters (out):	s	A span of chars where the read characters shall be stored.
Return value:	pos_type	Actual number of characters 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.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00132*)

8.2.9.4 ReadAccessor::getline

[SWS_PER_00119]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::ReadAccessor::getline(ara::core::Span< char > s, char const delim= '\n')
Scope:	class ara::per::ReadAccessor
Syntax:	`pos_type getline (ara::core::Span< char > s, char const delim= '\n') noexcept;`

▽

△

Parameters (in):	delim	The character that is used as delimiter.
Parameters (out):	s	A span of chars where the read line shall be stored.
Return value:	pos_type	Actual number of characters that have been read, including the delimiter.
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.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00129*, *RS_AP_00132*)

8.2.10 ReadWriteAccessor class

This section shows the methods available for a ReadWriteAccessor object obtained from a call to 8.2.6.11 or 8.2.6.9.

[SWS_PER_00343]{DRAFT} ⌈

Kind:	class
Symbol:	ara::per::ReadWriteAccessor
Scope:	namespace ara::per
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 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.

⌋(*RS_PER_00004*, *RS_AP_00122*)

8.2.10.1 ReadWriteAccessor::fsync

[SWS_PER_00122]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::ReadWriteAccessor::fsync()
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.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00128*, *RS_AP_00127*, *RS_AP_00129*, *RS_AP_00132*)

8.2.10.2 ReadWriteAccessor::write

[SWS_PER_00166]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::ReadWriteAccessor::write(ara::core::Span< char > s)	
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.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00132*)

8.2.10.3 ReadWriteAccessor::flush

[SWS_PER_00124]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::ReadWriteAccessor::flush()
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.

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00120*, *RS_AP_00132*)

8.2.10.4 ReadWriteAccessor::operator«

[SWS_PER_00125]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::ReadWriteAccessor::operator<<(ara::core::StringView s)	
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.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00132*)

[SWS_PER_00126]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::ReadWriteAccessor::operator<<(ReadWriteAccessor &(*op)	
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.	

⌋(*RS_PER_00001*, *RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

8.3 Update and Removal of Persistent Data

The `Persistency` cluster allows for updating and resetting/removing all installed `Key-Value Storages` and `File Storages`. And the application may also register a callback function that is called after the update of any `Key-Value Storage` and `File Storage`.

8.3.1 RegisterApplicationDataUpdateCallback

[SWS_PER_00356]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::RegisterApplicationDataUpdateCallback(std::function< void(ara::core::InstanceSpecifier, ara::core::String)	
Scope:	namespace ara::per	
Syntax:	`void RegisterApplicationDataUpdateCallback (std::function< void(ara::core::InstanceSpecifier, ara::core::String)> appDataUpdateCallback) noexcept;`	
Parameters (in):	appDataUpdateCallback	The callback function to be called by Persistency after an update of persistent data took place. The function will be called with the shortName path of an updated key-value storage or file storage, and with the Executable version with which the Persistency was last accessed.
Return value:	None	
Exception Safety:	noexcept	
Thread Safety:	no	
Header file:	#include "ara/per/update.h"	
Description:	Register an application data update callback with persistency. The provided callback function will be called by persistency if an update of stored application data might be necessary. This decision is based on the Executable versions. The version that last accessed Persistency is provided as an argument to the callback, as well as the Instance Specifier referring to the updated key-value storage or file storage. The provided function will be called from the context of UpdatePersistency(), OpenKeyValueStorage(), or OpenFileStorage().	

⌋(*RS_PER_00013*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00127*, *RS_AP_00132*)

8.3.2 UpdatePersistency

[SWS_PER_00357]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::UpdatePersistency()
Scope:	namespace ara::per
Syntax:	`ara::core::Result<void> UpdatePersistency () 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/update.h"	
Description:	Update all persistency file and key-value storages after a new manifest was installed. This method can be used to update the persistent data of the application during verification phase.	

⌋(*RS_PER_00013*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00127*, *RS_AP_00128*, *RS_AP_00132*)

8.3.3 ResetPersistency

[SWS_PER_00358]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::ResetPersistency()	
Scope:	namespace ara::per	
Syntax:	`ara::core::Result<void> ResetPersistency () 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/update.h"	
Description:	Remove all file and key-value storages. This method can be used to restore the initial state or to prepare removal of the application.	

⌋(*RS_PER_00015*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00127*, *RS_AP_00128*, *RS_AP_00132*)

8.4 Handle Classes

This section contains the definition of the handles used in the API of the `Persistency` cluster. The shared handle (section 8.4.1) is used to provide shared access to the `Key-Value Storage` and `File Storage`, while the unique handle (section 8.4.2) is used to provide non-shared access to ReadAccessors and ReadWriteAccessors of the `File Storage`.

8.4.1 SharedHandle Class

[SWS_PER_00362]{DRAFT} ⌈

Kind:	class
Symbol:	ara::per::SharedHandle
Scope:	namespace ara::per
Syntax:	`template <typename T >` `class SharedHandle final {...};`
Template param:	typename T
Header file:	#include "ara/per/shared_handle.h"
Description:	Handle to a file storage or key-value storage. This is returned by the functions OpenFile Storage() and OpenKeyValueStorage() and can be passed between threads as needed. It provides the abstraction that is necessary to allow thead-safe implementation of OpenFile Storage() and OpenKeyValueStorage().

⌋(*RS_PER_00002*, *RS_AP_00122*)

8.4.1.1 SharedHandle::SharedHandle

[SWS_PER_00367]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::SharedHandle::SharedHandle(SharedHandle &&fsh)	
Scope:	class ara::per::SharedHandle	
Syntax:	`ara::per::SharedHandle< T >::SharedHandle (SharedHandle &&fsh) noexcept;`	
Parameters (in):	fsh	The SharedHandle object to be moved.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/shared_handle.h"	
Description:	Move constructor for SharedHandle.	

⌋(*RS_PER_00004*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00129*, *RS_AP_00132*)

[SWS_PER_00369]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::SharedHandle::SharedHandle(SharedHandle const &fsh)	
Scope:	class ara::per::SharedHandle	
Syntax:	`ara::per::SharedHandle< T >::SharedHandle (SharedHandle const &fsh) noexcept;`	
Parameters (in):	fsh	The SharedHandle object to be moved.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/shared_handle.h"	
Description:	Copy constructor for SharedHandle.	

⌋(*RS_PER_00004*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00129*, *RS_AP_00132*)

8.4.1.2 SharedHandle::operator=

[SWS_PER_00368]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::SharedHandle::operator=(SharedHandle &&fsh)	
Scope:	class ara::per::SharedHandle	
Syntax:	`SharedHandle& ara::per::SharedHandle< T >::operator= (SharedHandle &&fsh) noexcept;`	
Parameters (in):	fsh	The SharedHandle object to be moved.
Return value:	SharedHandle &	The moved SharedHandle object.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/shared_handle.h"	
Description:	Move assignment operator for SharedHandle.	

⌋(*RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

[SWS_PER_00370]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::SharedHandle::operator=(SharedHandle const &fsh)	
Scope:	class ara::per::SharedHandle	
Syntax:	`SharedHandle& ara::per::SharedHandle< T >::operator= (SharedHandle const &fsh) noexcept;`	
Parameters (in):	fsh	The SharedHandle object to be moved.
Return value:	SharedHandle &	The moved SharedHandle object.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/shared_handle.h"	
Description:	Copy assignment operator for SharedHandle.	

⌋(*RS_PER_00004*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

8.4.1.3 SharedHandle::Operator->

[SWS_PER_00363]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::SharedHandle::operator->()	
Scope:	class ara::per::SharedHandle	
Syntax:	`T* ara::per::SharedHandle< T >::operator-> () noexcept;`	
Return value:	T *	–
Exception Safety:	noexcept	

▽

△

Thread Safety:	re-entrant
Header file:	#include "ara/per/shared_handle.h"
Description:	Non-constant arrow operator.

⌋(*RS_PER_00001*, *RS_PER_00002*, *RS_PER_00003*, *RS_AP_00119*, *RS_AP_00129*, *RS_AP_00132*)

[SWS_PER_00364]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::SharedHandle::operator->()
Scope:	class ara::per::SharedHandle
Syntax:	`T const* ara::per::SharedHandle< T >::operator-> () const noexcept;`
Return value:	T const * | –
Exception Safety:	noexcept
Thread Safety:	re-entrant
Header file:	#include "ara/per/shared_handle.h"
Description:	Constant arrow operator.

⌋(*RS_PER_00001*, *RS_PER_00002*, *RS_PER_00003*, *RS_AP_00119*, *RS_AP_00129*, *RS_AP_00132*)

8.4.2 UniqueHandle Class

[SWS_PER_00359]{DRAFT} ⌈

Kind:	class
Symbol:	ara::per::UniqueHandle
Scope:	namespace ara::per
Syntax:	`template <typename T >` `class UniqueHandle final {...};`
Template param:	typename T
Header file:	#include "ara/per/unique_handle.h"
Description:	Handle to a ReadAccessor or ReadWriteAccessor. This is returned by the functions OpenFile ReadOnly(), OpenFileReadWrite(), and OpenFileReadWrite().

⌋(*RS_PER_00002*, *RS_AP_00122*)

8.4.2.1 UniqueHandle::UniqueHandle

[SWS_PER_00371]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::UniqueHandle::UniqueHandle(UniqueHandle &&kvsh)	
Scope:	class ara::per::UniqueHandle	
Syntax:	`ara::per::UniqueHandle< T >::UniqueHandle (UniqueHandle &&kvsh) noexcept;`	
Parameters (in):	kvsh	The UniqueHandle object to be moved.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/unique_handle.h"	
Description:	Move constructor for UniqueHandle.	

⌋(*RS_PER_00002*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00129*, *RS_AP_00132*)

[SWS_PER_00373]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::UniqueHandle::UniqueHandle(UniqueHandle const &)
Scope:	class ara::per::UniqueHandle
Syntax:	`ara::per::UniqueHandle< T >::UniqueHandle (UniqueHandle const &)=delete;`
Header file:	#include "ara/per/unique_handle.h"
Description:	The copy constructor for UniqueHandle shall not be used.

⌋(*RS_PER_00002*, *RS_AP_00120*)

8.4.2.2 UniqueHandle::operator=

[SWS_PER_00372]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::UniqueHandle::operator=(UniqueHandle &&kvsh)	
Scope:	class ara::per::UniqueHandle	
Syntax:	`UniqueHandle& ara::per::UniqueHandle< T >::operator= (UniqueHandle &&kvsh) noexcept;`	
Parameters (in):	kvsh	The UniqueHandle object to be moved.
Return value:	UniqueHandle &	The moved UniqueHandle object.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/unique_handle.h"	
Description:	Move assignment operator for UniqueHandle.	

⌋(*RS_PER_00002*, *RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

[SWS_PER_00374]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::UniqueHandle::operator=(UniqueHandle const &)
Scope:	class ara::per::UniqueHandle
Syntax:	`UniqueHandle& ara::per::UniqueHandle< T >::operator= (UniqueHandle const &)=delete;`
Header file:	#include "ara/per/unique_handle.h"
Description:	The copy assignment operator for UniqueHandle shall not be used.

⌋(*RS_PER_00002*, *RS_AP_00120*)

8.4.2.3 UniqueHandle::Operator->

[SWS_PER_00360]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::UniqueHandle::operator->()	
Scope:	class ara::per::UniqueHandle	
Syntax:	`T* ara::per::UniqueHandle< T >::operator-> () noexcept;`	
Return value:	T *	–
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/unique_handle.h"	
Description:	Non-constant arrow operator.	

⌋(*RS_PER_00001*, *RS_PER_00002*, *RS_PER_00003*, *RS_AP_00119*, *RS_AP_00129*, *RS_AP_00132*)

[SWS_PER_00361]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::UniqueHandle::operator->()	
Scope:	class ara::per::UniqueHandle	
Syntax:	`T const* ara::per::UniqueHandle< T >::operator-> () const noexcept;`	
Return value:	T const *	–
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/unique_handle.h"	
Description:	Constant arrow operator.	

⌋(*RS_PER_00001*, *RS_PER_00002*, *RS_PER_00003*, *RS_AP_00119*, *RS_AP_00129*, *RS_AP_00132*)

8.4.2.4 UniqueHandle::Operator*

[SWS_PER_00400]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::UniqueHandle::operator*()
Scope:	class ara::per::UniqueHandle
Syntax:	`T& ara::per::UniqueHandle< T >::operator* () noexcept;`
Return value:	T & –
Exception Safety:	noexcept
Thread Safety:	re-entrant
Header file:	#include "ara/per/unique_handle.h"
Description:	Non-constant dereference operator.

⌋(*RS_PER_00001*, *RS_PER_00002*, *RS_PER_00003*, *RS_AP_00119*, *RS_AP_00129*, *RS_AP_00132*)

[SWS_PER_00401]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::UniqueHandle::operator*()
Scope:	class ara::per::UniqueHandle
Syntax:	`T const& ara::per::UniqueHandle< T >::operator* () const noexcept;`
Return value:	T const & –
Exception Safety:	noexcept
Thread Safety:	re-entrant
Header file:	#include "ara/per/unique_handle.h"
Description:	Constant dereference operator.

⌋(*RS_PER_00001*, *RS_PER_00002*, *RS_PER_00003*, *RS_AP_00119*, *RS_AP_00129*, *RS_AP_00132*)

8.5 Errors

The `Persistency` cluster implements an error handling based on `ara::core::Result`. The errors supported by the `Persistency` cluster are listed in section 8.5.1.

8.5.1 PerErrc

[SWS_PER_00311]{DRAFT} ⌈

Kind:	enumeration	
Symbol:	ara::per::PerErrc	
Scope:	namespace ara::per	
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.
	kOutOfMemoryError= 12	The allocated storage quota was exceeded, or memory could not be allocated.
	kFileNotFoundError= 13	The file was not found.
Header file:	#include "ara/per/per_error_domain.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.	

⌋(*RS_AP_00122*, *RS_AP_00127*)

8.5.2 GetPerDomain

[SWS_PER_00352]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::GetPerDomain()	
Scope:	namespace ara::per	
Syntax:	`constexpr ara::core::ErrorDomain const& GetPerDomain () noexcept;`	
Return value:	ara::core::ErrorDomain const &	The global PerErrorDomain object.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/per_error_domain.h"	
Description:	Returns the global PerErrorDomain object.	

⌋(*RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

8.5.3 MakeErrorCode

[SWS_PER_00351]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::MakeErrorCode(PerErrc code, ara::core::ErrorDomain::SupportDataType data, char const *message)	
Scope:	namespace ara::per	
Syntax:	`constexpr ara::core::ErrorCode MakeErrorCode (PerErrc code, ara::core::ErrorDomain::SupportDataType data, char const *message) noexcept;`	
Parameters (in):	code	Error code number.
	data	Vendor defined data associated with the error.
	message	Human readable message explaining the error.
Return value:	ara::core::ErrorCode	An ErrorCode object.
Exception Safety:	noexcept	
Thread Safety:	re-entrant	
Header file:	#include "ara/per/per_error_domain.h"	
Description:	Creates an error code.	

⌋(*RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

8.5.4 PerException

[SWS_PER_00354]{DRAFT} ⌈

Kind:	class
Symbol:	ara::per::PerException
Scope:	namespace ara::per
Base class:	ara::core::Exception
Syntax:	`class PerException : public Exception {...};`
Header file:	#include "ara/per/per_error_domain.h"
Description:	Exception type thrown by persistency classes.

⌋(*RS_AP_00122*, *RS_AP_00127*)

8.5.4.1 PerException::PerException

[SWS_PER_00355]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::PerException::PerException(ara::core::ErrorCode errorCode)	
Scope:	class ara::per::PerException	
Syntax:	`explicit PerException (ara::core::ErrorCode errorCode) noexcept;`	
Parameters (in):	errorCode	The error code.
Exception Safety:	noexcept	
Header file:	#include "ara/per/per_error_domain.h"	
Description:	Construct a new persistency exception object containing an error code.	

⌋(*RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

8.5.5 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]{DRAFT} ⌈

Kind:	class
Symbol:	ara::per::PerErrorDomain
Scope:	namespace ara::per
Base class:	ara::core::ErrorDomain
Syntax:	`class PerErrorDomain final : public ErrorDomain {...};`
Header file:	#include "ara/per/per_error_domain.h"
Description:	Defines the error domain for Persistency.

⌋(*RS_AP_00122*, *RS_AP_00127*)

[SWS_PER_00349]{DRAFT} ⌈ The numerical ID of the `PerErrorDomain` shall be 0x8000'0000'0000'0101. ⌋(*RS_PER_00001*)

8.5.5.1 PerErrorDomain::PerErrorDomain

[SWS_PER_00313]{DRAFT} ⌈

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

⌋(*RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

8.5.5.2 PerErrorDomain::Name

[SWS_PER_00314]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::PerErrorDomain::Name()	
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:	re-entrant	
Header file:	#include "ara/per/per_error_domain.h"	
Description:	Returns the name of the error domain.	

⌋(*RS_AP_00119*, *RS_AP_00120*, *RS_AP_00132*)

[SWS_PER_00353]{DRAFT} ⌈ PerErrorDomain::Name shall return the NUL-terminated string "Per". ⌋(*RS_PER_00001*)

8.5.5.3 PerErrorDomain::Message

[SWS_PER_00315]{DRAFT} ⌈

Kind:	function
Symbol:	ara::per::PerErrorDomain::Message(CodeType errorCode)
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/per_error_domain.h"	
Description:	Returns the message associated with the error code.	

⌋(*RS_AP_00119*, *RS_AP_00120*, *RS_AP_00121*, *RS_AP_00132*)

8.5.5.4 PerErrorDomain::ThrowAsException

[SWS_PER_00350]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::per::PerErrorDomain::ThrowAsException(ara::core::ErrorCode const &errorCode)	
Scope:	class ara::per::PerErrorDomain	
Syntax:	`void ThrowAsException (ara::core::ErrorCode const &errorCode) const override;`	
Parameters (in):	errorCode	The error to throw.
Return value:	None	
Thread Safety:	no	
Header file:	#include "ara/per/per_error_domain.h"	
Description:	Throws the exception associated with the error code.	

⌋(*RS_AP_00120*, *RS_AP_00121*)

A Not applicable requirements

[SWS_PER_NA]{DRAFT} ⌈ These requirements are not applicable to this specification. ⌋(*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	***CppImplementationDataType*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::CppImplementationDataType			
Note	This meta-class represents the way to specify a reusable data type definition taken as a the basis for a C++ language binding **Tags:** atp.Status=draft			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	CustomCppImplementationDataType, StdCppImplementationDataType			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
arraySize	PositiveInteger	0..1	attr	This attribute can be used to specify the array size if the enclosing CppImplementationDataType has array semantics. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
namespace (ordered)	SymbolProps	*	aggr	This aggregation allows for the definition an own namespace for the enclosing CppImplementationDataType. **Tags:** atp.Status=draft
subElement (ordered)	CppImplementationDataTypeElement	*	aggr	This represents the collection of sub-elements of the enclosing CppImplementationDataType **Tags:** atp.Status=draft
templateArgument (ordered)	CppTemplateArgument	*	aggr	This aggreation allows for the specification of properties of template arguments **Tags:** atp.Status=draft
typeEmitter	NameToken	0..1	attr	This attribute can be taken to control how the respective CppImplementationDataType is contributed to the language binding.

▽

△

Class	***CppImplementationDataType*** (abstract)			
typeReference	CppImplementation DataType	0..1	ref	This reference shall be defined to define a type reference (a.k.a. typedef). **Tags:** atp.Status=draft

Table B.2: CppImplementationDataType

Class	**Executable**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure			
Note	This meta-class represents an executable program. **Tags:** atp.Status=draft atp.recommendedPackage=Executables			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
buildType	BuildTypeEnum	0..1	attr	This attribute describes the buildType of a module and/or platform implementation.
minimumTimer Granularity	TimeValue	0..1	attr	This attribute describes the minimum timer resolution (TimeValue of one tick) that is required by the Executable. **Tags:** atp.Status=draft
rootSw Component Prototype	RootSwComponent Prototype	0..1	aggr	This represents the root SwCompositionPrototype of the Executable. This aggregation is required (in contrast to a direct reference of a SwComponentType) in order to support the definition of instanceRefs in Executable context. **Tags:** atp.Status=draft
version	StrongRevisionLabel String	0..1	attr	Version of the executable. **Tags:** atp.Status=draft

Table B.3: Executable

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*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*

▽

△

Class	**PRPortPrototype**			
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*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
updateStrategy	PersistencyElementLevelUpdateStrategyEnum	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	PersistencyDataElement	1	ref	This refrence represents the PersistencyDataElement for which the PersistencyDataRequiredComSpec applies. **Tags:** atp.Status=draft

▽

△

Class	**PersistencyDataRequiredComSpec**			
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.
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	PersistencyElement LevelUpdateStrategy Enum	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.

△

Class	PersistencyFileProxy			
updateStrategy	PersistencyElement LevelUpdateStrategy Enum	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.

▽

△

Class	***PersistencyInterface*** (abstract)			
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
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*

▽

△

Class	**PersistencyPortPrototypeToKeyValueDatabaseMapping**			
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			
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

▽

△

Class	***PortPrototype*** (abstract)			
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*			
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).
preMapping	Boolean	0..1	attr	This attribute describes whether the executable is preloaded into the memory.
processState Machine	ModeDeclarationGroup Prototype	0..1	aggr	Set of Process States that are defined for the process. **Tags:** atp.Status=draft
stateDependent StartupConfig	StateDependentStartup Config	*	aggr	Applicable startup configurations. **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*, NmNetworkHandle, 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

Class	**SoftwareCluster**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage			
Note	This meta-class represents the ability to define an uploadable software-package, i.e. the SoftwareCluster shall contain all software and configuration for a given purpose. **Tags:** atp.Status=draft atp.recommendedPackage=SoftwareClusters			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
conflictsTo	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation handles conflicts. If it yields true then the SoftwareCluster shall not be installed. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=conflictsTo atp.Status=draft

▽

△

Class	**SoftwareCluster**			
contained ARElement	ARElement	*	ref	This reference represents the collection of model elements that cannot derive from UploadablePackage Element and that contribute to the completeness of the definition of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft
containedFibex Element	FibexElement	*	ref	This allows for referencing FibexElements that need to be considered in the context of a SoftwareCluster. **Tags:** atp.Status=draft
contained Package Element	UploadablePackage Element	*	ref	This reference identifies model elements that are required to complete the manifest content. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=containedPackageElement atp.Status=draft
contained Process	Process	*	ref	This reference represent the processes contained in the enclosing SoftwareCluster. **Tags:** atp.Status=draft
dependsOn	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation can be taken to identify a dependency for the enclosing SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=dependsOn atp.Status=draft
design	SoftwareClusterDesign	*	ref	This reference represents the identification of all Software ClusterDesigns applicable for the enclosing Software Cluster. **Stereotypes:** atpUriDef **Tags:** atp.Status=draft
diagnostic Address	SoftwareCluster DiagnosticAddress	*	aggr	This aggregation represents the collection of diagnostic addresses that apply for the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=diagnosticAddress atp.Status=draft
diagnostic Extract	DiagnosticContribution Set	0..1	ref	This reference represents the definition of the diagnostic extract applicable to the referencing SoftwareCluster **Tags:** atp.Status=draft
license	Documentation	*	ref	This attribute allows for the inclusion of the the full text of a license of the enclosing SoftwareCluster. In many cases open source licenses require the inclusion of the full license text to any software that is released under the respective license. **Tags:** atp.Status=draft
module Instantiation	AdaptiveModule Instantiation	*	ref	This reference identifies AdaptiveModuleInstantiations that need to be included with the SoftwareCluster in order to establish infrastructure required for the installation of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=moduleInstantiation atp.Status=draft

▽

△

Class	**SoftwareCluster**			
releaseNotes	Documentation	0..1	ref	This attribute allows for the explanations of changes since the previous version. The list of changes might require the creation of multiple paragraphs of test. **Tags:** atp.Status=draft
subSoftware Cluster	SoftwareCluster	*	ref	This reference is used to identify the sub-Software Clusters of an "umbrella" SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=subSoftwareCluster atp.Status=draft
vendorId	PositiveInteger	1	attr	Vendor ID of this Implementation according to the AUTOSAR vendor list.
vendor Signature	CryptoService Certificate	1	ref	This reference identifies the certificate that represents the vendor's signature. **Tags:** atp.Status=draft
version	StrongRevisionLabel String	1	attr	This attribute can be used to describe a version information for the enclosing SoftwareCluster.

Table B.29: SoftwareCluster

Class	**SoftwarePackage**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage			
Note	This meta-class represents the ability to formalize the content of a software package. **Tags:** atp.Status=draft atp.recommendedPackage=SoftwarePackages			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
actionType	SoftwarePackageAction TypeEnum	1	attr	This attribute defines the action to be taken in the step of processing the enclosing SoftwarePackage.
activationAction	SoftwarePackage ActivationActionEnum	1	attr	This attribute governs the action to be taken after the installation of the SoftwareCluster completed.
compressed Software PackageSize	PositiveInteger	1	attr	This size represents the size of the compressed Software Package.
isDeltaPackage	Boolean	1	attr	This attribute denotes whether the SoftwarePackage is only able to update but not for initial installation.
maximum SupportedUcm Version	RevisionLabelString	1	attr	This attribute identifies the maximum supported version of the UCM for this SoftwarePackage.
minimum SupportedUcm Version	RevisionLabelString	1	attr	This attribute identifies the minimum supported version of the UCM for this SoftwarePackage.
packagerId	PositiveInteger	1	attr	This attribute identifies Id of the organization that provides the packager generating the SoftwarePackage.
packager Signature	CryptoService Certificate	1	ref	This reference identifies the certificate that represents the packager's signature. **Tags:** atp.Status=draft

▽

△

Class	**SoftwarePackage**			
softwareCluster	SoftwareCluster	1	ref	This reference identifies the SoftwareCluster that belongs to the SoftwarePackage. The nature of this relation is actually more like an aggregation than a reference. But the relation is still modelled as a reference because two ARElements cannot aggregate each other. **Tags:** atp.Status=draft
typeApproval	String	0..1	attr	This attribute carries the homologation information that may be specific for a given country.
uncompressed SoftwareCluster Size	PositiveInteger	1	attr	This attribute gives an indication about the storage that has to be available on the target.

Table B.30: SoftwarePackage

Primitive	**StrongRevisionLabelString**
Package	M2::AUTOSARTemplates::AdaptivePlatform::UploadableSoftwarePackage
Note	This primitive represents a revision label which identifies an engineering object. It represents a pattern which requires four integer numbers separated by a dot, representing from left to right MajorVersion, MinorVersion, PatchVersion, and BuildVersion. Legal patterns are for example: 4.0.0.3456 4.0.0.1234565 **Tags:** atp.Status=draft xml.xsd.customType=STRONG-REVISION-LABEL-STRING xml.xsd.pattern=[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+ xml.xsd.type=string

Table B.31: StrongRevisionLabelString