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

Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R24-11

Document Change History

Date	Release	Changed by	Description
2024-11-27	R24-11	AUTOSAR Release Management	• Improved update sequence, allowing for independent updates of configuration • Storage access type no longer dependent on port type • Renamed error kIsEof as `kEof`, added production errors and log messages • Clarified synchronous behavior of `ara::per::KeyValueStorage::SyncToStorage` and `ara::per::ReadWriteAccessor::SyncToFile` • `ara::per::ReadWriteAccessor` changed to “final”
2023-11-23	R23-11	AUTOSAR Release Management	• Improved the clean up and the version handling during update • Removed local handling of `minimumSustainedSize` • Removed constraints regarding storage sync • Formal description of dependencies to other Functional Clusters

△

2022-11-24	R22-11	AUTOSAR Release Management	• Fixed security and improved distribution of redundancy • Improved update scenarios, introducing data type migration • Improved handling of large data in Key-Value Storages • Extended and improved error handling, splitting `kOutOfStorageSpace`
2021-11-25	R21-11	AUTOSAR Release Management	• Clarified and extended specification of Persistency behavior • Improved configuration of storage location and versioning • kNotInitialized was removed • Deleted move constructors/operators
2020-11-30	R20-11	AUTOSAR Release Management	• Replaced POSIX based file access API and improved error handling and symmetry of other APIs • Full support for encryption and redundancy by hashes using Crypto API • Added information to application about safety related problems • Improved installation/update and redundancy
2019-11-28	R19-11	AUTOSAR Release Management	• Introduced reset and restore of storages • Introduced storage statistics • Improved compliance with general AUTOSAR concepts • Improved naming and consistency of classes / methods / functions / constants • Changed Document Status from Final to published

▽

△

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

Contents

1 Introduction and Functional Overview

This specification describes the functionality, API and the configuration for the Functional Cluster Persistency.

The Persistency Functional Cluster will be referenced as Persistency in the remainder of this document.

Persistency offers mechanisms to Adaptive Applications and other Functional Clusters to store information in the non-volatile memory of a machine. The data is available over boot and ignition cycles.

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

To avoid redundancy in the specification, in the remainder of this document, Adaptive Applications and Functional Clusters that use Persistency will be called Persistency users, while the term Adaptive Application will be also be used to denote a Functional Cluster implemented as a Daemon, which is handled similarly to an Adaptive Application.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations that are only relevant within this specification. A general list of acronyms and abbreviations is available in [1].

Abbreviation / Acronym	Description
FS	File Storage
KVS	Key-Value Storage
MAC	Message Authentication Code

Table 2.1: Acronyms and Abbreviations used in the Scope of this Document

Term	Description
Adaptive Application	Refers to the Adaptive Application defined in [1], or to a Functional Cluster implemented as a Daemon.
Adaptive Platform	Refers to the AUTOSAR Adaptive Platform defined in [1].
Adaptive Platform Foundation	Refers to the Adaptive Platform Foundation defined in [1].
Contract Version	This version identifies the structural layout of Persistency. The contract version is expected to change when Storages are added, removed, or renamed, when their access mode changes, or when the available data types of a Key-Value Storage or the data types of storage elements change.
Daemon	Refers to a Functional Cluster implemented as a Daemon.
Deployment Version	This version identifies the pre-installed content of Persistency. The deployment version is expected to change when pre-installed elements of otherwise unchanged storages are added, removed, or renamed.
Element	Refers to either a key-value pair of a Key-Value Storage or a file of a File Storage. Used in the specification where something applies to all kinds of storage elements.
Execution Manifest	Refers to the Execution Manifest defined in [1].
File	A binary or text file to be stored in a File Storage.
File Name	The file name uniquely identifies a file within a File Storage.
File Storage	A set of files that are stored persistently.
Functional Cluster	Refers to the Functional Cluster defined in [1].
Integrity	Persistency distinguishes data integrity, which is ensured by the configured redundancy, from structural integrity, i.e. the readability of the structure of a Key-Value Storage or File Storage.
Key	The key uniquely identifies a key-value pair within a Key-Value Storage.
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.
Metadata	Additional data about a storage. This metadata is distributed over the storage location and a central location for all storages of a Process.
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.
Persistency User	An Adaptive Application or another Functional Cluster that uses Persistency.

▽

△

Physical Storage	The actual physical memory on which `persistent data` is stored. This could be a flash device that is accessed directly by `Persistency`, or a file system of the OS that uses an SSD.
Redundancy	Redundancy is used by `Persistency` to ensure the `integrity` of stored data. It can be configured to use replication of stored data, CRCs, or Hashes. Typically, only replication will allow to repair corrupted data.
Service Interface	Refers to the `Service Interface` defined in [1].
Software Package	Refers to the `Software Package` defined in [1].
Storage	Refers to either a `Key-Value Storage` or a `File Storage`. Used in the specification where something applies to all kinds of `storages`.
Update Strategy	`Persistency` uses update strategies configured on `element` and `storage` level. The actual `update strategy` of an `element` is derived from `manifest` parameters for the `element` and the containing `storage` defined during design and deployment phase.
Value	A `value` of a `key-value pair` stored in a `Key-Value Storage`.

Table 2.2: Terms used in the Scope of this Document

3 Related Documentation

3.1 Input Documents & Related Standards and Norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] Specification of Adaptive Platform Core
AUTOSAR_AP_SWS_Core

[3] Explanation of Adaptive Platform Software Architecture
AUTOSAR_AP_EXP_SWArchitecture

[4] Specification of Execution Management
AUTOSAR_AP_SWS_ExecutionManagement

[5] Requirements on Persistency
AUTOSAR_AP_RS_Persistency

[6] General Requirements specific to Adaptive Platform
AUTOSAR_AP_RS_General

[7] Specification of Update and Configuration Management
AUTOSAR_AP_SWS_UpdateAndConfigurationManagement

[8] Specification of Manifest
AUTOSAR_AP_TPS_ManifestSpecification

[9] Explanation of Adaptive Platform Design
AUTOSAR_AP_EXP_PlatformDesign

[10] Specification of Cryptography
AUTOSAR_AP_SWS_Cryptography

[11] Specification of Platform Types for Adaptive Platform
AUTOSAR_AP_SWS_PlatformTypes

[12] Specification of Language Binding for modeled AP data types
AUTOSAR_AP_SWS_LanguageBindingForModeledAPdatatypes

3.2 Further Applicable Specifications

AUTOSAR provides a core specification [2] which is also applicable for this functional cluster. The chapter "General requirements for all Functional Clusters" of [2] shall be considered an additional and required specification for implementing this functional cluster.

4 Constraints and Assumptions

4.1 Known Limitations

Although a Key-Value Storage and File Storage can be configured as write-only, the current API always allows read access. Read access is even possible when a file has been opened with ara::per::FileStorage::OpenFileWriteOnly.

4.2 Assumptions about the Implementation of Persistency

Persistency can be implemented with different internal architectures, and at the same time, Persistency users expect a well defined behavior that is independent of the actual implementation of Persistency. Therefore, the specification in this document needs to be sufficiently generic to cover the underlying differences. As usual in AUTOSAR Adaptive, the actual implementation is not defined by the specification and is therefore entirely up to the vendor of Persistency.

The following subsections list some of the possible implementations of Persistency.

4.2.1 Library Only

In this scenario, Persistency consists solely of a library. The manifest is partly used to configure the compilation of the library (design time parts) and partly transferred to a configuration file that is read by Persistency at run-time (deployment parts).

There is still some freedom regarding the storage of metadata and actual data, and regarding the points in time when elements of a storage are checked regarding redundancy or authenticity or when they are decrypted. E.g. the whole File Storage could be checked when it is opened or the single files could be checked when they are opened.

4.2.2 Using a Central Daemon

In this scenario, the Persistency library connects in the background to a daemon via some IPC mechanism. The manifest influences again the compilation of the library part and is also transferred to configuration files for the library part and possibly also the daemon.

An implementer has the same freedom regarding data storage, encryption, and checks for redundancy and authenticity as in the last scenario. But an additional challenge is that the communication channel between library and daemon can be lost at any time, resulting in additional potential errors.

4.2.3 Based on a File System

Both a library and a daemon can access storage using the file system services of the operating system. In this case, because AUTOSAR Adaptive is based on POSIX, the file systems will be mounted with specific options that are out of control of the Persistency implementation, but influence the point in time when written data actually appears on the disc. These are further detailed in Appendix B. Without following these advices, it might not be possible to guarantee synchronous behavior for calls like ara::per::KeyValueStorage::SyncToStorage or ara::per::ReadWriteAccessor::SyncToFile.

4.2.4 Direct Access to Storage Hardware

Independently of whether a daemon is used or not, Persistency can be implemented to access hardware devices like flash memory directly. 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 Persistency 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 Functional Clusters

This chapter defines the dependencies of this functional cluster to other functional clusters. AUTOSAR decided not to standardize interfaces which are exclusively used between functional clusters to allow efficient implementations which might depend e.g., on the used operating system. The goal of this chapter is to provide an informative guideline for the interactions between functional clusters without specifying syntactical details. This ensures compatibility between documents specifying different functional clusters and supports parallel implementation of different functional clusters. Details of internal interfaces are up to the platform provider. Additional internal interfaces, parameters, and return values can be added. A detailed technical architecture documentation of the overall AUTOSAR Adaptive Platform is provided in [3].

5.1 Provided Interfaces

This section provides an overview of the public interfaces provided by this functional cluster towards other functional clusters.

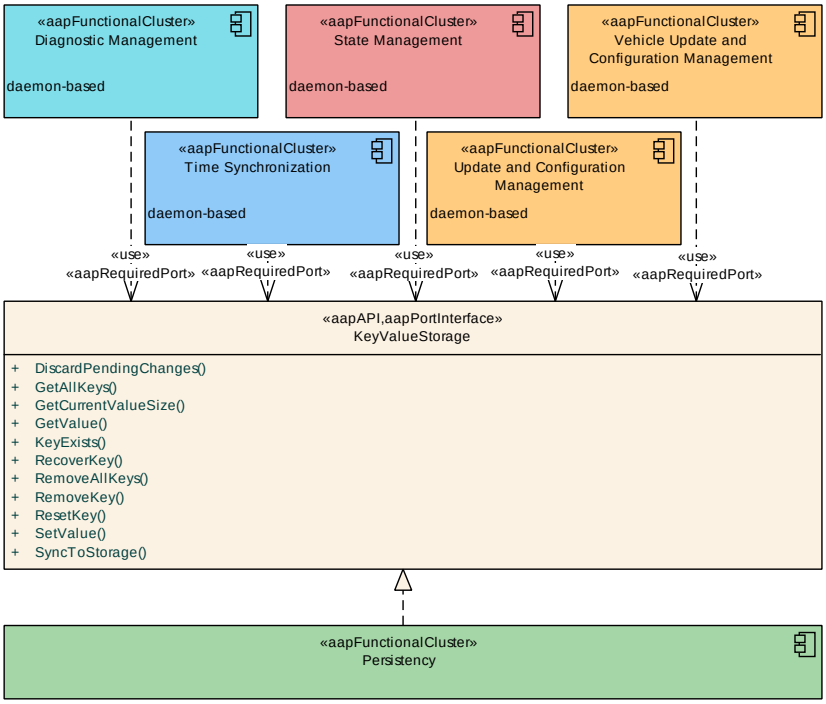


Figure 5.1: Interfaces Provided by Persistency to Other Functional Clusters

Figure 5.1 shows the Key-Value Storage interfaces provided by Persistency to other Functional Clusters within the AUTOSAR Adaptive Platform.

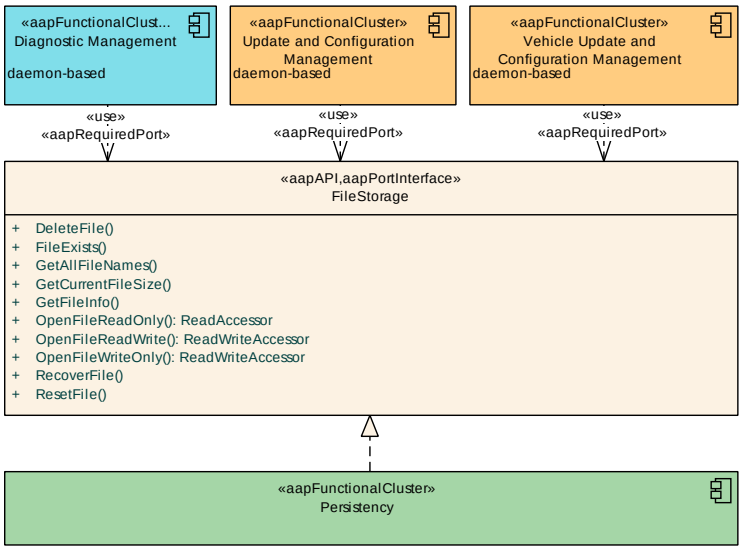


Figure 5.2: Interfaces Provided by Persistency to Other Functional Clusters

Figure 5.2 shows the File Storage interfaces provided by Persistency to other Functional Clusters within the AUTOSAR Adaptive Platform.

Table 5.1 lists the interfaces provided to other Functional Clusters within the AUTOSAR Adaptive Platform and provides a rationale.

Interface	*Functional Cluster*	*Purpose*
FileStorage	Diagnostic Management	Used to store associated data of diagnostic trouble codes (e.g., freeze frames).
	Update and Configuration Management	Used to store files of received software packages.
	Vehicle Update and Configuration Management	Used to store files of received vehicle packages.
KeyValueStorage	Diagnostic Management	Used to store properties of diagnostic trouble codes and diagnostic sessions.
	State Management	Used to store the internal state of State Management.
	Time Synchronization	Used to store the last received timestamp to enable a faster startup.
	Update and Configuration Management	Used to store the internal state of Update and Configuration Management.
	Vehicle Update and Configuration Management	Used to store the internal state of Vehicle Update and Configuration Management.

Table 5.1: Interfaces provided to other Functional Clusters

5.2 Required Interfaces

This section provides an overview of the public interfaces required by this functional cluster from other functional clusters.

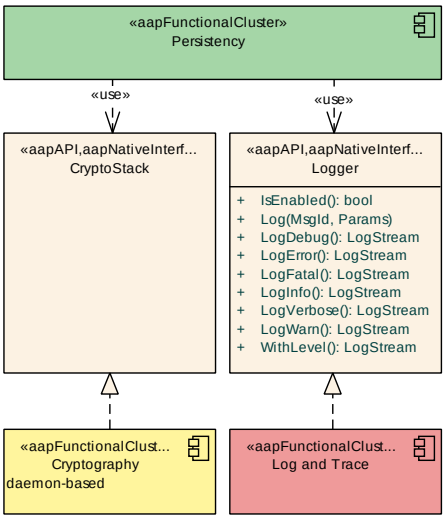


Figure 5.3: Interfaces required by Persistency from other Functional Clusters

Figure 5.3 shows the interfaces required by `Persistency` from other `Functional Clusters` within the AUTOSAR `Adaptive Platform`.

Table 5.2 lists the interfaces required from other `Functional Clusters` within the AUTOSAR `Adaptive Platform` and provides a rationale.

Functional Cluster	*Interface*	*Purpose*
Cryptography	CryptoStack	Used to ensure confidentiality and integrity of the persisted data.
Log and Trace	Logger	Used to provide information about errors and internal states and actions of `Persistency`.

Table 5.2: Interfaces required from other Functional Clusters

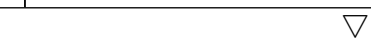
5.3 Persistency Specifics

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` (see [4, SWS Execution Management]).

6 Requirements Tracing

The following tables reference the requirements specified in the [5, RS Persistency] and the [6, RS General], and links to the fulfillments of these. Please note that if column “Satisfied by” is empty for a specific requirement, this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[RS_AP_00115]	Public namespaces	[SWS_PER_00002]
[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_00107] [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_00144] [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_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_00398] [SWS_PER_00399] [SWS_PER_00400] [SWS_PER_00401] [SWS_PER_00402] [SWS_PER_00403] [SWS_PER_00405] [SWS_PER_00406] [SWS_PER_00407] [SWS_PER_00414] [SWS_PER_00416] [SWS_PER_00418] [SWS_PER_00419] [SWS_PER_00420] [SWS_PER_00421] [SWS_PER_00422] [SWS_PER_00423] [SWS_PER_00424] [SWS_PER_00426] [SWS_PER_00427] [SWS_PER_00428] [SWS_PER_00429] [SWS_PER_00430] [SWS_PER_00431] [SWS_PER_00434] [SWS_PER_00438] [SWS_PER_00554] [SWS_PER_00567] [SWS_PER_00576] [SWS_PER_00577]



△

Requirement	Description	Satisfied by
[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_00107] [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_00144] [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_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] [SWS_PER_00405] [SWS_PER_00406] [SWS_PER_00407] [SWS_PER_00413] [SWS_PER_00414] [SWS_PER_00415] [SWS_PER_00416] [SWS_PER_00417] [SWS_PER_00418] [SWS_PER_00419] [SWS_PER_00420] [SWS_PER_00421] [SWS_PER_00422] [SWS_PER_00423] [SWS_PER_00424] [SWS_PER_00426] [SWS_PER_00427] [SWS_PER_00428] [SWS_PER_00429] [SWS_PER_00430] [SWS_PER_00431] [SWS_PER_00433] [SWS_PER_00434] [SWS_PER_00438] [SWS_PER_00459] [SWS_PER_00460] [SWS_PER_00461] [SWS_PER_00462] [SWS_PER_00554] [SWS_PER_00567] [SWS_PER_00568] [SWS_PER_00569] [SWS_PER_00576] [SWS_PER_00577] [SWS_PER_00578]

▽

△

Requirement	Description	Satisfied by
[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_00144] [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_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] [SWS_PER_00405] [SWS_PER_00406] [SWS_PER_00407] [SWS_PER_00413] [SWS_PER_00414] [SWS_PER_00420] [SWS_PER_00421] [SWS_PER_00422] [SWS_PER_00423] [SWS_PER_00424] [SWS_PER_00426] [SWS_PER_00427] [SWS_PER_00429] [SWS_PER_00430] [SWS_PER_00431] [SWS_PER_00433] [SWS_PER_00434] [SWS_PER_00438] [SWS_PER_00554] [SWS_PER_00578]
[RS_AP_00122]	Type names	[SWS_PER_00146] [SWS_PER_00147] [SWS_PER_00311] [SWS_PER_00312] [SWS_PER_00339] [SWS_PER_00340] [SWS_PER_00342] [SWS_PER_00343] [SWS_PER_00354] [SWS_PER_00359] [SWS_PER_00362] [SWS_PER_00411] [SWS_PER_00412] [SWS_PER_00432] [SWS_PER_00435] [SWS_PER_00436] [SWS_PER_00437]
[RS_AP_00125]	Enumerator and constant names	[SWS_PER_00146] [SWS_PER_00147] [SWS_PER_00311] [SWS_PER_00432] [SWS_PER_00435] [SWS_PER_00436]
[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] ▽

▽

△

Requirement	Description	Satisfied by
		△ [SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377] [SWS_PER_00405] [SWS_PER_00406] [SWS_PER_00407] [SWS_PER_00420] [SWS_PER_00421] [SWS_PER_00422] [SWS_PER_00423] [SWS_PER_00424] [SWS_PER_00426] [SWS_PER_00427] [SWS_PER_00428] [SWS_PER_00429] [SWS_PER_00430] [SWS_PER_00431] [SWS_PER_00433] [SWS_PER_00438] [SWS_PER_00554] [SWS_PER_00567] [SWS_PER_00576] [SWS_PER_00577] [SWS_PER_00578]
[RS_AP_00128]	Error reporting	[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_00353] [SWS_PER_00357] [SWS_PER_00358] [SWS_PER_00365] [SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377] [SWS_PER_00405] [SWS_PER_00406] [SWS_PER_00407] [SWS_PER_00424] [SWS_PER_00426] [SWS_PER_00427] [SWS_PER_00428] [SWS_PER_00429] [SWS_PER_00430] [SWS_PER_00431] [SWS_PER_00438] [SWS_PER_00472] [SWS_PER_00473] [SWS_PER_00474] [SWS_PER_00475] [SWS_PER_00476] [SWS_PER_00536] [SWS_PER_00537] [SWS_PER_00538] [SWS_PER_00539] [SWS_PER_00540] [SWS_PER_00541] [SWS_PER_00542] [SWS_PER_00543] [SWS_PER_00544] [SWS_PER_00545] [SWS_PER_00546] [SWS_PER_00547] [SWS_PER_00548] [SWS_PER_00549] [SWS_PER_00550] [SWS_PER_00551] [SWS_PER_00552] [SWS_PER_00553] [SWS_PER_00554] [SWS_PER_00564] [SWS_PER_00567] [SWS_PER_00576] [SWS_PER_00577]
[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_00360] [SWS_PER_00361] [SWS_PER_00363] [SWS_PER_00364] [SWS_PER_00365] [SWS_PER_00367] [SWS_PER_00369] [SWS_PER_00371] ▽

▽

△

Requirement	Description	Satisfied by
		△ [SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377] [SWS_PER_00398] [SWS_PER_00399] [SWS_PER_00400] [SWS_PER_00401] [SWS_PER_00402] [SWS_PER_00403] [SWS_PER_00405] [SWS_PER_00406] [SWS_PER_00407] [SWS_PER_00413] [SWS_PER_00417] [SWS_PER_00424] [SWS_PER_00426] [SWS_PER_00427] [SWS_PER_00428] [SWS_PER_00429] [SWS_PER_00430] [SWS_PER_00431] [SWS_PER_00438] [SWS_PER_00459] [SWS_PER_00460] [SWS_PER_00461] [SWS_PER_00462] [SWS_PER_00554] [SWS_PER_00568] [SWS_PER_00569]
[RS_AP_00134]	noexcept behavior of class destructors	[SWS_PER_00050] [SWS_PER_00330] [SWS_PER_00417] [SWS_PER_00568] [SWS_PER_00569]
[RS_AP_00135]	Avoidance of shared ownership	[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_00163] [SWS_PER_00164] [SWS_PER_00165] [SWS_PER_00166] [SWS_PER_00167] [SWS_PER_00168] [SWS_PER_00332] [SWS_PER_00333] [SWS_PER_00334] [SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00337] [SWS_PER_00338] [SWS_PER_00351] [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_00375] [SWS_PER_00376] [SWS_PER_00377] [SWS_PER_00405] [SWS_PER_00406] [SWS_PER_00407] [SWS_PER_00418] [SWS_PER_00419] [SWS_PER_00420] [SWS_PER_00421] [SWS_PER_00422] [SWS_PER_00423] [SWS_PER_00426] [SWS_PER_00427] [SWS_PER_00428] [SWS_PER_00429] [SWS_PER_00430] [SWS_PER_00431] [SWS_PER_00433] [SWS_PER_00438] [SWS_PER_00554] [SWS_PER_00567] [SWS_PER_00576] [SWS_PER_00577] [SWS_PER_00578]
[RS_AP_00136]	Usage of string types	[SWS_PER_00044] [SWS_PER_00046] [SWS_PER_00119] [SWS_PER_00165] [SWS_PER_00166] [SWS_PER_00332] [SWS_PER_00420] [SWS_PER_00524]

▽

△

Requirement	Description	Satisfied by
[RS_AP_00137]	Connecting run-time interface with model	[SWS_PER_00052] [SWS_PER_00116] [SWS_PER_00333] [SWS_PER_00334] [SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00356] [SWS_PER_00405] [SWS_PER_00406] [SWS_PER_00433] [SWS_PER_00578]
[RS_AP_00139]	Return type of synchronous function calls	[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_00163] [SWS_PER_00164] [SWS_PER_00165] [SWS_PER_00166] [SWS_PER_00167] [SWS_PER_00168] [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] [SWS_PER_00405] [SWS_PER_00406] [SWS_PER_00407] [SWS_PER_00418] [SWS_PER_00419] [SWS_PER_00420] [SWS_PER_00421] [SWS_PER_00422] [SWS_PER_00423] [SWS_PER_00426] [SWS_PER_00427] [SWS_PER_00428] [SWS_PER_00429] [SWS_PER_00430] [SWS_PER_00431] [SWS_PER_00438] [SWS_PER_00554] [SWS_PER_00567] [SWS_PER_00576] [SWS_PER_00577]
[RS_AP_00140]	Usage of "final specifier"	[SWS_PER_00312] [SWS_PER_00339] [SWS_PER_00340] [SWS_PER_00359] [SWS_PER_00362]
[RS_AP_00141]	Usage of out parameters	[SWS_PER_00044]
[RS_AP_00143]	Use 32-bit integral types by default	[SWS_PER_00146] [SWS_PER_00147] [SWS_PER_00432] [SWS_PER_00435] [SWS_PER_00436]
[RS_AP_00144]	Availability of a named constructor	[SWS_PER_00052] [SWS_PER_00113] [SWS_PER_00114] [SWS_PER_00115] [SWS_PER_00116] [SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377] [SWS_PER_00429] [SWS_PER_00430] [SWS_PER_00431]
[RS_AP_00145]	Availability of special member functions	[SWS_PER_00050] [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_00367] [SWS_PER_00368] [SWS_PER_00369] [SWS_PER_00370] [SWS_PER_00371] [SWS_PER_00372] [SWS_PER_00373] [SWS_PER_00374] [SWS_PER_00413] [SWS_PER_00414] [SWS_PER_00415] [SWS_PER_00416] [SWS_PER_00417] [SWS_PER_00568] [SWS_PER_00569]

▽

△

Requirement	Description	Satisfied by
[RS_AP_00146]	Classes whose construction requires interaction by the ARA framework	[SWS_PER_00339] [SWS_PER_00340] [SWS_PER_00342] [SWS_PER_00343] [SWS_PER_00459] [SWS_PER_00460] [SWS_PER_00461] [SWS_PER_00462]
[RS_AP_00147]	Classes that are created with an InstanceSpecifier as an argument are not copyable, but at most movable.	[SWS_PER_00052] [SWS_PER_00116]
[RS_AP_00149]	Error handling for non-initialized Functional Cluster	[SWS_PER_00311] [SWS_PER_00571]
[RS_AP_00153]	Assignment operators should restrict "this" to lvalues	[SWS_PER_00368] [SWS_PER_00370] [SWS_PER_00372]
[RS_AP_00159]	usage of "noexcept" specifier	[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_00107] [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_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_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_00398] [SWS_PER_00399] [SWS_PER_00400] [SWS_PER_00401] [SWS_PER_00402] [SWS_PER_00403] [SWS_PER_00405] [SWS_PER_00406] [SWS_PER_00407] [SWS_PER_00413] [SWS_PER_00414] [SWS_PER_00417] [SWS_PER_00418] [SWS_PER_00419] [SWS_PER_00420] [SWS_PER_00421] [SWS_PER_00422] [SWS_PER_00423] [SWS_PER_00424] [SWS_PER_00426] [SWS_PER_00427] [SWS_PER_00428] [SWS_PER_00429] [SWS_PER_00430] [SWS_PER_00431] [SWS_PER_00433] [SWS_PER_00438] [SWS_PER_00554] [SWS_PER_00567] [SWS_PER_00568] [SWS_PER_00569] [SWS_PER_00576] [SWS_PER_00577] [SWS_PER_00578]

▽

△

Requirement	Description	Satisfied by
[RS_PER_00001]	Storage of Persistent Data	[SWS_PER_00107] [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_00144] [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_00360] [SWS_PER_00361] [SWS_PER_00363] [SWS_PER_00364] [SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377] [SWS_PER_00398] [SWS_PER_00399] [SWS_PER_00400] [SWS_PER_00401] [SWS_PER_00402] [SWS_PER_00403] [SWS_PER_00418] [SWS_PER_00419] [SWS_PER_00420] [SWS_PER_00421] [SWS_PER_00422] [SWS_PER_00423] [SWS_PER_00425] [SWS_PER_00428] [SWS_PER_00429] [SWS_PER_00430] [SWS_PER_00431] [SWS_PER_00434] [SWS_PER_00494] [SWS_PER_00495] [SWS_PER_00499] [SWS_PER_00501] [SWS_PER_00502] [SWS_PER_00503] [SWS_PER_00534] [SWS_PER_00535] [SWS_PER_00582]
[RS_PER_00002]	Retrieval of Persistent Data	[SWS_PER_00049] [SWS_PER_00050] [SWS_PER_00322] [SWS_PER_00323] [SWS_PER_00324] [SWS_PER_00325] [SWS_PER_00339] [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_00398] [SWS_PER_00399] [SWS_PER_00400] [SWS_PER_00401] [SWS_PER_00402] [SWS_PER_00403] [SWS_PER_00459] [SWS_PER_00496] [SWS_PER_00497] [SWS_PER_00498] [SWS_PER_00506] [SWS_PER_00569] [SWS_PER_00582]

▽

△

Requirement	Description	Satisfied by
[RS_PER_00003]	Key-Value Storage	[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_00331] [SWS_PER_00332] [SWS_PER_00333] [SWS_PER_00334] [SWS_PER_00360] [SWS_PER_00361] [SWS_PER_00363] [SWS_PER_00364] [SWS_PER_00398] [SWS_PER_00399] [SWS_PER_00400] [SWS_PER_00401] [SWS_PER_00402] [SWS_PER_00403] [SWS_PER_00426] [SWS_PER_00427] [SWS_PER_00496] [SWS_PER_00497] [SWS_PER_00498] [SWS_PER_00499] [SWS_PER_00501] [SWS_PER_00502] [SWS_PER_00504] [SWS_PER_00505] [SWS_PER_00534] [SWS_PER_00535] [SWS_PER_00565] [SWS_PER_00566]
[RS_PER_00004]	File Storage	[SWS_PER_00107] [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_00144] [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] [SWS_PER_00413] [SWS_PER_00414] [SWS_PER_00415] [SWS_PER_00416] [SWS_PER_00417] [SWS_PER_00418] [SWS_PER_00419] [SWS_PER_00420] [SWS_PER_00421] [SWS_PER_00422] [SWS_PER_00423] [SWS_PER_00428] [SWS_PER_00429] [SWS_PER_00430] [SWS_PER_00431] [SWS_PER_00434] [SWS_PER_00435] [SWS_PER_00436] [SWS_PER_00437] [SWS_PER_00438] [SWS_PER_00440] [SWS_PER_00441] [SWS_PER_00442] [SWS_PER_00443] [SWS_PER_00444] [SWS_PER_00445] [SWS_PER_00457] [SWS_PER_00458] [SWS_PER_00460] [SWS_PER_00461] [SWS_PER_00462] [SWS_PER_00507] [SWS_PER_00508] [SWS_PER_00509] [SWS_PER_00510] [SWS_PER_00511] [SWS_PER_00512] [SWS_PER_00513] [SWS_PER_00514] [SWS_PER_00515] [SWS_PER_00516] [SWS_PER_00517] [SWS_PER_00518] [SWS_PER_00519] [SWS_PER_00520] [SWS_PER_00521] [SWS_PER_00522] [SWS_PER_00523] ▽

▽

△

Requirement	Description	Satisfied by
		△ [SWS_PER_00524] [SWS_PER_00525] [SWS_PER_00526] [SWS_PER_00527] [SWS_PER_00528] [SWS_PER_00529] [SWS_PER_00530] [SWS_PER_00531] [SWS_PER_00532] [SWS_PER_00533] [SWS_PER_00557] [SWS_PER_00568] [SWS_PER_00570]
[RS_PER_00005]	Encryption and Decryption	[SWS_PER_00210] [SWS_PER_00211] [SWS_PER_00464] [SWS_PER_00465] [SWS_PER_00559] [SWS_PER_00562] [SWS_PER_00563]
[RS_PER_00008]	Detection of Data Corruption	[SWS_PER_00221] [SWS_PER_00317] [SWS_PER_00318] [SWS_PER_00319] [SWS_PER_00432] [SWS_PER_00433] [SWS_PER_00439] [SWS_PER_00447] [SWS_PER_00448] [SWS_PER_00480] [SWS_PER_00481] [SWS_PER_00482] [SWS_PER_00483] [SWS_PER_00484] [SWS_PER_00485] [SWS_PER_00486] [SWS_PER_00487] [SWS_PER_00488] [SWS_PER_00489] [SWS_PER_00490] [SWS_PER_00555] [SWS_PER_00558] [SWS_PER_00560] [SWS_PER_00561]
[RS_PER_00009]	Recovery of Corrupted Data	[SWS_PER_00317] [SWS_PER_00318] [SWS_PER_00319] [SWS_PER_00333] [SWS_PER_00334] [SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00337] [SWS_PER_00338] [SWS_PER_00358] [SWS_PER_00426] [SWS_PER_00427] [SWS_PER_00439] [SWS_PER_00447] [SWS_PER_00448] [SWS_PER_00452] [SWS_PER_00453] [SWS_PER_00454] [SWS_PER_00455] [SWS_PER_00456] [SWS_PER_00477] [SWS_PER_00478] [SWS_PER_00479] [SWS_PER_00555] [SWS_PER_00556] [SWS_PER_00572] [SWS_PER_00573]
[RS_PER_00010]	Configurable Layout of Persistent Data	[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_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_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] ▽

▽

△

Requirement	Description	Satisfied by
		△ [SWS_PER_00392] [SWS_PER_00393] [SWS_PER_00394] [SWS_PER_00395] [SWS_PER_00426] [SWS_PER_00427] [SWS_PER_00429] [SWS_PER_00430] [SWS_PER_00431] [SWS_PER_00439] [SWS_PER_00447] [SWS_PER_00448] [SWS_PER_00449] [SWS_PER_00450] [SWS_PER_00451] [SWS_PER_00456] [SWS_PER_00463] [SWS_PER_00464] [SWS_PER_00465] [SWS_PER_00466] [SWS_PER_00467] [SWS_PER_00468] [SWS_PER_00469] [SWS_PER_00470] [SWS_PER_00471] [SWS_PER_00477] [SWS_PER_00478] [SWS_PER_00479] [SWS_PER_00555] [SWS_PER_00556] [SWS_PER_00558] [SWS_PER_00559] [SWS_PER_00560] [SWS_PER_00561] [SWS_PER_00562] [SWS_PER_00563] [SWS_PER_00572] [SWS_PER_00573] [SWS_PER_00580] [SWS_PER_00581]
[RS_PER_00011]	Storage Quota	[SWS_PER_00321] [SWS_PER_00491] [SWS_PER_00492] [SWS_PER_00493]
[RS_PER_00012]	Installation	[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_00382] [SWS_PER_00383] [SWS_PER_00384] [SWS_PER_00385] [SWS_PER_00456] [SWS_PER_00463] [SWS_PER_00469] [SWS_PER_00470] [SWS_PER_00471] [SWS_PER_00477] [SWS_PER_00478] [SWS_PER_00479] [SWS_PER_00556] [SWS_PER_00558] [SWS_PER_00559] [SWS_PER_00572] [SWS_PER_00573]
[RS_PER_00013]	Update	[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_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_00463] [SWS_PER_00469] [SWS_PER_00470] [SWS_PER_00471] [SWS_PER_00560] [SWS_PER_00561] [SWS_PER_00562] [SWS_PER_00563] [SWS_PER_00576] [SWS_PER_00577] [SWS_PER_00578] [SWS_PER_00579] [SWS_PER_00580] [SWS_PER_00581]
[RS_PER_00014]	Rollback	[SWS_PER_00378] [SWS_PER_00396] [SWS_PER_00463] [SWS_PER_00469] [SWS_PER_00470] [SWS_PER_00471]
[RS_PER_00016]	Cleanup	[SWS_PER_00446] [SWS_PER_00463] [SWS_PER_00470] [SWS_PER_00471] [SWS_PER_00567]

▽

△

Requirement	Description	Satisfied by
[RS_PER_00017]	Storage Size	[SWS_PER_00405] [SWS_PER_00406] [SWS_PER_00407] [SWS_PER_00424] [SWS_PER_00554]
[RS_PER_00018]	Initialization and Shutdown	[SWS_PER_00408] [SWS_PER_00409] [SWS_PER_00574]
[RS_PER_00019]	Authentication	[SWS_PER_00449] [SWS_PER_00450] [SWS_PER_00451] [SWS_PER_00466] [SWS_PER_00467] [SWS_PER_00468]

Table 6.1: Requirements Tracing

7 Functional Specification

7.1 The Architecture of Persistency

The Persistency offers two different mechanisms to access persistent memory: Key-Value Storages offer access to a set of keys with associated values (similar to a database), while File Storages offer access to a set of files (similar to a directory of a file system).

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. Of course, the same applies to other Functional Clusters using Persistency.

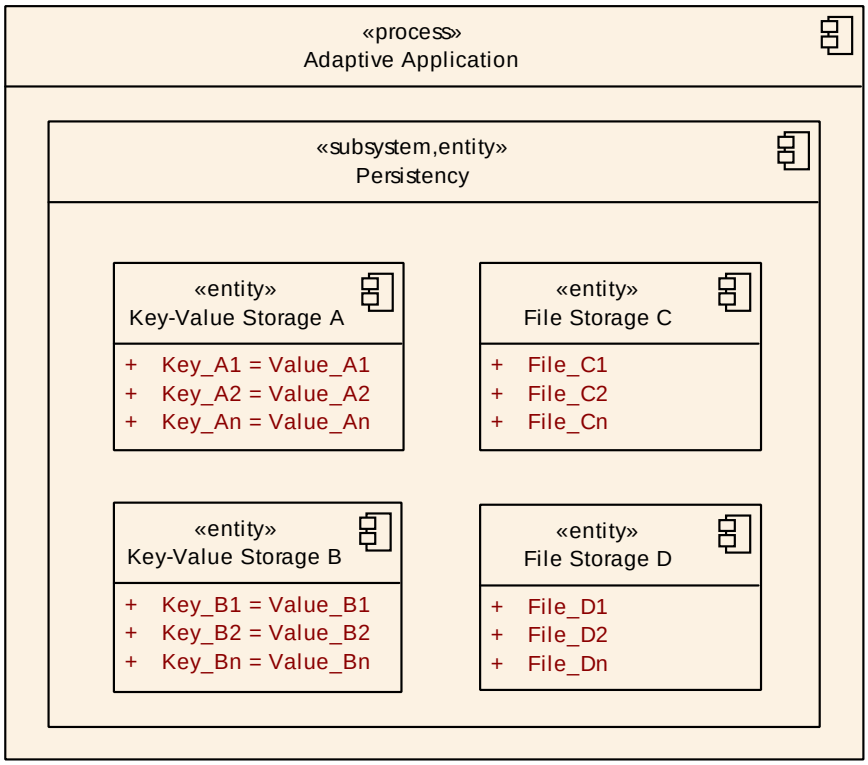


Figure 7.1: Typical usage of Persistency within an Adaptive Application

Persistency can also be used directly by other Functional Clusters without involvement of the Adaptive Application. The library part of these Functional Clusters will use dedicated deployment information of Persistency, while Daemons of Functional Clusters can be modeled similarly to Adaptive Applications.

7.1.1 Persistency in the Manifest

The Persistency usage of an Adaptive Application is modeled in the Execution Manifest (furtheron simply referred to as the "manifest") as part of the AdaptiveApplicationSwComponentTypes of an Executable. The model has two principal parts: The application design information, aggregated by the PersistencyKeyValueStorageInterface and the PersistencyFileStorageInterface, and the deployment information, aggregated by the PersistencyKeyValueStorage and the PersistencyFileStorage. The latter is also used when the Persistency is accessed directly by another Functional Cluster.

The API specification holds the classes ara::per::KeyValueStorage and ara::per::FileStorage for access to a Key-Value Storage or a File Storage, respectively. The global functions of these classes receive either the identifier (the fully qualified shortName path) of an RPortPrototype typed by a PersistencyInterface, or the shortName path of a FunctionalClusterInteractsWithPersistencyDeploymentMapping. Both are then used as an ara::core::InstanceSpecifier input parameter (see Section 8.2.1 and Section 8.3.1). Depending on the RPortPrototype.requiredComSpec.accessMode or on the value of FunctionalClusterInteractsWithPersistencyDeploymentMapping.persistencyAccess, the Key-Value Storage or File Storage will be accessible as:

Read Only if the PersistencyAccessEnum is read or FunctionalClusterInteractsWithPersistencyDeploymentMapping.persistencyAccess is read, or

Read/Write if the PersistencyAccessEnum is readWrite or FunctionalClusterInteractsWithPersistencyDeploymentMapping.persistencyAccess is readWrite, or

Write Only if the PersistencyAccessEnum is write or FunctionalClusterInteractsWithPersistencyDeploymentMapping.persistencyAccess is write. Please note that currently read access is not technically constrained on write only Storages.

In case of access to Persistency from an Adaptive Application, the manifest contains separate deployment data for each Process that references the Executable. The Process is bound to the deployment data by specialization of the class PersistencyPortPrototypeToDeploymentMapping, which refers to the RPortPrototype typed by a PersistencyInterface, to the PersistencyDeployment, and to the Process.

Usage of base classes in the manifest

For simplification reasons, the information that applies to both the Key-Value Storages and the File Storages is collected in base classes in the manifest, namely in PersistencyInterface for PersistencyKeyValueStorageInterface and

PersistencyFileStorageInterface, and in PersistencyDeployment for PersistencyKeyValueStorage and PersistencyFileStorage.

Likewise, the common information about key-value pairs and files is collected in PersistencyInterfaceElement for PersistencyDataElement and PersistencyFileElement, and in PersistencyDeploymentElement for PersistencyKeyValuePair and PersistencyFile.

And the link between application design and deployment information, represented by PersistencyPortPrototypeToDeploymentMapping, is specialized as PersistencyPortPrototypeToKeyValueStorageMapping and PersistencyPortPrototypeToFileStorageMapping.

In case of access to Persistency from the library part of a Functional Cluster, only the deployment data exists, without a mapping to an RPortPrototype. Instead, the Functional Cluster (represented by a NonOsModuleInstantiation) is linked to the PersistencyDeployment with a FunctionalClusterInteractsWithPersistencyDeploymentMapping, which also refers to a Process.

7.1.2 Key-Value Storages in the Manifest

Every Key-Value Storage is either represented by an RPortPrototype typed by a PersistencyKeyValueStorageInterface in the application design for the respective AdaptiveApplicationSwComponentType, or by a FunctionalClusterInteractsWithPersistencyDeploymentMapping, and in both cases by a PersistencyKeyValueStorage containing deployment information. Every Key-Value Storage can hold multiple key-value pairs. Key-value pairs can be added and removed at run-time by the Persistency user using the Persistency API (see Section 8.2.5.8 and Section 8.2.5.9).

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 (indirectly) triggered by the Update and Configuration Management [7] during installation or update using the deployment information and data provided by the software package of the Adaptive Application. See Section 7.2.5.

The link between application design and deployment information of a Key-Value Storage is represented by PersistencyPortPrototypeToKeyValueStorageMapping, which refers to an RPortPrototype typed by a PersistencyKeyValueStorageInterface, the corresponding PersistencyKeyValueStorage, and a Process.

7.1.3 File Storages in the Manifest

Every File Storage is represented by an RPortPrototype typed by a PersistencyFileStorageInterface in the application design for the respective

AdaptiveApplicationSwComponentType, or by a FunctionalClusterInteractsWithPersistencyDeploymentMapping, and in both cases by a PersistencyFileStorage containing deployment information. Every File Storage can hold multiple files as described in [8]. Similar to the key-value pairs mentioned above, files can be created and deleted at run-time by the Persistency user using the Persistency API (see Section 8.3.11.11, Section 8.3.11.13, and Section 8.3.11.5).

A File Storage with predefined files with initial content can be deployed during installation or update. This operation is also (indirectly) triggered by the Update and Configuration Management [7]. All needed deployment information and files come with the software package of the Adaptive Application. See Section 7.2.5.

The link between application design and deployment information of a File Storage is represented by PersistencyPortPrototypeToFileStorageMapping, which refers to an RPortPrototype typed by a PersistencyFileStorageInterface, the corresponding PersistencyFileStorage, and a Process.

7.2 General Features of Persistency

[SWS_PER_00002]

Upstream requirements: RS_AP_00115

⌈All specified classes within the Persistency shall reside within the C++ namespace ara::per.⌋

7.2.1 Error Handling

Error handling in Persistency is aligned with the guidelines described in [2]. To this end, the Persistency has to implement a set of standard classes and APIs, which are described in this section.

[SWS_PER_00472]

Upstream requirements: RS_AP_00128

⌈Persistency shall use the error codes defined in ara::per::PerErrc to report problems to the calling Persistency user via ara::core::Result. Vendors of Persistency may add their own errors to ara::per::PerErrc, using codes above 255.⌋

ara::per::PerErrc belongs to the ara::per::PerErrorDomain, which can be used by a Persistency user to classify returned errors.

[SWS_PER_00473]

Upstream requirements: RS_AP_00128

⌈ara::per::GetPerDomain shall return the global ara::per::PerErrorDomain object.⌋

To create its own Persistency error codes, the Persistency user may use ara::per::MakeErrorCode.

[SWS_PER_00474]

Upstream requirements: RS_AP_00128

⌈ara::per::MakeErrorCode shall return an ara::core::ErrorCode when called with an error code from ara::per::PerErrc.⌋

[SWS_PER_00353]

Upstream requirements: RS_AP_00128

⌈`ara::per::PerErrorDomain::Name` shall return the NUL-terminated string "Per".⌋

[SWS_PER_00475]

Upstream requirements: RS_AP_00128

⌈`ara::per::PerErrorDomain::Message` shall return the error message associated with the passed `ara::core::ErrorCode`.⌋

The whole `Persistency` API has been designed to be exception-less. If a `Persistency user` prefers to use exceptions, it may use `ara::per::PerErrorDomain::ThrowAsException`, or simply `ara::core::ErrorDomain::ThrowAsException` or `ara::core::ErrorCode::ThrowAsException`.

[SWS_PER_00476]

Upstream requirements: RS_AP_00128

⌈`ara::per::PerErrorDomain::ThrowAsException` shall throw an `ara::per::PerException` that is created from the passed error code.⌋

[SWS_PER_00571]

Upstream requirements: RS_AP_00149

⌈When `Persistency` detects an inconsistency in the `manifest` including invalid or non-existing URIs, it shall treat this situation as a violation as defined in [2] and call `ara::core::Abort`.⌋

7.2.1.1 Handling of General Errors

[SWS_PER_00536]

Upstream requirements: RS_AP_00128

⌈When a function or method of `Persistency` encounters a problem with the hardware or the underlaying operating system services during the access to a `storage` or an `element` of a `storage`, `Persistency` shall return the error `kPhysicalStorageFailure`.⌋

When `kPhysicalStorageFailure` occurs, the `Persistency user` cannot access the affected `storage` any more. Depending on the system, a reboot might restore the access.

[SWS_PER_00537]

Upstream requirements: RS_AP_00128

⌈When a method of Persistency would modify the underlying storage, but the storage is configured as read-only, Persistency shall return the error kIllegalWriteAccess.⌋

[SWS_PER_00538]

Upstream requirements: RS_AP_00128

⌈When a function of Persistency is called with an ara::core::InstanceSpecifier that is not available in the manifest, Persistency shall return the error kStorageNotFound.⌋

The two previous errors (kIllegalWriteAccess and kStorageNotFound) occur when the configuration does not match the implemented access to a storage. A Persistency user might be implemented to be aware of this possibility and react accordingly to the error by avoiding (write) accesses to the storage. If the Persistency user depends on (write) access to the storage, these errors would hint at an incorrect configuration of the storage.

[SWS_PER_00539]

Upstream requirements: RS_AP_00128

⌈When a function or method of Persistency detects a fundamental problem in the structure of the storage that prevents accessing the storage, Persistency shall return the error kIntegrityCorrupted.⌋

The Persistency user can restore a corrupted storage by calling ara::per::RecoverKeyValueStorage, ara::per::ResetKeyValueStorage, ara::per::RecoverAllFiles, or ara::per::ResetAllFiles. When the kIntegrityCorrupted is reported for an element of a storage, integrity can be restored by calling ara::per::KeyValueStorage::RecoverKey, ara::per::KeyValueStorage::ResetKey, ara::per::FileStorage::RecoverFile, or ara::per::FileStorage::ResetFile.

When Persistency detects insufficient or broken redundancy, it will report kValidationFailed as described in [SWS_PER_00221].

[SWS_PER_00540]

Upstream requirements: RS_AP_00128

⌈When encryption or decryption of persistent data fails within a function or method of Persistency, Persistency shall return the error kEncryptionFailed.⌋

[SWS_PER_00564]

Upstream requirements: RS_AP_00128

⌈When the calculation or verification of the MAC of persistent data fails within a function or method of Persistency, Persistency shall return the error kAuthenticationFailed.⌋

The two previous errors can occur when the configured cryptographic keys or algorithms are not available at run time, or when the storage or the element of a storage is corrupted. The latter case could be detected or even avoided by configuring redundancy.

[SWS_PER_00541]

Upstream requirements: RS_AP_00128

⌈When a method of Persistency tries to read data from a file, but the position is already at the end, Persistency shall return the error kEof.⌋

This error can typically be dealt with by the Persistency user, and can be avoided by using ara::per::ReadAccessor::IsEof.

[SWS_PER_00542]

Upstream requirements: RS_AP_00128

⌈When a function or method of Persistency would create a file, but the number of existing files already equals the configured PersistencyFileStorageInterface.maxNumberOfFiles of the corresponding File Storage, Persistency shall return the error kTooManyFiles.⌋

This error might occur when a new file is opened for writing, or when a File Storage is updated or when it is recovered after an error. A Persistency user seeing this error might delete some files to be able to create the new file (or files), or reset the File Storage to the initial state using ara::per::ResetAllFiles or ara::per::ResetPersistency.

[SWS_PER_00543]

Upstream requirements: RS_AP_00128

⌈When a function or method of Persistency would increase the size of a storage or an element of a storage such that the total storage size would exceed the configured PersistencyDeployment.maximumAllowedSize of the storage, Persistency shall return the error kQuotaExceeded.⌋

This error means that the Persistency user tried to store data that exceeds the configured quota of a storage. The Persistency user has to be able

to deal with this situation, and either write less data to the affected storage or remove outdated data from this storage, or reset the affected element/storage with a call to ara::per::ResetKeyValueStorage, ara::per::ResetAllFiles, ara::per::KeyValueStorage::ResetKey, ara::per::FileStorage::ResetFile, or even ara::per::ResetPersistency.

[SWS_PER_00544]

Upstream requirements: RS_AP_00128

⌈When a function or method of Persistency cannot increase the size of a storage because the physical storage is not sufficient (e.g. file system quoata exceeded or partition full), Persistency shall return the error kOutOfStorageSpace.⌋

This error means that some other storage or even a completely independent process occupies so much physical storage that Persistency cannot store additional data. This can only happen if PersistencyDeployment.maximumAllowedSize is configured to a higher value than PersistencyDeployment.minimumSustainedSize. The Persistency user has to be able to deal with this situation, and either write less data to the affected storage or remove outdated data from this or another storage, or reset this or another element/storage with a call to ara::per::ResetKeyValueStorage, ara::per::ResetAllFiles, ara::per::KeyValueStorage::ResetKey, ara::per::FileStorage::ResetFile, or even ara::per::ResetPersistency.

7.2.2 Parallel Access to Persistent Data

According to [9], the persistent data is local to one Process. Therefore, Persistency will never share persistent data between two (or more) Processes, even of the same Executable. The background of this decision is that Persistency should not provide an additional communication path for Adaptive Applications besides the mechanisms provided by the Functional Cluster Communication Management (e.g. using ara::com).

[SWS_PER_00309]

Upstream requirements: RS_PER_00001

⌈Persistent data shall always be local to one Process.⌋

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

Persistency is, on the other hand, prepared to handle concurrent access from multiple threads of the same Adaptive Application, running in the context of the same Process. To create shared access to a Key-Value Storage or File Storage, either the ara::per::SharedHandle returned by ara::per::OpenKeyValueStorage and ara::per::OpenFileStorage can be passed on (i.e. copied) to another thread, or ara::per::OpenKeyValueStorage and ara::per::OpenFileStorage can be called in independent threads for the same Key-Value Storage or File Storage, respectively. All operations of the Key-Value Storage and File Storage support concurrent access from multiple threads, though operations like ara::per::RecoverKeyValueStorage and ara::per::ResetKeyValueStorage or ara::per::RecoverAllFiles and ara::per::ResetAllFiles will only succeed when the corresponding Key-Value Storage or File Storage is not opened.

[SWS_PER_00545]

Upstream requirements: RS_AP_00128

⌈When a function of Persistency that modifies a storage is called while another function that modifies the same storage is being executed, Persistency shall return the error kResourceBusy.⌋

This restriction also applies to global functions like ara::per::UpdatePersistency, which will only succeed if no storage is open, and no other thread is currently modifying a storage with functions like ara::per::RecoverKeyValueStorage or ara::per::ResetAllFiles.

Access to single key-value pairs of a Key-Value Storage is possible from multiple threads at the same time, because the operation of

ara::per::KeyValueStorage::GetValue and ara::per::KeyValueStorage::SetValue are atomic, as are those of ara::per::KeyValueStorage::RemoveKey, ara::per::KeyValueStorage::RemoveAllKeys, ara::per::KeyValueStorage::SyncToStorage, and ara::per::KeyValueStorage::DiscardPendingChanges.

Access to single files of a File Storage cannot be shared between multiple threads, because it would be impossible to synchronize read and write accesses and the corresponding change of the seek position in a file. Accordingly, the ara::per::UniqueHandle returned by the OpenFile* APIs can only be moved to another thread, and trying to open an already opened file will fail. Likewise, operations like ara::per::FileStorage::DeleteFile, ara::per::FileStorage::RecoverFile, and ara::per::FileStorage::ResetFile will also not be possible on open files.

[SWS_PER_00546]

Upstream requirements: RS_AP_00128

⌈When a method of ara::per::FileStorage that opens or modifies a file is called while the same file is currently open or being modified by another method, Persistency shall return the error kResourceBusy.⌋

Files are implicitly closed when their ara::per::UniqueHandle goes out of scope, or when the File Storage to which they belong is closed.

[SWS_PER_00425]

Upstream requirements: RS_PER_00001

⌈When a File Storage is closed, because all related ara::per::SharedHandles go out of scope, any files which are still open are also closed.⌋

Accessing a ara::per::UniqueHandle of a file of a closed File Storage will result in undefined behavior.

7.2.3 Security Concepts

The Persistency supports protection of the authenticity and confidentiality of data stored in a Key-Value Storage or File Storage. Which kind of protection is applied and the used algorithms are decided at deployment time. The Persistency user is not aware of this fact.

If confidentiality of data shall be protected, a storage or an element of a storage are encrypted after the creation of the storage and when the storage is saved, and are decrypted when a storage is opened. Therefore, only encrypted data is stored persistently. In case of a read-only storage, encryption is done only once during installation, decryption is done every time the storage is opened.

If authenticity of data shall be protected, a message authentication code (MAC) is generated after the creation of a storage and when the storage is saved, and verified when the storage is opened. Therefore, unauthorized modifications to the storage can be detected.

In case of a read-only storage, it is also possible to protect the authenticity of the storage(or an element therein) by calculating a hash value of the data to be protected and comparing this calculated value to a hash value provided in the manifest. This assumes that the authenticity of the processed manifest is protected using other mechanisms (like secure boot).

If authenticity and confidentiality shall be protected, authenticated encryption schemes (like AES GCM) can be used.

[SWS_PER_00210]

Upstream requirements: RS_PER_00005, RS_PER_00010

⌈If a PersistencyDeploymentToCryptoKeySlotMapping exists in the manifest, and PersistencyDeploymentToCryptoKeySlotMapping.keySlotUsage is set to encryption, the Persistency shall encrypt all data related to the storage using the algorithm specified by PersistencyDeploymentToCryptoKeySlotMapping.cryptoAlgorithmString before storing it to the persistent memory.⌋

[SWS_PER_00464]

Upstream requirements: RS_PER_00005, RS_PER_00010

⌈If a PersistencyDeploymentElementToCryptoKeySlotMapping exists in the manifest, and PersistencyDeploymentElementToCryptoKeySlotMapping.keySlotUsage is set to encryption, the Persistency shall encrypt the element data using the algorithm specified by PersistencyDeploymentElementToCryptoKeySlotMapping.cryptoAlgorithmString before storing it to the persistent memory.⌋

[SWS_PER_00211]

Upstream requirements: RS_PER_00005, RS_PER_00010

⌈If a PersistencyDeploymentToCryptoKeySlotMapping exists in the manifest, and PersistencyDeploymentToCryptoKeySlotMapping.keySlotUsage is set to encryption, the Persistency shall decrypt all data related to the storage using the algorithm specified by PersistencyDeploymentToCryptoKeySlotMapping.cryptoAlgorithmString after reading it from persistent memory.⌋

[SWS_PER_00465]

Upstream requirements: RS_PER_00005, RS_PER_00010

⌈If a PersistencyDeploymentElementToCryptoKeySlotMapping exists in the manifest, and PersistencyDeploymentElementToCryptoKeySlotMapping.keySlotUsage is set to encryption, the Persistency shall decrypt the element data using the algorithm specified by PersistencyDeploymentElementToCryptoKeySlotMapping.cryptoAlgorithmString after reading it from persistent memory.⌋

[SWS_PER_00449]

Upstream requirements: RS_PER_00019, RS_PER_00010

⌈If a PersistencyDeploymentToCryptoKeySlotMapping exists in the manifest, and PersistencyDeploymentToCryptoKeySlotMapping.keySlotUsage is set to verification, the Persistency shall create and store a message authentication code (MAC) for all data related to the storage using the algorithm specified by PersistencyDeploymentToCryptoKeySlotMapping.cryptoAlgorithmString when storing it to the persistent memory.⌋

[SWS_PER_00466]

Upstream requirements: RS_PER_00019, RS_PER_00010

⌈If a PersistencyDeploymentElementToCryptoKeySlotMapping exists in the manifest, and PersistencyDeploymentElementToCryptoKeySlotMapping.keySlotUsage is set to verification, the Persistency shall create and store a message authentication code (MAC) for the element data using the algorithm specified by PersistencyDeploymentElementToCryptoKeySlotMapping.cryptoAlgorithmString when storing it to the persistent memory.⌋

[SWS_PER_00450]

Upstream requirements: RS_PER_00019, RS_PER_00010

⌈If a PersistencyDeploymentToCryptoKeySlotMapping exists in the manifest, and PersistencyDeploymentToCryptoKeySlotMapping.keySlotUsage is set to verification, the Persistency shall verify the MAC of all data

related to the storage using the algorithm specified by PersistencyDeployment-ToCryptoKeySlotMapping.cryptoAlgorithmString after reading it from persistent memory.⌋

[SWS_PER_00467]

Upstream requirements: RS_PER_00019, RS_PER_00010

⌈If a PersistencyDeploymentElementToCryptoKeySlotMapping exists in the manifest, and PersistencyDeploymentElementToCryptoKeySlotMapping.keySlotUsage is set to verification, the Persistency shall verify the MAC of the element data using the algorithm specified by PersistencyDeployment-ElementToCryptoKeySlotMapping.cryptoAlgorithmString after reading it from persistent memory.⌋

[SWS_PER_00451]

Upstream requirements: RS_PER_00019, RS_PER_00010

⌈If PersistencyDeploymentToCryptoKeySlotMapping.verificationHash is available, the Persistency shall calculate a hash value over all data related to the storage using the hash algorithm specified by PersistencyDeploymentTo-CryptoKeySlotMapping.cryptoAlgorithmString and verify that the calculated hash value matches PersistencyDeploymentToCryptoKeySlotMapping.ver-ificationHash.⌋

[SWS_PER_00468]

Upstream requirements: RS_PER_00019, RS_PER_00010

⌈If PersistencyDeploymentElementToCryptoKeySlotMapping.verifica-tionHash is available, the Persistency shall calculate a hash value over the element data using the hash algorithm specified by PersistencyDeployment-ElementToCryptoKeySlotMapping.cryptoAlgorithmString and verify that the calculated hash value matches PersistencyDeploymentElementToCrypto-KeySlotMapping.verificationHash.⌋

The Persistency will use the services of the Cryptography [10] for all cryptographic operations.

7.2.4 Redundancy Concepts

The Persistency shall take care of the integrity of the stored data, both for safety purposes and to prevent data loss. This can be achieved by calculating CRCs or hash values of the stored data, and by creating redundant copies. All these measures effectively create some redundancy for the stored data. The concrete measures to be taken are configurable: The application designer can use PersistencyInterface.redundancy to request redundancy (by setting it to redundant or redundantPerElement), or use PersistencyInterface.redundancyHandling to preselect the actual measures to be taken. During deployment, the integrator can define the actual measures taken to ensure data integrity using PersistencyDeployment.redundancyHandling. If PersistencyInterface.redundancyHandling is configured, the integrator shall use it as a guidance, but may also choose other, more appropriate measures based on superior knowledge of the final system.

[SWS_PER_00317]

Upstream requirements: RS_PER_00008, RS_PER_00009, RS_PER_00010

⌈The Persistency shall store redundant information for every storage represented by a PersistencyDeployment where PersistencyDeployment.redundancyHandling is configured.⌋

The actual handling of the redundancy configured during deployment is described in the following sections, see also [SWS_PER_00318], [SWS_PER_00319], and [SWS_PER_00447].

[SWS_PER_00221]

Upstream requirements: RS_PER_00008

⌈Persistency shall check the redundant data when accessing stored data. When the stored data is corrupted, Persistency shall try to restore it using the available redundancy. If Persistency is not able to recover using the redundancy, it shall report kValidationFailed.⌋

Depending on the actual implementation, Persistency might access the stored data at different times, e.g. when ara::core::Initialize is called, when a Key-Value Storage is opened, or when a file is accessed. The question whether the redundancy is sufficient for recovery is also implementation specific and can only be safely assumed for PersistencyRedundancyMOutOfN.

When the recovery failed, the Persistency user can choose to use ara::per::RecoverKeyValueStorage, ara::per::KeyValueStorage::RecoverKey, ara::per::RecoverAllFiles, or ara::per::FileStorage::RecoverFile to recover as much as possible and set the corresponding Key-Value Storage or File Storage again into a consistent state.

[SWS_PER_00452]

Upstream requirements: RS_PER_00009

⌈When `ara::per::RecoverKeyValueStorage` is called, `Persistency` shall restore the `Key-Value Storage` to a consistent state, including `redundancy`. First, the infrastructure of the whole `Key-Value Storage` shall be restored, then `Persistency` shall try to recover all `key-value pairs` available in the `Key-Value Storage` as described in [SWS_PER_00453]. Depending on available information, the whole `Key-Value Storage` might be reset to the initial state as described in [SWS_PER_00456], losing all updated `values` of its `key-value pairs`, or may contain outdated `key-value pairs` after the operation.⌋

[SWS_PER_00547]

Upstream requirements: RS_AP_00128

⌈When `ara::per::KeyValueStorage::RecoverKey` is called, `Persistency` shall first check whether the `key-value pair` is present in any instance of the `Key-Value Storage`, and otherwise return directly with `kKeyNotFound`.⌋

[SWS_PER_00453]

Upstream requirements: RS_PER_00009

⌈When `ara::per::KeyValueStorage::RecoverKey` is called for an existing `key-value pair`, `Persistency` shall try to restore the given `key` to a consistent state, including `redundancy`. Depending on available information, the `key` might be removed, reset to the initial `value` as described in [SWS_PER_00477], or might contain an outdated `value` after the operation.⌋

[SWS_PER_00454]

Upstream requirements: RS_PER_00009

⌈When `ara::per::RecoverAllFiles` is called, `Persistency` shall restore the `File Storage` to a consistent state, including `redundancy`. First, the infrastructure of the whole `File Storage` shall be restored as described in [SWS_PER_00478], then `Persistency` shall try to recover all currently available `files` as described in [SWS_PER_00455]. Depending on available information, the whole `File Storage` might be reset to the initial state, losing all updated content of its `files`, or may contain outdated `files` after the operation.⌋

[SWS_PER_00548]

Upstream requirements: RS_AP_00128

⌈When `ara::per::FileStorage::RecoverFile` is called, `Persistency` shall first check whether the `file` is present in the `File Storage`, and otherwise return directly with `kFileNotFound`.⌋

[SWS_PER_00455]

Upstream requirements: RS_PER_00009

⌈When `ara::per::FileStorage::RecoverFile` is called for an existing `file`, `Persistency` shall try to restore the given `file` to a consistent state, including `redundancy`. Depending on available information, the `file` might be removed, reset to the initial state as described in [SWS_PER_00479], or might contain outdated content after the operation.⌋

Of course the `Persistency user` has to validate the restored data in this case.

Or it can use `ara::per::ResetPersistency` (see [SWS_PER_00556]), `ara::per::ResetKeyValueStorage` (see [SWS_PER_00456]), `ara::per::KeyValueStorage::ResetKey` (see [SWS_PER_00477]), `ara::per::ResetAllFiles` (see [SWS_PER_00478]), or `ara::per::FileStorage::ResetFile` (see [SWS_PER_00479]) to reset the corrupted item to the initial state according to the current `manifest`.

The `Adaptive Application` may want to monitor its `storages` for any problem detected by `redundancy`, even if `Persistency` is able to recover by itself. This might be required to e.g. get an early indication of hardware problems or for safety critical `Adaptive Applications`. This monitoring is supported by `Persistency`, which will trigger a callback function of the `Adaptive Application` in case of any problems with the `storages`. To activate this monitoring, the `Adaptive Application` has to register that callback function using `ara::per::RegisterRecoveryReportCallback`.

[SWS_PER_00480]

Upstream requirements: RS_PER_00008

⌈When `ara::per::RegisterRecoveryReportCallback` is called, `Persistency` shall register the provided function and enable reporting of `redundancy` problems in all `storages` of this `Persistency user`.⌋

`Persistency` may check `redundancy` at different places, e.g. when `ara::core::Initialize` is called, when a `storage` is opened, or when `elements` of the `storage` are accessed. Whenever a problem is detected with `redundancy`, independently of the situation in which the problem appeared or whether the problem could be handled, `Persistency` will inform the `Adaptive Application` about these problems via the registered callback, stating `kKeyValueStorageRecovered`, `kKeyRecovered`, `kFileStorageRecovered`, or `kFileRecovered` when recovery of a `Key-Value Storage`, a `File Storage`, a `key-value pair`, or a `file` was possible, and `kKeyValueStorageRecoveryFailed`, `kKeyRecoveryFailed`, `kFileStorageRecoveryFailed`, or `kFileRecoveryFailed` if not. The callback also reports the affected `storage`, the affected `elements`, and how many copies of these `elements` were affected (the latter only in case `PersistencyRedundancyMOutOfN` is configured).

[SWS_PER_00481]

Upstream requirements: RS_PER_00008

⌈When a Key-Value Storage is accessed, and a redundancy problem affecting the whole Key-Value Storage is detected that cannot be handled by Persistency (i.e. kValidationFailed is returned), Persistency shall call the registered callback with storage set to the ara::core::InstanceSpecifier of the Key-Value Storage, recoveryReportKind set to kKeyValueStorageRecoveryFailed, an empty ara::core::Vector for reportedElements, and an ara::core::Vector with the indices of the affected redundant instances of the Key-Value Storage in reportedInstances.⌋

[SWS_PER_00482]

Upstream requirements: RS_PER_00008

⌈When a Key-Value Storage is accessed, and a redundancy problem affecting the whole Key-Value Storage is detected that can be handled by Persistency (i.e. the operation succeeds), Persistency shall call the registered callback with storage set to the ara::core::InstanceSpecifier of the Key-Value Storage, recoveryReportKind set to kKeyValueStorageRecovered, an empty ara::core::Vector for reportedElements, and an ara::core::Vector with the indices of the affected redundant instances of the Key-Value Storage in reportedInstances.⌋

[SWS_PER_00483]

Upstream requirements: RS_PER_00008

⌈When a File Storage is accessed, and a redundancy problem affecting the whole File Storage is detected that cannot be handled by Persistency (i.e. kValidationFailed is returned), Persistency shall call the registered callback with storage set to the ara::core::InstanceSpecifier of the File Storage, recoveryReportKind set to kFileStorageRecoveryFailed, an empty ara::core::Vector for reportedElements, and an ara::core::Vector with the indices of the affected redundant instances of the File Storage in reportedInstances.⌋

[SWS_PER_00484]

Upstream requirements: RS_PER_00008

⌈When a File Storage is accessed, and a redundancy problem affecting the whole File Storage is detected that can be handled by Persistency (i.e. the operation succeeds), Persistency shall call the registered callback with storage set to the ara::core::InstanceSpecifier of the File Storage, recoveryReportKind set to kFileStorageRecovered, an empty ara::core::Vector for reportedElements, and an ara::core::Vector with the indices of the affected redundant instances of the File Storage in reportedInstances.⌋

[SWS_PER_00485]

Upstream requirements: RS_PER_00008

⌈When a Key-Value Storage or one of its keys is accessed, and a redundancy problem affecting a set of keys is detected that cannot be handled by Persistency (i.e. kValidationFailed is returned), Persistency shall call the registered callback with storage set to the ara::core::InstanceSpecifier of the Key-Value Storage, recoveryReportKind set to kKeyRecoveryFailed, an ara::core::Vector with the affected keys in reportedElements, and an ara::core::Vector with the indices of the affected redundant instances of the keys in reportedInstances.⌋

[SWS_PER_00486]

Upstream requirements: RS_PER_00008

⌈When a Key-Value Storage or one of its keys is accessed, and a redundancy problem affecting a set of keys is detected that can be handled by Persistency (i.e. the operation succeeds), Persistency shall call the registered callback with storage set to the ara::core::InstanceSpecifier of the Key-Value Storage, recoveryReportKind set to kKeyRecovered, an ara::core::Vector with the affected keys in reportedElements, and an ara::core::Vector with the indices of the affected redundant instances of the keys in reportedInstances.⌋

[SWS_PER_00487]

Upstream requirements: RS_PER_00008

⌈When a redundancy problem of single keys is reported according to [SWS_PER_00485] or [SWS_PER_00486], Persistency shall in general ensure that each entry in reportedElements matches an entry in reportedInstances at the same positions, the two ara::core::Vectors shall have the same size. If several instances of a key are affected, the key may appear several times in reportedElements. As an optimization, if only one key is affected, reportedElements may contain the affected key as single entry, related to all entries of reportedInstances.⌋

[SWS_PER_00488]

Upstream requirements: RS_PER_00008

⌈When a File Storage or one of its files is accessed, and a redundancy problem affecting a set of files is detected that cannot be handled by Persistency (i.e. kValidationFailed is returned), Persistency shall call the registered callback with storage set to the ara::core::InstanceSpecifier of the File Storage, recoveryReportKind set to kFileRecoveryFailed, an ara::core::Vector with the affected files in reportedElements, and an ara::core::Vector with the indices of the affected redundant instances of the files in reportedInstances.⌋

[SWS_PER_00489]

Upstream requirements: RS_PER_00008

⌈When a File Storage or one of its files is accessed, and a redundancy problem affecting a set of files is detected that can be handled by Persistency (i.e. the operation succeeds), Persistency shall call the registered callback with storage set to the ara::core::InstanceSpecifier of the File Storage, recoveryReportKind set to kFileRecovered, an ara::core::Vector with the affected files in reportedElements, and an ara::core::Vector with the indices of the affected redundant instances of the files in reportedInstances.⌋

[SWS_PER_00490]

Upstream requirements: RS_PER_00008

⌈When a redundancy problem of single file is reported according to [SWS_PER_00488] or [SWS_PER_00489], Persistency shall in general ensure that each entry in reportedElements matches an entry in reportedInstances at the same positions, the two ara::core::Vectors shall have the same size. If several instances of a file are affected, the file may appear several times in reportedElements. As an optimization, if only one file is affected, reportedElements may contain the affected file as single entry, related to all entries of reportedInstances.⌋

7.2.4.1 Redundancy Types

The type of redundancy that is applied by the Persistency is defined by the set of PersistencyRedundancyHandling classes aggregated as PersistencyDeployment.redundancyHandling. The level to which redundancy is applied is defined by the possible values of the PersistencyRedundancyHandlingScopeEnum, which are persistencyRedundancyHandlingScopeStorage and persistencyRedundancyHandlingScopeElement for a Key-Value Storage and its key-value pairs, or a File Storage and its files, respectively.

[SWS_PER_00318]

Upstream requirements: RS_PER_00008, RS_PER_00009, RS_PER_00010

⌈In case a PersistencyRedundancyHandling aggregated as PersistencyDeployment.redundancyHandling is derived as PersistencyRedundancyCrc, the Persistency shall calculate a CRC value when persisting the storage or an element of the storage (depending on PersistencyDeployment.redundancyHandling.scope), and shall use this CRC to check the storage or the element when it is read back.⌋

[SWS_PER_00439]

Upstream requirements: RS_PER_00008, RS_PER_00009, RS_PER_00010

⌈`Persistency` shall calculate the CRC value using the algorithm defined by `PersistencyRedundancyCrc.algorithmFamily` with the bit width defined by `PersistencyRedundancyCrc.length`.⌋

[SWS_PER_00319]

Upstream requirements: RS_PER_00008, RS_PER_00009, RS_PER_00010

⌈In case a `PersistencyRedundancyHandling` aggregated as `PersistencyDeployment.redundancyHandling` is derived as `PersistencyRedundancyMOutOfN`, the `Persistency` shall store N copies when persisting the `storage` or an `element` of the `storage` (depending on `PersistencyDeployment.redundancyHandling.scope`), and shall check that at least M of the N copies of the `storage` or the `element` are identical when it is read back. N is defined by `n`, and M is defined by `m`.⌋

It is possible to configure more than one `PersistencyDeployment.deploymentUri` for a `storage` that uses `PersistencyRedundancyMOutOfN`. In this case, the copies will be distributed over the different locations. The existence of multiple URIs for a single `storage` is limited to this case by [constr_10366].

[SWS_PER_00555]

Upstream requirements: RS_PER_00008, RS_PER_00009, RS_PER_00010

⌈In case multiple `PersistencyDeployment.deploymentUri`s exist, and `PersistencyDeployment.redundancyHandling.scope` is `persistencyRedundancyHandlingScopeStorage`, `Persistency` shall store the copies of the `storage` in the different locations as follows:

2 The first location contains the main copy, the second location contains all other copies.

n (== `PersistencyRedundancyMOutOfN.n`) Each copy is placed in a separate location.

⌋

[SWS_PER_00447]

Upstream requirements: RS_PER_00008, RS_PER_00009, RS_PER_00010

⌈In case a `PersistencyRedundancyHandling` aggregated as `PersistencyDeployment.redundancyHandling` is derived as `PersistencyRedundancyHash`, the `Persistency` shall calculate a hash value when persisting the `storage` or an `element` of the `storage` (depending on `PersistencyDeployment.redundancyHandling.scope`), and shall use this hash value to check the `storage` or the `element` when it is read back.⌋

[SWS_PER_00448]

Upstream requirements: RS_PER_00008, RS_PER_00009, RS_PER_00010

⌈`Persistency` shall calculate the hash value using the algorithm defined by `PersistencyRedundancyHash.algorithmFamily` with the bit width defined by `PersistencyRedundancyHash.length`. If `PersistencyRedundancyHash.initializationVectorLength` is configured, an initialization vector of this length shall be calculated containing random data and passed to the hash algorithm.⌋

A possible approach to calculate the hash value and the random data would be to use the `Cryptography` [10]. The integration will have to take care that the configured `PersistencyRedundancyHash.length` and `PersistencyRedundancyHash.initializationVectorLength` are supported by the configured `PersistencyRedundancyHash.algorithmFamily`.

7.2.5 Installation and Update of Persistent Data

The Update and Configuration Management [7] 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 Adaptive Application. The Adaptive Application may trigger this handling explicitly by calling ara::per::UpdatePersistency during the verification phase that follows the installation or update, or rely on the Persistency to do this implicitly when persistent data is accessed (ara::per::OpenKeyValueStorage/ ara::per::OpenFileStorage). In both cases, the Persistency will compare a previously stored contract version and deployment version against the contract version and deployment version of the current manifest, and perform the required action.

Persistency stores information about already installed storages together with version information in a central location.

[SWS_PER_00463]

Upstream requirements: RS_PER_00010, RS_PER_00012, RS_PER_00013, RS_PER_00014, RS_PER_00016

⌈Persistency shall store information about the installed Key-Value Storages and File Storages in the location denoted by ProcessToMachineMapping.persistencyCentralStorageURI of the ProcessToMachineMapping that refers to the Process that is referenced by PersistencyPortPrototypeToDeploymentMappings. It shall also store the contract version and deployment version of the current manifest in this location.⌋

The Update and Configuration Management can also deploy independent updates of configuration data, if SoftwarePackage.actionType is set to updateConfiguration. Such a configuration update can also include Persistency manifest data. To detect this situation, an Adaptive Application can poll for updates of the manifest using ara::per::CheckForManifestUpdate.

[SWS_PER_00580] Check for Manifest Update

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈When `ara::per::CheckForManifestUpdate` is called, `Persistency` shall compare the `deployment version` of the stored `manifest` against the version that was read during initialization. If the stored version is higher, `ara::per::CheckForManifestUpdate` shall return true, otherwise false.⌋

When an update is reported, the `Adaptive Application` can use `ara::per::ReloadPersistencyManifest` to initiate a re-read of the manifest data, and after that it may trigger an update using `ara::per::UpdatePersistency` after closing all storages, or simply close and re-open each storage individually.

[SWS_PER_00581] Reload Persistency Manifest

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈When `ara::per::ReloadPersistencyManifest` is called, `Persistency` shall reload the `manifest` data and flag all `storages` for an update if the `deployment version` was incremented.⌋

[SWS_PER_00469]

Upstream requirements: RS_PER_00010, RS_PER_00012, RS_PER_00013, RS_PER_00014

⌈When `ara::per::UpdatePersistency` is called, the `Persistency` shall follow [SWS_PER_00382] (for installation), [SWS_PER_00386] and [SWS_PER_00387] (for update), or [SWS_PER_00396] (for roll-back) for each `storage` configured as `PersistencyDeployment` in the deployment data.⌋

[SWS_PER_00470]

Upstream requirements: RS_PER_00010, RS_PER_00012, RS_PER_00013, RS_PER_00014, RS_PER_00016

⌈When a `Key-Value Storage` is opened by the `Persistency user` using `ara::per::OpenKeyValueStorage`, the `Persistency` shall follow [SWS_PER_00382] (for installation), [SWS_PER_00386] and [SWS_PER_00387] (for update), [SWS_PER_00446] (for finalization), or [SWS_PER_00396] (for roll-back) for this `Key-Value Storage` configured as `PersistencyKeyValueStorage` in the deployment data.⌋

[SWS_PER_00471]

Upstream requirements: RS_PER_00010, RS_PER_00012, RS_PER_00013, RS_PER_00014, RS_PER_00016

⌈When a `File Storage` is opened by the `Persistency user` using `ara::per::OpenFileStorage`, the `Persistency` shall follow [SWS_PER_00382] (for installation), [SWS_PER_00386] and [SWS_PER_00387] (for update), [SWS_PER_00446]

(for finalization), or [SWS_PER_00396] (for roll-back) for each File Storage configured as PersistencyFileStorage in the deployment data.⌋

[SWS_PER_00378]

Upstream requirements: RS_PER_00010, RS_PER_00013, RS_PER_00014

⌈Persistency shall extract the contract version (PersistencyInterface.contractVersion or FunctionalClusterInteractsWithPersistencyDeploymentMapping.contractVersion) and the deployment version (PersistencyDeployment.version) from the manifest, and store them persistently in the location denoted by ProcessToMachineMapping.persistencyCentralStorageURI.⌋

The contract version is used by Persistency to detect a change of the Adaptive Application (see [SWS_PER_00387]), while the deployment version is used to detect a change of the deployed persistent data (see [SWS_PER_00386] and [SWS_PER_00396]).

[SWS_PER_CONSTR_00001] ⌈When the contract version (PersistencyInterface.contractVersion or FunctionalClusterInteractsWithPersistencyDeploymentMapping.contractVersion) is increased, the deployment version (PersistencyDeployment.version) needs to be increased, too.⌋

The deployment version and contract version are StrongRevisionLabelStrings. These strings consists of a MajorVersion, a MinorVersion, a PatchVersion, and additional labels for pre-release versions and build information. It is assumed that at least one of the first three will be incremented when the version is changed, while the additional labels might be arbitrary.

[SWS_PER_CONSTR_00002] ⌈When the deployment version (PersistencyDeployment.version) or contract version (PersistencyInterface.contractVersion or FunctionalClusterInteractsWithPersistencyDeploymentMapping.contractVersion) is increased, the MajorVersion, MinorVersion, or PatchVersion have to be incremented.⌋

After installation of the Adaptive Application, the Persistency will install predefined 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 PersistencyKeyValueStorageInterface or PersistencyFileStorageInterface.

- Persistent data that was defined by an application designer can be changed and fine-tuned by an integrator within PersistencyKeyValueStorage or PersistencyFileStorage.
- Persistent data can be directly defined by an integrator within PersistencyKeyValueStorage or PersistencyFileStorage.

[SWS_PER_00379]

Upstream requirements: RS_PER_00010, RS_PER_00012, RS_PER_00013

⌈Elements defined in the deployment data (PersistencyDeploymentElement) shall always be preferred over the elements defined in the application design (PersistencyInterfaceElement). The latter shall only be used if the former does not exist.⌋

After an update of the Adaptive Application or the manifest, the Persistency 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-value pair or file (PersistencyElementLevelUpdateStrategyEnum – keepExisting, overwrite, or delete).

[SWS_PER_00251]

Upstream requirements: RS_PER_00010, RS_PER_00012, RS_PER_00013

⌈An update strategy defined in the deployment data (PersistencyDeploymentElement.updateStrategy) shall always be preferred over the update strategy defined in the application design (PersistencyInterfaceElement.updateStrategy). The latter shall only be used if the former does not exist.⌋

PersistencyDeployment.updateStrategy is a mandatory attribute and therefore PersistencyInterface.updateStrategy is just a recommendation for the deployment and never used by Persistency.

[SWS_PER_00380]

Upstream requirements: RS_PER_00010, RS_PER_00012, RS_PER_00013

⌈An update strategy defined for a single element (PersistencyDeploymentElement.updateStrategy, PersistencyInterfaceElement.updateStrategy) shall always be preferred over the update strategy defined for the enclosing storage (PersistencyDeployment.updateStrategy, PersistencyInterface.updateStrategy). The latter shall only be used if the former does not exist.⌋

When the update succeeded, the Update and Configuration Management will finalize the new Adaptive Application. The Persistency will free the resources allocated by the last backup when it is opened the first time after the update succeeded.

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

The Adaptive Application can always reset its storages or single elements of the storages to their initial state, using ara::per::ResetPersistency, ara::per::ResetKeyValueStorage, ara::per::KeyValueStorage::ResetKey, ara::per::ResetAllFiles, or ara::per::FileStorage::ResetFile.

[SWS_PER_00556]

Upstream requirements: RS_PER_00009, RS_PER_00010, RS_PER_00012

⌈When ara::per::ResetPersistency is called, Persistency shall reset all Key-Value Storages and File Storages to the state they would have after installation of the Adaptive Application using the current manifest information.⌋

[SWS_PER_00456]

Upstream requirements: RS_PER_00009, RS_PER_00010, RS_PER_00012

⌈When ara::per::ResetKeyValueStorage is called, Persistency shall reset the Key-Value Storage to the state it would have after installation of the Adaptive Application using the current manifest information.⌋

[SWS_PER_00572]

Upstream requirements: RS_PER_00009, RS_PER_00010, RS_PER_00012

⌈When ara::per::KeyValueStorage::ResetKey is called, Persistency shall first check whether the key-value pair is present in the manifest, and otherwise return directly with kInitValueNotAvailable.⌋

[SWS_PER_00477]

Upstream requirements: RS_PER_00009, RS_PER_00010, RS_PER_00012

⌈When `ara::per::KeyValueStorage::ResetKey` is called for a `key-value pair` that is present in the `manifest`, `Persistency` shall reset the given `key` to the state it would have after installation of the `Adaptive Application` using the current `manifest` information.⌋

[SWS_PER_00478]

Upstream requirements: RS_PER_00009, RS_PER_00010, RS_PER_00012

⌈When `ara::per::ResetAllFiles` is called, `Persistency` shall reset the `File Storage` to the state it would have after installation of the `Adaptive Application` using the current `manifest` information.⌋

[SWS_PER_00573]

Upstream requirements: RS_PER_00009, RS_PER_00010, RS_PER_00012

⌈When `ara::per::FileStorage::ResetFile` is called, `Persistency` shall first check whether the `file` is present in the `manifest`, and otherwise return directly with `kInitValueNotAvailable`.⌋

[SWS_PER_00479]

Upstream requirements: RS_PER_00009, RS_PER_00010, RS_PER_00012

⌈When `ara::per::FileStorage::ResetFile` is called for a `file` that is present in the `manifest`, `Persistency` shall reset the given `file` to the state it would have after installation of the `Adaptive Application` using the current `manifest` information.⌋

Finally, when the `Adaptive Application` is removed, the `Update and Configuration Management` is responsible to remove the related `persistent data` as well.

7.2.5.1 Installation of Persistent Data

[SWS_PER_00382]

Upstream requirements: RS_PER_00010, RS_PER_00012

⌈When a `storage` is opened by the `Persistency user`, the `Persistency` shall check for the existence of any `persistent data` of this `Process`. If no `persistent data` is found, the `Persistency` shall initialize the `persistent data`.⌋

Initialization of `persistent data` is described in Section 7.2.5.1.1 and Section 7.2.5.1.2.

To be able to update Persistency with changed redundancy, Persistency stores the information about the used redundancy measures within the metadata of each storage. The same applies to the used keys and algorithms for encryption and authentication.

[SWS_PER_00558]

Upstream requirements: RS_PER_00008, RS_PER_00010, RS_PER_00012

⌈When a new storage is installed, Persistency shall store the information about the used redundancy in the metadata of the storage.⌋

[SWS_PER_00559]

Upstream requirements: RS_PER_00005, RS_PER_00010, RS_PER_00012

⌈When a new storage is installed, Persistency shall store the information about keys and algorithms used for encryption and authentication in the metadata of the storage.⌋

7.2.5.1.1 Installation of Key-Value Storage

[SWS_PER_00383]

Upstream requirements: RS_PER_00010, RS_PER_00012

⌈Persistency shall create a Key-Value Storage for each RPortPrototype typed by a PersistencyKeyValueStorageInterface that is found in the manifest of a newly installed Adaptive Application.⌋

The Key-Value Storages created by [SWS_PER_00383] are identified at run-time by the shortName path of the RPortPrototype, passed as ara::core::InstanceSpecifier to ara::per::OpenKeyValueStorage.

[SWS_PER_00252]

Upstream requirements: RS_PER_00010, RS_PER_00012

⌈Persistency shall create an entry in the Key-Value Storage for each PersistencyKeyValueStorageInterface.dataElement and PersistencyKeyValueStorage.keyValuePair that is found in the manifest of a newly installed or updated Adaptive Application, and for which the update strategy is not delete.⌋

Key-Value Storage entries are identified by the key. An entry with identical key might be defined both in the PersistencyKeyValueStorageInterface as dataElement and the PersistencyKeyValueStorage as keyValuePair, in

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

[SWS_PER_00253]

Upstream requirements: RS_PER_00010, RS_PER_00012

⌈Entries in the `Key-Value Storage` shall use the `shortName` of the `PersistencyDataElement` and/or `PersistencyKeyValuePair` as `key`.⌋

[SWS_PER_00254]

Upstream requirements: RS_PER_00010, RS_PER_00012

⌈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`.⌋

[SWS_PER_00384]

Upstream requirements: RS_PER_00010, RS_PER_00012

⌈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`.⌋

[SWS_PER_CONSTR_00003] ⌈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.⌋

Invalid `manifests` should be rejected by the tooling.

7.2.5.1.2 Installation of File Storage

[SWS_PER_00385]

Upstream requirements: RS_PER_00010, RS_PER_00012

⌈`Persistency` shall create a `File Storage` for each `RPortPrototype` typed by a `PersistencyFileStorageInterface` that is found in the `manifest` of a newly installed `Adaptive Application`.⌋

The `File Storages` created by [SWS_PER_00385] are identified at run-time by the `shortName` path of the `RPortPrototype`, passed as `ara::core::InstanceSpecifier` to `ara::per::OpenFileStorage`.

[SWS_PER_00265]

Upstream requirements: RS_PER_00010, RS_PER_00012

⌈`Persistency` shall create a `file` in the `File Storage` for each `PersistencyFileStorageInterface.fileElement` and `PersistencyFileStorage.file` that is found in the `manifest` of a newly installed or updated `Adaptive Application`, and for which the update strategy is not `delete`.⌋

The `files` within a `File Storage` are identified by their `file name`. A `file` with the same `file name` might be defined both in the `PersistencyFileStorageInterface` as `fileElement` and the `PersistencyFileStorage` as `file`, in which case [SWS_PER_00379] applies. The update strategy is determined according to [SWS_PER_00251] and [SWS_PER_00380].

[SWS_PER_00266]

Upstream requirements: RS_PER_00010, RS_PER_00012

⌈`Files` in the `File Storage` shall use the `file name` identified by `PersistencyFileElement.fileName` and/or `PersistencyFile.fileName`.⌋

[SWS_PER_00267]

Upstream requirements: RS_PER_00010, RS_PER_00012

⌈`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 `PersistencyFileElement.contentUri`. If that does not exist either, an empty `file` shall be created.⌋

[SWS_PER_CONSTR_00004] ⌈A `manifest` is invalid if the `shortName`s of a `PersistencyFileElement` and a `PersistencyFile` with the same `file name` differs.⌋

Invalid `manifests` should be rejected by the tooling.

7.2.5.2 Update of Persistent Data

[SWS_PER_00386]

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈When a `storage` is opened by the `Persistency user`, the `Persistency` shall compare the `deployment version` (`PersistencyDeployment.version`) in the `manifest` against the stored `deployment version`. If the `deployment version` in the `manifest` is higher than the stored `deployment version`, the `Persistency`

shall first create a backup of all the persistent data of this Process and then update the data.⌋

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 Section 7.2.5.2.1 and Section 7.2.5.2.2.

[SWS_PER_00579] Register Data Update Indication

Upstream requirements: RS_PER_00013

⌈If the Adaptive Application registered a function using ara::per::RegisterDataUpdateIndication, and if the Persistency had to update at least one of its storages according to [SWS_PER_00386], it shall call the registered function for each updated element of each storage that was updated according to [SWS_PER_00386], subject to the registered monitored storages and elements.⌋

[SWS_PER_00387]

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈If the Adaptive Application registered a function using ara::per::RegisterApplicationDataUpdateCallback, and if the Persistency had to update at least one of its storages according to [SWS_PER_00386], it shall compare the contract version (PersistencyInterface.contractVersion or FunctionalClusterInteractsWithPersistencyDeploymentMapping.contractVersion) in the manifest against the stored contract version. If the contract version in the manifest is higher than the stored contract version, the Persistency shall call the registered function for each storage that was updated according to [SWS_PER_00386].⌋

The function registered by the Adaptive Application using ara::per::RegisterApplicationDataUpdateCallback can be used by the Adaptive Application to update elements of a storage manually. The storage is identified by the ara::core::InstanceSpecifier provided to this function. The Adaptive Application might then, based on the contract version of the stored data provided as second argument to the function, read in the stored data in the old format or with the old type, convert the data, and store it again with the new format or new type expected by the current contract version.

Example: Version 1 of the Adaptive Application stored the maximum speed in mph as uint8, but version 2 expects the maximum speed in km/h as uint16. The update callback function will then see that a Key-Value Storage from version 1 of the Persistency has been updated to the current version, and can read in the old maximum speed by ara::per::KeyValueStorage::GetValue as uint8, convert it, and store it as uint16 with ara::per::KeyValueStorage::SetValue after removing the old value with ara::per::KeyValueStorage::RemoveKey.

In case the redundancy configuration or the configuration of encryption and authentication of the updated manifest differs from the old manifest, special care has to be taken to keep the data consistent and readable.

[SWS_PER_00560]

Upstream requirements: RS_PER_00008, RS_PER_00010, RS_PER_00013

⌈During the update, when the old storage is read, Persistency shall check the redundancy according to the information stored in the metadata of the storage.⌋

[SWS_PER_00561]

Upstream requirements: RS_PER_00008, RS_PER_00010, RS_PER_00013

⌈When the storage is persisted after the update, Persistency shall use the redundancy configured in the manifest, and store the information about the used redundancy in the metadata of the storage.⌋

Please note that this means that in some situations redundant information might become obsolete and can be removed, e.g. when the new manifest has a lower n for PersistencyRedundancyMOutOfN.

[SWS_PER_00562]

Upstream requirements: RS_PER_00005, RS_PER_00010, RS_PER_00013

⌈During the update, when the old storage is read, Persistency shall decrypt and verify the signature of the storage or the elements of the storage according to the information stored in the metadata of the storage.⌋

[SWS_PER_00563]

Upstream requirements: RS_PER_00005, RS_PER_00010, RS_PER_00013

⌈When the storage is persisted after the update, Persistency shall encrypt and sign the storage or the elements of the storage as configured in the manifest, and store the information about the used keys and algorithms in the metadata of the storage.⌋

7.2.5.2.1 Update of Key-Value Storage

[SWS_PER_00388]

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈When a new RPortPrototype typed by a PersistencyKeyValueStorageInterface is detected in an updated manifest, the Persistency shall create a Key-Value Storage as specified in [SWS_PER_00383].⌋

[SWS_PER_00389]

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈When an RPortPrototype typed by a PersistencyKeyValueStorageInterface is missing in an updated manifest, the Persistency shall remove the corresponding Key-Value Storage.⌋

[SWS_PER_00390]

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈When a PersistencyKeyValueStorageInterface.dataElement and/or a PersistencyKeyValueStorage.keyValuePair with a new key is detected in an updated manifest for which the update strategy is not delete, 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].⌋

[SWS_PER_00391]

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈When an existing key-value pair cannot be associated with any PersistencyKeyValueStorageInterface.dataElement or PersistencyKeyValueStorage.keyValuePair in an updated manifest, and the update strategy of the PersistencyKeyValueStorage corresponding to the Key-Value Storage is delete, the Persistency shall remove that key-value pair from the Key-Value Storage.⌋

The update strategy is determined according to [SWS_PER_00251].

[SWS_PER_00275]

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈When an existing key-value pair can be associated with a PersistencyKeyValueStorageInterface.dataElement or PersistencyKeyValueStorage.keyValuePair in an updated manifest, and the update strategy is overwrite, the Persistency shall replace that key-value pair with the new type and value as specified in [SWS_PER_00254] and [SWS_PER_00384].⌋

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

[SWS_PER_00277]

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈When an existing key-value pair can be associated with a PersistencyKeyValueStorageInterface.dataElement or PersistencyKeyValueStorage.keyValuePair in an updated manifest, and the update strategy is delete, the Persistency shall remove that key-value pair from the Key-Value Storage.⌋

Updated key-value pairs 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.

To support the conversion from one CppImplementationDataType to another (or to a different version of the same type) in the function registered using ara::per::RegisterApplicationDataUpdateCallback, Persistency provides the ability to read data types from a storage that are no longer used by the Adaptive Application. These types are configured in the manifest as previousDataType of a PersistencyKeyValueDataTypeMapping that references a currently used type as currentDataType.

There are two scenarios in which such a conversion is necessary:

1. An existing data type has been modified for the new Adaptive Application. The data type still uses the same identifier.
2. A new data type was introduced that replaces a data type that is no longer used in the new Adaptive Application. The two data types have different identifiers.

[SWS_PER_CONSTR_00005] ⌈In case an existing data type is changed in a new Adaptive Application, Persistency expects PersistencyKeyValueDataTypeMappings referring to a copy of the old data type as previousDataType and the new data type as currentDataType. The name of the old data type shall be formed as follows:

<PersistencyKeyValueDataTypeMapping.currentDataType.shortName>_<PersistencyKeyValueDataTypeMapping.previousContractVersion>.⌋

7.2.5.2.2 Update of File Storage

[SWS_PER_00392]

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈When a new RPortPrototype typed by a PersistencyFileStorageInterface is detected in an updated manifest, the Persistency shall create a File Storage as specified in [SWS_PER_00385].⌋

[SWS_PER_00393]

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈When an RPortPrototype typed by a PersistencyFileStorageInterface is missing in an updated manifest, the Persistency shall remove the corresponding File Storage.⌋

[SWS_PER_00394]

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈When a PersistencyFileStorageInterface.fileElement and/or PersistencyFileStorage.file with a new file name is detected in an updated manifest for which the update strategy is not delete, the Persistency shall create a new file in the File Storage as specified in [SWS_PER_00265], [SWS_PER_00266], and [SWS_PER_00267].⌋

[SWS_PER_00395]

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈When an existing file cannot be associated with any PersistencyFileStorageInterface.fileElement or PersistencyFileStorage.file in an updated manifest, and the update strategy of the PersistencyFileStorage corresponding to the File Storage is delete, the Persistency shall remove that file from the File Storage.⌋

The update strategy is determined according to [SWS_PER_00251].

[SWS_PER_00281]

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈When an existing file can be associated with a PersistencyFileStorageInterface.fileElement or PersistencyFileStorage.file in an updated manifest, and the update strategy is overwrite, the Persistency shall replace the content of that file with the new content as specified in [SWS_PER_00267].⌋

A file with the same file name might be defined both in the PersistencyFileStorageInterface and the PersistencyFileStorage, in which case [SWS_PER_00379] applies. The update strategy is determined according to [SWS_PER_00251] and [SWS_PER_00380].

[SWS_PER_00283]

Upstream requirements: RS_PER_00010, RS_PER_00013

⌈When an existing file can be associated with a PersistencyFileStorageInterface.fileElement or PersistencyFileStorage.file in an updated manifest, and the update strategy is delete, the Persistency shall remove that file from the File Storage.⌋

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

7.2.5.3 Finalization of Persistent Data after Successful Update

After installation and update, Persistency will usually be called with ara::per::UpdatePersistency within the verification phase of the Adaptive Application. When this succeeded, the Adaptive Application will be finalized by Update and Configuration Management and then started again in normal execution mode. In this case, Persistency may remove any backups that were created during a preceding update. Because Persistency has different update strategies (all storages with ara::per::UpdatePersistency or single storages with ara::per::OpenKeyValueStorage and ara::per::OpenFileStorage), it is also up to the Adaptive Application to decide when the backup data shall be removed, by calling ara::per::CleanUpPersistency.

[SWS_PER_00446]

Upstream requirements: RS_PER_00016

⌈When ara::per::CleanUpPersistency is called by the Adaptive Application, the Persistency shall remove all backup data of the storage.⌋

Update of persistent data is described in Section 7.2.5.2.

7.2.5.4 Roll-Back of Persistent Data after Failed Update

[SWS_PER_00396]

Upstream requirements: RS_PER_00014

⌈When a storage is opened by the Persistency user, the Persistency shall compare the deployment version (PersistencyDeployment.version) in the manifest against the stored deployment version. If the deployment version in the manifest is lower than the stored deployment version, the Persistency shall compare the deployment version in the manifest against the deployment

version stored in backup data. If the deployment versions match, the Persistency shall restore the backup. Otherwise, it shall remove all storages, and re-install the persistent data from the manifest.⌋

Initialization of persistent data is described in Section 7.2.5.1.

7.2.5.5 Removal of Persistent Data

Persistency is not able to remove its own data when the Update and Configuration Management removes an Adaptive Application, because the Adaptive Application will not be executed in this case, and therefore Persistency does not run. On the other hand, the Update and Configuration Management may use the information in the manifest (ProcessToMachineMapping.persistencyCentralStorageURI and PersistencyDeployment.deploymentUri) to obtain the locations of persistent data, and, if it has access to the locations, remove it.

7.2.6 Resource Management Concepts

The Persistency 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.

The Persistency will actively monitor the upper limit, while the lower limit is expected to be checked during installation of the SoftwarePackage that contains the Executable which uses Persistency.

[SWS_PER_00321]

Upstream requirements: RS_PER_00010, RS_PER_00011

⌈The Persistency shall ensure that the space actually allocated by a storage never surpasses the amount configured by PersistencyDeployment.maximumAllowedSize.⌋

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

The Persistency user can also poll the amount of storage space currently occupied by a complete Key-Value Storage or File Storage by using ara::per::GetCurrentKeyValueStorageSize or ara::per::GetCurrentFileStorageSize, respectively. Naturally, the returned values will not drop below a configured minimum size (PersistencyDeployment.minimumSustainedSize) or rise above a configured maximum size (PersistencyDeployment.maximumAllowedSize).

[SWS_PER_00491]

Upstream requirements: RS_PER_00011

⌈ara::per::GetCurrentKeyValueStorageSize shall return the total size of the storage space currently allocated to a Key-Value Storage, including administrative data (apart from data stored in ProcessToMachineMapping.persistencyCentralStorageURI), redundant data, and backup data. The reported size may be inaccurate if the Key-Value Storage is currently open, or if another operation is currently being executed on the same storage.⌋

The inaccuracy is mainly due to the fact that metadata of the storage is only updated when the storage is fully synchronized, and predicting the metadata size is sometimes very difficult.

[SWS_PER_00492]

Upstream requirements: RS_PER_00011

⌈`ara::per::GetCurrentFileStorageSize` shall return the total size of the storage space currently allocated to a `File Storage`, including administrative data (apart from data stored in `ProcessToMachineMapping`.`persistencyCentralStorageURI`), all its `files`, redundant data, and backup data. The reported size may be inaccurate if the `File Storage` is currently open, or if another operation is currently being executed on the same `storage`.⌋

7.3 Key-Value Storage specific Features

To access a Key-Value Storage, the Persistency user has to call ara::per::OpenKeyValueStorage with the ara::core::InstanceSpecifier derived from the manifest (a shortName path from the Executable to a PortPrototype or a mapping derived from FunctionalClusterInteractsWithFunctionalClusterMapping). This call will return an ara::per::SharedHandle of an ara::per::KeyValueStorage. The Key-Value Storage is closed when the ara::per::SharedHandle and all of its copies go out of scope, or when ara::core::Deinitialize is called.

[SWS_PER_00506]

Upstream requirements: RS_PER_00002

⌈When ara::per::OpenKeyValueStorage is called, and Persistency is properly initialized as described in [SWS_PER_00408], Persistency shall create a temporary storage that provides access to the Key-Value Storage identified by the ara::core::InstanceSpecifier, and shall create and return an ara::per::SharedHandle of an ara::per::KeyValueStorage.⌋

If ara::per::OpenKeyValueStorage is called without proper initialization, [SWS_PER_00574] applies.

All operations on a Key-Value Storage will be done in a temporary storage created during the call to ara::per::OpenKeyValueStorage, which the Persistency user can persist using ara::per::KeyValueStorage::SyncToStorage, or reset to the last stored state with ara::per::KeyValueStorage::DiscardPendingChanges.

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 Persistency user will see the last saved state.

[SWS_PER_00331]

Upstream requirements: RS_PER_00003

⌈Modifications of a Key-Value Storage that have not been persisted with a call to ara::per::KeyValueStorage::SyncToStorage shall be discarded when the Key-Value Storage is closed or the system is restarted, just as if ara::per::KeyValueStorage::DiscardPendingChanges had been called.⌋

Changes done by any thread (using a copy of the ara::per::SharedHandle) will be immediately visible in all other threads. This also applies to ara::per::KeyValueStorage::DiscardPendingChanges, which resets the key-value pairs in

all threads, and to ara::per::KeyValueStorage::SyncToStorage, which persists all changes done by any thread.

[SWS_PER_00494]

Upstream requirements: RS_PER_00001

⌈When ara::per::KeyValueStorage::SyncToStorage is called, Persistency shall store all changes permanently that have been done to the Key-Value Storage since the last call to this method or since the Key-Value Storage was opened. Persistency shall also update any configured redundancy within this call.⌋

Please note that depending on the implementation of Persistency, the configuration of an optionally used file system, and the capabilities of the used physical storage device, the actual storage of key-value pairs may be delayed. So, in case of an immediate power down directly after this call, the physical data may be updated only partially or not at all. Therefore, data may be lost or, without redundancy, data may even be corrupted in this case.

The handling of redundancy is described in detail in Section 7.2.4.

[SWS_PER_00495]

Upstream requirements: RS_PER_00001

⌈When ara::per::KeyValueStorage::DiscardPendingChanges is called, Persistency shall reset the Key-Value Storage to the last persisted state, which is the state after the last call to ara::per::KeyValueStorage::SyncToStorage or after opening the Key-Value Storage.⌋

Single key-value pairs of the Key-Value Storage are accessed using ara::per::KeyValueStorage::GetValue and ara::per::KeyValueStorage::SetValue. ara::per::KeyValueStorage::SetValue may also be used to create a key-value pair.

[SWS_PER_00496]

Upstream requirements: RS_PER_00002, RS_PER_00003

⌈When ara::per::KeyValueStorage::GetValue is called, Persistency shall first check whether the key-value pair is present in the temporary storage, and otherwise return directly with kKeyNotFound.⌋

[SWS_PER_00497]

Upstream requirements: RS_PER_00002, RS_PER_00003

⌈When `ara::per::KeyValueStorage::GetValue` is called for an existing `key-value pair`, `Persistency` shall check whether the templated data type matches the stored data type, and otherwise return directly with `kDataTypeMismatch`.⌋

[SWS_PER_00498]

Upstream requirements: RS_PER_00002, RS_PER_00003

⌈When `ara::per::KeyValueStorage::GetValue` is called for an existing `key-value pair` with the correct templated data type, `Persistency` shall return the stored `value` of the `key-value pair`, or, if the `value` was recently changed by `ara::per::KeyValueStorage::SetValue` (also in another thread), this new temporary `value`.⌋

[SWS_PER_00499]

Upstream requirements: RS_PER_00001, RS_PER_00003

⌈When `ara::per::KeyValueStorage::SetValue` is called for an existing `key-value pair`, `Persistency` shall check whether the templated data type matches the stored data type, and otherwise return directly with `kDataTypeMismatch`.⌋

[SWS_PER_00534]

Upstream requirements: RS_PER_00001, RS_PER_00003

⌈When `ara::per::KeyValueStorage::SetValue` is called for an existing `key-value pair` with the correct templated data type, `Persistency` shall store the new `value` of the `key-value pair` in the temporary storage.⌋

[SWS_PER_00501]

Upstream requirements: RS_PER_00001, RS_PER_00003

⌈When `ara::per::KeyValueStorage::SetValue` is called, and the `key-value pair` does not exist in the temporary storage, `Persistency` shall create the `key-value pair` with the templated data type and the provided `value` in the temporary storage.⌋

To remove a single `key-value pair`, the `Persistency user` may use `ara::per::KeyValueStorage::RemoveKey`, while `ara::per::KeyValueStorage::RemoveAllKeys` empties the `Key-Value Storage`. The type of a `key-value pair` may be changed by first removing it, and then creating it with the new type.

[SWS_PER_00502]

Upstream requirements: RS_PER_00001, RS_PER_00003

⌈When `ara::per::KeyValueStorage::RemoveKey` is called, `Persistency` shall first check whether the `key-value pair` is present in the temporary storage, and otherwise return directly with `kKeyNotFound`.⌋

[SWS_PER_00535]

Upstream requirements: RS_PER_00001, RS_PER_00003

⌈When `ara::per::KeyValueStorage::RemoveKey` is called for an existing `key-value pair`, `Persistency` shall remove the `key-value pair` from the temporary storage.⌋

[SWS_PER_00503]

Upstream requirements: RS_PER_00001

⌈When `ara::per::KeyValueStorage::RemoveAllKeys` is called, `Persistency` shall remove all `key-value pairs` from the temporary storage, resulting in an empty `Key-Value Storage`.⌋

Finally, the `Persistency user` can check for the existence of a single `key` with `ara::per::KeyValueStorage::KeyExists`, check the current size of a `value` using `ara::per::KeyValueStorage::GetCurrentValueSize`, and acquire a list of all currently available `keys` using `ara::per::KeyValueStorage::GetAllKeys`.

[SWS_PER_00504]

Upstream requirements: RS_PER_00003

⌈`ara::per::KeyValueStorage::KeyExists` shall return true if the `key` is present in the temporary storage, otherwise it shall return false.⌋

[SWS_PER_00565]

Upstream requirements: RS_PER_00003

⌈When `ara::per::KeyValueStorage::GetCurrentValueSize` is called, `Persistency` shall first check whether the `key-value pair` is present in the temporary `storage`, and otherwise return directly with `kKeyNotFound`.⌋

[SWS_PER_00566]

Upstream requirements: RS_PER_00003

⌈When `ara::per::KeyValueStorage::GetCurrentValueSize` is called for an existing `key-value pair`, `Persistency` shall return the current size of its `value`.⌋

[SWS_PER_00505]

Upstream requirements: RS_PER_00003

⌈`ara::per::KeyValueStorage::GetAllKeys` shall return an `ara::core::Vector` of `ara::core::String`, containing all the `keys` that are present in the temporary storage. If the temporary storage is empty, an empty `ara::core::Vector` shall be returned.⌋

7.3.1 Supported Data Types in Key-Value Storages

The `Persistency` supports the following classes of data types in the functions `ara::per::KeyValueStorage::GetValue` (templated via `T`), `ara::per::KeyValueStorage::GetValue` with `out` parameter (templated via `T`), and `ara::per::KeyValueStorage::SetValue` (templated via `T`) of a `Key-Value Storage`.

[SWS_PER_00302]

Upstream requirements: RS_PER_00001

⌈The `Persistency` shall be able to store all data types described in [11] in a `Key-Value Storage`.⌋

[SWS_PER_00303]

Upstream requirements: RS_PER_00001

⌈The `Persistency` shall be able to store serialized binary data in a `Key-Value Storage`. `Persistency` shall accept serialized binary data in the form of an `ara::core::Vector` of `ara::core::Byte` or an `ara::core::Span` of `ara::core::Byte`.⌋

This allows the `Adaptive Application` to store custom data types.

Please note that an `ara::core::Span` of `ara::core::Byte` cannot be returned by `ara::per::KeyValueStorage::GetValue`. It can only be passed in to `ara::per::KeyValueStorage::SetValue` and `ara::per::KeyValueStorage::GetValue` with `out` parameter.

[SWS_PER_00304]

Upstream requirements: RS_PER_00001, RS_PER_00010

⌈The `Persistency` shall be able to store all `CppImplementationDataType`s referenced by `PersistencyKeyValueStorageInterface.dataTypeForSerialization` or as `PersistencyKeyValueStorageInterface.dataElement` in the application design of a `PersistencyKeyValueStorage` in the corresponding `Key-Value Storage`.⌋

The definitions of these data types are generated as described in [12]. Typically, `Persistency` will generate serializers and deserializers for these types.

7.4 File Storage specific Features

To access a File Storage, the Persistency user has to call ara::per::OpenFileStorage with the ara::core::InstanceSpecifier derived from the manifest (a shortName path from the Executable to an RPortPrototype or a mapping derived from FunctionalClusterInteractsWithFunctionalClusterMapping). This call will return an ara::per::SharedHandle of an ara::per::FileStorage. The File Storage is closed when the ara::per::SharedHandle and all of its copies go out of scope, or when ara::core::Deinitialize is called.

[SWS_PER_00507]

Upstream requirements: RS_PER_00004

⌈When ara::per::OpenFileStorage is called, and Persistency is properly initialized as described in [SWS_PER_00408], Persistency shall create the necessary structures to access the File Storage identified by the ara::core::InstanceSpecifier, and create and return an ara::per::SharedHandle of an ara::per::FileStorage.⌋

If ara::per::OpenFileStorage is called without proper initialization, [SWS_PER_00574] applies.

To check for the existence of a single file, the Persistency user may call ara::per::FileStorage::FileExists, and ara::per::FileStorage::GetAllFileNames will return a list of all currently available files of the File Storage.

[SWS_PER_00508]

Upstream requirements: RS_PER_00004

⌈ara::per::FileStorage::FileExists shall return true if the file is present in the File Storage, otherwise it shall return false.⌋

[SWS_PER_00509]

Upstream requirements: RS_PER_00004

⌈ara::per::FileStorage::GetAllFileNames shall return an ara::core::Vector of ara::core::String, containing the file names of all the files that are present in the File Storage. If the File Storage is empty, an empty ara::core::Vector shall be returned.⌋

Files may be have been installed with the Adaptive Application or may have been created during an update. To create new files, the Persistency user may use ara::per::FileStorage::OpenFileReadWrite or ara::per::FileStorage::OpenFileWriteOnly, and it can use ara::per::FileStorage::DeleteFile to remove any file.

[SWS_PER_00510]

Upstream requirements: RS_PER_00004

⌈When `ara::per::FileStorage::DeleteFile` is called, `Persistency` shall first check whether the `file` is present in the `File Storage`, and otherwise return directly with `kFileNotFound`.⌋

[SWS_PER_00511]

Upstream requirements: RS_PER_00004

⌈When `ara::per::FileStorage::DeleteFile` is called for an existing `file`, `Persistency` shall remove the `file` from the `File Storage`.⌋

To access a `file` of a `File Storage`, the `Persistency user` has to call `ara::per::FileStorage::OpenFileReadWrite`, `ara::per::FileStorage::OpenFileReadOnly`, or `ara::per::FileStorage::OpenFileWriteOnly` with the `file name` of the `file`. These calls will return an `ara::per::UniqueHandle` of an `ara::per::ReadAccessor` or `ara::per::ReadWriteAccessor`.

[SWS_PER_00551]

Upstream requirements: RS_AP_00128

⌈When `ara::per::FileStorage::OpenFileReadOnly` (or one of the overloaded versions `ara::per::FileStorage::OpenFileReadOnly` with `ara::per::OpenMode` or `ara::per::FileStorage::OpenFileReadOnly` with `ara::per::OpenMode` and separate buffer) is called, `Persistency` shall first check whether the `file` is present in the `File Storage`, and otherwise return directly with `kFileNotFound`.⌋

[SWS_PER_00552]

Upstream requirements: RS_AP_00128

⌈When `ara::per::FileStorage::OpenFileReadOnly` with `ara::per::OpenMode` (or the overloaded version `ara::per::FileStorage::OpenFileReadOnly` with `ara::per::OpenMode` and separate buffer) is called, `Persistency` shall first check whether the provided mode consists only of either `kAtTheBeginning` or `kAtTheEnd`, and otherwise return directly with `kInvalidOpenMode`.⌋

[SWS_PER_00512]

Upstream requirements: RS_PER_00004

⌈When `ara::per::FileStorage::OpenFileReadOnly` (or one of the overloaded versions `ara::per::FileStorage::OpenFileReadOnly` with `ara::per::OpenMode` or `ara::per::FileStorage::OpenFileReadOnly` with `ara::per::OpenMode` and separate buffer) is called for an existing `file` and with a valid

ara::per::OpenMode, Persistency shall create the necessary structures to access the file identified by the file name, and create and return an ara::per::UniqueHandle of an ara::per::ReadAccessor.⌋

[SWS_PER_00553]

Upstream requirements: RS_AP_00128

⌈When ara::per::FileStorage::OpenFileReadWrite with ara::per::OpenMode or ara::per::FileStorage::OpenFileWriteOnly with ara::per::OpenMode (or one of the overloaded versions ara::per::FileStorage::OpenFileReadWrite with ara::per::OpenMode and separate buffer or ara::per::FileStorage::OpenFileWriteOnly with ara::per::OpenMode and separate buffer) is called, Persistency shall first check whether the provided mode contains either kAtTheBeginning, possibly combined with kTruncate and kAppend, or kAtTheEnd, possibly combined with kAppend, or only kTruncate. Otherwise, the call shall return directly with kInvalidOpenMode.⌋

[SWS_PER_00513]

Upstream requirements: RS_PER_00004

⌈When ara::per::FileStorage::OpenFileReadWrite or ara::per::FileStorage::OpenFileWriteOnly (or one of their overloaded versions ara::per::FileStorage::OpenFileReadWrite with ara::per::OpenMode, ara::per::FileStorage::OpenFileReadWrite with ara::per::OpenMode and separate buffer, ara::per::FileStorage::OpenFileWriteOnly with ara::per::OpenMode, or ara::per::FileStorage::OpenFileWriteOnly with ara::per::OpenMode and separate buffer) are called with a valid ara::per::OpenMode, Persistency shall create the necessary structures to access the file identified by the file name, and create and return an ara::per::UniqueHandle of an ara::per::ReadWriteAccessor.⌋

The file is closed when the ara::per::UniqueHandle goes out of scope, or when ara::core::Deinitialize is called.

[SWS_PER_00457]

Upstream requirements: RS_PER_00004

⌈When a file is closed, Persistency shall ensure that all changes to the file are persisted. This does not need to be done immediately like when ara::per::ReadWriteAccessor::SyncToFile is called, but may happen at a later time, latest when the file is opened again, or ara::core::Deinitialize is called.⌋

Some of the overloads of the file opening functions receive an ara::per::OpenMode as an argument. OpenModes can be combined using the operators "|" and "|=".

[SWS_PER_00514]

Upstream requirements: RS_PER_00004

⌈`ara::per::operator` "|" and `ara::per::operator` "|=" take two `ara::per::OpenMode` arguments and return the combined `ara::per::OpenMode`.⌋

All `files` of `Persistency` are implicitly readable, even when opened as "write only", which is expressed by `ara::per::ReadWriteAccessor` inheriting from `ara::per::ReadAccessor`. The `ara::per::ReadAccessor` class consequently also offers the methods related to `file` positions.

[SWS_PER_00515]

Upstream requirements: RS_PER_00004

⌈`ara::per::ReadAccessor::SetPosition` shall set the `file` position to the provided position. If the provided position is located outside of the current content of the `file` (including the position at the end of the `file`), `ara::per::ReadAccessor::SetPosition` shall keep the previous `file` position and return `kInvalidPosition`.⌋

[SWS_PER_00516]

Upstream requirements: RS_PER_00004

⌈`ara::per::ReadAccessor::MovePosition` shall move the `file` position to offset bytes according to the provided origin. If the new position would be located outside of the current content of the `file` (including the position at the end of the `file`), `ara::per::ReadAccessor::MovePosition` shall keep the previous `file` position and return `kInvalidPosition`.⌋

[SWS_PER_00517]

Upstream requirements: RS_PER_00004

⌈`ara::per::ReadAccessor::GetPosition` shall return the current read/write position in the `file`. In the case of an empty `file`, the position shall be returned as `0`.⌋

[SWS_PER_00518]

Upstream requirements: RS_PER_00004

⌈`ara::per::ReadAccessor::IsEof` shall return true if the position is the last possible position in the `file`, otherwise false.⌋

`ara::per::ReadAccessor::IsEof` will return true if the current position corresponds to the total `file` size, which can be obtained separately using `ara::per::ReadAccessor::GetSize`. In other words, true is only returned when the current position is at the position directly after the last character in the `file`, or if the `file` is empty.

[SWS_PER_00519]

Upstream requirements: RS_PER_00004

⌈`ara::per::ReadAccessor::GetSize` shall return the current total size of the `file`.⌋

`Persistency` does not care whether the content of a `file` is text or some binary data, and therefore offers separate methods to access the `file` content as text or as binary data. To read content from a text `file`, the `Persistency user` may use one of the following methods of the `ara::per::ReadAccessor` class:

[SWS_PER_00520]

Upstream requirements: RS_PER_00004

⌈`ara::per::ReadAccessor::PeekChar` shall return the character at the current `file` position without changing the position.⌋

[SWS_PER_00521]

Upstream requirements: RS_PER_00004

⌈`ara::per::ReadAccessor::GetChar` shall return the character at the current `file` position and advance the position by one.⌋

[SWS_PER_00522]

Upstream requirements: RS_PER_00004

⌈`ara::per::ReadAccessor::ReadText` shall read the text from the current position to the end of the `file` and return it as an `ara::core::String`. The position shall be set to the end of the `file`.⌋

[SWS_PER_00523]

Upstream requirements: RS_PER_00004

⌈`ara::per::ReadAccessor::ReadText` shall read the `n` characters of text from the current position and return them as an `ara::core::String`. The position shall be incremented by `n`. In case the end of the `file` is reached during this operation, the available characters shall be returned, and the position shall be set to the end of the `file`.⌋

[SWS_PER_00524]

Upstream requirements: RS_PER_00004, RS_AP_00136

⌈`ara::per::ReadAccessor::ReadLine` shall read all characters until the delimiter (defaulting to the newline character) or the end of the `file` is reached, and return them as a `ara::core::String`. The delimiter shall not be included in the returned `ara::core::String`. The position shall be set to the character following the delimiter or the end of the `file`.⌋

All these methods return characters with a size of eight bits, which are just so-called code units in case of UTF-8, not code points. Therefore, these methods might return incomplete code points. Persistency itself does not change or interpret the content of a file when accessing it in text mode. It is assumed, though, that files in the File Storage are encoded as UTF-8 (see [RS_AP_00136]). It is also assumed that line endings are handled according to UNIX conventions, i.e. just LF ("\n").

[SWS_PER_CONSTR_00006] ⌈If a CppImplementationDataType with category equal to STRING is used as PersistencyDataElement, then the encoding of this string data type is expected to be UTF-8.⌋

This means that a CppImplementationDataType can only be mapped to an ApplicationDataType of category STRING where attribute swDataDefProps.swTextProps.baseType.baseTypeDefinition.baseTypeEncoding is set to the value UTF-8 as defined by [constr_5035]. If a CppImplementationDataType without an ApplicationDataType is used there is no formal description about the UTF-8 encoding in the ServiceInterface description.

The following methods of the ara::per::ReadAccessor class can be used by a Persistency user to read binary content from a file:

[SWS_PER_00525]

Upstream requirements: RS_PER_00004

⌈ara::per::ReadAccessor::PeekByte shall return the byte at the current file position without changing the position.⌋

[SWS_PER_00526]

Upstream requirements: RS_PER_00004

⌈ara::per::ReadAccessor::GetByte shall return the byte at the current file position and advance the position by one.⌋

[SWS_PER_00527]

Upstream requirements: RS_PER_00004

⌈ara::per::ReadAccessor::ReadBinary shall read binary data from the current position to the end of the file and return it as an ara::core::Vector of ara::core::Byte. The position shall be set to the end of the file.⌋

[SWS_PER_00528]

Upstream requirements: RS_PER_00004

⌈ara::per::ReadAccessor::ReadBinary shall read the n characters of text from the current position and return them as an ara::core::Vector of ara::core::

Byte. The position shall be incremented by n. In case the end of the file is reached during this operation, the available bytes shall be returned, and the position shall be set to the end of the file.⌋

To write text to files, the Persistency user may use the ara::per::ReadWriteAccessor::WriteText method or the ara::per::ReadWriteAccessor::operator<< of the ara::per::ReadWriteAccessor class, which treat text in the same way as described above for e.g. ara::per::ReadAccessor::ReadText.

[SWS_PER_00557]

Upstream requirements: RS_PER_00004

⌈If the file was opened with ara::per::OpenMode set to kAppend, Persistency shall always set the current position to the end of the file before writing data to the file according to [SWS_PER_00529], [SWS_PER_00530], or [SWS_PER_00531].⌋

[SWS_PER_00529]

Upstream requirements: RS_PER_00004

⌈ara::per::ReadWriteAccessor::WriteText shall write the characters provided as ara::core::StringView to the file at the current position, possibly overwriting current content and advancing the end of the file if necessary. The position shall be set to the character following the last character that was written during this operation, or to the end of the file.⌋

[SWS_PER_00530]

Upstream requirements: RS_PER_00004

⌈ara::per::ReadWriteAccessor::operator<< shall write the characters provided as ara::core::StringView to the file at the current position, possibly overwriting current content and advancing the end of the file if necessary. The position shall be set to the character following the last character that was written during this operation, or to the end of the file. If an error occurs during this operation, the file content might be partially updated and the resulting file position might not be as expected.⌋

To write binary data to a file, the Persistency user may use the method ara::per::ReadWriteAccessor::WriteBinary of the ara::per::ReadWriteAccessor class. See also [SWS_PER_00557] for ara::per::OpenMode kAppend.

[SWS_PER_00531]

Upstream requirements: RS_PER_00004

⌈ara::per::ReadWriteAccessor::WriteBinary shall write the bytes provided as ara::core::Span of ara::core::Byte to the file at the current position,

possibly overwriting current content and advancing the end of the file if necessary. The position shall be set to the byte following the last byte that was written during this operation, or to the end of the file.⌋

The Persistency user may use ara::per::ReadWriteAccessor::SetFileSize to explicitly set the file size to a defined value in order to truncate a file or to empty it. Enlarging files is not supported by ara::per::ReadWriteAccessor::SetFileSize.

[SWS_PER_00532]

Upstream requirements: RS_PER_00004

⌈ara::per::ReadWriteAccessor::SetFileSize shall set the file size to the provided value. The read/write position shall be set to the end of the file if the current position is higher than the new file size. If the provided value is larger than the current file size, ara::per::ReadWriteAccessor::SetFileSize shall return kInvalidSize.⌋

When a Persistency user changed a file, Persistency will ensure that these changes are persisted. This can happen at any time, and latest when the file is closed. To trigger an additional synchronization of the file content to the persistent storage, the Persistency user may call ara::per::ReadWriteAccessor::SyncToFile.

[SWS_PER_00533]

Upstream requirements: RS_PER_00004

⌈When ara::per::ReadWriteAccessor::SyncToFile is called, Persistency shall store the content of the file. Persistency shall also update any configured redundancy within this call.⌋

Please note that depending on the implementation of Persistency, the configuration of an optionally used file system, and the capabilities of the used physical storage device, the actual storage of the file content may be delayed. So, in case of an immediate power down directly after this call, the physical data may be updated only partially or not at all. Therefore, data may be lost or, without redundancy, data may even be corrupted in this case.

The handling of redundancy is described in detail in Section 7.2.4.

7.4.1 Access to Additional Information about Files

To gain information about stored files, the Persistency provides the methods ara::per::FileStorage::GetCurrentFileSize and ara::per::FileStorage::GetFileInfo.

The Persistency user can poll the amount of storage space currently occupied by a single file using ara::per::FileStorage::GetCurrentFileSize of an open File Storage.

[SWS_PER_00549]

Upstream requirements: RS_AP_00128

⌈When ara::per::FileStorage::GetCurrentFileSize is called, Persistency shall first check whether the file is present in the File Storage, and otherwise return directly with kFileNotFound.⌋

[SWS_PER_00493]

Upstream requirements: RS_PER_00011

⌈When ara::per::FileStorage::GetCurrentFileSize is called for an existing file, Persistency shall return the current size of the passed file. This size shall reflect only the data contained in the file. In case the file is currently open, Persistency shall return the current size of the file, which might differ from the size of the file in the storage if the last changes are not yet synchronized. Otherwise, if the file is not open, Persistency shall return the size of any existing instance of the file without checking the consistency/validity of the file.⌋

Please note that administrative and redundant information is not included in the file size reported by ara::per::FileStorage::GetCurrentFileSize, while it is included in the total size of the File Storage returned by ara::per::GetCurrentFileStorageSize (see [SWS_PER_00492]).

Using ara::per::FileStorage::GetFileInfo, a Persistency user can acquire information about the time the file was created (creationTime), last modified (modificationTime), and last accessed (accessTime), and how and by whom it was created (fileCreationState) and last modified (fileModificationState).

[SWS_PER_00550]

Upstream requirements: RS_AP_00128

⌈When ara::per::FileStorage::GetFileInfo is called, Persistency shall first check whether the file is present in the File Storage, and otherwise return directly with kFileNotFound.⌋

[SWS_PER_00440]

Upstream requirements: RS_PER_00004

⌈When `ara::per::FileStorage::GetFileInfo` is called for an existing but currently not opened `file`, `Persistency` shall gather the required information from any existing instance of the `file` (without checking the consistency/validity of this `file`) into a `ara::per::FileInfo` struct and return it to the `Persistency user`.⌋

[SWS_PER_00570]

Upstream requirements: RS_PER_00004

⌈When `ara::per::FileStorage::GetFileInfo` is called for an existing and currently opened `file`, `Persistency` shall return the time when the `file` was opened for `accessTime` and `creationTime` (the latter only if the `file` was also created at the same time).

For the `modificationTime`, before the `Persistency user` wrote anything to the `file`, `Persistency` shall return the original modification time. Afterwards, `Persistency` shall return the time the `file` was last modified by the `Persistency user` or `ara::per::ReadWriteAccessor::SyncToFile` was last called.⌋

In case the `Persistency` uses a file system of the underlying OS, part of that information (like the creation or access time) can be obtained from the file system. Please note that the time stamps returned by `ara::per::FileStorage::GetFileInfo` on a closed `file` might be later than the ones returned by `Persistency` while the `file` was still opened, because `Persistency` has no control over the actual point in time at which a file system (if used) updates these time stamps.

[SWS_PER_00458]

Upstream requirements: RS_PER_00004

⌈If `creationTime`, `modificationTime`, or `accessTime` are not available, they shall be set to 0.⌋

As an example, the `accessTime` is not available for a read-only `File Storage`, and would therefore be reported as “midnight 1970-01-01”.

7.5 Functional Cluster Life-Cycle

Using `ara::core::Initialize` and `ara::core::Deinitialize` defined in [2, SWS Core], the `Adaptive Application` initializes and de-initializes all `Functional Clusters` with direct ARA interfaces (i.e. the `Adaptive Platform Foundation`).

The `Persistency user` is expected not to call any API of `Persistency` (directly or indirectly through other `Functional Clusters`) before initialization with `ara::core::Initialize` or after de-initialization with `ara::core::Deinitialize`, but `Persistency` needs to protect itself against such eventualities.

`ara::per::OpenKeyValueStorage` and `ara::per::OpenFileStorage` as well as other global functions that have the `ara::core::InstanceSpecifier` as an input parameter will check whether they are called before `ara::core::Initialize` was called or after `ara::core::Deinitialize` was called by the `Adaptive Application` (see [SWS_CORE_90021]). In this case, it is handled as violation according to [SWS_CORE_00021]. All other functions/methods are unprotected, and will therefore result in undefined behavior according to [SWS_CORE_90022].

[SWS_PER_00574] Check for Initialization

Upstream requirements: RS_PER_00018

⌈`ara::per::UpdatePersistency`, `ara::per::ResetPersistency`, `ara::per::OpenKeyValueStorage`, `ara::per::RecoverKeyValueStorage`, `ara::per::ResetKeyValueStorage`, `ara::per::GetCurrentKeyValueStorageSize`, `ara::per::OpenFileStorage`, `ara::per::RecoverAllFiles`, `ara::per::ResetAllFiles`, and `ara::per::GetCurrentFileStorageSize` shall check whether they are called before `ara::core::Initialize` was called or after `ara::core::Deinitialize` was called by the `Adaptive Application`. If not, `Persistency` shall handle this situation as a violation according to [SWS_CORE_00021].⌋

7.5.1 Startup

[SWS_PER_00408]

Upstream requirements: RS_PER_00018

⌈When `ara::core::Initialize` is called, the `Persistency` shall read in the `manifest` information and prepare the access structures to all `Key-Value Storages` and `File Storages` that are defined in the `manifest`.⌋

7.5.2 Shutdown

[SWS_PER_00409]

Upstream requirements: RS_PER_00018

⌈When `ara::core::Deinitialize` is called, the `Persistency` shall implicitly ensure that all open `files` of all `File Storages` are persisted as though `ara::per::ReadWriteAccessor::SyncToFile` was called and closed as though the `ara::per::UniqueHandle`s were destructed, and that not persisted `values` in all `Key-Value Storages` are dropped as though `ara::per::KeyValueStorage::DiscardPendingChanges` was called. Afterwards, `Persistency` shall free remaining resources that are not exposed by objects held by the `Persistency user`, including but not limited to the configuration of `Persistency`.⌋

7.6 Reporting

7.6.1 Security Events

This functional cluster does not define any security events.

7.6.2 Log Messages

This section lists all non-verbose log messages (i.e., modelled DLT messages) defined by this functional cluster.

7.6.2.1 Log Messages with Details for all Reported Errors

[SWS_PER_20001] LogMessage StorageNotFound

Status: DRAFT

⌈

Dlt-Message	StorageNotFound		
Description	The requested Key-Value Storage or File Storage is not configured in the AUTOSAR model.		
MessageId	0x80008000		
MessageType Info	DLT_LOG_ERROR		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20002] LogMessage KeyNotFound

Status: DRAFT

⌈

Dlt-Message	KeyNotFound		
Description	The provided key cannot be not found in the Key-Value Storage.		
MessageId	0x80008001		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
KeyName	Affected Key-Value Pair	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20003] LogMessage IllegalWriteAccess

Status: DRAFT

⌈

Dlt-Message	IllegalWriteAccess		
Description	Synchronizing a Key-Value Pair of the Key-Value Storage failed, or opening a file of the File Storage for writing or changing failed, because the Key-Value Storage or File Storage is configured read-only.		
MessageId	0x80008002		
MessageType Info	DLT_LOG_ERROR		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20004] LogMessage PhysicalStorageFailure

Status: DRAFT

⌈

Dlt-Message	PhysicalStorageFailure		
Description	An error occurred when accessing the physical storage, e.g. because of a corrupted file system or corrupted hardware, or because of insufficient access rights.		
MessageId	0x80008003		
MessageType Info	DLT_LOG_ERROR		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
StorageLocation	Location of the storage in the file system or the address in a flash device	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20005] LogMessage IntegrityCorrupted

Status: DRAFT

⌈

Dlt-Message	IntegrityCorrupted		
Description	The structural integrity of the Key-Value Storage or File Storage could not be established. This can happen when the internal structure of a Key-Value Storage or the metadata of a File Storage is corrupted.		
MessageId	0x80008004		
MessageType Info	DLT_LOG_ERROR		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
StorageLocation	Location of the storage in the file system or the address in a flash device	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20006] LogMessage ValidationFailed

Status: DRAFT

⌈

Dlt-Message	ValidationFailed		
Description	The validation of redundancy measures failed for a single key-value pair, for the whole Key-Value Storage, for a single file, or for the whole File Storage.		
MessageId	0x80008005		
MessageType Info	DLT_LOG_WARN		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
ElementName	Affected storage element (Key-Value Pair or file)	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20007] LogMessage EncryptionFailed

Status: DRAFT

⌈

Dlt-Message	EncryptionFailed		
Description	The encryption or decryption failed for a single key-value pair, for the whole Key-Value Storage, for a single file, or for the whole File Storage.		
MessageId	0x80008006		
MessageType Info	DLT_LOG_WARN		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
ElementName	Affected storage element (Key-Value Pair or file)	uint8 [encoding UTF-8]	
KeySlotName	Affected key slot	uint8 [encoding UTF-8]	
AlgorithmName	Applied algorithm	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20008] LogMessage DataTypeMismatch

Status: DRAFT

⌈

Dlt-Message	DataTypeMismatch		
Description	The provided data type does not match the stored data type.		
MessageId	0x80008007		
MessageType Info	DLT_LOG_ERROR		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	

▽

△

KeyName	Affected Key-Value Pair	uint8 [encoding UTF-8]	
ProvidedType	Data type used by the adaptive application to access the Key-Value Pair	uint8 [encoding UTF-8]	
StoredType	Data type of the stored Key-Value Pair	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20009] LogMessage InitValueNotAvailable

Status: DRAFT

⌈

Dlt-Message	InitValueNotAvailable		
Description	The operation could not be performed because no initial value is available.		
MessageId	0x80008008		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
ElementName	Affected storage element (Key-Value Pair or file)	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20010] LogMessage ResourceBusy

Status: DRAFT

⌈

Dlt-Message	ResourceBusy		
Description	The operation could not be performed because the resource is currently busy.		
MessageId	0x80008009		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FunctionName	Affected function or method of the Persistency API	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20011] LogMessage OutOfStorageSpace

Status: DRAFT

⌈

Dlt-Message	OutOfStorageSpace		
Description	The physical storage space was exceeded.		
MessageId	0x8000800a		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20012] LogMessage FileNotFound

Status: DRAFT

⌈

Dlt-Message	FileNotFound		
Description	The requested file name cannot be not found in the File Storage.		
MessageId	0x8000800b		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20013] LogMessage InvalidPosition

Status: DRAFT

⌈

Dlt-Message	InvalidPosition		
Description	SetPosition tried to move to a position that is not reachable (i.e. which is smaller than zero or greater than the current size of the file).		
MessageId	0x8000800c		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	
Position	Read/write position in the file	uint64	

⌋

[SWS_PER_20014] LogMessage Eof

Status: DRAFT

⌈

Dlt-Message	Eof		
Description	The Persistency user tried to read from the end of the file or from an empty file.		
MessageId	0x8000800d		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	
Position	Read/write position in the file	uint64	

⌋

[SWS_PER_20015] LogMessage InvalidOpenMode

Status: DRAFT

⌈

Dlt-Message	InvalidOpenMode		
Description	Opening a file failed because the requested combination of Open Modes is invalid.		
MessageId	0x8000800e		
MessageType Info	DLT_LOG_ERROR		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	
OpenMode	Requested Open Mode for the file	uint8	

⌋

[SWS_PER_20016] LogMessage InvalidSize

Status: DRAFT

⌈

Dlt-Message	InvalidSize		
Description	SetFileSize tried to set a new size that is bigger than the current file size.		
MessageId	0x8000800f		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	
FileSize	Current size of the file	uint64	

⌋

[SWS_PER_20017] LogMessage TooManyFiles

Status: DRAFT

⌈

Dlt-Message	TooManyFiles		
Description	The maximum number of files was exceeded.		
MessageId	0x80008010		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	
FileNumber	Maximum allowed number of files	uint32	

⌋

[SWS_PER_20018] LogMessage QuotaExceeded

Status: DRAFT

⌈

Dlt-Message	QuotaExceeded		
Description	The allocated storage quota was exceeded.		
MessageId	0x80008011		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
Quota	Maximum allowed size of the storage	uint64	

⌋

[SWS_PER_20019] LogMessage AuthenticationFailed

Status: DRAFT

⌈

Dlt-Message	AuthenticationFailed		
Description	Calculating or checking of the MAC failed for a single key-value pair, for the whole Key-Value Storage, for a single file, or for the whole File Storage.		
MessageId	0x80008012		
MessageType Info	DLT_LOG_WARN		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
ElementName	Affected storage element (Key-Value Pair or file)	uint8 [encoding UTF-8]	
KeySlotName	Affected key slot	uint8 [encoding UTF-8]	

▽

△

AlgorithmName	Applied algorithm	uint8 [encoding UTF-8]	

⌋

7.6.2.2 Log Messages for Reported Redundancy Loss

[SWS_PER_20020] LogMessage KeyValueStorageRecoveryFailed

Status: DRAFT

⌈

Dlt-Message	KeyValueStorageRecoveryFailed		
Description	A Key-Value Storage was corrupted, an insufficient number of valid copies existed. storage contains the short-name path of the Key-Value Storage, reportedElements is empty, reportedInstances contains the indices of the affected Key-Value Storage copies.		
MessageId	0x80008013		
MessageType Info	DLT_LOG_WARN		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
InstanceIndices	Set of affected instances	uint8 [42]	

⌋

[SWS_PER_20021] LogMessage KeyValueStorageRecovered

Status: DRAFT

⌈

Dlt-Message	KeyValueStorageRecovered		
Description	A Key-Value Storage was corrupted, but a sufficient number of valid copies existed. storage contains the short-name path of the Key-Value Storage, reportedElements is empty, reportedInstances contains the indices of the affected Key-Value Storage copies.		
MessageId	0x80008014		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
InstanceIndices	Set of affected instances	uint8 [42]	

⌋

[SWS_PER_20022] LogMessage KeyRecoveryFailed

Status: DRAFT

⌈

Dlt-Message	KeyRecoveryFailed		
Description	A set of key-value pairs was corrupted, an insufficient number of valid copies existed. storage contains the short-name path of the Key-Value Storage, reportedElements contains the list of affected keys, reportedInstances contains the indices of the affected Key-Value Storage or key-value pair copies. In general, the nth key in reportedElements corresponds to the nth index in reported Instances, i.e. a key may be reported several times if several copies are broken. In case only one key-value pair is affected, reported Elements may be provided containing just this key.		
MessageId	0x80008015		
MessageType Info	DLT_LOG_WARN		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
KeyNames	Set of affected Key-Value Pairs	uint8 [42, encoding UTF-8]	
InstanceIndices	Set of affected instances	uint8 [42]	

⌋

[SWS_PER_20023] LogMessage KeyRecovered

Status: DRAFT

⌈

Dlt-Message	KeyRecovered		
Description	A set of key-value pairs was corrupted, but a sufficient number of valid copies existed. storage contains the short-name path of the Key-Value Storage, reportedElements contains the list of affected keys, reportedInstances contains the indices of the affected Key-Value Storage or key-value pair copies. In general, the nth key in reportedElements corresponds to the nth index in reported Instances, i.e. a key may be reported several times if several copies are broken. In case only one key-value pair is affected, reported Elements may be provided containing just this key.		
MessageId	0x80008016		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
KeyNames	Set of affected Key-Value Pairs	uint8 [42, encoding UTF-8]	
InstanceIndices	Set of affected instances	uint8 [42]	

⌋

[SWS_PER_20024] LogMessage FileStorageRecoveryFailed

Status: DRAFT

⌈

Dlt-Message	FileStorageRecoveryFailed		
Description	A File Storage was corrupted, an insufficient number of valid copies existed. storage contains the short-name path of the File Storage, reportedElements is empty, reportedInstances contains the indices of the affected File Storage copies.		
MessageId	0x80008017		
MessageType Info	DLT_LOG_WARN		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
InstanceIndices	Set of affected instances	uint8 [42]	

⌋

[SWS_PER_20025] LogMessage FileStorageRecovered

Status: DRAFT

⌈

Dlt-Message	FileStorageRecovered		
Description	A File Storage was corrupted, but a sufficient number of valid copies existed. storage contains the short-name path of the File Storage, reportedElements is empty, reportedInstances contains the indices of the affected File Storage copies.		
MessageId	0x80008018		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
InstanceIndices	Set of affected instances	uint8 [42]	

⌋

[SWS_PER_20026] LogMessage FileRecoveryFailed

Status: DRAFT

⌈

Dlt-Message	FileRecoveryFailed
Description	A set of files was corrupted, an insufficient number of valid copies existed. storage contains the short-name path of the File Storage, reportedElements contains the list of affected file names, reported Instances contains the indices of the affected File Storage or file copies. In general, the nth file name in reportedElements corresponds to the nth index in reportedInstances, i.e. a file name may be reported several times if several copies are broken. In case only one file is affected, reportedElements may be provided containing just this file name.
MessageId	0x80008019
MessageType Info	DLT_LOG_WARN

▽

△

Dlt-Argument	*ArgumentDescription*	*ArgumentType*	*ArgumentUnit*
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileNames	Set of affected files	uint8 [42, encoding UTF-8]	
InstanceIndices	Set of affected instances	uint8 [42]	

⌋

[SWS_PER_20027] LogMessage FileRecovered

Status: DRAFT

⌈

Dlt-Message	FileRecovered		
Description	A set of files was corrupted, but a sufficient number of valid copies existed. storage contains the short-name path of the File Storage, reportedElements contains the list of affected file names, reported Instances contains the indices of the affected File Storage or file copies. In general, the nth file name in reportedElements corresponds to the nth index in reportedInstances, i.e. a file name may be reported several times if several copies are broken. In case only one file is affected, reportedElements may be provided containing just this file name.		
MessageId	0x8000801a		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	*ArgumentDescription*	*ArgumentType*	*ArgumentUnit*
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileNames	Set of affected files	uint8 [42, encoding UTF-8]	
InstanceIndices	Set of affected instances	uint8 [42]	

⌋

7.6.2.3 Log Messages for Potentially Unexpected Behavior

[SWS_PER_20031] LogMessage KeyImplicitlyDeleted

Status: DRAFT

⌈

Dlt-Message	KeyImplicitlyDeleted		
Description	A Key-Value Pair was deleted during recovery.		
MessageId	0x8000801b		
MessageType Info	DLT_LOG_WARN		
Dlt-Argument	*ArgumentDescription*	*ArgumentType*	*ArgumentUnit*
StorageName	Affected storage	uint8 [encoding UTF-8]	

▽

△

KeyName	Affected Key-Value Pair	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20032] LogMessage KeyImplicitlyReset

Status: DRAFT

⌈

Dlt-Message	KeyImplicitlyReset		
Description	A Key-Value Pair was set to its initial value during recovery.		
MessageId	0x8000801c		
MessageType Info	DLT_LOG_WARN		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
KeyName	Affected Key-Value Pair	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20033] LogMessage FileImplicitlyDeleted

Status: DRAFT

⌈

Dlt-Message	FileImplicitlyDeleted		
Description	A file was deleted during recovery.		
MessageId	0x8000801d		
MessageType Info	DLT_LOG_WARN		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20034] LogMessage FileImplicitlyReset

Status: DRAFT

⌈

Dlt-Message	FileImplicitlyReset
Description	A file was set to its initial value during recovery.
MessageId	0x8000801e
MessageType Info	DLT_LOG_WARN

▽

△

Dlt-Argument	*ArgumentDescription*	*ArgumentType*	*ArgumentUnit*
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	

⌋

7.6.2.4 Log Messages for State Changes

[SWS_PER_20035] LogMessage PersistencyInitialized

Status: DRAFT

⌈

Dlt-Message	PersistencyInitialized		
Description	Persistency was initialized via ara::core::Initialize.		
MessageId	0x8000801f		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	*ArgumentDescription*	*ArgumentType*	*ArgumentUnit*
No Dlt-arguments available			

⌋

[SWS_PER_20036] LogMessage PersistencyDeInitialized

Status: DRAFT

⌈

Dlt-Message	PersistencyDeInitialized		
Description	Persistency was de-initialized via ara::core::Deinitialize.		
MessageId	0x80008020		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	*ArgumentDescription*	*ArgumentType*	*ArgumentUnit*
No Dlt-arguments available			

⌋

7.6.2.5 Log Messages for Storage Access

[SWS_PER_20037] LogMessage ConfigAccessed

Status: DRAFT

⌈

Dlt-Message	ConfigAccessed		
Description	The Persistency configuration was accessed.		
MessageId	0x80008021		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
ConfigLocation	Location of the configuration in the file system or the address in a flash device	uint8 [encoding UTF-8]	
ContractVersion	Contract version of Persistency in the configuration	uint8 [encoding UTF-8]	
Deployment Version	Deployment version of Persistency in the configuration	uint8 [encoding UTF-8]	
CheckSum	Checksum of the configuration, 0 if not available	uint64	

⌋

[SWS_PER_20038] LogMessage KvsConfigAccessed

Status: DRAFT

⌈

Dlt-Message	KvsConfigAccessed		
Description	A Key-Value Storage configuration was accessed.		
MessageId	0x80008022		
MessageType Info	DLT_LOG_VERBOSE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
IsWriteable	The storage is writable	boolean	
Copies	Number of configured copies of the storage	uint8	
HasCRC	The storage uses a CRC	boolean	
HasHash	The storage uses a Hash	boolean	
IsEncrypted	The storage is encrypted	boolean	
IsAuthenticated	The storage uses MAC based authentication	boolean	
HasPortAccess	The storage was configured for access by an adaptive application	boolean	

⌋

[SWS_PER_20039] LogMessage FsConfigAccessed

Status: DRAFT

⌈

Dlt-Message	FsConfigAccessed		
Description	A File Storage configuration was accessed.		
MessageId	0x80008023		
MessageType Info	DLT_LOG_VERBOSE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
IsWriteable	The storage is writable	boolean	
Copies	Number of configured copies of the storage	uint8	
HasCRC	The storage uses a CRC	boolean	
HasHash	The storage uses a Hash	boolean	
IsEncrypted	The storage is encrypted	boolean	
IsAuthenticated	The storage uses MAC based authentication	boolean	
HasPortAccess	The storage was configured for access by an adaptive application	boolean	

⌋

[SWS_PER_20040] LogMessage CentralLocationAccessed

Status: DRAFT

⌈

Dlt-Message	CentralLocationAccessed		
Description	The central storage location of Persistency was accessed.		
MessageId	0x80008024		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
Location	Location of the central storage in the file system or the address in a flash device	uint8 [encoding UTF-8]	
ContractVersion	Stored contract version of Persistency	uint8 [encoding UTF-8]	
Deployment Version	Stored deployment version of Persistency	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20041] LogMessage KvsLocationAccessed

Status: DRAFT

⌈

Dlt-Message	KvsLocationAccessed		
Description	A Key-Value Storage location was accessed.		
MessageId	0x80008025		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
KvsLocations	Set of locations of copies of the storage in the file system or the addresses in a flash device	uint8 [42, encoding UTF-8]	
StorageSize	Current size of the storage	uint64	
Keys	Current number of Key-Value Pairs in the storage	uint32	

⌋

[SWS_PER_20042] LogMessage KvsAccessShared

Status: DRAFT

⌈

Dlt-Message	KvsAccessShared		
Description	A handle to a Key-Value Storage was shared.		
MessageId	0x80008026		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20043] LogMessage KvsAccessFinished

Status: DRAFT

⌈

Dlt-Message	KvsAccessFinished		
Description	A Key-Value Storage was closed.		
MessageId	0x80008027		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20044] LogMessage ValueChanged

Status: DRAFT

⌈

Dlt-Message	ValueChanged		
Description	The value of a Key-Value Pair changed.		
MessageId	0x80008028		
MessageType Info	DLT_LOG_VERBOSE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
KeyName	Affected Key-Value Pair	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20045] LogMessage KeyCreated

Status: DRAFT

⌈

Dlt-Message	KeyCreated		
Description	A Key-Value Pair was created.		
MessageId	0x80008029		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
KeyName	Affected Key-Value Pair	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20046] LogMessage KeyDeleted

Status: DRAFT

⌈

Dlt-Message	KeyDeleted		
Description	A Key-Value Pair was deleted.		
MessageId	0x8000802a		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
KeyName	Affected Key-Value Pair	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20047] LogMessage KeyAccessed

Status: DRAFT

⌈

Dlt-Message	KeyAccessed		
Description	A Key-Value Pair was accessed.		
MessageId	0x8000802b		
MessageType Info	DLT_LOG_VERBOSE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
KeyName	Affected Key-Value Pair	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20048] LogMessage FsLocationAccessed

Status: DRAFT

⌈

Dlt-Message	FsLocationAccessed		
Description	A File Storage location was accessed.		
MessageId	0x8000802c		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FsLocations	Set of locations of copies of the storage in the file system or the addresses in a flash device	uint8 [42, encoding UTF-8]	
StorageSize	Current size of the storage	uint64	
Files	Current number of files in the storage	uint32	

⌋

[SWS_PER_20049] LogMessage FsAccessShared

Status: DRAFT

⌈

Dlt-Message	FsAccessShared		
Description	A handle to a File Storage was shared.		
MessageId	0x8000802d		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20050] LogMessage FsAccessFinished

Status: DRAFT

⌈

Dlt-Message	FsAccessFinished		
Description	A File Storage was closed.		
MessageId	0x8000802e		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
StorageSize	Current size of the storage	uint64	
Files	Current number of files in the storage	uint32	

⌋

[SWS_PER_20051] LogMessage FileLocationAccessed

Status: DRAFT

⌈

Dlt-Message	FileLocationAccessed		
Description	A file location was accessed.		
MessageId	0x8000802f		
MessageType Info	DLT_LOG_VERBOSE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	
OpenMode	Requested Open Mode for the file	uint8	
FileSize	Current size of the file	uint64	

⌋

[SWS_PER_20052] LogMessage FileCreated

Status: DRAFT

⌈

Dlt-Message	FileCreated		
Description	A file was created.		
MessageId	0x80008030		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20053] LogMessage FileDeleted

Status: DRAFT

⌈

Dlt-Message	FileDeleted		
Description	A file was deleted.		
MessageId	0x80008031		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20054] LogMessage FileResized

Status: DRAFT

⌈

Dlt-Message	FileResized		
Description	A file was resized, either explicitly by a call to SetFileSize or implicitly while writing to a file.		
MessageId	0x80008032		
MessageType Info	DLT_LOG_VERBOSE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	
FileSize	Current size of the file	uint64	

⌋

[SWS_PER_20055] LogMessage FileRead

Status: DRAFT

⌈

Dlt-Message	FileRead		
Description	Data was read from a file.		
MessageId	0x80008033		
MessageType Info	DLT_LOG_VERBOSE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	
Position	Read/write position in the file	uint64	
BytesRead	Bytes of data actually read from the file	uint64	

▽

△

IsBinary	Binary access used	boolean	

⌋

[SWS_PER_20056] LogMessage FileWritten

Status: DRAFT

⌈

Dlt-Message	FileWritten		
Description	Data was written to a file.		
MessageId	0x80008034		
MessageType Info	DLT_LOG_VERBOSE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	
Position	Read/write position in the file	uint64	
BytesWritten	Bytes of data actually written to the file	uint64	
IsBinary	Binary access used	boolean	

⌋

[SWS_PER_20057] LogMessage FilePosition

Status: DRAFT

⌈

Dlt-Message	FilePosition		
Description	The current position of a file was changed.		
MessageId	0x80008035		
MessageType Info	DLT_LOG_VERBOSE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	
Position	Read/write position in the file	uint64	

⌋

7.6.2.6 Log Messages for Synchronization

[SWS_PER_20058] LogMessage SyncToFile

Status: DRAFT

⌈

Dlt-Message	SyncToFile		
Description	A file was synchronized to the physical device. Note: Also reported by ara:core::Deinitialize.		
MessageId	0x80008036		
MessageType Info	DLT_LOG_VERBOSE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20059] LogMessage SyncToStorage

Status: DRAFT

⌈

Dlt-Message	SyncToStorage		
Description	A Key-Value Storage was synchronized to the physical device.		
MessageId	0x80008037		
MessageType Info	DLT_LOG_VERBOSE		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
SyncedKeys	Set of updated Key-Value Pairs	uint8 [42, encoding UTF-8]	
StorageSize	Current size of the storage	uint64	
Keys	Current number of Key-Value Pairs in the storage	uint32	

⌋

[SWS_PER_20060] LogMessage DiscardPendingChanges

Status: DRAFT

⌈

Dlt-Message	DiscardPendingChanges
Description	Unsynchronized changes of a Key-Value Storage were reverted. Note: Also reported by ara:core::Deinitialize.
MessageId	0x80008038
MessageType Info	DLT_LOG_VERBOSE

▽

△

Dlt-Argument	*ArgumentDescription*	*ArgumentType*	*ArgumentUnit*
StorageName	Affected storage	uint8 [encoding UTF-8]	
DiscardedKeys	Set of changed Key-Value Pairs that were reverted	uint8 [42, encoding UTF-8]	

⌋

7.6.2.7 Log Messages for Information about Updates

[SWS_PER_20061] LogMessage PersistencyInstalled

Status: DRAFT

⌈

Dlt-Message	PersistencyInstalled		
Description	An adaptive application using Persistency was installed.		
MessageId	0x80008039		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	*ArgumentDescription*	*ArgumentType*	*ArgumentUnit*
No Dlt-arguments available			

⌋

[SWS_PER_20062] LogMessage PersistencyUpdated

Status: DRAFT

⌈

Dlt-Message	PersistencyUpdated		
Description	The Persistency configuration of an adaptive application was updated.		
MessageId	0x8000803a		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	*ArgumentDescription*	*ArgumentType*	*ArgumentUnit*
No Dlt-arguments available			

⌋

[SWS_PER_20063] LogMessage PersistencyRolledBack

Status: DRAFT

⌈

Dlt-Message	PersistencyRolledBack		
Description	The Persistency of an adaptive application was rolled back.		
MessageId	0x8000803b		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
No Dlt-arguments available			

⌋

[SWS_PER_20064] LogMessage PersistencyReset

Status: DRAFT

⌈

Dlt-Message	PersistencyReset		
Description	All Persistency storages of an adaptive application were reset.		
MessageId	0x8000803c		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
No Dlt-arguments available			

⌋

[SWS_PER_20065] LogMessage KvsInstalled

Status: DRAFT

⌈

Dlt-Message	KvsInstalled		
Description	A Key-Value Storage was added during installation or update of an adaptive application.		
MessageId	0x8000803d		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20066] LogMessage KvsRemoved

Status: DRAFT

⌈

Dlt-Message	KvsRemoved		
Description	A Key-Value Storage was removed during update of an adaptive application.		
MessageId	0x8000803e		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20067] LogMessage KvsReset

Status: DRAFT

⌈

Dlt-Message	KvsReset		
Description	A Key-Value Storage was reset.		
MessageId	0x8000803f		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20068] LogMessage KeyReset

Status: DRAFT

⌈

Dlt-Message	KeyReset		
Description	A Key-Value Pair was reset.		
MessageId	0x80008040		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
KeyName	Affected Key-Value Pair	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20069] LogMessage FsInstalled

Status: DRAFT

⌈

Dlt-Message	FsInstalled		
Description	A File Storage was added during installation or update of an adaptive application.		
MessageId	0x80008041		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20070] LogMessage FsRemoved

Status: DRAFT

⌈

Dlt-Message	FsRemoved		
Description	A File Storage was removed during update of an adaptive application.		
MessageId	0x80008042		
MessageType Info	DLT_LOG_INFO		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20071] LogMessage FsReset

Status: DRAFT

⌈

Dlt-Message	FsReset		
Description	A File Storage was reset.		
MessageId	0x80008043		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	

⌋

[SWS_PER_20072] LogMessage FileReset

Status: DRAFT

⌈

Dlt-Message	FileReset		
Description	A file was reset.		
MessageId	0x80008044		
MessageType Info	DLT_LOG_DEBUG		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
StorageName	Affected storage	uint8 [encoding UTF-8]	
FileName	Affected file	uint8 [encoding UTF-8]	

⌋

Some of the log messages that are logged during ordinary access to `Persistency` (see Section 7.6.2.5) are also logged during installation or update of `Persistency`, namely `CentralLocationAccessed`, `KvsLocationAccessed`, `ValueChanged`, `KeyCreated`, `KeyDeleted`, `FsLocationAccessed`, `FileLocationAccessed`, `FileDeleted`, `FileWritten`.

7.6.3 Violation Messages

This section lists all violation messages (i.e., DLT messages logged for Violations according to [SWS_CORE_00021]) defined by this functional cluster.

7.6.3.1 OutOfMemory

[SWS_PER_20028] ViolationMessage OutOfMemoryViolation

Status: DRAFT

⌈

Dlt-Message	OutOfMemoryViolation		
Description	Memory allocation failed. String format: "Violation detected in {processIdentifier} at {location} where the memory allocation of the affected {VariableName} with size of {Size} failed.".		
MessageId	0x80008fff		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
processIdentifier	Meta-model identifier of the process that caused the violation, i.e. short name path with '/' as a separator.	uint8 [encoding UTF-8]	NoUnit

▽

△

location	An implementation-defined identifier of the location where the violation was detected, for example \\\{filename}:\\\{linenumber}.	uint8 [encoding UTF-8]	NoUnit
VariableName	Affected variable	uint8 [encoding UTF-8]	
Size	Size of the variable	uint64	

⌋

7.6.3.2 CalledWhileUnititialized

[SWS_PER_20029] ViolationMessage CalledWhileUnititializedViolation

Status: DRAFT

⌈

Dlt-Message	CalledWhileUnititializedViolation		
Description	A named constructor or global function of the Persistency API was called before ara::core::Initialize. String format: "Violation detected in {processIdentifier} at {location} in function or method {Function Name} of the Persistency API.".		
MessageId	0x80008ffe		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
processIdentifier	Meta-model identifier of the process that caused the violation, i.e. short name path with '/' as a separator.	uint8 [encoding UTF-8]	NoUnit
location	An implementation-defined identifier of the location where the violation was detected, for example \\\{filename}:\\\{linenumber}.	uint8 [encoding UTF-8]	NoUnit
FunctionName	Affected function or method of the Persistency API	uint8 [encoding UTF-8]	

⌋

7.6.3.3 InvalidConfiguration

[SWS_PER_20030] ViolationMessage InvalidConfigurationViolation

Status: DRAFT

⌈

Dlt-Message	InvalidConfigurationViolation
Description	The installed Persistency configuration is unreadable, most likely it is corrupted. String format: "Violation detected in {processIdentifier} at {location}:"
MessageId	0x80008ffd

▽

△

MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
processIdentifier	Meta-model identifier of the process that caused the violation, i.e. short name path with '/' as a separator.	uint8 [encoding UTF-8]	NoUnit
location	An implementation-defined identifier of the location where the violation was detected, for example \\\{filename}:\\\{linenumber}.	uint8 [encoding UTF-8]	NoUnit

⌋

7.6.4 Production Errors

This section lists all production errors (i.e., Diagnostic Events) defined by this functional cluster.

7.6.4.1 PER_E_HARDWARE

[SWS_PER_10001] Diagnostic Event: Reading from or writing to the Storage failed ⌈

Diagnostic Event (Error Name)	PER_E_HARDWARE
Description	Reading from or writing to the Storage failed
Failed condition	Persistency returns kPhysicalStorageFailure from any executed function or method, or a daemon has problems accessing the stored data.
Passed condition	Persistency executes a function or method successfully that can report kPhysical StorageFailure, or a daemon accessed the stored data successfully.

⌋

7.6.4.2 PER_E_INTEGRITY_FAILED

[SWS_PER_10002] Diagnostic Event: The structure of the Storage was destroyed ⌈

Diagnostic Event (Error Name)	PER_E_INTEGRITY_FAILED
Description	The structure of the Storage was destroyed
Failed condition	Persistency returns kIntegrityCorrupted from any executed function or method, or a daemon detects a structural corruption in the stored data.

▽

△

Passed condition	Persistency executes a function or method successfully that can report kIntegrity Corrupted, or a daemon accessed the stored data successfully.

⌋

7.6.4.3 PER_E_LOSS_OF_REDUNDANCY

[SWS_PER_10003] Diagnostic Event: Redundancy of the Storage was lost ⌈

Diagnostic Event (Error Name)	PER_E_LOSS_OF_REDUNDANCY
Description	Redundancy of the Storage was lost
Failed condition	Persistency returns kValidationFailed from any executed function or method, or a daemon detects loss of redundancy of the stored data.
Passed condition	Persistency executes a function or method successfully that can report kValidation Failed, or a daemon accessed the stored data successfully.

⌋

8 API Specification

This chapter provides a reference of the APIs defined by this functional cluster. The API is described in the following chapters in tables. Table 8.1 explains the content that is described in such an API table.

Kind:	Defines the kind of the declaration that this API table describes. The following values are supported: • class (Declaration of a class) • function (Declaration of a member or non-member function) • struct (Declaration of a structure) • type alias (Declaration of a type alias) • enumeration (Declaration of an enumeration) • variable (Declaration of a variable)	
Header File:	Defines the header file to be included according to [SWS_CORE_90001]	
Forwarding Header File:	Defines the forwarding header file to be included according to [SWS_CORE_90001]	
Scope:	Defines the scope that may be a namespace (in case of a class or non-member function) or a class declaration (in case of a member)	
Symbol:	Entity name	
Thread Safety:	Defines whether a function is thread-safe, not thread-safe, or conditional according to [SWS_CORE_13200] and [SWS_CORE_13202]	
Syntax:	Description of C++ syntax	
Template Param:	Template parameter (0..*)	Template parameter(s) used to parametrize the template
Parameters (in):	Parameter declaration (0..*)	Parameter(s) that are passed to the function
Parameters (out):	Parameter declaration (0..*)	Parameter(s) that are returned to the caller
Return Value:	Return type	Type of the value that the function returns
Exception Safety:	Defines whether a function is exception-safe, not exception safe or conditionally exception safe	
Exceptions:	List of exceptions that may be thrown from the function	
Violations:	List of violations that may occur in the function	
Errors:	Error type (0..*)	List of defined error codes that may be returned by the function with their recoverability class defined in [RS_AP_00160]. APIs can be extended with vendor-specific error codes. These are not part of the AUTOSAR SWS specifications
Description:	Brief description of the function	

Table 8.1: Explanation of an API table

The APIs for accessing `File Storages` and `Key-Value Storage` are completely separate, and therefore divided into separate sections. Additional sections describe common functionality.

The API of `Persistency` is designed around the `ara::per::SharedHandle` and `ara::per::UniqueHandle`, which are returned by factory functions like `ara::per::OpenKeyValueStorage` or `ara::per::FileStorage::OpenFileReadWrite`. The classes defined in this chapter cannot be constructed directly by the `Persistency user`, and consequently the default constructors are considered to be not publicly accessible (i.e. to be deleted, private, or protected).

8.1 General Features of Persistency

8.1.1 ara::core Types

The `ara::per` API is based heavily on the `ara::core` types defined in [2].

`ara::core::Result` is used wherever possible, and because of this, most methods are defined as `noexcept`.

Consequently, in situations where memory cannot be allocated for new objects, the `Persistency` shall terminate the process by calling `ara::core::Abort` (see [2]).

8.1.2 Installation and Update of Persistent Data

The Persistency installs/updates its Key-Value Storages and File Storages either during the validation of a freshly installed/updated Adaptive Application in the ara::per::UpdatePersistency call, or when the storages are opened for the first time. It allows to monitor these operations with the callback functions installed with ara::per::RegisterDataUpdateIndication and ara::per::RegisterApplicationDataUpdateCallback.

The Adaptive Application can also monitor the manifest for an independent update using ara::per::CheckForManifestUpdate and then tell Persistency to re-read the manifest with ara::per::ReloadPersistencyManifest. The actual update will then happen when a Storage is closed and re-openend.

The Adaptive Application may also restore the Persistency to the initial state (i.e. the state that it would have if it was installed with the current manifest) with a call to ara::per::ResetPersistency.

8.1.2.1 RegisterDataUpdateIndication

[SWS_PER_00578] Definition of API function ara::per::RegisterDataUpdateIndication

Upstream requirements: RS_PER_00013, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00135, RS_AP_00137, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/update.h"	
Scope:	`namespace ara::per`	
Syntax:	`void RegisterDataUpdateIndication (std::function< void(const ara::core::InstanceSpecifier &updatedStorage, ara::core::StringView updatedElement)> dataUpdateIndication, ara::core::InstanceSpecifier monitoredStorage, ara::core::Vector< ara::core::String > monitoredElements) noexcept;`	
Parameters (in):	dataUpdateIndication	The callback function to be called by Persistency after an update of persistent data took place. The function will be called with the short Name path of an updated Key-Value Storage or File Storage, and with the the updated elements of the storage.
	monitoredStorage	A Key-Value Storage or File Storage for which updates shall be reported.
	monitoredElements	A set of elements for which updates shall be reported.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	

▽

△

Description:	Registers a data update indication callback with Persistency. The provided callback function will be called by Persistency when an update of stored data happened. It will be called from the context of UpdatePersistency(), OpenKeyValueStorage(), or OpenFileStorage().

⌋

8.1.2.2 RegisterApplicationDataUpdateCallback

[SWS_PER_00356] Definition of API function ara::per::RegisterApplicationDataUpdateCallback

Upstream requirements: RS_PER_00013, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00135, RS_AP_00137, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/update.h"	
Scope:	`namespace ara::per`	
Syntax:	`void RegisterApplicationDataUpdateCallback (std::function< void(const ara::core::InstanceSpecifier &storage, ara::core::String contract Version)> 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 short Name path of an updated Key-Value Storage or File Storage, and with the contract version with which the Persistency was last accessed.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Registers 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 contract versions. The contract version that last accessed Persistency is provided as an argument to the callback, as well as the InstanceSpecifier referring to the updated Key-Value Storage or File Storage. Based on this information, the Adaptive Application can decide which updates are actually necessary, e.g. a migration from any older contract version could be supported, with different steps required for each of these. The provided function will be called from the context of UpdatePersistency(), OpenKeyValue Storage(), or OpenFileStorage().	

⌋

8.1.2.3 UpdatePersistency

[SWS_PER_00357] Definition of API function ara::per::UpdatePersistency

Upstream requirements: RS_PER_00013, RS_AP_00119, RS_AP_00120, RS_AP_00127, RS_AP_00128, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/update.h"	
Scope:	`namespace ara::per`	
Syntax:	`ara::core::Result< void > UpdatePersistency () noexcept;`	
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails during the update operation.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the encryption or decryption of stored data fails during the update operation.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if CleanUpPersistency or ResetPersistency is currently being executed, or if RecoverKeyValueStorage or ResetKeyValue Storage is currently being executed for any Key-Value Storage, or if RecoverAllFiles or ResetAllFiles is currently being executed for any File Storage, or a SharedHandle of a Key-Value Storage or a File Storage is currently in use.
	PerErrc::kOutOfStorage Space	rollback_semantics
		Returned if the available physical storage space is insufficient for the update.
	PerErrc::kTooManyFiles	rollback_semantics
		Returned if the files added during the update of any File Storage would exceed the configured maxNumberOfFiles.
	PerErrc::kQuota Exceeded	rollback_semantics
		Returned if the update would exceed the configured maximum AllowedSize of any Key-Value Storage or File Storage.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if calculating or checking the MAC of stored data fails.

▽

△

Description:	Updates all Persistency Key-Value Storages and File Storages after a new manifest was installed. This method can be used to update the persistent data of the Adaptive Application during verification phase.

⌋

8.1.2.4 CleanUpPersistency

[SWS_PER_00567] Definition of API function ara::per::CleanUpPersistency

Upstream requirements: RS_PER_00016, RS_AP_00119, RS_AP_00120, RS_AP_00127, RS_AP_00128, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/update.h"	
Scope:	`namespace ara::per`	
Syntax:	`ara::core::Result< void > CleanUpPersistency () noexcept;`	
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails during the update operation.
	PerErrc::kIntegrityCorrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidationFailed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the encryption or decryption of stored data fails during the update operation.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if UpdatePersistency or ResetPersistency is currently being executed, or if RecoverKeyValueStorage or ResetKeyValue Storage is currently being executed for any Key-Value Storage, or if RecoverAllFiles or ResetAllFiles is currently being executed for any File Storage, or a SharedHandle of a Key-Value Storage or a File Storage is currently in use.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if calculating or checking the MAC of stored data fails.
Description:	Removes backup data and other unused data of all Persistency Key-Value Storages and File Storages.	

⌋

8.1.2.5 ResetPersistency

[SWS_PER_00358] Definition of API function ara::per::ResetPersistency

Upstream requirements: RS_PER_00009, RS_AP_00119, RS_AP_00120, RS_AP_00127, RS_AP_00128, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/update.h"	
Scope:	`namespace ara::per`	
Syntax:	`ara::core::Result< void > ResetPersistency () noexcept;`	
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails during the reset operation.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if UpdatePersistency or CleanUpPersistency is currently being executed, or if RecoverKeyValueStorage or ResetKeyValue Storage is currently being executed for any Key-Value Storage, or if RecoverAllFiles or ResetAllFiles is currently being executed for any File Storage, or a SharedHandle of a Key-Value Storage or a File Storage is currently in use.
Description:	Resets all Key-Value Storages and File Storages by entirely removing their content. The Key-Value Storages and File Storages will be re-created when OpenFileStorage or Open KeyValueStorage is called next time.	

⌋

8.1.2.6 CheckForManifestUpdate

[SWS_PER_00576] Definition of API function ara::per::CheckForManifestUpdate

Upstream requirements: RS_PER_00013, RS_AP_00119, RS_AP_00120, RS_AP_00127, RS_AP_00128, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function
Header file:	#include "ara/per/update.h"
Scope:	`namespace ara::per`
Syntax:	`ara::core::Result< bool > CheckForManifestUpdate () noexcept;`

▽

△

Return value:	ara::core::Result< bool >	A Result containing true if the manifest was updated or false if it is unchanged since the last time the manifest was read. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails during the checking operation.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
Description:	Checks for updates of the Persistency manifest. This method can be used to detect configuration updates that are installed by UCM while the Adaptive Application that uses Persistency is running. To re-read the updated manifest, the Adaptive Application needs to call ReloadPersistencyManifest.	

⌋

8.1.2.7 ReloadPersistencyManifest

[SWS_PER_00577] Definition of API function ara::per::ReloadPersistencyManifest

Upstream requirements: RS_PER_00013, RS_AP_00119, RS_AP_00120, RS_AP_00127, RS_AP_00128, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/update.h"	
Scope:	`namespace ara::per`	
Syntax:	`ara::core::Result< void > ReloadPersistencyManifest () noexcept;`	
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails during the read operation.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.

▽

△

Description:	Re-reads the manifest. This method can be used to prepare for updates of the Key-Value Storages and File Storages of Persistency after a new manifest was installed by UCM while the Adaptive Application that uses Persistency is running. Storages are only updated after closing them, either when they are opened again or by an explicit call to UpdatePersistency.

⌋

8.1.3 Redundancy Handling

The Persistency supports redundant storage of Key-Value Storages, File Storages, and the key-value pairs and files contained in these. An error in the stored data that can be fixed using the redundantly stored data will be implicitly fixed when the Key-Value Storage or File Storage is accessed, an error is only returned by Persistency when the redundancy fails. To be able to track whether storage errors have been fixed using the available redundancy, the Adaptive Application can register the following callback function.

8.1.3.1 RecoveryReportKind

[SWS_PER_00432] Definition of API enum ara::per::RecoveryReportKind

Upstream requirements: RS_PER_00008, RS_AP_00122, RS_AP_00125, RS_AP_00143

⌈

Kind:	enumeration	
Header file:	#include "ara/per/recovery.h"	
Forwarding header file:	#include "ara/per/per_fwd.h"	
Scope:	`namespace ara::per`	
Symbol:	RecoveryReportKind	
Underlying type:	std::uint32_t	
Syntax:	`enum class RecoveryReportKind : std::uint32_t {...};`	
Values:	kKeyValueStorage RecoveryFailed= 1	A Key-Value Storage was corrupted, an insufficient number of valid copies existed. storage contains the short-name path of the Key-Value Storage, reportedElements is empty, reportedInstances contains the indices of the affected Key-Value Storage copies.
	kKeyValueStorage Recovered= 2	A Key-Value Storage was corrupted, but a sufficient number of valid copies existed. storage contains the short-name path of the Key-Value Storage, reportedElements is empty, reportedInstances contains the indices of the affected Key-Value Storage copies.
	kKeyRecoveryFailed= 3	A set of key-value pairs was corrupted, an insufficient number of valid copies existed. storage contains the short-name path of the Key-Value Storage, reportedElements contains the list of affected keys, reportedInstances contains the indices of the affected Key-Value Storage or key-value pair copies. In general, the nth key in reportedElements corresponds to the nth index in reported Instances, i.e. a key may be reported several times if several copies are broken. In case only one key-value pair is affected, reported Elements may be provided containing just this key.
	kKeyRecovered= 4	A set of key-value pairs was corrupted, but a sufficient number of valid copies existed. storage contains the short-name path of the Key-Value Storage, reportedElements contains the list of affected keys, reportedInstances contains the indices of the affected Key-Value Storage or key-value pair copies. In general, the nth key in reportedElements corresponds to the nth index in reported Instances, i.e. a key may be reported several times if several copies are broken. In case only one key-value pair is affected, reported Elements may be provided containing just this key.

▽

△

	kFileStorageRecovery Failed= 5	A File Storage was corrupted, an insufficient number of valid copies existed. storage contains the short-name path of the File Storage, reportedElements is empty, reportedInstances contains the indices of the affected File Storage copies.
	kFileStorageRecovered= 6	A File Storage was corrupted, but a sufficient number of valid copies existed. storage contains the short-name path of the File Storage, reportedElements is empty, reportedInstances contains the indices of the affected File Storage copies.
	kFileRecoveryFailed= 7	A set of files was corrupted, an insufficient number of valid copies existed. storage contains the short-name path of the File Storage, reportedElements contains the list of affected file names, reported Instances contains the indices of the affected File Storage or file copies. In general, the nth file name in reportedElements corresponds to the nth index in reportedInstances, i.e. a file name may be reported several times if several copies are broken. In case only one file is affected, reportedElements may be provided containing just this file name.
	kFileRecovered= 8	A set of files was corrupted, but a sufficient number of valid copies existed. storage contains the short-name path of the File Storage, reportedElements contains the list of affected file names, reported Instances contains the indices of the affected File Storage or file copies. In general, the nth file name in reportedElements corresponds to the nth index in reportedInstances, i.e. a file name may be reported several times if several copies are broken. In case only one file is affected, reportedElements may be provided containing just this file name.
Description:	Defines the reported recovery actions.	

⌋

8.1.3.2 RegisterRecoveryReportCallback

[SWS_PER_00433] Definition of API function ara::per::RegisterRecoveryReportCallback

Upstream requirements: RS_PER_00008, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00135, RS_AP_00137, RS_AP_00159

⌈

Kind:	function
Header file:	#include "ara/per/recovery.h"
Scope:	`namespace ara::per`
Syntax:	`void RegisterRecoveryReportCallback (std::function< void(const ara::core::InstanceSpecifier &storage, ara::per::RecoveryReportKind recoveryReportKind, ara::core::Vector< ara::core::String > reported Elements, ara::core::Vector< std::uint8_t > reportedInstances)> recoveryReportCallback) noexcept;`

▽

△

Parameters (in):	recoveryReportCallback	The callback function to be called by Persistency to report errors in the stored data that were corrected using the available redundancy. The function will be called with the shortName path of the affected Key-Value Storage or File Storage in storage and information on what has been corrected, placed in the parameters recoveryReport Kind, reportedElements, and reportedInstances.
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Register a recovery reporting callback with Persistency. This callback can be used in safety-aware Adaptive Applications to detect actions of the Persistency that are related to the correctness of the persisted data and the reliability of the storage.	

⌋

8.1.4 Handle Classes

This section contains the definition of the handle classes used in the API of the Persistency. The ara::per::SharedHandle (templated via typenameT) is used to provide shared access to either a ara::per::KeyValueStorage or a ara::per::FileStorage, while the ara::per::UniqueHandle (templated via typenameT) is used to provide non-shared access to either a ara::per::ReadAccessor or a ara::per::ReadWriteAccessor to a File Storage.

8.1.4.1 SharedHandle Class

[SWS_PER_00362] Definition of API class ara::per::SharedHandle

Upstream requirements: RS_PER_00002, RS_AP_00122, RS_AP_00140

⌈

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

⌋

[SWS_PER_00582] Thread-Safety of the SharedHandle

Upstream requirements: RS_PER_00001, RS_PER_00002

⌈The copy assignment operator (operator=), the move assignment operator (operator=), the copy constructor (SharedHandle), the move constructor (SharedHandle), and the destructor (~SharedHandle) for SharedHandle shall be implemented such that all these operations can take place at the same time in different threads without risking to lose track of the usage of the contained object, which could result in undefined behavior or memory leaks.⌋

Please note: This behavior can be achieved by protecting a counter that is used internally to keep track of the currently existing copies, so that the last one destructing the SharedHandle can safely destruct the embedded object. This is similar to how the C++ shared pointer handles copying/destruction.

8.1.4.1.1 SharedHandle::SharedHandle

[SWS_PER_00367] Definition of API function ara::per::SharedHandle::SharedHandle

Upstream requirements: RS_PER_00004, RS_AP_00120, RS_AP_00121, RS_AP_00129, RS_AP_00135, RS_AP_00145, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/shared_handle.h"	
Scope:	`class ara::per::SharedHandle`	
Syntax:	`SharedHandle (SharedHandle &&sh) noexcept;`	
Parameters (in):	sh	The SharedHandle object to be moved.
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructor for SharedHandle. The source handle object is invalidated and cannot be used anymore. The operator bool() shall be used to check the state of a handle object before using any other operators of the handle object.	

⌋

[SWS_PER_00369] Definition of API function ara::per::SharedHandle::SharedHandle

Upstream requirements: RS_PER_00004, RS_AP_00120, RS_AP_00121, RS_AP_00129, RS_AP_00135, RS_AP_00145, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/shared_handle.h"	
Scope:	`class ara::per::SharedHandle`	
Syntax:	`SharedHandle (const SharedHandle &sh) noexcept;`	
Parameters (in):	sh	The SharedHandle object to be copied.
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Copy constructor for SharedHandle.	

⌋

8.1.4.1.2 SharedHandle::operator=

[SWS_PER_00368] Definition of API function ara::per::SharedHandle::operator=

Upstream requirements: RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00135, RS_AP_00145, RS_AP_00153, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/shared_handle.h"	
Scope:	`class ara::per::SharedHandle`	
Syntax:	`SharedHandle & operator= (SharedHandle &&sh) & noexcept;`	
Parameters (in):	sh	The SharedHandle object to be moved.
Return value:	SharedHandle &	The moved SharedHandle object.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator for SharedHandle. The source handle object is invalidated and cannot be used anymore. The operator bool() shall be used to check the state of a handle object before using any other operators of the handle object.	

⌋

[SWS_PER_00370] Definition of API function ara::per::SharedHandle::operator=

Upstream requirements: RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00135, RS_AP_00145, RS_AP_00153, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/shared_handle.h"	
Scope:	`class ara::per::SharedHandle`	
Syntax:	`SharedHandle & operator= (const SharedHandle &sh) & noexcept;`	
Parameters (in):	sh	The SharedHandle object to be copied.
Return value:	SharedHandle &	The copied SharedHandle object.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Copy assignment operator for SharedHandle.	

⌋

8.1.4.1.3 SharedHandle::~SharedHandle

[SWS_PER_00568] Definition of API function ara::per::SharedHandle::~SharedHandle

Upstream requirements: RS_PER_00004, RS_AP_00120, RS_AP_00129, RS_AP_00134, RS_AP_00145, RS_AP_00159

⌈

Kind:	function
Header file:	#include "ara/per/shared_handle.h"
Scope:	`class ara::per::SharedHandle`
Syntax:	`~SharedHandle () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor for SharedHandle.

⌋

8.1.4.1.4 SharedHandle::operator bool

[SWS_PER_00398] Definition of API function ara::per::SharedHandle::operator bool

Upstream requirements: RS_PER_00001, RS_PER_00002, RS_PER_00003, RS_AP_00119, RS_AP_00129, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/shared_handle.h"	
Scope:	`class ara::per::SharedHandle`	
Syntax:	`explicit operator bool () const noexcept;`	
Return value:	bool	False if the handle is empty, otherwise true.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Handle state. True if the handle represents a valid object of the templated class, False if the handle is empty (e.g. after a move operation). Using other operators than bool() of an empty handle will result in undefined behavior.	

⌋

8.1.4.1.5 SharedHandle::Operator->

[SWS_PER_00363] Definition of API function ara::per::SharedHandle::operator->

Upstream requirements: RS_PER_00001, RS_PER_00002, RS_PER_00003, RS_AP_00119, RS_AP_00129, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/shared_handle.h"	
Scope:	`class ara::per::SharedHandle`	
Syntax:	`T * operator-> () noexcept;`	
Return value:	T *	A pointer to the SharedHandle object.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Non-constant arrow operator.	

⌋

[SWS_PER_00364] Definition of API function ara::per::SharedHandle::operator->

Upstream requirements: RS_PER_00001, RS_PER_00002, RS_PER_00003, RS_AP_00119, RS_AP_00129, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/shared_handle.h"	
Scope:	`class ara::per::SharedHandle`	
Syntax:	`const T * operator-> () const noexcept;`	
Return value:	const T *	A constant pointer to the SharedHandle object.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Constant arrow operator.	

⌋

8.1.4.1.6 SharedHandle::Operator*

[SWS_PER_00402] Definition of API function ara::per::SharedHandle::operator*

Upstream requirements: RS_PER_00001, RS_PER_00002, RS_PER_00003, RS_AP_00119, RS_AP_00129, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/shared_handle.h"	
Scope:	`class ara::per::SharedHandle`	
Syntax:	`T & operator* () noexcept;`	
Return value:	T &	A reference to the SharedHandle object.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Non-constant dereference operator.	

⌋

[SWS_PER_00403] Definition of API function ara::per::SharedHandle::operator*

Upstream requirements: RS_PER_00001, RS_PER_00002, RS_PER_00003, RS_AP_00119, RS_AP_00129, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/shared_handle.h"	
Scope:	`class ara::per::SharedHandle`	
Syntax:	`const T & operator* () const noexcept;`	
Return value:	const T &	A constant reference to the SharedHandle object.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Constant dereference operator.	

⌋

8.1.4.2 UniqueHandle Class

[SWS_PER_00359] Definition of API class ara::per::UniqueHandle

Upstream requirements: RS_PER_00002, RS_AP_00122, RS_AP_00140

⌈

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

⌋

8.1.4.2.1 UniqueHandle::UniqueHandle

[SWS_PER_00371] Definition of API function ara::per::UniqueHandle::UniqueHandle

Upstream requirements: RS_PER_00002, RS_AP_00120, RS_AP_00121, RS_AP_00129, RS_AP_00135, RS_AP_00145, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/unique_handle.h"	
Scope:	`class ara::per::UniqueHandle`	
Syntax:	`UniqueHandle (UniqueHandle &&uh) noexcept;`	
Parameters (in):	uh	The UniqueHandle object to be moved.
Exception Safety:	exception safe	
Thread Safety:	implementation defined	
Description:	Move constructor for UniqueHandle. The source handle object is invalidated and cannot be used anymore. The operator bool() shall be used to check the state of a handle object before using any other operators of the handle object.	

⌋

[SWS_PER_00373] Definition of API function ara::per::UniqueHandle::UniqueHandle

Upstream requirements: RS_PER_00002, RS_AP_00120, RS_AP_00145

⌈

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

⌋

8.1.4.2.2 UniqueHandle::operator=

[SWS_PER_00372] Definition of API function ara::per::UniqueHandle::operator=

Upstream requirements: RS_PER_00002, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00135, RS_AP_00145, RS_AP_00153, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/unique_handle.h"	
Scope:	`class ara::per::UniqueHandle`	
Syntax:	`UniqueHandle & operator= (UniqueHandle &&uh) & noexcept;`	
Parameters (in):	uh	The UniqueHandle object to be moved.
Return value:	UniqueHandle &	The moved UniqueHandle object.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Move assignment operator for UniqueHandle. The source handle object is invalidated and cannot be used anymore. The operator bool() shall be used to check the state of a handle object before using any other operators of the handle object.	

⌋

[SWS_PER_00374] Definition of API function ara::per::UniqueHandle::operator=

Upstream requirements: RS_PER_00002, RS_AP_00120, RS_AP_00145

⌈

Kind:	function
Header file:	#include "ara/per/unique_handle.h"
Scope:	`class ara::per::UniqueHandle`

▽

△

Syntax:	`UniqueHandle & operator= (const UniqueHandle &)=delete;`
Description:	The copy assignment operator for UniqueHandle shall not be used.

⌋

8.1.4.2.3 UniqueHandle::~UniqueHandle

[SWS_PER_00569] Definition of API function ara::per::UniqueHandle::~UniqueHandle

Upstream requirements: RS_PER_00002, RS_AP_00120, RS_AP_00129, RS_AP_00134, RS_AP_00145, RS_AP_00159

⌈

Kind:	function
Header file:	#include "ara/per/unique_handle.h"
Scope:	`class ara::per::UniqueHandle`
Syntax:	`~UniqueHandle () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor for UniqueHandle.

⌋

8.1.4.2.4 UniqueHandle::operator bool

[SWS_PER_00399] Definition of API function ara::per::UniqueHandle::operator bool

Upstream requirements: RS_PER_00001, RS_PER_00002, RS_PER_00003, RS_AP_00119, RS_AP_00129, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/unique_handle.h"	
Scope:	`class ara::per::UniqueHandle`	
Syntax:	`explicit operator bool () const noexcept;`	
Return value:	bool	False if the handle is empty, otherwise true.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	

▽

△

Description:	Handle state. True if the handle represents a valid object of the templated class, False if the handle is empty (e.g. after a move operation). Using other operators than bool() of an empty handle will result in undefined behavior.

⌋

8.1.4.2.5 UniqueHandle::Operator->

[SWS_PER_00360] Definition of API function ara::per::UniqueHandle::operator->

Upstream requirements: RS_PER_00001, RS_PER_00002, RS_PER_00003, RS_AP_00119, RS_AP_00129, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/unique_handle.h"	
Scope:	`class ara::per::UniqueHandle`	
Syntax:	`T * operator-> () noexcept;`	
Return value:	T *	A pointer to the UniqueHandle object.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Non-constant arrow operator.	

⌋

[SWS_PER_00361] Definition of API function ara::per::UniqueHandle::operator->

Upstream requirements: RS_PER_00001, RS_PER_00002, RS_PER_00003, RS_AP_00119, RS_AP_00129, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/unique_handle.h"	
Scope:	`class ara::per::UniqueHandle`	
Syntax:	`const T * operator-> () const noexcept;`	
Return value:	const T *	A constant pointer to the UniqueHandle object.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Constant arrow operator.	

⌋

8.1.4.2.6 UniqueHandle::Operator*

[SWS_PER_00400] Definition of API function ara::per::UniqueHandle::operator*

Upstream requirements: RS_PER_00001, RS_PER_00002, RS_PER_00003, RS_AP_00119, RS_AP_00129, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/unique_handle.h"	
Scope:	`class ara::per::UniqueHandle`	
Syntax:	`T & operator* () noexcept;`	
Return value:	T &	A reference to the UniqueHandle object.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Non-constant dereference operator.	

⌋

[SWS_PER_00401] Definition of API function ara::per::UniqueHandle::operator*

Upstream requirements: RS_PER_00001, RS_PER_00002, RS_PER_00003, RS_AP_00119, RS_AP_00129, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/unique_handle.h"	
Scope:	`class ara::per::UniqueHandle`	
Syntax:	`const T & operator* () const noexcept;`	
Return value:	const T &	A constant reference to the UniqueHandle object.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Constant dereference operator.	

⌋

8.1.5 Errors

The `Persistency` implements an error handling based on `ara::core::Result`. The errors supported by the `Persistency` are listed in Section 8.1.5.1.

8.1.5.1 PerErrc

[SWS_PER_00311] Definition of API enum ara::per::PerErrc

Upstream requirements: RS_AP_00122, RS_AP_00125, RS_AP_00127, RS_AP_00149

⌈

Kind:	enumeration	
Header file:	#include "ara/per/per_error_domain.h"	
Forwarding header file:	#include "ara/per/per_fwd.h"	
Scope:	`namespace ara::per`	
Symbol:	PerErrc	
Underlying type:	ara::core::ErrorDomain::CodeType	
Syntax:	`enum class PerErrc : ara::core::ErrorDomain::CodeType {...};`	
Values:	kStorageNotFound= 1	The requested Key-Value Storage or File Storage is not configured in the AUTOSAR model.
	kKeyNotFound= 2	The provided key cannot be not found in the Key-Value Storage.
	kIllegalWriteAccess= 3	Synchronizing a Key-Value Pair of the Key-Value Storage failed, or opening a file of the File Storage for writing or changing failed, because the Key-Value Storage or File Storage is configured read-only.
	kPhysicalStorage Failure= 4	An error occurred when accessing the physical storage, e.g. because of a corrupted file system or corrupted hardware, or because of insufficient access rights.
	kIntegrityCorrupted= 5	The structural integrity of the Key-Value Storage or File Storage could not be established. This can happen when the internal structure of a Key-Value Storage or the metadata of a File Storage is corrupted.
	kValidationFailed= 6	The validation of redundancy measures failed for a single key-value pair, for the whole Key-Value Storage, for a single file, or for the whole File Storage.
	kEncryptionFailed= 7	The encryption or decryption failed for a single key-value pair, for the whole Key-Value Storage, for a single file, or for the whole File Storage.
	kDataTypeMismatch= 8	The provided data type does not match the stored data type.
	kInitValueNotAvailable= 9	The operation could not be performed because no initial value is available.
	kResourceBusy= 10	The operation could not be performed because the resource is currently busy.
	kOutOfStorageSpace= 12	The physical storage space was exceeded.
	kFileNotFound= 13	The requested file name cannot be not found in the File Storage.
	kInvalidPosition= 15	SetPosition tried to move to a position that is not reachable (i.e. which is smaller than zero or greater than the current size of the file).

▽

△

	kEof= 16	The Persistency user tried to read from the end of the file or from an empty file.
	kInvalidOpenMode= 17	Opening a file failed because the requested combination of Open Modes is invalid.
	kInvalidSize= 18	SetFileSize tried to set a new size that is bigger than the current file size.
	kTooManyFiles= 19	The maximum number of files was exceeded.
	kQuotaExceeded= 20	The allocated storage quota was exceeded.
	kAuthenticationFailed= 21	Calculating or checking of the MAC failed for a single key-value pair, for the whole Key-Value Storage, for a single file, or for the whole File Storage.
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.	

⌋

8.1.5.2 GetPerDomain

[SWS_PER_00352] Definition of API function ara::per::GetPerDomain

Upstream requirements: RS_AP_00119, RS_AP_00120, RS_AP_00159

⌈

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

⌋

8.1.5.3 MakeErrorCode

[SWS_PER_00351] Definition of API function ara::per::MakeErrorCode

Upstream requirements: RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00135, RS_AP_00159

⌈

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

⌋

8.1.5.4 PerException Class

[SWS_PER_00354] Definition of API class ara::per::PerException

Upstream requirements: RS_AP_00122, RS_AP_00127

⌈

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

⌋

8.1.5.4.1 PerException::PerException

[SWS_PER_00355] Definition of API function ara::per::PerException::PerException

Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00135, RS_AP_00159

⌈

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

⌋

8.1.5.5 PerErrorDomain Class

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

[SWS_PER_00312] Definition of API class ara::per::PerErrorDomain

Upstream requirements: RS_AP_00122, RS_AP_00127, RS_AP_00140

⌈

Kind:	class
Header file:	#include "ara/per/per_error_domain.h"
Forwarding header file:	#include "ara/per/per_fwd.h"
Scope:	`namespace ara::per`
Symbol:	PerErrorDomain
Base class:	ara::core::ErrorDomain
Syntax:	`class PerErrorDomain final :  public ara::core::ErrorDomain {...};`
Unique ID:	As per `ara::per::PerErrorDomain` in [SWS_CORE_90023]
Description:	Defines the error domain for Persistency.

⌋

8.1.5.5.1 PerErrorDomain::Errc

[SWS_PER_00411] Definition of API type ara::per::PerErrorDomain::Errc

Upstream requirements: RS_AP_00122

⌈

Kind:	type alias
Header file:	#include "ara/per/per_error_domain.h"
Scope:	`class ara::per::PerErrorDomain`
Symbol:	Errc
Syntax:	`using Errc = PerErrc;`
Description:	Alias for the error code value enumeration.

⌋

8.1.5.5.2 PerErrorDomain::Exception

[SWS_PER_00412] Definition of API type ara::per::PerErrorDomain::Exception

Upstream requirements: RS_AP_00122

⌈

Kind:	type alias
Header file:	#include "ara/per/per_error_domain.h"
Scope:	`class ara::per::PerErrorDomain`
Symbol:	Exception
Syntax:	`using Exception = PerException;`
Description:	Alias for the exception base class.

⌋

8.1.5.5.3 PerErrorDomain::PerErrorDomain

[SWS_PER_00313] Definition of API function ara::per::PerErrorDomain::PerErrorDomain

Upstream requirements: RS_AP_00119, RS_AP_00120, RS_AP_00159

⌈

Kind:	function
Header file:	#include "ara/per/per_error_domain.h"
Scope:	`class ara::per::PerErrorDomain`
Syntax:	`PerErrorDomain () noexcept;`
Exception Safety:	exception safe
Thread Safety:	thread-safe
Description:	Creates a PerErrorDomain instance.

⌋

8.1.5.5.4 PerErrorDomain::Name

[SWS_PER_00314] Definition of API function ara::per::PerErrorDomain::Name

Upstream requirements: RS_AP_00119, RS_AP_00120, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/per_error_domain.h"	
Scope:	`class ara::per::PerErrorDomain`	
Syntax:	`const char * Name () const noexcept override;`	
Return value:	const char *	The name of the error domain.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the name of the error domain.	

⌋

8.1.5.5.5 PerErrorDomain::Message

[SWS_PER_00315] Definition of API function ara::per::PerErrorDomain::Message

Upstream requirements: RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/per_error_domain.h"	
Scope:	`class ara::per::PerErrorDomain`	
Syntax:	`const char * Message (CodeType errorCode) const noexcept override;`	
Parameters (in):	errorCode	The error code number.
Return value:	const char *	The message associated with the error code.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the message associated with the error code.	

⌋

8.1.5.5.6 PerErrorDomain::ThrowAsException

[SWS_PER_00350] Definition of API function ara::per::PerErrorDomain::ThrowAsException

Upstream requirements: RS_AP_00120, RS_AP_00121

⌈

Kind:	function	
Header file:	#include "ara/per/per_error_domain.h"	
Scope:	`class ara::per::PerErrorDomain`	
Syntax:	`void ThrowAsException (const ara::core::ErrorCode &errorCode) const noexcept(false) override;`	
Parameters (in):	errorCode	The error to throw.
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Throws the exception associated with the error code. As per [SWS_CORE_10304], this function does not participate in overload resolution when C++ exceptions are disabled in the compiler toolchain.	

⌋

8.2 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, to reset it to the deployed defaults, and to get the amount of storage space allocated to the Key-Value Storage.

8.2.1 OpenKeyValueStorage

[SWS_PER_00052] Definition of API function ara::per::OpenKeyValueStorage

Upstream requirements: 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_00135, RS_AP_00137, RS_AP_00139, RS_AP_00144, RS_AP_00147, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/key_value_storage.h"	
Scope:	`namespace ara::per`	
Syntax:	`ara::core::Result< SharedHandle< KeyValueStorage > > OpenKeyValue Storage (const ara::core::InstanceSpecifier &kvs) noexcept;`	
Parameters (in):	kvs	The shortName path of a PortPrototype typed by a PersistencyKey ValueStorageInterface.
Return value:	ara::core::Result< SharedHandle< Key ValueStorage > >	A Result containing a SharedHandle for the KeyValueStorage. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kStorageNot Found	rollback_semantics
		Returned if the passed InstanceSpecifier does not match any PersistencyKeyValueStorageInterface configured for this Executable.
	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics

▽

△

		Returned if CleanUpPersistency, or ResetPersistency is currently being executed, or if RecoverKeyValueStorage or ResetKeyValue Storage is currently being executed for the same Key-Value Storage. Also, while UpdatePersistency is currently being executed, this error is returned when OpenKeyValueStorage is called outside of the callback registered via RegisterApplicationDataUpdate Callback, or if the used InstanceSpecifier differs from the one provided by the callback.
	PerErrc::kOutOfStorage Space	rollback_semantics
		Returned if the available physical storage space is insufficient for the values that are added/updated during an implicit update of the Key-Value Storage.
	PerErrc::kQuota Exceeded	rollback_semantics
		Returned if the values that are added/updated during an implicit update of the Key-Value Storage would exceed the configured maximumAllowedSize.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Opens a Key-Value Storage. OpenKeyValueStorage will fail with kResourceBusy when the Key-Value Storage is currently being modified by a call from another thread to UpdatePersistency, CleanUpPersistency, Reset Persistency, RecoverKeyValueStorage, or ResetKeyValueStorage. Because multiple threads can access the same Key-Value Storage concurrently, the Key-Value Storage might not be closed when the SharedHandle returned by this function goes out of scope. It will only be closed when all SharedHandles that refer to the same Key-Value Storage went out of scope.	

⌋

8.2.2 RecoverKeyValueStorage

[SWS_PER_00333] Definition of API function ara::per::RecoverKeyValueStorage

Upstream requirements: RS_PER_00003, RS_PER_00009, RS_PER_00010, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00135, RS_AP_00137, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/key_value_storage.h"	
Scope:	`namespace ara::per`	
Syntax:	`ara::core::Result< void > RecoverKeyValueStorage (const ara::core::InstanceSpecifier &kvs) noexcept;`	
Parameters (in):	kvs	The shortName path of a PortPrototype typed by a PersistencyKey ValueStorageInterface.
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe	
Errors:	PerErrc::kStorageNotFound	rollback_semantics
		Returned if the passed InstanceSpecifier does not match any PersistencyKeyValueStorageInterface configured for this Executable.
	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the encryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if UpdatePersistency, CleanUpPersistency, or Reset Persistency is currently being executed, or if ResetKeyValue Storage is currently being executed for the same Key-Value Storage, or a SharedHandle of the same Key-Value Storage is currently in use.
	PerErrc::kOutOfStorageSpace	rollback_semantics
		Returned if the available physical storage space is insufficient for the recovered values.
	PerErrc::kQuotaExceeded	rollback_semantics
		Returned if the recovered values would exceed the configured maximumAllowedSize of the Key-Value Storage.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if calculating the MAC of stored data fails.
Description:	Recovers a Key-ValueStorage. RecoverKeyValueStorage allows to recover a Key-Value Storage when the redundancy checks fail. It will fail with kResourceBusy when the Key-Value Storage is currently open, or when it is modified by a call from another thread to UpdatePersistency, CleanUpPersistency, Reset Persistency, RecoverKeyValueStorage, or ResetKeyValueStorage. This method does a best-effort recovery of all key-value pairs. After recovery, keys might show outdated or initial value, or might be lost.	

⌋

8.2.3 ResetKeyValueStorage

[SWS_PER_00334] Definition of API function ara::per::ResetKeyValueStorage

Upstream requirements: RS_PER_00003, RS_PER_00009, RS_PER_00010, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00135, RS_AP_00137, RS_AP_00139, RS_AP_00159

⌈

Kind:	function
Header file:	#include "ara/per/key_value_storage.h"
Scope:	`namespace ara::per`

▽

△

Syntax:	`ara::core::Result< void > ResetKeyValueStorage (const ara::core::InstanceSpecifier &kvs) noexcept;`	
Parameters (in):	kvs	The shortName path of a PortPrototype typed by a PersistencyKeyValueStorageInterface.
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kStorageNotFound	rollback_semantics
		Returned if the passed InstanceSpecifier does not match any PersistencyKeyValueStorageInterface configured for this Executable.
	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the encryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if UpdatePersistency, CleanUpPersistency, or Reset Persistency is currently being executed, or if RecoverKeyValue Storage is currently being executed for the same Key-Value Storage, or a SharedHandle of the same Key-Value Storage is currently in use.
	PerErrc::kOutOfStorageSpace	rollback_semantics
		Returned if the available physical storage space is insufficient for the initial values.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if calculating the MAC of stored data fails.
Description:	Resets a Key-Value Storage to the initial state. ResetKeyValueStorage allows to reset a Key-Value Storage to the initial state, containing only key-value pairs which were deployed from the manifest, with their initial values. Afterwards, the Key-Value Storage will appear as if it was newly installed from the current manifest. It will fail with kResourceBusy when the Key-Value Storage is currently open, or when it is modified by a call from another thread to UpdatePersistency, CleanUpPersistency, Reset Persistency, RecoverKeyValueStorage, or ResetKeyValueStorage.	

⌋

8.2.4 GetCurrentKeyValueStorageSize

[SWS_PER_00405] Definition of API function ara::per::GetCurrentKeyValueStorageSize

Upstream requirements: RS_PER_00017, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00135, RS_AP_00137, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/key_value_storage.h"	
Scope:	`namespace ara::per`	
Syntax:	`ara::core::Result< std::uint64_t > GetCurrentKeyValueStorageSize (const ara::core::InstanceSpecifier &kvs) noexcept;`	
Parameters (in):	kvs	The shortName path of a PortPrototype typed by a PersistencyKeyValueStorageInterface.
Return value:	ara::core::Result< std::uint64_t >	A Result containing the occupied space in bytes. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kStorageNotFound	rollback_semantics
		Returned if the passed InstanceSpecifier does not match any PersistencyKeyValueStorageInterface configured for this Executable.
	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
Description:	Returns the space in bytes currently occupied by a Key-Value Storage. The returned size includes all metadata and the space used for redundancy and backups. The returned size is only guaranteed to be accurate if the Key-Value Storage is not opened and no other operation on the Key-Value Storage takes place at the same time.	

⌋

8.2.5 KeyValueStorage Class

This section shows the methods available for a `ara::per::KeyValueStorage` object obtained from a call to `ara::per::OpenKeyValueStorage`.

[SWS_PER_00339] Definition of API class ara::per::KeyValueStorage

Upstream requirements: RS_PER_00002, RS_AP_00122, RS_AP_00140, RS_AP_00146

⌈

Kind:	class
Header file:	#include "ara/per/key_value_storage.h"
Forwarding header file:	#include "ara/per/per_fwd.h"
Scope:	`namespace ara::per`
Symbol:	KeyValueStorage
Syntax:	`class KeyValueStorage final {...};`
Description:	The Key-Value Storage contains a set of keys with associated values.

⌋

8.2.5.1 KeyValueStorage::KeyValueStorage

[SWS_PER_00459] Definition of API function ara::per::KeyValueStorage::KeyValueStorage

Upstream requirements: RS_PER_00002, RS_AP_00120, RS_AP_00129, RS_AP_00146

⌈

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

⌋

[SWS_PER_00322] Definition of API function ara::per::KeyValueStorage::KeyValueStorage

Upstream requirements: RS_PER_00002, RS_AP_00120, RS_AP_00121, RS_AP_00129, RS_AP_00145, RS_AP_00159

⌈

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

⌋

[SWS_PER_00324] Definition of API function ara::per::KeyValueStorage::KeyValueStorage

Upstream requirements: RS_PER_00002, RS_AP_00120, RS_AP_00145

⌈

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

⌋

8.2.5.2 KeyValueStorage::operator=

[SWS_PER_00323] Definition of API function ara::per::KeyValueStorage::operator=

Upstream requirements: RS_PER_00002, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00145, RS_AP_00159

⌈

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

⌋

[SWS_PER_00325] Definition of API function ara::per::KeyValueStorage::operator=

Upstream requirements: RS_PER_00002, RS_AP_00119, RS_AP_00120, RS_AP_00145

⌈

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

⌋

8.2.5.3 KeyValueStorage::~KeyValueStorage

[SWS_PER_00050] Definition of API function ara::per::KeyValueStorage::~KeyValueStorage

Upstream requirements: RS_PER_00002, RS_AP_00120, RS_AP_00129, RS_AP_00134, RS_AP_00145, RS_AP_00159

⌈

Kind:	function
Header file:	#include "ara/per/key_value_storage.h"
Scope:	`class ara::per::KeyValueStorage`
Syntax:	`~KeyValueStorage () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor for KeyValueStorage.

⌋

8.2.5.4 KeyValueStorage::GetAllKeys

[SWS_PER_00042] Definition of API function ara::per::KeyValueStorage::GetAllKeys

Upstream requirements: RS_PER_00003, RS_AP_00119, RS_AP_00120, RS_AP_00127, RS_AP_00129, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/key_value_storage.h"	
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. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.

▽

△

	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Returns a list of all currently available keys of this Key-Value Storage. The list of keys is only accurate if no key-value pair is added or deleted at the same time.	

⌋

8.2.5.5 KeyValueStorage::KeyExists

[SWS_PER_00043] Definition of API function ara::per::KeyValueStorage::KeyExists

Upstream requirements: RS_PER_00003, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/key_value_storage.h"	
Scope:	`class ara::per::KeyValueStorage`	
Syntax:	`ara::core::Result< bool > KeyExists (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. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Checks if a key-value pair exists in this Key-Value Storage. The result is only accurate if no key-value pair is added or deleted at the same time. E.g. when a key-value pair is removed in another thread directly after this function returned "true", the result is not valid anymore.	

⌋

8.2.5.6 KeyValueStorage::GetCurrentValueSize

[SWS_PER_00554] Definition of API function ara::per::KeyValueStorage::GetCurrentValueSize

Upstream requirements: RS_PER_00017, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/key_value_storage.h"	
Scope:	`class ara::per::KeyValueStorage`	
Syntax:	`ara::core::Result< std::uint64_t > GetCurrentValueSize (ara::core::StringView key) const noexcept;`	
Parameters (in):	key	The key to look up.
Return value:	ara::core::Result< std::uint64_t >	A Result containing the size of the value in bytes. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kKeyNotFound	rollback_semantics
		Returned if the provided key does not exist in the Key-Value Storage.
	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrityCorrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidationFailed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Returns the size (in bytes) of the value assigned to a key of this Key-Value Storage. GetCurrentValueSize may be delayed by an ongoing call from another thread to RemoveAllKeys or DiscardPendingChanges, or to SetValue, RemoveKey, RecoverKey, or ResetKey for the same key-value pair.	

⌋

8.2.5.7 KeyValueStorage::GetValue

[SWS_PER_00332] Definition of API function ara::per::KeyValueStorage::GetValue

Upstream requirements: 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_00135, RS_AP_00136, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/key_value_storage.h"	
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 containing the retrieved value. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kKeyNotFound	rollback_semantics
		Returned if the provided key does not exist in the Key-Value Storage.
	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrityCorrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidationFailed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kDataTypeMismatch	rollback_semantics
		Returned if the data type of stored value does not match the templated type.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Returns the value assigned to a key of this Key-Value Storage. GetValue may be delayed by an ongoing call from another thread to RemoveAllKeys or DiscardPendingChanges, or to SetValue, RemoveKey, RecoverKey, or ResetKey for the same key-value pair.	

⌋

[SWS_PER_00044] Definition of API function ara::per::KeyValueStorage::GetValue

Upstream requirements: 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_00135, RS_AP_00136, RS_AP_00139, RS_AP_00141, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/key_value_storage.h"	
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 of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kKeyNotFound	rollback_semantics
		Returned if the provided key does not exist in the Key-Value Storage.
	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrityCorrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidationFailed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kDataTypeMismatch	rollback_semantics
		Returned if the data type of stored value does not match the templated type.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Returns the value assigned to a key of this KeyValueStorage. This method should only be used to access very large values repeatedly. GetValue may be delayed by an ongoing call from another thread to RemoveAllKeys or DiscardPendingChanges, or to SetValue, RemoveKey, RecoverKey, or ResetKey for the same key-value pair.	

⌋

8.2.5.8 KeyValueStorage::SetValue

[SWS_PER_00046] Definition of API function ara::per::KeyValueStorage::SetValue

Upstream requirements: 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_00135, RS_AP_00136, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/key_value_storage.h"	
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 of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kIllegalWrite Access	rollback_semantics
		Returned if the Key-Value Storage is configured as read- only.
	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be written because the structural integrity is corrupted.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the encryption or decryption of stored data fails.
	PerErrc::kDataType Mismatch	rollback_semantics
		Returned if the data type of an already stored value does not match the templated type.
	PerErrc::kOutOfStorage Space	rollback_semantics
		Returned if the available physical storage space is insufficient for the new value.
	PerErrc::kQuota Exceeded	rollback_semantics
		Returned if the new value would exceed the configured maximum AllowedSize of the Key-Value Storage.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if calculating or checking the MAC of stored data fails.

▽

△

Description:	Stores a key-value pair in this Key-Value Storage. If a value already exists and has the same data type as the new value, it is overwritten. If the new value has a different data type than the stored value, kDataTypeMismatch is returned. SetValue may be delayed by an ongoing call from another thread to RemoveAllKeys, SyncTo Storage, or DiscardPendingChanges, or to SetValue, GetValue, GetCurrentValueSize, Remove Key, RecoverKey, or ResetKey for the same key-value pair.

⌋

8.2.5.9 KeyValueStorage::RemoveKey

[SWS_PER_00047] Definition of API function ara::per::KeyValueStorage::RemoveKey

Upstream requirements: 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_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/key_value_storage.h"	
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 of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kKeyNotFound	rollback_semantics
		Returned if the provided key does not exist in the Key-Value Storage.
	PerErrc::kIllegalWrite Access	rollback_semantics
		Returned if the Key-Value Storage is configured as read- only.
	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be written because the structural integrity is corrupted.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the encryption or decryption of stored data fails.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if calculating or checking the MAC of stored data fails.

▽

△

Description:	Removes a key and the associated value from this Key-Value Storage. RemoveKey may be delayed by an ongoing call from another thread to RemoveAllKeys, SyncTo Storage, or DiscardPendingChanges, or to SetValue, GetValue, GetCurrentValueSize, Remove Key, RecoverKey, or ResetKey for the same key-value pair.

⌋

8.2.5.10 KeyValueStorage::RecoverKey

[SWS_PER_00427] Definition of API function ara::per::KeyValueStorage::RecoverKey

Upstream requirements: RS_PER_00003, RS_PER_00009, RS_PER_00010, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/key_value_storage.h"	
Scope:	`class ara::per::KeyValueStorage`	
Syntax:	`ara::core::Result< void > RecoverKey (ara::core::StringView key) noexcept;`	
Parameters (in):	key	The key to be recovered.
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kKeyNotFound	rollback_semantics
		Returned if the provided key does not exist in the Key-Value Storage.
	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be written because the structural integrity is corrupted.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the encryption or decryption of stored data fails.
	PerErrc::kOutOfStorage Space	rollback_semantics
		Returned if the available physical storage space is insufficient for the recovered value.
	PerErrc::kQuota Exceeded	rollback_semantics
		Returned if the recovered value would exceed the configured maximumAllowedSize of the Key-Value Storage.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if calculating or checking the MAC of stored data fails.

▽

△

Description:	Recovers a single key-value pair of this Key Value Storage. This method allows to recover a single key-value pair when the redundancy checks fail. This method does a best-effort recovery of the key-value pair. After recovery, the key-value pair might contain outdated or initial content, or might be lost. RecoverKey may be delayed by an ongoing call from another thread to RemoveAllKeys, SyncTo Storage, or DiscardPendingChanges, or to SetValue, GetValue, GetCurrentValueSize, Remove Key, RecoverKey, or ResetKey for the same key-value pair.

⌋

8.2.5.11 KeyValueStorage::ResetKey

[SWS_PER_00426] Definition of API function ara::per::KeyValueStorage::Reset Key

Upstream requirements: RS_PER_00003, RS_PER_00009, RS_PER_00010, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_-AP_00129, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/key_value_storage.h"	
Scope:	`class ara::per::KeyValueStorage`	
Syntax:	`ara::core::Result< void > ResetKey (ara::core::StringView key) noexcept;`	
Parameters (in):	key	The key to be reset.
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kIllegalWrite Access	rollback_semantics
		Returned if the Key-Value Storage is configured as read- only.
	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be written because the structural integrity is corrupted.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the encryption or decryption of stored data fails.
	PerErrc::kInitValueNot Available	rollback_semantics
		Returned if no intitial value was configured for this key.
	PerErrc::kOutOfStorage Space	rollback_semantics
		Returned if the available physical storage space is insufficient for the initial value.
	PerErrc::kQuota Exceeded	rollback_semantics

▽

△

		Returned if the initial value would exceed the configured maximum AllowedSize of the Key-Value Storage.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if calculating or checking the MAC of stored data fails.
Description:	Resets a key of this Key-Value Storage to its initial value. ResetKey allows to reset a single key to its initial value. If the key is currently not available in the Key-Value Storage, it is re-created. Afterwards, the key-value pair will appear in both cases as if it was newly installed from the current manifest. ResetKey will fail with kInitValueNotAvailable when neither design nor deployment define an initial value for the key. ResetKey may be delayed by an ongoing call from another thread to RemoveAllKeys, SyncTo Storage, or DiscardPendingChanges, or to SetValue, GetValue, GetCurrentValueSize, Remove Key, RecoverKey, or ResetKey for the same key-value pair.	

⌋

8.2.5.12 KeyValueStorage::RemoveAllKeys

[SWS_PER_00048] Definition of API function ara::per::KeyValueStorage::RemoveAllKeys

Upstream requirements: RS_PER_00003, RS_PER_00010, RS_AP_00119, RS_AP_00120, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/key_value_storage.h"	
Scope:	`class ara::per::KeyValueStorage`	
Syntax:	`ara::core::Result< void > RemoveAllKeys () noexcept;`	
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kIllegalWrite Access	rollback_semantics
		Returned if the Key-Value Storage is configured as read- only.
	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be written because the structural integrity is corrupted.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the encryption or decryption of stored data fails.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if calculating or checking the MAC of stored data fails.

▽

△

Description:	Removes all key-value pairs and associated values from this Key-Value Storage. RemoveAllKeys may be delayed by an ongoing call from another thread to RemoveAllKeys, SyncToStorage, DiscardPendingChanges, SetValue, GetValue, GetCurrentValueSize, RemoveKey, RecoverKey, or ResetKey.

⌋

8.2.5.13 KeyValueStorage::SyncToStorage

[SWS_PER_00049] Definition of API function ara::per::KeyValueStorage::SyncToStorage

Upstream requirements: RS_PER_00002, RS_AP_00119, RS_AP_00120, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/key_value_storage.h"	
Scope:	`class ara::per::KeyValueStorage`	
Syntax:	`ara::core::Result< void > SyncToStorage () noexcept;`	
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kIllegalWriteAccess	rollback_semantics
		Returned if the Key-Value Storage is configured as read- only.
	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrityCorrupted	rollback_semantics
		Returned if stored data cannot be written because the structural integrity is corrupted.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the encryption of stored data fails.
	PerErrc::kOutOfStorageSpace	rollback_semantics
		Returned if the available physical storage space is insufficient for the added/changed values.
	PerErrc::kQuotaExceeded	rollback_semantics
		Returned if the added/changed values would exceed the configured maximumAllowedSize of the Key-Value Storage.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if calculating the MAC of stored data fails.

▽

△

Description:	Flushes changed key-value pairs of the Key-Value Storage to the physical storage. SyncToStorage may be delayed by an ongoing call from another thread to RemoveAllKeys, DiscardPendingChanges, SetValue, RemoveKey, RecoverKey, or ResetKey.

⌋

8.2.5.14 KeyValueStorage::DiscardPendingChanges

[SWS_PER_00365] Definition of API function ara::per::KeyValueStorage::DiscardPendingChanges

Upstream requirements: RS_PER_00002, RS_AP_00119, RS_AP_00120, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/key_value_storage.h"	
Scope:	`class ara::per::KeyValueStorage`	
Syntax:	`ara::core::Result< void > DiscardPendingChanges () noexcept;`	
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrityCorrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidationFailed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Removes all pending changes to this Key-Value Storage since the last call to SyncToStorage() or since this Key-Value Storage was opened using OpenKeyValueStorage(). DiscardPendingChanges may be delayed by an ongoing call from another thread to RemoveAllKeys, SyncToStorage, DiscardPendingChanges, SetValue, GetValue, GetCurrentValueSize, RemoveKey, RecoverKey, or ResetKey.	

⌋

8.3 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, to reset it to the deployed defaults, and to get the amount of storage space allocated to the `File Storage`. In addition, operators are present to combine the `ara::per::OpenMode` values passed as `mode` to the `OpenFile*` functions.

8.3.1 OpenFileStorage

[SWS_PER_00116] Definition of API function ara::per::OpenFileStorage

Upstream requirements: 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_00135, RS_AP_00137, RS_AP_00139, RS_AP_00144, RS_AP_00147, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`namespace ara::per`	
Syntax:	`ara::core::Result< SharedHandle< FileStorage > > OpenFileStorage (const ara::core::InstanceSpecifier &fs) noexcept;`	
Parameters (in):	fs	The shortName path of a PortPrototype typed by a PersistencyFileStorageInterface.
Return value:	ara::core::Result< SharedHandle< FileStorage > >	A Result containing a SharedHandle for the File Storage. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kStorageNotFound	rollback_semantics
		Returned if the passed InstanceSpecifier does not match any PersistencyFileStorageInterface configured for this Executable.
	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrityCorrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidationFailed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics

▽

△

		Returned if CleanUpPersistency, or ResetPersistency is currently being executed, or if RecoverAllFiles or ResetAllFiles is currently being executed for the same File Storage. Also, while Update Persistency is currently being executed, this error is returned when OpenFileStorage is called outside of the callback registered via RegisterApplicationDataUpdateCallback, or if the used Instance Specifier differs from the one provided by the callback.
	PerErrc::kOutOfStorage Space	rollback_semantics
		Returned if the available physical storage space is insufficient for the files that are added/updated during an implicit update of the File Storage.
	PerErrc::kTooManyFiles	rollback_semantics
		Returned if the files that are added during an implicit update of the File Storage would exceed the configured maxNumberOfFiles.
	PerErrc::kQuota Exceeded	rollback_semantics
		Returned if the files that are added/updated during an implicit update of the File Storage would exceed the configured maximum AllowedSize.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Opens a File Storage. OpenFileStorage will fail with kResourceBusy when the File Storage is currently being modified by a call from another thread to UpdatePersistency, CleanUpPersistency, ResetPersistency, RecoverAllFiles, or ResetAllFiles. Because multiple threads can access the same File Storage concurrently, the File Storage might not be closed when the SharedHandle returned by this function goes out of scope. It will only be closed when all SharedHandles that refer to the same File Storage went out of scope.	

⌋

8.3.2 RecoverAllFiles

[SWS_PER_00335] Definition of API function ara::per::RecoverAllFiles

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_PER_00009, RS_PER_00010, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00135, RS_AP_00137, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`namespace ara::per`	
Syntax:	`ara::core::Result< void > RecoverAllFiles (const ara::core::InstanceSpecifier &fs) noexcept;`	
Parameters (in):	fs	The shortName path of a PortPrototype typed by a PersistencyFileStorageInterface.

▽

△

Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kStorageNot Found	rollback_semantics
		Returned if the passed InstanceSpecifier does not match any PersistencyFileStorageInterface configured for this Executable.
	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the encryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if UpdatePersistency, CleanUpPersistency, or Reset Persistency is currently being executed, or if ResetAllFiles is currently being executed for the same File Storage, or a Shared Handle of the same File Storage is currently in use.
	PerErrc::kOutOfStorage Space	rollback_semantics
		Returned if the available physical storage space is insufficient for the recovered files.
	PerErrc::kQuota Exceeded	rollback_semantics
		Returned if the recovered files would exceed the configured maximumAllowedSize of the File Storage.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if calculating the MAC of stored data fails.
Description:	Recovers a File Storage, including all files. RecoverAllFiles recovers a File Storage when the redundancy checks fail. It will fail with kResourceBusy when the File Storage is currently open, or when it is modified by a call from another thread to UpdatePersistency, CleanUpPersistency, ResetPersistency, RecoverAllFiles, or ResetAllFiles. This method does a best-effort recovery of all files. After recovery, files might show outdated or initial content, or might be lost.	

⌋

8.3.3 ResetAllFiles

[SWS_PER_00336] Definition of API function ara::per::ResetAllFiles

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_PER_00009, RS_PER_00010, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00135, RS_AP_00137, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`namespace ara::per`	
Syntax:	`ara::core::Result< void > ResetAllFiles (const ara::core::InstanceSpecifier &fs) noexcept;`	
Parameters (in):	fs	The shortName path of a PortPrototype typed by a PersistencyFileStorageInterface.
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kStorageNotFound	rollback_semantics
		Returned if the passed InstanceSpecifier does not match any PersistencyFileStorageInterface configured for this Executable.
	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the encryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if UpdatePersistency, CleanUpPersistency, or ResetPersistency is currently being executed, or if RecoverAllFiles is currently being executed for the same File Storage, or a Shared Handle of the same File Storage is currently in use.
	PerErrc::kOutOfStorageSpace	rollback_semantics
		Returned if the available physical storage space is insufficient for the initial files.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if calculating the MAC of stored data fails.
Description:	Resets a File Storage, including all files. ResetAllFiles resets a File Storage to the initial state, containing only the files which were deployed from the manifest, with their initial content. Afterwards, the File Storage will appear as if it was newly installed from the current manifest. It will fail with kResourceBusy when the File Storage is currently open, or when it is modified by a call from another thread to UpdatePersistency, CleanUpPersistency, ResetPersistency, RecoverAllFiles, or ResetAllFiles.	

⌋

8.3.4 GetCurrentFileStorageSize

[SWS_PER_00406] Definition of API function ara::per::GetCurrentFileStorage Size

Upstream requirements: RS_PER_00017, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00135, RS_AP_00137, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`namespace ara::per`	
Syntax:	`ara::core::Result< std::uint64_t > GetCurrentFileStorageSize (const ara::core::InstanceSpecifier &fs) noexcept;`	
Parameters (in):	fs	The shortName path of a PortPrototype typed by a PersistencyFile StorageInterface.
Return value:	ara::core::Result< std::uint64_t >	A Result containing the occupied space in bytes. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kStorageNot Found	rollback_semantics
		Returned if the passed InstanceSpecifier does not match any PersistencyFileStorageInterface configured for this Executable.
	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
Description:	Returns the space in bytes currently occupied by a File Storage. The returned size includes all metadata and the space used for redundancy and backups. The returned size is only guaranteed to be accurate if the File Storage is not opened and no other operation on the File Storage takes place at the same time.	

⌋

8.3.5 OpenMode

[SWS_PER_00147] Definition of API enum ara::per::OpenMode

Upstream requirements: RS_PER_00003, RS_AP_00122, RS_AP_00125, RS_AP_00143

⌈

Kind:	enumeration
Header file:	#include "ara/per/file_storage.h"
Forwarding header file:	#include "ara/per/per_fwd.h"
Scope:	`namespace ara::per`
Symbol:	OpenMode

▽

△

Underlying type:	std::uint32_t	
Syntax:	`enum class OpenMode : std::uint32_t {...};`	
Values:	kAtTheBeginning= 1 << 0	Sets the seek position to the beginning of the file when the file is opened. This mode cannot be combined with kAtTheEnd.
	kAtTheEnd= 1 << 1	Sets the seek position to the end of the file when the file is opened. This mode cannot be combined with kAtTheBeginning or kTruncate.
	kTruncate= 1 << 2	Removes existing content when the file is opened. This mode cannot be combined with kAtTheEnd.
	kAppend= 1 << 3	Append to the end. Always seeks to the end of the file before writing.
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.	

⌋

8.3.6 operator| for FileStorage::OpenMode

[SWS_PER_00144] Definition of API function ara::per::operator|

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`namespace ara::per`	
Syntax:	`constexpr OpenMode operator| (OpenMode left, OpenMode right);`	
Parameters (in):	left	First OpenMode modifiers.
	right	Second OpenMode modifiers.
Return value:	OpenMode	returns Merged OpenMode modifiers.
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Merges two OpenMode modifiers into one.	

⌋

8.3.7 operator|= for FileStorage::OpenMode

[SWS_PER_00434] Definition of API function ara::per::operator|=

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`namespace ara::per`	
Syntax:	`OpenMode & operator|= (OpenMode &left, const OpenMode &right);`	
Parameters (in):	left	Left OpenMode modifiers.
	right	Right OpenMode modifiers.
Return value:	OpenMode &	returns The modified OpenMode.
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Description:	Merges an OpenMode modifier into this OpenMode.	

⌋

8.3.8 FileCreationState

[SWS_PER_00435] Definition of API enum ara::per::FileCreationState

Upstream requirements: RS_PER_00004, RS_AP_00122, RS_AP_00125, RS_AP_00143

⌈

Kind:	enumeration	
Header file:	#include "ara/per/file_storage.h"	
Forwarding header file:	#include "ara/per/per_fwd.h"	
Scope:	`namespace ara::per`	
Symbol:	FileCreationState	
Underlying type:	std::uint32_t	
Syntax:	`enum class FileCreationState : std::uint32_t {...};`	
Values:	kCreatedDuring Installation= 1	The file was created by Persistency after installation of the Adaptive Application or after ResetPersistency.
	kCreatedDuringUpdate= 2	The file was created by Persistency during an update.
	kCreatedDuringReset= 3	The file was re-created due to a call to ResetFile or ResetAllFiles.
	kCreatedDuring Recovery= 4	The file was re-created by Persistency after a corruption was detected.
	kCreatedByApplication= 5	The file was created by the Persistency user.
Description:	This enumeration describes how and when a file was created.	

⌋

8.3.9 FileModificationState

[SWS_PER_00436] Definition of API enum ara::per::FileModificationState

Upstream requirements: RS_PER_00004, RS_AP_00122, RS_AP_00125, RS_AP_00143

⌈

Kind:	enumeration	
Header file:	#include "ara/per/file_storage.h"	
Forwarding header file:	#include "ara/per/per_fwd.h"	
Scope:	`namespace ara::per`	
Symbol:	FileModificationState	
Underlying type:	std::uint32_t	
Syntax:	`enum class FileModificationState : std::uint32_t {...};`	
Values:	kModifiedDuringUpdate= 2	The file was last modified by Persistency during an update.
	kModifiedDuringReset= 3	The file was last modified by Persistency due to a call to ResetFile or ResetAllFiles.
	kModifiedDuring Recovery= 4	The file was last modified by Persistency after a corruption was detected.
	kModifiedByApplication= 5	The file was last modified by the Persistency user.
Description:	This enumeration describes how and when a file was last modified.	

⌋

8.3.10 FileInfo

[SWS_PER_00437] Definition of API class ara::per::FileInfo

Upstream requirements: RS_PER_00004, RS_AP_00122

⌈

Kind:	struct
Header file:	#include "ara/per/file_storage.h"
Forwarding header file:	#include "ara/per/per_fwd.h"
Scope:	`namespace ara::per`
Symbol:	FileInfo
Syntax:	`struct FileInfo {...};`
Description:	This structure contains additional information on a file returned by GetFileInfo.

⌋

8.3.10.1 FileInfo.creationTime

[SWS_PER_00441] Definition of API variable ara::per::FileInfo::creationTime

Upstream requirements: RS_PER_00004

⌈

Kind:	variable
Header file:	#include "ara/per/file_storage.h"
Scope:	`struct ara::per::FileInfo`
Symbol:	creationTime
Type:	`std::uint64_t`
Syntax:	`std::uint64_t creationTime;`
Description:	Time in nanoseconds since midnight 1970-01-01 UTC at which the file was created.

⌋

8.3.10.2 FileInfo.modificationTime

[SWS_PER_00442] Definition of API variable ara::per::FileInfo::modificationTime

Upstream requirements: RS_PER_00004

⌈

Kind:	variable
Header file:	#include "ara/per/file_storage.h"
Scope:	`struct ara::per::FileInfo`
Symbol:	modificationTime
Type:	`std::uint64_t`
Syntax:	`std::uint64_t modificationTime;`
Description:	Time in nanoseconds since midnight 1970-01-01 UTC at which the file was last modified.

⌋

8.3.10.3 FileInfo.accessTime

[SWS_PER_00443] Definition of API variable ara::per::FileInfo::accessTime

Upstream requirements: RS_PER_00004

⌈

Kind:	variable
Header file:	#include "ara/per/file_storage.h"
Scope:	`struct ara::per::FileInfo`
Symbol:	accessTime
Type:	`std::uint64_t`
Syntax:	`std::uint64_t accessTime;`
Description:	Time in nanoseconds since midnight 1970-01-01 UTC at which the file was last accessed.

⌋

8.3.10.4 FileInfo.fileCreationState

[SWS_PER_00444] Definition of API variable ara::per::FileInfo::fileCreationState

Upstream requirements: RS_PER_00004

⌈

Kind:	variable
Header file:	#include "ara/per/file_storage.h"
Scope:	`struct ara::per::FileInfo`
Symbol:	fileCreationState
Type:	`FileCreationState`
Syntax:	`FileCreationState fileCreationState;`
Description:	Information on how and by whom the file was created.

⌋

8.3.10.5 FileInfo.fileModificationState

[SWS_PER_00445] Definition of API variable ara::per::FileInfo::fileModification State

Upstream requirements: RS_PER_00004

⌈

Kind:	variable
Header file:	#include "ara/per/file_storage.h"
Scope:	`struct ara::per::FileInfo`
Symbol:	fileModificationState
Type:	`FileModificationState`
Syntax:	`FileModificationState fileModificationState;`
Description:	Information on how and by whom the file was last modified.

⌋

8.3.11 FileStorage Class

This section shows the methods available for a `ara::per::FileStorage` object obtained from a call to `ara::per::OpenFileStorage`.

[SWS_PER_00340] Definition of API class ara::per::FileStorage

Upstream requirements: RS_PER_00004, RS_AP_00122, RS_AP_00140, RS_AP_00146

⌈

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

⌋

8.3.11.1 FileStorage::FileStorage

[SWS_PER_00460] Definition of API function ara::per::FileStorage::FileStorage

Upstream requirements: RS_PER_00004, RS_AP_00120, RS_AP_00129, RS_AP_00146

⌈

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

⌋

[SWS_PER_00326] Definition of API function ara::per::FileStorage::FileStorage

Upstream requirements: RS_PER_00004, RS_AP_00120, RS_AP_00121, RS_AP_00129, RS_AP_00145, RS_AP_00159

⌈

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

⌋

[SWS_PER_00328] Definition of API function ara::per::FileStorage::FileStorage

Upstream requirements: RS_PER_00004, RS_AP_00120, RS_AP_00145

⌈

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

⌋

8.3.11.2 FileStorage::operator=

[SWS_PER_00327] Definition of API function ara::per::FileStorage::operator=

Upstream requirements: RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00145, RS_AP_00159

⌈

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

⌋

[SWS_PER_00329] Definition of API function ara::per::FileStorage::operator=

Upstream requirements: RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00145

⌈

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

⌋

8.3.11.3 FileStorage::~FileStorage

[SWS_PER_00330] Definition of API function ara::per::FileStorage::~FileStorage

Upstream requirements: RS_PER_00004, RS_AP_00120, RS_AP_00129, RS_AP_00134, RS_AP_00145, RS_AP_00159

⌈

Kind:	function
Header file:	#include "ara/per/file_storage.h"
Scope:	`class ara::per::FileStorage`
Syntax:	`~FileStorage () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor for FileStorage.

⌋

8.3.11.4 FileStorage::GetAllFileNames

[SWS_PER_00110] Definition of API function ara::per::FileStorage::GetAllFileNames

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00127, RS_AP_00129, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
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 available file names. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrityCorrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidationFailed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Returns a list of all currently available file names of this File Storage. The list of file names is only accurate if no file is added or deleted at the same time.	

⌋

8.3.11.5 FileStorage::DeleteFile

[SWS_PER_00111] Definition of API function ara::per::FileStorage::DeleteFile

Upstream requirements: 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_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`class ara::per::FileStorage`	
Syntax:	`ara::core::Result< void > DeleteFile (ara::core::StringView fileName) noexcept;`	
Parameters (in):	fileName	File name of the file. May correspond to the PersistencyFile.file Name of a configured file.
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kIllegalWrite Access	rollback_semantics
		Returned if the File Storage is configured as read-only.
	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be written because the structural integrity is corrupted.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the encryption or decryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if the file is open, or if RecoverFile or ResetFile with the same file name is currently being executed.
	PerErrc::kFileNotFound	rollback_semantics
		Returned if the provided file does not exist in the File Storage.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if calculating or checking the MAC of stored data fails.
Description:	Deletes a file from this File Storage. This operation will fail with kResourceBusy when the file is currently open.	

⌋

8.3.11.6 FileStorage::FileExists

[SWS_PER_00112] Definition of API function ara::per::FileStorage::FileExists

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`class ara::per::FileStorage`	
Syntax:	`ara::core::Result< bool > FileExists (ara::core::StringView fileName) const noexcept;`	
Parameters (in):	fileName	File name of the file. May correspond to the PersistencyFile.file Name of a configured file.
Return value:	ara::core::Result< bool >	A Result containing true if the file could be located or false if it couldn't. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Checks if a file exists in this File Storage. The result is only accurate if no file is added or deleted at the same time. E.g. when a file is removed in another thread directly after this function returned "true", the result is not valid anymore.	

⌋

8.3.11.7 FileStorage::RecoverFile

[SWS_PER_00337] Definition of API function ara::per::FileStorage::RecoverFile

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_PER_00009, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`class ara::per::FileStorage`	
Syntax:	`ara::core::Result< void > RecoverFile (ara::core::StringView fileName) noexcept;`	
Parameters (in):	fileName	File name of the file. May correspond to the PersistencyFile.file Name of a configured file.
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the encryption or decryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if the file is open, or if DeleteFile or ResetFile with the same file name is currently being executed.
	PerErrc::kOutOfStorageSpace	rollback_semantics
		Returned if the available physical storage space is insufficient for the recovered file.
	PerErrc::kFileNotFound	rollback_semantics
		Returned if the provided file does not exist in the File Storage.
	PerErrc::kQuotaExceeded	rollback_semantics
		Returned if the recovered file would exceed the configured maximumAllowedSize of the File Storage.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if calculating or checking the MAC of stored data fails.
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 kResourceBusy 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.	

⌋

8.3.11.8 FileStorage::ResetFile

[SWS_PER_00338] Definition of API function ara::per::FileStorage::ResetFile

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_PER_00009, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`class ara::per::FileStorage`	
Syntax:	`ara::core::Result< void > ResetFile (ara::core::StringView fileName) noexcept;`	
Parameters (in):	fileName	File name of the file. May correspond to the PersistencyFile.file Name of a configured file.
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the encryption or decryption of stored data fails.
	PerErrc::kInitValueNotAvailable	rollback_semantics
		Returned if no intitial value was configured for this file.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if the file is open, or if DeleteFile or RecoverFile with the same file name is currently being executed.
	PerErrc::kOutOfStorageSpace	rollback_semantics
		Returned if the available physical storage space is insufficient for the initial file.
	PerErrc::kTooManyFiles	rollback_semantics
		Returned if the number of files would get larger than the configured maxNumberOfFiles when the file is restored.
	PerErrc::kQuotaExceeded	rollback_semantics
		Returned if the initial file would exceed the configured maximum AllowedSize of the File Storage.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if calculating or checking the MAC of stored data fails.
Description:	Resets a file of this File Storage to its initial content. ResetFile allows to reset a single file to its initial content. If the file is currently not available in the File Storage, it is re-created. Afterwards, the file will appear in both cases as if it was newly installed from the current manifest. It will fail with kResourceBusy when the file is currently open, and with kInitValueNotAvailable when neither design nor deployment define an initial content for the file.	

⌋

8.3.11.9 FileStorage::GetCurrentFileSize

[SWS_PER_00407] Definition of API function ara::per::FileStorage::GetCurrentFileSize

Upstream requirements: RS_PER_00017, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`class ara::per::FileStorage`	
Syntax:	`ara::core::Result< std::uint64_t > GetCurrentFileSize (ara::core::StringView fileName) const noexcept;`	
Parameters (in):	fileName	File name of the file. May correspond to the PersistencyFile.fileName of a configured file.
Return value:	ara::core::Result< std::uint64_t >	A Result containing the occupied space in bytes. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrityCorrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kFileNotFound	rollback_semantics
		Returned if the provided file does not exist in the File Storage.
Description:	Returns the space in bytes currently occupied by the content of a file of this File Storage. The returned size might be inaccurate if any of the instances of a file is invalid or if another operation on the file takes place at the same time.	

⌋

8.3.11.10 FileStorage::GetFileInfo

[SWS_PER_00438] Definition of API function ara::per::FileStorage::GetFileInfo

Upstream requirements: RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`class ara::per::FileStorage`	
Syntax:	`ara::core::Result< FileInfo > GetFileInfo (ara::core::StringView fileName) const noexcept;`	
Parameters (in):	fileName	File name of the file. May correspond to the PersistencyFile.fileName of a configured file.
Return value:	ara::core::Result< FileInfo >	A Result containing a FileInfo struct. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrityCorrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kFileNotFound	rollback_semantics
		Returned if the provided file does not exist in the File Storage.
Description:	Returns additional information on a file of this File Storage. The returned FileInfo struct contains information about the times when the file was created, last modified, and last accessed, and about how and by whom the file was created and last modified. The modificationTime, accessTime, and fileModificationState returned in the FileInfo are only accurate if the file is currently not open.	

⌋

8.3.11.11 FileStorage::OpenFileReadWrite

[SWS_PER_00375] Definition of API function ara::per::FileStorage::OpenFileReadWrite

Upstream requirements: 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_00135, RS_AP_00139, RS_AP_00144, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`class ara::per::FileStorage`	
Syntax:	`ara::core::Result< UniqueHandle< ReadWriteAccessor > > OpenFileReadWrite (ara::core::StringView fileName) noexcept;`	
Parameters (in):	fileName	File name of the file. May correspond to the PersistencyFile.fileName of a configured file.
Return value:	ara::core::Result< UniqueHandle< ReadWriteAccessor > >	A Result containing a UniqueHandle for the file. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kIllegalWriteAccess	rollback_semantics
		Returned if the File Storage is configured as read-only.
	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrityCorrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidationFailed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if the file is already open, or if DeleteFile, RecoverFile, or ResetFile with the same file name is currently being executed.
	PerErrc::kOutOfStorageSpace	rollback_semantics
		Returned if the available physical storage space is insufficient for a new file.
	PerErrc::kTooManyFiles	rollback_semantics
		Returned if the number of files would get larger than the configured maxNumberOfFiles when the file is created.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if checking the MAC of stored data fails.

▽

△

Description:	Opens a file of this File Storage for reading and writing. The file is opened with the seek position set to the beginning (corresponding to kAtThe Beginning). If the file does not exist, it is created. The file will be closed when the returned UniqueHandle goes out of scope.

⌋

[SWS_PER_00113] Definition of API function ara::per::FileStorage::OpenFile ReadWrite

Upstream requirements: 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_00135, RS_AP_00139, RS_AP_00144, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`class ara::per::FileStorage`	
Syntax:	`ara::core::Result< UniqueHandle< ReadWriteAccessor > > OpenFileReadWrite (ara::core::StringView fileName, OpenMode mode) noexcept;`	
Parameters (in):	fileName	File name of the file. May correspond to the PersistencyFile.file Name 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 for the file. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kIllegalWriteAccess	rollback_semantics
		Returned if the File Storage is configured as read-only.
	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrityCorrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidationFailed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if the file is already open, or if DeleteFile, RecoverFile, or ResetFile with the same file name is currently being executed.
	PerErrc::kOutOfStorageSpace	rollback_semantics
		Returned if the available physical storage space is insufficient for a new file.
	PerErrc::kInvalidOpenMode	rollback_semantics

▽

△

		Returned if the passed mode contains an invalid combination of modes.
	PerErrc::kTooManyFiles	rollback_semantics
		Returned if the number of files would get larger than the configured maxNumberOfFiles when the file is created.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Opens a file of this File Storage for reading and writing with a defined mode. If not otherwise specified by the provided mode, the file is opened with the seek position set to the beginning (corresponding to kAtTheBeginning). If the file does not exist, it is created. The file will be closed when the returned UniqueHandle goes out of scope.	

⌋

[SWS_PER_00429] Definition of API function ara::per::FileStorage::OpenFile ReadWrite

Upstream requirements: 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_00135, RS_AP_00139, RS_AP_00144, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`class ara::per::FileStorage`	
Syntax:	`ara::core::Result< UniqueHandle< ReadWriteAccessor > > OpenFileReadWrite (ara::core::StringView fileName, OpenMode mode, ara::core::Span< ara::core::Byte > buffer) noexcept;`	
Parameters (in):	fileName	File name of the file. May correspond to the PersistencyFile.file Name of a configured file.
	mode	Mode with which the file shall be opened.
	buffer	Memory to be used for block-wise reading/writing.
Return value:	ara::core::Result< UniqueHandle< ReadWriteAccessor > >	A Result containing a UniqueHandle for the file. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kIllegalWrite Access	rollback_semantics
		Returned if the File Storage is configured as read-only.
	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.

▽

△

	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if the file is already open, or if DeleteFile, RecoverFile, or ResetFile with the same file name is currently being executed.
	PerErrc::kOutOfStorage Space	rollback_semantics
		Returned if the available physical storage space is insufficient for a new file.
	PerErrc::kInvalidOpen Mode	rollback_semantics
		Returned if the passed mode contains an invalid combination of modes.
	PerErrc::kTooManyFiles	rollback_semantics
		Returned if the number of files would get larger than the configured maxNumberOfFiles when the file is created.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Opens a file of this File Storage for reading and writing with a user provided buffer. If not otherwise specified by the provided mode, the file is opened with the seek position set to the beginning (corresponding to kAtTheBeginning). The provided buffer will be used by the ReadWriteAccessor to implement block-wise reading and writing to speed up multiple small accesses to the file. If the file does not exist, it is created. The file will be closed when the returned UniqueHandle goes out of scope.	

⌋

8.3.11.12 FileStorage::OpenFileReadOnly

[SWS_PER_00376] Definition of API function ara::per::FileStorage::OpenFileReadOnly

Upstream requirements: 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_00135, RS_AP_00139, RS_AP_00144, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`class ara::per::FileStorage`	
Syntax:	`ara::core::Result< UniqueHandle< ReadAccessor > > OpenFileReadOnly (ara::core::StringView fileName) noexcept;`	
Parameters (in):	fileName	File name of the file. May correspond to the PersistencyFile.file Name of a configured file.

▽

△

Return value:	ara::core::Result< UniqueHandle< Read Accessor > >	A Result containing a UniqueHandle for the file. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if the file is already open, or if DeleteFile, RecoverFile, or ResetFile with the same file name is currently being executed.
	PerErrc::kFileNotFound	rollback_semantics
		Returned if the provided file does not exist in the File Storage.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Opens a file of this File Storage for reading. The file is opened with the seek position set to the beginning (corresponding to kAtThe Beginning). The file will be closed when the returned UniqueHandle goes out of scope.	

⌋

[SWS_PER_00114] Definition of API function ara::per::FileStorage::OpenFile ReadOnly

Upstream requirements: 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_00135, RS_AP_00139, RS_AP_00144, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`class ara::per::FileStorage`	
Syntax:	`ara::core::Result< UniqueHandle< ReadAccessor > > OpenFileReadOnly (ara::core::StringView fileName, OpenMode mode) noexcept;`	
Parameters (in):	fileName	File name of the file. May correspond to the PersistencyFile.file Name of a configured file.
	mode	Mode with which the file shall be opened.
Return value:	ara::core::Result< UniqueHandle< Read Accessor > >	A Result containing a UniqueHandle for the file. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if the file is already open, or if DeleteFile, RecoverFile, or ResetFile with the same file name is currently being executed.
	PerErrc::kFileNotFound	rollback_semantics
		Returned if the provided file does not exist in the File Storage.
	PerErrc::kInvalidOpen Mode	rollback_semantics
		Returned if the passed mode contains an invalid combination of modes.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Opens a file of this File Storage for reading with a defined mode. If not otherwise specified by the provided mode, the file is opened with the seek position set to the beginning (corresponding to kAtTheBeginning). The file will be closed when the returned UniqueHandle goes out of scope.	

⌋

[SWS_PER_00430] Definition of API function ara::per::FileStorage::OpenFile ReadOnly

Upstream requirements: 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_00135, RS_AP_00139, RS_AP_00144, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`class ara::per::FileStorage`	
Syntax:	`ara::core::Result< UniqueHandle< ReadAccessor > > OpenFileReadOnly (ara::core::StringView fileName, OpenMode mode, ara::core::Span< ara::core::Byte > buffer) noexcept;`	
Parameters (in):	fileName	File name of the file. May correspond to the PersistencyFile.file Name of a configured file.
	mode	Mode with which the file shall be opened.
	buffer	Memory to be used for block-wise reading.

▽

△

Return value:	ara::core::Result< UniqueHandle< Read Accessor > >	A Result containing a UniqueHandle for the file. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if the file is already open, or if DeleteFile, RecoverFile, or ResetFile with the same file name is currently being executed.
	PerErrc::kFileNotFound	rollback_semantics
		Returned if the provided file does not exist in the File Storage.
	PerErrc::kInvalidOpen Mode	rollback_semantics
		Returned if the passed mode contains an invalid combination of modes.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Opens a file of this File Storage for reading with a user provided buffer. If not otherwise specified by the provided mode, the file is opened with the seek position set to the beginning (corresponding to kAtTheBeginning). The provided buffer will be used by the ReadAccessor to implement block-wise reading to speed up multiple small accesses to the file. The file will be closed when the returned UniqueHandle goes out of scope.	

⌋

8.3.11.13 FileStorage::OpenFileWriteOnly

[SWS_PER_00377] Definition of API function ara::per::FileStorage::OpenFileWriteOnly

Upstream requirements: 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_00135, RS_AP_00139, RS_AP_00144, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`class ara::per::FileStorage`	
Syntax:	`ara::core::Result< UniqueHandle< ReadWriteAccessor > > OpenFileWriteOnly (ara::core::StringView fileName) noexcept;`	
Parameters (in):	fileName	File name of the file. May correspond to the PersistencyFile.fileName of a configured file.
Return value:	ara::core::Result< UniqueHandle< ReadWriteAccessor > >	A Result containing a UniqueHandle for the file. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kIllegalWriteAccess	rollback_semantics
		Returned if the File Storage is configured as read-only.
	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrityCorrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidationFailed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if the file is already open, or if DeleteFile, RecoverFile, or ResetFile with the same file name is currently being executed.
	PerErrc::kOutOfStorageSpace	rollback_semantics
		Returned if the available physical storage space is insufficient for a new file.
	PerErrc::kTooManyFiles	rollback_semantics
		Returned if the number of files would get larger than the configured maxNumberOfFiles when the file is created.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if checking the MAC of stored data fails.

▽

△

Description:	Opens a file of this File Storage for writing. The file is truncated (corresponding to kTruncate). If the file does not exist, it is created. The file will be closed when the returned UniqueHandle goes out of scope.

⌋

[SWS_PER_00115] Definition of API function ara::per::FileStorage::OpenFileWriteOnly

Upstream requirements: 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_00135, RS_AP_00139, RS_AP_00144, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`class ara::per::FileStorage`	
Syntax:	`ara::core::Result< UniqueHandle< ReadWriteAccessor > > OpenFileWriteOnly (ara::core::StringView fileName, OpenMode mode) noexcept;`	
Parameters (in):	fileName	File name of the file. May correspond to the PersistencyFile.file Name of a configured file.
	mode	Mode with which the file shall be opened.
Return value:	ara::core::Result< UniqueHandle< Read WriteAccessor > >	A Result containing a UniqueHandle for the file. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kIllegalWrite Access	rollback_semantics
		Returned if the File Storage is configured as read-only.
	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if the file is already open, or if DeleteFile, RecoverFile, or ResetFile with the same file name is currently being executed.
	PerErrc::kOutOfStorage Space	rollback_semantics
		Returned if the available physical storage space is insufficient for a new file.
	PerErrc::kInvalidOpen Mode	rollback_semantics

▽

△

		Returned if the passed mode contains an invalid combination of modes.
	PerErrc::kTooManyFiles	rollback_semantics
		Returned if the number of files would get larger than the configured maxNumberOfFiles when the file is created.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Opens a file of this File Storage for writing with a defined mode. If not otherwise specified by the provided mode, the file is truncated (corresponding to k Truncate). If the file does not exist, it is created. The file will be closed when the returned UniqueHandle goes out of scope.	

⌋

[SWS_PER_00431] Definition of API function ara::per::FileStorage::OpenFile WriteOnly

Upstream requirements: 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_00135, RS_AP_00139, RS_AP_00144, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/file_storage.h"	
Scope:	`class ara::per::FileStorage`	
Syntax:	`ara::core::Result< UniqueHandle< ReadWriteAccessor > > OpenFileWrite Only (ara::core::StringView fileName, OpenMode mode, ara::core::Span< ara::core::Byte > buffer) noexcept;`	
Parameters (in):	fileName	File name of the file. May correspond to the PersistencyFile.file Name of a configured file.
	mode	Mode with which the file shall be opened.
	buffer	Memory to be used for block-wise writing.
Return value:	ara::core::Result< UniqueHandle< Read WriteAccessor > >	A Result containing a UniqueHandle for the file. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	PerErrc::kIllegalWrite Access	rollback_semantics
		Returned if the File Storage is configured as read-only.
	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kIntegrity Corrupted	rollback_semantics
		Returned if stored data cannot be read because the structural integrity is corrupted.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.

▽

△

	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kResourceBusy	rollback_semantics
		Returned if the file is already open, or if DeleteFile, RecoverFile, or ResetFile with the same file name is currently being executed.
	PerErrc::kOutOfStorage Space	rollback_semantics
		Returned if the available physical storage space is insufficient for a new file.
	PerErrc::kInvalidOpen Mode	rollback_semantics
		Returned if the passed mode contains an invalid combination of modes.
	PerErrc::kTooManyFiles	rollback_semantics
		Returned if the number of files would get larger than the configured maxNumberOfFiles when the file is created.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Opens a file of this File Storage for writing with a user provided buffer. If not otherwise specified by the provided mode, the file is truncated (corresponding to k Truncate). The provided buffer will be used by the ReadWriteAccessor to implement block-wise writing to speed up multiple small accesses to the file. If the file does not exist, it is created. The file will be closed when the returned UniqueHandle goes out of scope.	

⌋

8.3.12 Origin

[SWS_PER_00146] Definition of API enum ara::per::Origin

Upstream requirements: RS_PER_00003, RS_AP_00122, RS_AP_00125, RS_AP_00143

⌈

Kind:	enumeration	
Header file:	#include "ara/per/read_accessor.h"	
Forwarding header file:	#include "ara/per/per_fwd.h"	
Scope:	`namespace ara::per`	
Symbol:	Origin	
Underlying type:	std::uint32_t	
Syntax:	`enum class Origin : std::uint32_t {...};`	
Values:	kBeginning= 0	Seek from the beginning of the file.
	kCurrent= 1	Seek from the current position.
	kEnd= 2	Seek from the end of the file.
Description:	Specification of origin used in MovePosition.	

⌋

8.3.13 ReadAccessor Class

This section shows the methods available for a `ara::per::ReadAccessor` object obtained from a call to `ara::per::FileStorage::OpenFileReadOnly`, and for the inheriting `ara::per::ReadWriteAccessor` object obtained from a call to `ara::per::FileStorage::OpenFileWriteOnly` or `ara::per::FileStorage::OpenFileReadWrite`.

[SWS_PER_00342] Definition of API class ara::per::ReadAccessor

Upstream requirements: RS_PER_00004, RS_AP_00122, RS_AP_00146

⌈

Kind:	class
Header file:	#include "ara/per/read_accessor.h"
Forwarding header file:	#include "ara/per/per_fwd.h"
Scope:	`namespace ara::per`
Symbol:	ReadAccessor
Syntax:	`class ReadAccessor {...};`
Description:	ReadAccessor is used to read file data. It provides binary and text mode methods for checking or getting the current byte/character (PeekByte/PeekChar, GetByte/GetChar) methods for reading a section of a binary/text file (Read Binary/ReadText), a method to read a line of text (ReadLine), and methods for checking and setting the current position in the file (GetPosition, SetPosition, MovePosition, IsEof) and for checking the current size of the file (GetSize).

⌋

8.3.13.1 ReadAccessor::ReadAccessor

[SWS_PER_00461] Definition of API function ara::per::ReadAccessor::ReadAccessor

Upstream requirements: RS_PER_00004, RS_AP_00120, RS_AP_00129, RS_AP_00146

⌈

Kind:	function
Header file:	#include "ara/per/read_accessor.h"
Scope:	`class ara::per::ReadAccessor`
Syntax:	`ReadAccessor ()=delete;`
Description:	The default constructor for ReadAccessor shall not be used.

⌋

[SWS_PER_00413] Definition of API function ara::per::ReadAccessor::ReadAccessor

Upstream requirements: RS_PER_00004, RS_AP_00120, RS_AP_00121, RS_AP_00129, RS_AP_00145, RS_AP_00159

⌈

Kind:	function
Header file:	#include "ara/per/read_accessor.h"
Scope:	`class ara::per::ReadAccessor`
Syntax:	`ReadAccessor (ReadAccessor &&ra)=delete;`
Description:	The move constructor for ReadAccessor shall not be used.

⌋

[SWS_PER_00415] Definition of API function ara::per::ReadAccessor::ReadAccessor

Upstream requirements: RS_PER_00004, RS_AP_00120, RS_AP_00145

⌈

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

⌋

8.3.13.2 ReadAccessor::operator=

[SWS_PER_00414] Definition of API function ara::per::ReadAccessor::operator=

Upstream requirements: RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00145, RS_AP_00159

⌈

Kind:	function
Header file:	#include "ara/per/read_accessor.h"
Scope:	`class ara::per::ReadAccessor`
Syntax:	`ReadAccessor & operator= (ReadAccessor &&ra)=delete;`
Description:	The move assignment operator for ReadAccessor shall not be used.

⌋

[SWS_PER_00416] Definition of API function ara::per::ReadAccessor::operator=

Upstream requirements: RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00145

⌈

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

⌋

8.3.13.3 ReadAccessor::~ReadAccessor

[SWS_PER_00417] Definition of API function ara::per::ReadAccessor::~ReadAccessor

Upstream requirements: RS_PER_00004, RS_AP_00120, RS_AP_00129, RS_AP_00134, RS_AP_00145, RS_AP_00159

⌈

Kind:	function
Header file:	#include "ara/per/read_accessor.h"
Scope:	`class ara::per::ReadAccessor`
Syntax:	`~ReadAccessor () noexcept;`
Exception Safety:	exception safe
Thread Safety:	not thread-safe
Description:	Destructor for ReadAccessor.

⌋

8.3.13.4 ReadAccessor::PeekChar

[SWS_PER_00167] Definition of API function ara::per::ReadAccessor::PeekChar

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_accessor.h"	
Scope:	`class ara::per::ReadAccessor`	
Syntax:	`ara::core::Result< char > PeekChar () const noexcept;`	
Return value:	ara::core::Result< char >	A Result containing a character. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kValidationFailed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kEof	rollback_semantics
		Returned if the current position is at the end of the file or if the file is empty.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Returns the character at the current position of the file. The current position is not changed.	

⌋

8.3.13.5 ReadAccessor::PeekByte

[SWS_PER_00418] Definition of API function ara::per::ReadAccessor::PeekByte

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function
Header file:	#include "ara/per/read_accessor.h"
Scope:	`class ara::per::ReadAccessor`
Syntax:	`ara::core::Result< ara::core::Byte > PeekByte () const noexcept;`

▽

△

Return value:	ara::core::Result< ara::core::Byte >	A Result containing a byte. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kEof	rollback_semantics
		Returned if the current position is at the end of the file or if the file is empty.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Returns the byte at the current position of the file. The current position is not changed.	

⌋

8.3.13.6 ReadAccessor::GetChar

[SWS_PER_00168] Definition of API function ara::per::ReadAccessor::GetChar

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_accessor.h"	
Scope:	`class ara::per::ReadAccessor`	
Syntax:	`ara::core::Result< char > GetChar () noexcept;`	
Return value:	ara::core::Result< char >	A Result containing a character. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kEof	rollback_semantics

▽

△

		Returned if the current position is at the end of the file or if the file is empty.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Returns the character at the current position of the file, advancing the current position. In case of an error, the current position is not changed.	

⌋

8.3.13.7 ReadAccessor::GetByte

[SWS_PER_00419] Definition of API function ara::per::ReadAccessor::GetByte

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_accessor.h"	
Scope:	`class ara::per::ReadAccessor`	
Syntax:	`ara::core::Result< ara::core::Byte > GetByte () noexcept;`	
Return value:	ara::core::Result< ara::core::Byte >	A Result containing a byte. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kEof	rollback_semantics
		Returned if the current position is at the end of the file or if the file is empty.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Returns the byte at the current position of the file, advancing the current position. In case of an error, the current position is not changed.	

⌋

8.3.13.8 ReadAccessor::ReadText

[SWS_PER_00420] Definition of API function ara::per::ReadAccessor::ReadText

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00135, RS_AP_00136, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_accessor.h"	
Scope:	`class ara::per::ReadAccessor`	
Syntax:	`ara::core::Result< ara::core::String > ReadText () noexcept;`	
Return value:	ara::core::Result< ara::core::String >	A Result containing a String. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kValidationFailed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kEof	rollback_semantics
		Returned if the current position is at the end of the file or if the file is empty.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Reads all remaining characters into a String, starting from the current position. The returned string may start with an incomplete Unicode code point in case the current read position is in the middle of a code point. The current position is set to the end of the file. In case of an error, the current position is not changed.	

⌋

[SWS_PER_00165] Definition of API function ara::per::ReadAccessor::ReadText

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00135, RS_AP_00136, RS_AP_00139, RS_AP_00159

⌈

Kind:	function
Header file:	#include "ara/per/read_accessor.h"
Scope:	`class ara::per::ReadAccessor`
Syntax:	`ara::core::Result< ara::core::String > ReadText (std::uint64_t n) noexcept;`

▽

△

Parameters (in):	n	Number of characters to read.
Return value:	ara::core::Result< ara::core::String >	A Result containing a String. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kEof	rollback_semantics
		Returned if the current position is at the end of the file or if the file is empty.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Reads a number of characters into a String, starting from the current position. The returned string may start and/or end with incomplete Unicode code points in case the current read position and/or the last read character (code unit) is in the middle of a code point. The current position is advanced accordingly. If the end of the file is reached, the number of returned characters can be less than the requested number, and the current position is set to the end of the file. In case of an error, the current position is not changed.	

⌋

8.3.13.9 ReadAccessor::ReadBinary

[SWS_PER_00421] Definition of API function ara::per::ReadAccessor::ReadBinary

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_accessor.h"	
Scope:	`class ara::per::ReadAccessor`	
Syntax:	`ara::core::Result< ara::core::Vector< ara::core::Byte > > ReadBinary () noexcept;`	
Return value:	ara::core::Result< ara::core::Vector< ara::core::Byte > >	A Result containing a Vector of Byte. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	

▽

△

Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kEof	rollback_semantics
		Returned if the current position is at the end of the file or if the file is empty.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if checking the MAC of stored data fails.
Description:	Reads all remaining bytes into a Vector of Byte, starting from the current position. The current position is set to the end of the file. In case of an error, the current position is not changed.	

⌋

[SWS_PER_00422] Definition of API function ara::per::ReadAccessor::ReadBinary

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_accessor.h"	
Scope:	`class ara::per::ReadAccessor`	
Syntax:	`ara::core::Result< ara::core::Vector< ara::core::Byte > > ReadBinary (std::uint64_t n) noexcept;`	
Parameters (in):	n	Number of bytes to read.
Return value:	ara::core::Result< ara::core::Vector< ara::core::Byte > >	A Result containing a Vector of Byte. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kEof	rollback_semantics
		Returned if the current position is at the end of the file or if the file is empty.
	PerErrc::kAuthentication Failed	rollback_semantics

▽

△

		Returned if checking the MAC of stored data fails.
Description:	Reads a number of bytes into a Vector of Byte, starting from the current position. The current position is advanced accordingly. If the end of the file is reached, the number of returned bytes can be less than the requested number, and the current position is set to the end of the file. In case of an error, the current position is not changed.	

⌋

8.3.13.10 ReadAccessor::ReadLine

[SWS_PER_00119] Definition of API function ara::per::ReadAccessor::ReadLine

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00129, RS_AP_00135, RS_AP_00136, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_accessor.h"	
Scope:	`class ara::per::ReadAccessor`	
Syntax:	`ara::core::Result< ara::core::String > ReadLine (char delimiter='\n') noexcept;`	
Parameters (in):	delimiter	The character that is used as delimiter.
Return value:	ara::core::Result< ara::core::String >	A Result containing a String. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kValidationFailed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the decryption of stored data fails.
	PerErrc::kEof	rollback_semantics
		Returned if the current position is at the end of the file or if the file is empty.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if checking the MAC of stored data fails.

▽

△

Description:	Reads a complete line of characters into a String, advancing the current position accordingly. The end of the line is demarcated by the delimiter, or by "\\n" (ASCII 0x0a) if that parameter is omitted. The delimiter itself is not included in the returned String. Only Unicode code points with one character (code unit) can be used as delimiters. The returned string may start with an incomplete Unicode code point in case the current read position is in the middle of a code point. If the end of the file is reached, the remaining characters are returned and the current position is set to the end of the file. In case of an error, the current position is not changed.

⌋

8.3.13.11 ReadAccessor::GetSize

[SWS_PER_00424] Definition of API function ara::per::ReadAccessor::GetSize

Upstream requirements: RS_PER_00017, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00128, RS_AP_00129, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_accessor.h"	
Scope:	`class ara::per::ReadAccessor`	
Syntax:	`std::uint64_t GetSize () const noexcept;`	
Return value:	std::uint64_t	The current size of the file in bytes.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Returns the current size of a file in bytes.	

⌋

8.3.13.12 ReadAccessor::GetPosition

[SWS_PER_00162] Definition of API function ara::per::ReadAccessor::GetPosition

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_accessor.h"	
Scope:	`class ara::per::ReadAccessor`	
Syntax:	`std::uint64_t GetPosition () const noexcept;`	
Return value:	std::uint64_t	The current position in the file in bytes from the beginning of the file.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Returns the current position relative to the beginning of the file. The returned position may be at the end of the file.	

⌋

8.3.13.13 ReadAccessor::SetPosition

[SWS_PER_00163] Definition of API function ara::per::ReadAccessor::SetPosition

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_accessor.h"	
Scope:	`class ara::per::ReadAccessor`	
Syntax:	`ara::core::Result< void > SetPosition (std::uint64_t position) noexcept;`	
Parameters (in):	position	Current position in the file in bytes from the beginning of the file.
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kInvalidPosition	rollback_semantics
		Returned if the given position is beyond the end of the file.
Description:	Sets the current position relative to the beginning of the file. In case of an error, the current position is not changed.	

⌋

8.3.13.14 ReadAccessor::MovePosition

[SWS_PER_00164] Definition of API function ara::per::ReadAccessor::MovePosition

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_accessor.h"	
Scope:	`class ara::per::ReadAccessor`	
Syntax:	`ara::core::Result< std::uint64_t > MovePosition (Origin origin, std::int64_t offset) noexcept;`	
Parameters (in):	origin	Starting point from which to move 'offset' bytes.
	offset	Offset in bytes relative to 'origin'. Can be positive in case of k Beginning and kCurrent and negative in case of kCurrent and k End. In case of kCurrent, an offset of zero will not change the current position. In case of kEnd, an offset of zero will set the position to the end of the file.
Return value:	ara::core::Result< std::uint64_t >	A Result containing the new position in bytes from the beginning of the file. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kInvalidPosition	rollback_semantics
		Returned if the resulting position is lower than zero or beyond the end of the file.
Description:	Moves the current position in the file relative to the Origin. In case of an error, the current position is not changed.	

⌋

8.3.13.15 ReadAccessor::IsEof

[SWS_PER_00107] Definition of API function ara::per::ReadAccessor::IsEof

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_accessor.h"	
Scope:	`class ara::per::ReadAccessor`	
Syntax:	`bool IsEof () const noexcept;`	
Return value:	bool	True if the current position is at the end of the file, false otherwise.
Exception Safety:	exception safe	

▽

△

Thread Safety:	not thread-safe
Description:	Checks if the current position is at end of file.

⌋

8.3.14 ReadWriteAccessor Class

This section shows the methods available for a `ara::per::ReadWriteAccessor` object obtained from a call to `ara::per::FileStorage::OpenFileWriteOnly` or `ara::per::FileStorage::OpenFileReadWrite`.

[SWS_PER_00343] Definition of API class ara::per::ReadWriteAccessor

Upstream requirements: RS_PER_00004, RS_AP_00122, RS_AP_00146

⌈

Kind:	class
Header file:	#include "ara/per/read_write_accessor.h"
Forwarding header file:	#include "ara/per/per_fwd.h"
Scope:	`namespace ara::per`
Symbol:	ReadWriteAccessor
Base class:	ReadAccessor
Syntax:	`class ReadWriteAccessor final : public ReadAccessor {...};`
Description:	ReadWriteAccessor is used to read and write file data. It provides the WriteBinary and WriteText methods featuring a Result for controlled, unformatted writing, and the operator<< method for simple formatted writing. It also provides SyncToFile() to flush the buffer of the operating system to the physical storage.

⌋

8.3.14.1 ReadWriteAccessor::ReadWriteAccessor

[SWS_PER_00462] Definition of API function ara::per::ReadWriteAccessor::ReadWriteAccessor

Upstream requirements: RS_PER_00004, RS_AP_00120, RS_AP_00129, RS_AP_00146

⌈

Kind:	function
Header file:	#include "ara/per/read_write_accessor.h"
Scope:	`class ara::per::ReadWriteAccessor`

▽

△

Syntax:	`ReadWriteAccessor ()=delete;`
Description:	The default constructor for ReadWriteAccessor shall not be used.

⌋

8.3.14.2 ReadWriteAccessor::SyncToFile

[SWS_PER_00122] Definition of API function ara::per::ReadWriteAccessor::SyncToFile

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00128, RS_AP_00127, RS_AP_00129, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_write_accessor.h"	
Scope:	`class ara::per::ReadWriteAccessor`	
Syntax:	`ara::core::Result< void > SyncToFile () noexcept;`	
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysicalStorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kEncryptionFailed	rollback_semantics
		Returned if the encryption of stored data fails.
	PerErrc::kOutOfStorageSpace	rollback_semantics
		Returned if the available physical storage space is insufficient for the changed file.
	PerErrc::kQuotaExceeded	rollback_semantics
		Returned if the changed file would exceed the configured maximum AllowedSize of the File Storage.
	PerErrc::kAuthenticationFailed	rollback_semantics
		Returned if calculating the MAC of stored data fails.
Description:	Flushes the current file content to the physical storage.	

⌋

8.3.14.3 ReadWriteAccessor::SetFileSize

[SWS_PER_00428] Definition of API function ara::per::ReadWriteAccessor::SetFileSize

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00128, RS_AP_00127, RS_AP_00129, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_write_accessor.h"	
Scope:	`class ara::per::ReadWriteAccessor`	
Syntax:	`ara::core::Result< void > SetFileSize (std::uint64_t size) noexcept;`	
Parameters (in):	size	New size of the file.
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the encryption or decryption of stored data fails.
	PerErrc::kInvalidSize	rollback_semantics
		Returned if the new size is larger than the current size.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if calculating or checking the MAC of stored data fails.
Description:	Reduces the size of the file to 'size', effectively removing the current content of the file beyond this size. The current file position is unchanged if it is lower than 'size', or set to the last valid position in the file otherwise. If 'size' is 0, the current file position will also be set to 0.	

⌋

8.3.14.4 ReadWriteAccessor::WriteText

[SWS_PER_00166] Definition of API function ara::per::ReadWriteAccessor::WriteText

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00135, RS_AP_00136, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_write_accessor.h"	
Scope:	`class ara::per::ReadWriteAccessor`	
Syntax:	`ara::core::Result< void > WriteText (ara::core::StringView s) noexcept;`	
Parameters (in):	s	A StringView containing the characters to be written.
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the encryption or decryption of stored data fails.
	PerErrc::kOutOfStorage Space	rollback_semantics
		Returned if the available physical storage space is insufficient for the changed file.
	PerErrc::kQuota Exceeded	rollback_semantics
		Returned if the changed file would exceed the configured maximum AllowedSize of the File Storage.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if calculating or checking the MAC of stored data fails.
Description:	Writes the content of a StringView to the file. The time when the content is persisted depends on the implementation of Persistency. SyncTo File can be used to force Persistency to persist the file content. In case of an error, the file content might be corrupted, and the current position might or might not have changed. The expected state of the file for each supported error can be expected to be as follows: • kPhysicalStorageFailure: The state of the file is unknown. It could have been entirely destroyed. • kEncryptionFailed: The content of the file and the current position will have been updated, but could not be persisted. The persisted file will reflect an older version of the file. • kOutOfStorageSpace: The content of the file will have been updated, but the part of the operation that exceeded the quota will have been discarded. The current position will be at the end of the file.	

⌋

8.3.14.5 ReadWriteAccessor::WriteBinary

[SWS_PER_00423] Definition of API function ara::per::ReadWriteAccessor::WriteBinary

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00135, RS_AP_00139, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_write_accessor.h"	
Scope:	`class ara::per::ReadWriteAccessor`	
Syntax:	`ara::core::Result< void > WriteBinary (ara::core::Span< const ara::core::Byte > b) noexcept;`	
Parameters (in):	b	A Span of Byte containing the bytes to be written.
Return value:	ara::core::Result< void >	A Result of void. In case of an error, it contains any of the errors defined below, or a vendor specific error.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	PerErrc::kPhysical StorageFailure	rollback_semantics
		Returned if access to the physical storage fails.
	PerErrc::kValidation Failed	rollback_semantics
		Returned if the validity of stored data cannot be ensured.
	PerErrc::kEncryption Failed	rollback_semantics
		Returned if the encryption or decryption of stored data fails.
	PerErrc::kOutOfStorage Space	rollback_semantics
		Returned if the available physical storage space is insufficient for the changed file.
	PerErrc::kQuota Exceeded	rollback_semantics
		Returned if the changed file would exceed the configured maximum AllowedSize of the File Storage.
	PerErrc::kAuthentication Failed	rollback_semantics
		Returned if calculating or checking the MAC of stored data fails.
Description:	Writes the content of a Span of Byte to the file. The time when the content is persisted depends on the implementation of Persistency. SyncTo File can be used to force Persistency to persist the file content. In case of an error, the file content might be corrupted, and the current position might or might not have changed. The expected state of the file for each supported error can be expected to be as follows: • kPhysicalStorageFailure: The state of the file is unknown. It could have been entirely destroyed. • kEncryptionFailed: The content of the file and the current position will have been updated, but could not be persisted. The persisted file will reflect an older version of the file. • kOutOfStorageSpace: The content of the file will have been updated, but the part of the operation that exceeded the quota will have been discarded. The current position will be at the end of the file.	

⌋

8.3.14.6 ReadWriteAccessor::operator<<

[SWS_PER_00125] Definition of API function ara::per::ReadWriteAccessor::operator<<

Upstream requirements: RS_PER_00001, RS_PER_00004, RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00127, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/per/read_write_accessor.h"	
Scope:	`class ara::per::ReadWriteAccessor`	
Syntax:	`ReadWriteAccessor & operator<< (ara::core::StringView s) noexcept;`	
Parameters (in):	s	The StringView containing the characters to be written.
Return value:	ReadWriteAccessor &	The ReadWriteAccessor object.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Writes the content of a StringView to the file. This operator is just a comfort feature for non-safety critical Persistency users. If an error occurs during this operation, it is silently ignored.	

⌋

9 Service Interfaces

This functional cluster does not define any provided or required service interfaces.

10 Configuration

The configuration model of this functional cluster is defined in [8]. This chapter defines the default values for attributes and semantic constraints for elements specified in [8] that are part of the configuration model of this functional cluster.

10.1 Default Values

This functional cluster does not define any default values for attributes specified in [8].

10.2 Semantic Constraints

This section defines semantic constraints for the configuration elements of `Persistency` defined in the [8, TPS Manifest Specification] that need to be observed by the tooling which creates/processes this part of the `Execution Manifest`. There are also several additional constraints on the `Persistency` configuration that are defined in the [8, TPS Manifest Specification].

[SWS_PER_CONSTR_00007] Configurable Namespace for Persistency ⌈`PersistencyInterface.namespace` shall never exist.⌋

A Mentioned Manifest Elements

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

This chapter is generated.

Class	***AbstractPersistencyRequireComSpec*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec			
Note	This meta-class acts as a bas-class of persistency-related ComSpec classes.			
Base	*ARObject*, *RPortComSpec*			
Subclasses	PersistencyDataRequiredComSpec, PersistencyFileRequiredComSpec			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
accessMode	PersistencyAccessEnum	0..1	attr	This attribute controls how persistent data can be accessed.

Table A.1: AbstractPersistencyRequireComSpec

Class	***AbstractRequiredPortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	This abstract class provides the ability to become a required PortPrototype.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Subclasses	PRPortPrototype, RPortPrototype			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
requiredComSpec	RPortComSpec	*	aggr	Required communication attributes, one for each interface element. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=requiredComSpec

Table A.2: AbstractRequiredPortPrototype

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.recommendedPackage=AdaptiveApplicationSwComponentTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

△

Class	AdaptiveApplicationSwComponentType			
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.shortName, internal Behavior.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table A.3: AdaptiveApplicationSwComponentType

Class	***ApplicationDataType*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	ApplicationDataType defines a data type from the application point of view. Especially it should be used whenever something "physical" is at stake. An ApplicationDataType represents a set of values as seen in the application model, such as measurement units. It does not consider implementation details such as bit-size, endianess, etc. It should be possible to model the application level aspects of a VFB system by using ApplicationData Types only.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*ApplicationCompositeDataType*, ApplicationPrimitiveDataType			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.4: ApplicationDataType

Class	***AutosarDataType*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	Abstract base class for user defined AUTOSAR data types for software.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractImplementationDataType*, *ApplicationDataType*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of this AutosarDataType. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=swDataDefProps

Table A.5: AutosarDataType

Class	***BaseType*** (abstract)
Package	M2::MSR::AsamHdo::BaseTypes
Note	This abstract meta-class represents the ability to specify a platform dependent base type.
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*
Subclasses	SwBaseType

▽

△

Class	***BaseType*** (abstract)			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
baseType Definition	BaseTypeDefinition	1	aggr	This is the actual definition of the base type. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

Table A.6: BaseType

Class	**BaseTypeDirectDefinition**			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This BaseType is defined directly (as opposite to a derived BaseType)			
Base	*ARObject*, *BaseTypeDefinition*			
Aggregated by	*BaseType*.baseTypeDefinition			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
baseType Encoding	BaseTypeEncoding String	0..1	attr	This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence. **Tags:** xml.sequenceOffset=90
baseTypeSize	PositiveInteger	0..1	attr	Describes the length of the data type specified in the container in bits. **Tags:** xml.sequenceOffset=70
byteOrder	ByteOrderEnum	0..1	attr	This attribute specifies the byte order of the base type. **Tags:** xml.sequenceOffset=110
memAlignment	PositiveInteger	0..1	attr	This attribute describes the alignment of the memory object in bits. E.g. "8" specifies, that the object in question is aligned to a byte while "32" specifies that it is aligned four byte. If the value is set to "0" the meaning shall be interpreted as "unspecified". **Tags:** xml.sequenceOffset=100
native Declaration	NativeDeclarationString	0..1	attr	This attribute describes the declaration of such a base type in the native programming language, primarily in the Programming language C. This can then be used by a code generator to include the necessary declarations into a header file. For example BaseType with shortName: "MyUnsignedInt" native Declaration: "unsigned short" Results in typedef unsigned short MyUnsignedInt; If the attribute is not defined the referring Implementation DataTypes will not be generated as a typedef by RTE. If a nativeDeclaration type is given it shall fulfill the characteristic given by basetypeEncoding and baseType Size. This is required to ensure the consistent handling and interpretation by software components, RTE, COM and MCM systems. **Tags:** xml.sequenceOffset=120

Table A.7: BaseTypeDirectDefinition

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			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *CppImplementationDataTypeContextTarget*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	CustomCppImplementationDataType, StdCppImplementationDataType			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arraySize	PositiveInteger	0..1	attr	This attribute can be used to specify the array size if the enclosing CppImplementationDataType has array semantics. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
headerFile	String	0..1	attr	Configuration of the Header File with the custom class declaration.
namespace (ordered)	SymbolProps	*	aggr	This aggregation allows for the definition an own namespace for the enclosing CppImplementationData Type.
subElement (ordered)	CppImplementation DataTypeElement	*	aggr	This represents the collection of sub-elements of the enclosing CppImplementationDataType
template Argument (ordered)	CppTemplateArgument	*	aggr	This aggregation allows for the specification of properties of template arguments
typeEmitter	NameToken	0..1	attr	This attribute can be taken to control how the respective CppImplementationDataType is contributed to the language binding.
typeReference	CppImplementation DataType	0..1	ref	This reference shall be defined to define a type reference (a.k.a. typedef).

Table A.8: CppImplementationDataType

Enumeration	**CryptoKeySlotUsageEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment
Note	This enum defines the possible roles of the keySlotUsage.
Aggregated by	PersistencyDeploymentElementToCryptoKeySlotMapping.keySlotUsage, PersistencyDeploymentTo CryptoKeySlotMapping.keySlotUsage
Literal	***Description***
encryption	Key slot usage for encryption **Tags:** atp.EnumerationLiteralIndex=1
verification	Key slot usage for verification **Tags:** atp.EnumerationLiteralIndex=0

Table A.9: CryptoKeySlotUsageEnum

Class	**Executable**
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ApplicationStructure
Note	This meta-class represents an executable program. **Tags:** atp.recommendedPackage=Executables

▽

△

Class	Executable			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
buildType	BuildTypeEnum	0..1	attr	This attribute describes the buildType of a module and/or platform implementation.
implementation Props	Executable ImplementationProps	*	aggr	This aggregation contains the collection of implementation-specific properties necessary to properly build the enclosing Executable.
minimumTimer Granularity	TimeValue	0..1	attr	This attribute describes the minimum timer resolution (TimeValue of one tick) that is required by the Executable.
reporting Behavior	ExecutionState ReportingBehavior Enum	0..1	attr	this attribute controls the execution state reporting behavior of the enclosing Executable.
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.
traceSwitch Configuration	TraceSwitch Configuration	*	aggr	Configuration of the MsgId based trace switch **Tags:** atp.Status=draft
version	StrongRevisionLabel String	0..1	attr	Version of the executable.

Table A.10: Executable

Class	***FunctionalClusterInteractsWithFunctionalClusterMapping*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::FunctionalClusterInteractsWithFunctionalClusterMapping			
Note	This meta-class identifies a relation between functional clusters on the adaptive platform such one functional cluster can call APIs of the other functional cluster.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Subclasses	ArtifactChecksumToCryptoProviderMapping, ComCertificateToCryptoCertificateMapping, ComKeyTo CryptoKeySlotMapping, ComSecOcToCryptoKeySlotMapping, FunctionalClusterInteractsWithDiagnostic EventMapping, FunctionalClusterInteractsWithPersistencyDeploymentMapping, FunctionalClusterTo SecurityEventDefinitionMapping, NmInteractsWithSmMapping, PersistencyDeploymentElementToCrypto KeySlotMapping, PersistencyDeploymentToCryptoKeySlotMapping, PersistencyDeploymentToDltLogSink Mapping, SmInteractsWithNmMapping, TimeBaseProviderToPersistencyMapping, UcmToTimeBase ResourceMapping			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.11: FunctionalClusterInteractsWithFunctionalClusterMapping

Class	**FunctionalClusterInteractsWithPersistencyDeploymentMapping**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency
Note	This meta-class represents the ability to define a mapping between any functional cluster modeled as a subclass of NonOsModuleInstantiation and a PersistencyDeployment. **Tags:** atp.recommendedPackage=FCInteractions

▽

△

Class	**FunctionalClusterInteractsWithPersistencyDeploymentMapping**			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contractVersion	StrongRevisionLabel String	0..1	attr	This attribute represents the contract version that is used to determine whether the Persistency configuration experienced structural changes and is also used for the check for data type compatibility.
functional Cluster	NonOsModule Instantiation	0..1	ref	This reference identifies the client functional cluster that wants to use persistency.
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 FunctionalClusterInteractsWithPersistency DeploymentMapping.
persistency Access	FunctionalCluster PersistencyAccess Enum	0..1	attr	This attribute represents the definition of the persistency access of all kinds of persisted data at run-time in the context of the enclosing FunctionalClusterInteractsWith PersistencyDeploymentMapping.
persistency Deployment	PersistencyDeployment	0..1	ref	This reference identifies the applicable Persistency Deployment.
process	Process	0..1	ref	"This reference identifies the applicable process.

Table A.12: FunctionalClusterInteractsWithPersistencyDeploymentMapping

Enumeration	**FunctionalClusterPersistencyAccessEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency
Note	This meta-class provides possible values about how functional clusters may use persistency with respect to the direction of access.
Aggregated by	FunctionalClusterInteractsWithPersistencyDeploymentMapping.persistencyAccess
Literal	***Description***
read	Functional Cluster wants to access persistency with a read semantics. **Tags:** atp.EnumerationLiteralIndex=0
readWrite	Functional Cluster wants to access persistency with a read and write semantics. **Tags:** atp.EnumerationLiteralIndex=2
write	Functional Cluster wants to access persistency with a write semantics. **Tags:** atp.EnumerationLiteralIndex=1

Table A.13: FunctionalClusterPersistencyAccessEnum

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

▽

△

Class	***Identifiable*** (abstract)			
Subclasses	ARPackage, *AbstractDoIpLogicAddressProps*, *AbstractEvent*, *AbstractFunctionalClusterDesign*, *AbstractImplementationDataTypeElement*, *AbstractSecurityEventFilter*, *AbstractSecurityIdsmInstance Filter*, *AbstractServiceInstance*, *AbstractSignalBasedToISignalTriggeringMapping*, AdaptiveSwcInternal Behavior, ApApplicationEndpoint, *ApmcAbstractDefinition*, *ApmcConfigurationElementDef*, *Apmc ContainerElementValue*, ApmcContainerValue, ApmcEnumerationLiteralDef, ApplicationEndpoint, ApplicationError, AppliedStandard, ArtifactChecksum, ArtifactLocator, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, AutosarOperationArgumentInstance, AutosarVariableInstance, *BuildAction Entity*, BuildActionEnvironment, Chapter, CheckpointTransition, ClassContentConditional, ClientId Definition, ClientServerOperation, Code, *CollectableElement*, ComManagementMapping, *Comm ConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEventGroup, CouplingPort, *CouplingPortAbstractShaper*, *CouplingPortStructuralElement*, CryptoCertificate, CryptoKeySlot, CryptoKeySlotDesign, CryptoKeySlotUsageDesign, CryptoProvider, *CryptoServiceMapping*, DataPrototypeGroup, DataPrototypeTransformationPropsIdent, Data Transformation, DdsCpDomain, DdsCpPartition, DdsCpQosProfile, DdsCpTopic, DdsDomainRange, DependencyOnArtifact, *DiagEventDebounceAlgorithm*, DiagnosticAuthTransmitCertificateEvaluation, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticDebounceAlgorithmProps, Diagnostic FunctionInhibitSource, DiagnosticParameterElement, *DiagnosticRoutineSubfunction*, DiagnosticSovd MethodPrimitive, DltApplication, DltArgument, DltMessage, DoIpInterface, DoIpLogicAddress, DoIp LogicalAddress, DoIpNetworkConfigurationDesign, DoIpRoutingActivation, E2EProfileConfiguration, End2EndEventProtectionProps, End2EndMethodProtectionProps, EndToEndProtection, Ethernet WakeupSleepOnDatalineConfig, EventHandler, EventMapping, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeatureMap Element, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FieldMapping, FireAndForget MethodMapping, FlexrayArTpNode, FlexrayTpPduPool, *FrameTriggering*, GeneralParameter, Global Supervision, GlobalTimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HealthChannel*, *HeapUsage*, HwAttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, *IEEE1722TpAcfBus*, *IEEE1722TpAcfBus Part*, IPSecRule, IPv6ExtHeaderFilterList, ISignalToIPduMapping, ISignalTriggering, *IdentCaption*, ImpositionTime, InternalTriggeringPoint, Keyword, LifeCycleState, Linker, MacAddressVlanMembership, MacMulticastGroup, MacSecKayParticipant, McDataInstance, MemorySection, MemoryUsage, Method Mapping, ModeDeclaration, ModeDeclarationMapping, ModeSwitchPoint, NetworkEndpoint, *NmCluster*, *NmNode*, *PackageableElement*, ParameterAccess, PduActivationRoutingGroup, PduToFrameMapping, PduTriggering, PerInstanceMemory, *PersistencyDeploymentElement*, *PersistencyInterfaceElement*, *Phm Supervision*, *PhysicalChannel*, PortGroup, *PortInterfaceMapping*, ProcessToMachineMapping, Processor, ProcessorCore, PskIdentityToKeySlotMapping, ResourceConsumption, ResourceGroup, RootSwClusterDesignComponentPrototype, RootSwComponentPrototype, RootSwComposition Prototype, RptComponent, RptContainer, RptExecutableEntity, RptExecutableEntityEvent, RptExecution Context, RptProfile, RptServicePoint, RunnableEntityGroup, *SdgAttribute*, SdgClass, SecOcJobMapping, SecOcJobRequirement, SecureCommunicationAuthenticationProps, *SecureCommunicationDeployment*, SecureCommunicationFreshnessProps, SecurityEventContextDataElement, SecurityEventContextProps, *ServiceEventDeployment*, *ServiceFieldDeployment*, ServiceInterfaceElementSecureComConfig, *Service MethodDeployment*, *ServiceNeeds*, SignalServiceTranslationEventProps, SignalServiceTranslation Props, SocketAddress, SoftwarePackageStep, SomeipEventGroup, SomeipProvidedEventGroup, SomeipTpChannel, *SpecElementReference*, *StackUsage*, *StateManagementActionItem*, State ManagementActionList, StateManagementStateNotification, *StateManagementStateRequest*, Static SocketConnection, StructuredReq, SupervisionCheckpoint, SupervisionMode, SupervisionMode Condition, SwGenericAxisParamType, SwServiceArg, SwcServiceDependency, SwitchAsynchronous TrafficShaperGroupEntry, SystemMapping, *TimeBaseResource*, *TimingClock*, TimingClockSync Accuracy, TimingCondition, *TimingConstraint*, *TimingDescription*, TimingExtensionResource, Timing ModeInstance, TlsCryptoCipherSuite, TlsCryptoCipherSuiteProps, TlsJobMapping, Topic1, TpAddress, TraceableTable, TraceableText, *TracedFailure*, TransformationISignalPropsIdent, *TransformationProps*, TransformationTechnology, Trigger, UcmDescription, UcmRetryStrategy, UcmStep, VariableAccess, VariationPointProxy, VehicleRolloutStep, ViewMap, VlanConfig, WaitPoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=adminData xml.sequenceOffset=-40

▽

△

Class	***Identifiable*** (abstract)			
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. **Tags:** xml.sequenceOffset=-25
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. **Tags:** xml.sequenceOffset=-50
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". **Tags:** xml.sequenceOffset=-60
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. **Tags:** xml.sequenceOffset=-30
uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp. **Tags:** xml.attribute=true

Table A.14: Identifiable

Class	***NonOsModuleInstantiation*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::AdaptiveModule Implementation
Note	This meta-class defines the abstract attributes for the configuration of an adaptive autosar module other than the OS module.
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	NonOsModuleInstantiation (abstract)			
Subclasses	AdaptiveFirewallModuleInstantiation, CmModuleInstantiation, CryptoModuleInstantiation, DoIp Instantiation, GenericDiagnosticTransportInstantiation, GenericModuleInstantiation, *IdsPlatform Instantiation*, LogAndTraceInstantiation, NmInstantiation, *SovdModuleInstantiation*, StateManagement ModuleInstantiation, TimeSyncModuleInstantiation, *UcmModuleInstantiation*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.15: NonOsModuleInstantiation

Enumeration	PersistencyAccessEnum
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::ComSpec
Note	This enumeration provides standardized ways of how persistent data can be accessed.
Aggregated by	*AbstractPersistencyRequireComSpec*.accessMode
Literal	**Description**
read	Access to persistent data is read-only. **Tags:** atp.EnumerationLiteralIndex=0
readWrite	Persistent data can both be written and read. **Tags:** atp.EnumerationLiteralIndex=3
write	Access to persistent data is write-only **Tags:** atp.EnumerationLiteralIndex=1

Table A.16: PersistencyAccessEnum

Enumeration	PersistencyCollectionLevelUpdateStrategyEnum
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface
Note	This enumeration provides possible values for the update strategy on interface/storage level.
Aggregated by	*PersistencyDeployment*.updateStrategy, *PersistencyInterface*.updateStrategy
Literal	**Description**
delete	The update strategy is to delete all values on the level of the respective collection. **Tags:** atp.EnumerationLiteralIndex=1
keepExisting	The update strategy is to keep the existing values on the level of the respective collection. **Tags:** atp.EnumerationLiteralIndex=0

Table A.17: PersistencyCollectionLevelUpdateStrategyEnum

Class	PersistencyDataElement			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::Persistency			
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 PersistencyKeyValueStorageInterface. PersistencyDataElement represents also a key-value pair of the deployed PersistencyKeyValueStorage and provides an initial value.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *Multilanguage Referrable*, *PersistencyInterfaceElement*, *Referrable*			
Aggregated by	*AtpClassifier*.atpFeature, PersistencyKeyValueStorageInterface.dataElement			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table A.18: 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.			
Base	*ARObject*, *AbstractPersistencyRequireComSpec*, *RPortComSpec*			
Aggregated by	*AbstractRequiredPortPrototype*.requiredComSpec, PortPrototypeBlueprint.requiredComSpec			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataElement	PersistencyData Element	0..1	ref	This refrence represents the PersistencyDataElement for which the PersistencyDataRequiredComSpec applies.
initValue	ValueSpecification	0..1	aggr	This aggregation represents the definition of an initial value for the PersistencyDataElement referenced by the enclosing PersistencyDataRequiredComSpec

Table A.19: 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.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusivePackageElement*, *UploadablePackageElement*			
Subclasses	PersistencyFileStorage, PersistencyKeyValueStorage			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
deploymentUri (ordered)	PersistencyDeployment Uri	*	aggr	This aggregation represents the collection of URIs relevant for the enclosing PersistencyDeployment.
maximum AllowedSize	PositiveUnlimitedInteger	0..1	attr	The value of this attribute represents the maximum size (unit: bytes) allowed at target-configuration time for the enclosing PersistencyDeployment.
minimum SustainedSize	PositiveInteger	0..1	attr	The value of this attribute represents the minimum size (unit: bytes) guaranteed at deployment time for the enclosing PersistencyDeployment.
redundancy Handling	PersistencyRedundancy Handling	*	aggr	This aggregation represents the chosen approaches to handle redundancy.
updateStrategy	PersistencyCollection LevelUpdateStrategy Enum	0..1	attr	This attribute shall be used to specify the update strategy of the respective PersistencyDeployment as a whole.
version	StrongRevisionLabel String	0..1	attr	The attribute represents the version of the `PersistencyFileStorage` or `PersistencyKeyValueStorage`.

Table A.20: PersistencyDeployment

Class	***PersistencyDeploymentElement*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This abstract meta-class serves as a base class for concrete classes representing different aspects of elements of a `PersistencyDeployment`.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	PersistencyFile, PersistencyKeyValuePair			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	***PersistencyDeploymentElement*** (abstract)			
updateStrategy	PersistencyElement LevelUpdateStrategy Enum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyDeploymentElement.

Table A.21: PersistencyDeploymentElement

Class	**PersistencyDeploymentElementToCryptoKeySlotMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to define a mapping between the PersistencyDeploymentElement and a CryptoKeySlot. **Tags:** atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cryptoAlgorithm String	String	0..1	attr	This attribute defines the cryptographic algorithm used for hashing, encryption, decryption, signature/MAC verification, or MAC generation.
cryptoKeySlot	CryptoKeySlot	0..1	ref	This reference represents the mapped CryptoKeySlot.
keySlotUsage	CryptoKeySlotUsage Enum	0..1	attr	This attribute defines the role of the keySlot assignment.
persistency Deployment Element	PersistencyDeployment Element	0..1	ref	This reference represents the mapped Persistency Deployment.
verificationHash	String	0..1	attr	This attribute defines the hash of the storage used in case of verification.

Table A.22: PersistencyDeploymentElementToCryptoKeySlotMapping

Class	**PersistencyDeploymentToCryptoKeySlotMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::CryptoDeployment			
Note	This meta-class represents the ability to define a mapping between the PersistencyDeployment and a CryptoKeySlot. **Tags:** atp.recommendedPackage=FCInteractions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
cryptoAlgorithm String	String	0..1	attr	This attribute defines the cryptographic algorithm used for hashing, encryption, decryption, signature/MAC verification, or MAC generation.
cryptoKeySlot	CryptoKeySlot	0..1	ref	This reference represents the mapped CryptoKeySlot.
keySlotUsage	CryptoKeySlotUsage Enum	0..1	attr	This attribute defines the role of the keySlot assignment.
persistency Deployment	PersistencyDeployment	0..1	ref	This reference represents the mapped Persistency Deployment.
verificationHash	String	0..1	attr	This attribute defines the hash of the storage used in case of verification.

Table A.23: PersistencyDeploymentToCryptoKeySlotMapping

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

Table A.24: 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.recommendedPackage=PersistencyFiles			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PersistencyDeploymentElement*, *Referrable*			
Aggregated by	PersistencyFileStorage.file			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contentUri	UriString	0..1	attr	This attribute represents the URI that identifies the initial content of the PersistencyFile.
fileName	String	0..1	attr	This attribute holds filename part of the storage location for the PersistencyFile, e.g. file on the file system.

Table A.25: PersistencyFile

Class	**PersistencyFileElement**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::Persistency			
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.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PersistencyInterfaceElement*, *Referrable*			
Aggregated by	PersistencyFileStorageInterface.fileElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
contentUri	UriString	0..1	attr	This attribute represents the URI that identifies the initial content of the PersistencyFile.
fileName	String	0..1	attr	This attribute holds the filename part of the storage location, e.g. file on the file system.

Table A.26: PersistencyFileElement

Class	**PersistencyFileStorage**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class comes with the ability to define a collection of single files (directory) that creates the deployment-side counterpart to a `PortPrototype` typed by a `PersistencyFileStorageInterface`. **Tags:** atp.recommendedPackage=PersistencyFileStorages			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PersistencyDeployment*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusive PackageElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
file	PersistencyFile	*	aggr	This aggregation represents the collection of files aggregated by the PersistencyFileStorage.

Table A.27: PersistencyFileStorage

Class	**PersistencyFileStorageInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::Persistency			
Note	This meta-class provides the ability to implement a PortInterface for supporting persistency use cases for files. **Tags:** atp.recommendedPackage=PersistencyFileStorageInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyInterface*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
fileElement	PersistencyFileElement	*	aggr	This aggregation represents the collection of Persistency FileStorages in the context of the enclosing Persistency FileStorageInterface.
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 PersistencyFileStorageInterface.

Table A.28: PersistencyFileStorageInterface

Class	***PersistencyInterface*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::Persistency			
Note	This meta-class provides the abstract ability to define a PortInterface for the support of persistency use cases.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	PersistencyFileStorageInterface, PersistencyKeyValueStorageInterface			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
contractVersion	StrongRevisionLabel String	0..1	attr	This attribute represents the contract version that is used to determine whether the Persistency configuration experienced structural changes and is also used for the check for data type compatibility.
minimum SustainedSize	PositiveInteger	0..1	attr	The value of this attribute represents the minimum size (unit: bytes) required at design time for the enclosing PersistencyInterface.
redundancy	PersistencyRedundancy Enum	0..1	attr	This attribute represents a requirement towards the redundancy of storage.

▽

△

Class	***PersistencyInterface*** (abstract)			
redundancy Handling	PersistencyRedundancy Handling	*	aggr	This aggregation represents the chosen approaches to handle redundancy for the various use cases implemented by subclasses
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 A.29: PersistencyInterface

Class	***PersistencyInterfaceElement*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::Persistency			
Note	This meta-class provides the abstract ability to define an element of a PortInterface for the support of persistency use cases.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	PersistencyDataElement, PersistencyFileElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
updateStrategy	PersistencyElement LevelUpdateStrategy Enum	0..1	attr	This attribute can be used to specify the update strategy of the respective PersistencyInterfaceElement.

Table A.30: PersistencyInterfaceElement

Class	**PersistencyKeyValueDataTypeMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::Persistency			
Note	This meta-class represents the ability to define a mapping between an existing data type in a key-value-storage stored by a previous version to a new data type used on application software level in the current version.			
Base	*ARObject*, *Describable*			
Aggregated by	PersistencyKeyValueStorageInterface.dataTypeMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
currentData Type	AutosarDataType	0..1	ref	This reference identifies the current data type for an existing key-value-pair in the context of the enclosing PersistencyKeyValueStorageInterface.
previous ContractVersion	StrongRevisionLabel String	0..1	attr	This attribute identifies the contract version in which the previousDataType was used.
previousData Type	AutosarDataType	0..1	ref	This reference identifies the previous data type in a key-value-pair existing in the context of the enclosing PersistencyKeyValueStorageInterface.

Table A.31: PersistencyKeyValueDataTypeMapping

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 target-configuration of persistency.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *PersistencyDeploymentElement*, *Referrable*			
Aggregated by	PersistencyKeyValueStorage.keyValuePair			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**PersistencyKeyValuePair**			
initValue	ValueSpecification	0..1	aggr	This aggregation represents the ability to define an initial value for the value side of the key-value pair. Please note that it does not make sense to configure an initial value if the `PersistencyDeploymentElement.updateStrategy` is set to the value `delete`.
valueDataType	AbstractImplementationDataType	0..1	ref	This reference represents the data type applicable for the value of the key-value pair.

Table A.32: PersistencyKeyValuePair

Class	**PersistencyKeyValueStorage**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to model a key-value storage on target-configuration level. **Tags:** atp.recommendedPackage=PersistencyKeyValueStorages			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyDeployment*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusivePackageElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
keyValuePair	PersistencyKeyValuePair	*	aggr	This aggregation represents the key-value-pairs owned by the enclosing `PersistencyKeyValueStorage`.

Table A.33: PersistencyKeyValueStorage

Class	**PersistencyKeyValueStorageInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface::Persistency			
Note	This meta-class provides the ability to implement a `PortInterface` for supporting persistency use cases for data. **Tags:** atp.recommendedPackage=PersistencyKeyValueStorageInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PersistencyInterface*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dataElement	PersistencyDataElement	*	aggr	This aggregation represents the collection of Persistency DataElements in the context of the enclosing Persistency KeyValueStorageInterface.
dataTypeForSerialization	AbstractImplementationDataType	*	ref	This reference identifies the AbstractImplementationData Types that shall be supported for storing in a key-value storage in addition to the types already determined from tha aggregation of PersistencyDataElement.
dataTypeMapping	PersistencyKeyValueDataTypeMapping	0..1	aggr	This aggregation provides a collection of replacement rules for data types used in the context of the enclosing PersistencyKeyValueStorageInterface.

Table A.34: PersistencyKeyValueStorageInterface

Class	***PersistencyPortPrototypeToDeploymentMapping*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This abstract bas class implements the shared functionality of all mapping between a `PortPrototype`, a `Process`, and a specific subclass of `PersistencyDeployment`.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusivePackageElement*, *UploadablePackageElement*			
Subclasses	PersistencyPortPrototypeToFileStorageMapping, PersistencyPortPrototypeToKeyValueStorageMapping			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
portPrototype	PortPrototype	0..1	iref	This reference represents the mapped PortPrototype. **InstanceRef implemented by:** PortPrototypeIn ExecutableInstanceRef
process	Process	0..1	ref	This reference represents the process required as context for the mapping.

Table A.35: PersistencyPortPrototypeToDeploymentMapping

Class	**PersistencyPortPrototypeToFileStorageMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to define a mapping between a collection of files on deployment level to a given `PortPrototype`. **Tags:** atp.recommendedPackage=PersistencyPortPrototypeToFileStorageMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PersistencyPortPrototypeToDeploymentMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusivePackageElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
fileStorage	PersistencyFileStorage	0..1	ref	This reference represents the mapped file storage.

Table A.36: PersistencyPortPrototypeToFileStorageMapping

Class	**PersistencyPortPrototypeToKeyValueStorageMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to define a mapping between a PortPrototype and a key-value storage. **Tags:** atp.recommendedPackage=PersistencyPortPrototypeToKeyValueStorageMappings			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *PersistencyPortPrototypeToDeploymentMapping*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusivePackageElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
keyValue Storage	PersistencyKeyValue Storage	0..1	ref	This reference represents the mapped key-value storage.

Table A.37: PersistencyPortPrototypeToKeyValueStorageMapping

Class	***PersistencyRedundancyChecksum*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	Abstract class that defines the common attributes for implementations of redundancy.			
Base	*ARObject*, *PersistencyRedundancyHandling*			
Subclasses	PersistencyRedundancyCrc, PersistencyRedundancyHash			
Aggregated by	*PersistencyDeployment*.redundancyHandling, *PersistencyInterface*.redundancyHandling			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
algorithmFamily	String	0..1	attr	This attribute identifies the algorithm family that is used to execute the CRC/Hash.
length	PositiveInteger	0..1	attr	This attribute describes the length of the CRC/Hash in the unit bits.

Table A.38: PersistencyRedundancyChecksum

Class	**PersistencyRedundancyCrc**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class formally describes the usage of a CRC for the implementation of redundancy.			
Base	*ARObject*, *PersistencyRedundancyChecksum*, *PersistencyRedundancyHandling*			
Aggregated by	*PersistencyDeployment*.redundancyHandling, *PersistencyInterface*.redundancyHandling			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table A.39: 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.
Aggregated by	*PersistencyInterface*.redundancy
Literal	***Description***
none	This value represents the requirement that redundancy measures are not applied on persistency storage level. **Tags:** atp.EnumerationLiteralIndex=1
redundant	This value represents the requirement that redundancy measures are applied on persistency storage level. The nature of the redundant persistent storage is not further qualified and subject to integrator decisions. **Tags:** atp.EnumerationLiteralIndex=0
redundantPer Element	This value represents the requirement that redundancy measures are applied on key-value level of a key-value storage or on file level of a file storage. The nature of the redundancy used on the persistent storage is not further qualified and subject to integrator decisions. **Tags:** atp.EnumerationLiteralIndex=2

Table A.40: PersistencyRedundancyEnum

Class	***PersistencyRedundancyHandling*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This abstract base class represents a formal description of redundancy.			
Base	*ARObject*			
Subclasses	*PersistencyRedundancyChecksum*, PersistencyRedundancyMOutOfN			
Aggregated by	*PersistencyDeployment*.redundancyHandling, *PersistencyInterface*.redundancyHandling			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
scope	PersistencyRedundancy HandlingScopeEnum	0..1	attr	This attribute controls the scope in which the redundancy handling is applied.

Table A.41: PersistencyRedundancyHandling

Enumeration	**PersistencyRedundancyHandlingScopeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency
Note	This meta-class provides values to control the scope of redundancy measures in the persistency target-configuration.
Aggregated by	*PersistencyRedundancyHandling*.scope
Literal	***Description***
persistency RedundancyHandlingScope Element	The redundancy handling shall be applied on element level (key-value pair and file). **Tags:** atp.EnumerationLiteralIndex=0
persistency RedundancyHandlingScope Storage	The redundancy handling shall be applied on storage (key-value storage and file storage) level. **Tags:** atp.EnumerationLiteralIndex=1

Table A.42: PersistencyRedundancyHandlingScopeEnum

Class	**PersistencyRedundancyHash**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class formally describes the usage of a Hash for the implementation of redundancy.			
Base	*ARObject*, *PersistencyRedundancyChecksum*, *PersistencyRedundancyHandling*			
Aggregated by	*PersistencyDeployment*.redundancyHandling, *PersistencyInterface*.redundancyHandling			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
initialization VectorLength	PositiveInteger	0..1	attr	Length of the initialization vector.

Table A.43: PersistencyRedundancyHash

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.			
Base	*ARObject*, *PersistencyRedundancyHandling*			
Aggregated by	*PersistencyDeployment*.redundancyHandling, *PersistencyInterface*.redundancyHandling			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
m	PositiveInteger	0..1	attr	This attribute represents the "M" coordinate in the "M out of N" scheme.
n	PositiveInteger	0..1	attr	This attribute represents the "N" coordinate in the "M out of N" scheme.

Table A.44: PersistencyRedundancyMOutOfN

Class	***PortInterface*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Abstract base class for an interface that is either provided or required by a port of a software component.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractRawDataStreamInterface*, *AbstractSynchronizedTimeBaseInterface*, ClientServerInterface, *CryptoInterface*, *DataInterface*, *DiagnosticPortInterface*, FirewallStateSwitchInterface, *IdsmAbstractPortInterface*, LogAndTraceInterface, ModeSwitchInterface, NetworkManagementPortInterface, *PersistencyInterface*, *PlatformHealthManagementInterface*, ServiceInterface, *StateManagementPortInterface*, TriggerInterface			
Aggregated by	ARPackage.element			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
namespace (ordered)	SymbolProps	*	aggr	This represents the SymbolProps used for the definition of a hierarchical namespace applicable for the generation of code artifacts out of the definition of a ServiceInterface. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=namespace.shortName

Table A.45: PortInterface

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*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**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.

▽

△

Class	***PortPrototype*** (abstract)			
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.
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table A.46: PortPrototype

Class	**Process**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ExecutionManifest			
Note	This meta-class provides information required to execute the referenced `Executable`. **Tags:** atp.recommendedPackage=Processes			
Base	*ARElement*, *ARObject*, *AbstractExecutionContext*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *Uploadable PackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ProcessDesign	0..1	ref	This reference represents the identification of the design-time representation for the Process that owns the reference.
executable	Executable	*	ref	Reference to executable that is executed in the process. **Stereotypes:** atpUriDef
functionCluster Affiliation	String	0..1	attr	This attribute specifies which functional cluster the Process is affiliated with.
numberOf RestartAttempts	PositiveInteger	0..1	attr	This attribute defines how often a process shall be restarted if the start fails. numberOfRestartAttempts = "0" OR Attribute not existing, start once numberOfRestartAttempts = "1", start a second time
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. This attribute is used to support the modeling of execution dependencies that utilize the condition of process state. Please note that the process states may not be modeled arbitrarily at any stage of the AUTOSAR workflow because the supported states are standardized in the context of the SWS Execution Management [4].
stateDependent StartupConfig	StateDependentStartup Config	*	aggr	Applicable startup configurations.

Table A.47: Process

Class	**ProcessToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SubSystemDesign::MachineManifest			
Note	This meta-class has the ability to associate a Process with a Machine. This relation involves the definition of further properties, e.g. timeouts.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ProcessToMachineMappingSet.processToMachineMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
design	ProcessDesignTo MachineDesignMapping	0..1	ref	This reference represents the identification of the design-time representation for the ProcessToMachine Mapping that owns the reference.
machine	Machine	0..1	ref	This reference identifies the Machine in the context of the ProcessToMachineMapping.
nonOsModule Instantiation	NonOsModule Instantiation	0..1	ref	This supports the optional case that the process represents a platform module.
persistency CentralStorage URI	UriString	0..1	attr	This attribute identifies a central place for the mapped Process to store the list of available storages and version information.
process	Process	0..1	ref	This reference identifies the Process in the context of the ProcessToMachineMapping.
shallNotRunOn	ProcessorCore	*	ref	This reference indicates a collection of cores onto which the mapped process shall not be executing.
shallRunOn	ProcessorCore	*	ref	This reference indicates a collection of cores onto which the mapped process shall be executing.

Table A.48: ProcessToMachineMapping

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*			
Aggregated by	*AtpClassifier*.atpFeature, *SwComponentType*.port			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
required Interface	PortInterface	0..1	tref	The interface that this port requires. **Stereotypes:** isOfType

Table A.49: 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, Bsw VariableAccess, CouplingPortTrafficClassAssignment, *CppImplementationDataTypeContextTarget*, *DiagnosticEnvModeElement*, EthernetPriorityRegeneration, ExclusiveAreaNestingOrder, *HwDescription Entity*, *ImplementationProps*, ModeTransition, *MultilanguageReferrable*, NmNetworkHandle, Pnc MappingIdent, *SingleLanguageReferrable*, SoConIPduIdentifier, SocketConnectionBundle, Someip RequiredEventGroup, TimeSyncServerConfiguration, TpConnectionIdent			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

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

Table A.50: Referrable

Class	**ServiceInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This represents the ability to define a PortInterface that consists of a heterogeneous collection of methods, events and fields. **Tags:** atp.recommendedPackage=ServiceInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
event	VariableDataPrototype	*	aggr	This represents the collection of events defined in the context of a ServiceInterface. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=event.shortName, event.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=30
field	Field	*	aggr	This represents the collection of fields defined in the context of a ServiceInterface. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=field.shortName, field.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=40
majorVersion	PositiveInteger	0..1	attr	Major version of the service contract. **Tags:** xml.sequenceOffset=10
method	ClientServerOperation	*	aggr	This represents the collection of methods defined in the context of a ServiceInterface. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=method.shortName, method.variationPoint.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=50
minorVersion	PositiveInteger	0..1	attr	Minor version of the service contract. **Tags:** xml.sequenceOffset=20

▽

△

Class	**ServiceInterface**			
trigger	Trigger	*	aggr	This represents the collection of triggers defined in the context of a ServiceInterface. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=trigger.shortName, trigger.variation Point.shortLabel vh.latestBindingTime=blueprintDerivationTime xml.sequenceOffset=60

Table A.51: ServiceInterface

Class	**SoftwarePackage**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to formalize the content of a software package. **Tags:** atp.recommendedPackage=SoftwarePackages			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
actionType	SoftwarePackageAction TypeEnum	0..1	attr	This attribute defines the action to be taken in the step of processing the enclosing SoftwarePackage.
activationAction	SoftwarePackage ActivationActionEnum	0..1	attr	This attribute governs the action to be taken after the installation of the SoftwareCluster completed.
artifactLocator	ArtifactLocator	0..1	aggr	This attribute identifies the software package at configuration time, out of the context of an AUTOSAR model.
compressed Software PackageSize	PositiveInteger	0..1	attr	This size represents the size of the compressed Software Package.
deltaPackage Applicable Version	StrongRevisionLabel String	0..1	attr	This attribute identifies the version of the included SoftwareCluster for which the enclosing SoftwarePackage can be used as a delta update
estimated DurationOf Operation	TimeValue	0..1	attr	This attribute provides an estimation about how long the operation of the SoftwarePackage is going to take for its transfer, processing and activation when updated standalone (not within an update campaign)
minimum SupportedUcm Version	RevisionLabelString	0..1	attr	This attribute identifies the minimum supported version of the UCM for this SoftwarePackage.
packagerId	PositiveInteger	0..1	attr	This attribute identifies Id of the organization that provides the packager generating the SoftwarePackage.
packager Signature	CryptoService Certificate	0..1	ref	This reference identifies the certificate that represents the packager's signature.
purposeOf Update	Documentation	0..1	ref	The referenced Documentation is supposed to provide a description of the purpose of the update.
softwareCluster	SoftwareCluster	0..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.
uncompressed SoftwareCluster Size	PositiveInteger	0..1	attr	This attribute gives an indication about the storage that has to be available on the target.

Table A.52: SoftwarePackage

Enumeration	**SoftwarePackageActionTypeEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This enumeration provides a choice of possible actions for the handling of a software package.
Aggregated by	SoftwarePackage.actionType
Literal	***Description***
install	Do a clean installation of a SoftwareCluster. **Tags:** atp.EnumerationLiteralIndex=1
remove	Remove a SoftwareCluster. **Tags:** atp.EnumerationLiteralIndex=2
update	Update a previously installed SoftwareCluster. **Tags:** atp.EnumerationLiteralIndex=0
update Configuration	Execute a configuration update. **Tags:** atp.EnumerationLiteralIndex=3

Table A.53: SoftwarePackageActionTypeEnum

Primitive	**StrongRevisionLabelString**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This primitive represents a revision label which identifies an object under version control. It represents a pattern which requires three integer numbers separated by a dot, representing from left to right Major Version, MinorVersion, PatchVersion and additional labels for pre-release version and build metadata. Legal patterns are for example: 1.0.0-alpha+001 1.0.0+20130313144700 1.0.0-beta+exp.sha.5114f85 **Tags:** xml.xsd.customType=STRONG-REVISION-LABEL-STRING xml.xsd.pattern=(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(-((0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*)(\.(0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(\+([0-9a-zA-Z-]+(\.[0-9a-zA-Z-]+)*))? xml.xsd.type=string

Table A.54: StrongRevisionLabelString

Class	«atpVariation» **SwDataDefProps**
Package	M2::MSR::DataDictionary::DataDefProperties
Note	This class is a collection of properties relevant for data objects under various aspects. One could consider this class as a "pattern of inheritance by aggregation". The properties can be applied to all objects of all classes in which SwDataDefProps is aggregated. Note that not all of the attributes or associated elements are useful all of the time. Hence, the process definition (e.g. expressed with an OCL or a Document Control Instance MSR-DCI) has the task of implementing limitations. SwDataDefProps covers various aspects: • Structure of the data element for calibration use cases: is it a single value, a curve, or a map, but also the recordLayouts which specify how such elements are mapped/converted to the DataTypes in the programming language (or in AUTOSAR). This is mainly expressed by properties like swRecordLayout and swCalprmAxisSet • Implementation aspects, mainly expressed by swImplPolicy, swVariableAccessImplPolicy, swAddr Method, swPointerTagetProps, baseType, implementationDataType and additionalNativeTypeQualifier • Access policy for the MCD system, mainly expressed by swCalibrationAccess • Semantics of the data element, mainly expressed by compuMethod and/or unit, dataConstr, invalid Value • Code generation policy provided by swRecordLayout **Tags:** vh.latestBindingTime=codeGenerationTime

▽

△

Class	«atpVariation» **SwDataDefProps**			
Base	*ARObject*			
Aggregated by	*AutosarDataType*.swDataDefProps, CompositeNetworkRepresentation.networkRepresentation, CppImplementationDataTypeElement.swDataDefProps, *DataPrototype*.swDataDefProps, DataPrototypeTransformationProps.networkRepresentationProps, DiagnosticDataElement.swDataDefProps, DiagnosticEnvDataElementCondition.swDataDefProps, DltArgument.networkRepresentation, FlatInstanceDescriptor.swDataDefProps, ImplementationDataTypeElement.swDataDefProps, InstantiationDataDefProps.swDataDefProps, ISignal.networkRepresentationProps, McDataInstance.resultingProperties, ParameterAccess.swDataDefProps, PerInstanceMemory.swDataDefProps, *ReceiverComSpec*.networkRepresentation, SecurityEventContextDataElement.networkRepresentation, *SenderComSpec*.networkRepresentation, SomeipDataPrototypeTransformationProps.networkRepresentation, SwPointerTargetProps.swDataDefProps, SwServiceArg.swDataDefProps, SwSystemconst.swDataDefProps, SystemSignal.physicalProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
additionalNativeTypeQualifier	NativeDeclarationString	0..1	attr	This attribute is used to declare native qualifiers of the programming language which can neither be deduced from the baseType (e.g. because the data object describes a pointer) nor from other more abstract attributes. Examples are qualifiers like "volatile", "strict" or "enum" of the C-language. All such declarations have to be put into one string. **Tags:** xml.sequenceOffset=235
annotation	Annotation	*	aggr	This aggregation allows to add annotations (yellow pads ...) related to the current data object. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false
baseType	SwBaseType	0..1	ref	Base type associated with the containing data object. **Tags:** xml.sequenceOffset=50
compuMethod	CompuMethod	0..1	ref	Computation method associated with the semantics of this data object. **Tags:** xml.sequenceOffset=180
dataConstr	DataConstr	0..1	ref	Data constraint for this data object. **Tags:** xml.sequenceOffset=190
displayFormat	DisplayFormatString	0..1	attr	This property describes how a number is to be rendered e.g. in documents or in a measurement and calibration system. **Tags:** xml.sequenceOffset=210
displayPresentation	DisplayPresentationEnum	0..1	attr	This attribute controls the presentation of the related data for measurement and calibration tools.

▽

△

Class	«atpVariation» **SwDataDefProps**			
implementation DataType	AbstractImplementation DataType	0..1	ref	This association denotes the ImplementationDataType of a data declaration via its aggregated SwDataDefProps. It is used whenever a data declaration is not directly referring to a base type. Especially • redefinition of an ImplementationDataType via a "typedef" to another ImplementationDatatype • the target type of a pointer (see SwPointerTarget Props), if it does not refer to a base type directly • the data type of an array or record element within an ImplementationDataType, if it does not refer to a base type directly • the data type of an SwServiceArg, if it does not refer to a base type directly **Tags:** xml.sequenceOffset=215
invalidValue	ValueSpecification	0..1	aggr	Optional value to express invalidity of the actual data element. **Tags:** xml.sequenceOffset=255
stepSize	Float	0..1	attr	This attribute can be used to define a value which is added to or subtracted from the value of a DataPrototype when using up/down keys while calibrating.
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this data object. Via an association to the same SwAddrMethod it can be specified that several DataPrototypes shall be located in the same memory without already specifying the memory section itself. **Tags:** xml.sequenceOffset=30
swAlignment	AlignmentType	0..1	attr	The attribute describes the intended typical alignment of the DataPrototype. If the attribute is not defined the alignment is determined by the swBaseType size and the memoryAllocationKeywordPolicy of the referenced Sw AddrMethod. **Tags:** xml.sequenceOffset=33
swBit Representation	SwBitRepresentation	0..1	aggr	Description of the binary representation in case of a bit variable. **Tags:** xml.sequenceOffset=60
swCalibration Access	SwCalibrationAccess Enum	0..1	attr	Specifies the read or write access by MCD tools for this data object. **Tags:** xml.sequenceOffset=70
swCalprmAxis Set	SwCalprmAxisSet	0..1	aggr	This specifies the properties of the axes in case of a curve or map etc. This is mainly applicable to calibration parameters. **Tags:** xml.sequenceOffset=90
swComparison Variable	SwVariableRefProxy	*	aggr	Variables used for comparison in an MCD process. **Tags:** xml.sequenceOffset=170 xml.typeElement=false
swData Dependency	SwDataDependency	0..1	aggr	Describes how the value of the data object has to be calculated from the value of another data object (by the MCD system). **Tags:** xml.sequenceOffset=200

▽

△

Class	«atpVariation» **SwDataDefProps**			
swHostVariable	SwVariableRefProxy	0..1	aggr	Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects. **Tags:** xml.sequenceOffset=220 xml.typeElement=false
swImplPolicy	SwImplPolicyEnum	0..1	attr	Implementation policy for this data object. **Tags:** xml.sequenceOffset=230
swIntended Resolution	Numerical	0..1	attr	The purpose of this element is to describe the requested quantization of data objects early on in the design process. The resolution ultimately occurs via the conversion formula present (compuMethod), which specifies the transition from the physical world to the standardized world (and vice-versa) (here, "the slope per bit" is present implicitly in the conversion formula). In the case of a development phase without a fixed conversion formula, a pre-specification can occur through swIntendedResolution. The resolution is specified in the physical domain according to the property "unit". **Tags:** xml.sequenceOffset=240
swInterpolation Method	Identifier	0..1	attr	This is a keyword identifying the mathematical method to be applied for interpolation. The keyword needs to be related to the interpolation routine which needs to be invoked. **Tags:** xml.sequenceOffset=250
swIsVirtual	Boolean	0..1	attr	This element distinguishes virtual objects. Virtual objects do not appear in the memory, their derivation is much more dependent on other objects and hence they shall have a swDataDependency . **Tags:** xml.sequenceOffset=260
swPointerTarget Props	SwPointerTargetProps	0..1	aggr	Specifies that the containing data object is a pointer to another data object. **Tags:** xml.sequenceOffset=280
swRecord Layout	SwRecordLayout	0..1	ref	Record layout for this data object. **Tags:** xml.sequenceOffset=290
swRefresh Timing	MultidimensionalTime	0..1	aggr	This element specifies the frequency in which the object involved shall be or is called or calculated. This timing can be collected from the task in which write access processes to the variable run. But this cannot be done by the MCD system. So this attribute can be used in an early phase to express the desired refresh timing and later on to specify the real refresh timing. **Tags:** xml.sequenceOffset=300
swTextProps	SwTextProps	0..1	aggr	the specific properties if the data object is a text object. **Tags:** xml.sequenceOffset=120

▽

△

Class	«atpVariation» **SwDataDefProps**			
swValueBlock Size	Numerical	0..1	attr	This represents the size of a Value Block **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=80
swValueBlock SizeMult (ordered)	Numerical	*	attr	This attribute is used to specify the dimensions of a value block (VAL_BLK) for the case that that value block has more than one dimension. The dimensions given in this attribute are ordered such that the first entry represents the first dimension, the second entry represents the second dimension, and so on. For one-dimensional value blocks the attribute swValue BlockSize shall be used and this attribute shall not exist. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime
unit	Unit	0..1	ref	Physical unit associated with the semantics of this data object. This attribute applies if no compuMethod is specified. If both units (this as well as via compuMethod) are specified the units shall be compatible. **Tags:** xml.sequenceOffset=350
valueAxisData Type	ApplicationPrimitive DataType	0..1	ref	The referenced ApplicationPrimitiveDataType represents the primitive data type of the value axis within a compound primitive (e.g. curve, map). It supersedes CompuMethod, Unit, and BaseType. **Tags:** xml.sequenceOffset=355

Table A.55: SwDataDefProps

Class	**SwTextProps**			
Package	M2::MSR::DataDictionary::DataDefProperties			
Note	This meta-class expresses particular properties applicable to strings in variables or calibration parameters.			
Base	*ARObject*			
Aggregated by	SwDataDefProps.swTextProps			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls the semantics of the arraysize for the array representing the string in an Implementation DataType. It is there to support a safe conversion between ApplicationDatatype and ImplementationDatatype, even for variable length strings as required e.g. for Support of SAE J1939.
baseType	SwBaseType	0..1	ref	This is the base type of one character in the string. In particular this baseType denotes the intended encoding of the characters in the string on level of ApplicationData Type. **Tags:** xml.sequenceOffset=30

▽

△

Class	**SwTextProps**			
swFillCharacter	Integer	0..1	attr	Filler character for text parameter to pad up to the maximum length swMaxTextSize. The value will be interpreted according to the encoding specified in the associated base type of the data object, e.g. 0x30 (hex) represents the ASCII character zero as filler character and 0 (dec) represents an end of string as filler character. The usage of the fill character depends on the arraySize Semantics. **Tags:** xml.sequenceOffset=40
swMaxTextSize	Integer	0..1	attr	Specifies the maximum text size in characters. Note the size in bytes depends on the encoding in the corresponding baseType. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=20

Table A.56: SwTextProps

B Demands and Constraints on Base Software (normative)

This chapter lists the demands or constraints of this functional cluster for the Base Software on which the AUTOSAR Adaptive Platform is running on (usually a POSIX-compatible operating system).

In many cases, `Persistency` uses advanced hardware and software facilities like caching and buffering, that may lead to data loss in case the ECU is switched off immediately after data was written. Using `redundancy`, data consistency can still be established, but to minimize the time in which a change of stored data can be lost, an integrator has to configure the system appropriately. A final guidance on how to configure the system has to be provided by a software supplier together with an implementation of `Persistency`, because it is highly dependent on the actual implementation. Still, this chapter gives some guidance on what is probably needed based on the implementation variants described in Section 4.2.

B.1 Based on a File System

In this scenario, `Persistency` will probably use the POSIX `fsync()` function. Unfortunately, `fsync()` just ensures that the POSIX subsystem flushes all buffers, but does not in all cases ensure that the file system also clears its caches, and in addition not all file systems are able to ensure that hardware device caches are flushed as well. Therefore, in some cases the actual transmission of the changed information to the hardware will happen asynchronously, i.e. after the methods `ara::per::KeyValueStorage::SyncToStorage` or `ara::per::ReadWriteAccessor::SyncToFile` return to the application.

[SWS_PER_00575] Synchronous Sync ⌈**Type:** Demand

Description: To achieve synchronous behavior of `ara::per::KeyValueStorage::SyncToStorage` or `ara::per::ReadWriteAccessor::SyncToFile`, an integrator of an adaptive application that uses `Persistency` has to ensure that a suitable file system is used and configured in a way that either all changes result in immediate changes to the hardware storage, or that `fsync()` has completely synchronous behavior.

Rationale: `Persistency` implementations that use a file system will usually rely on `fsync()` for ensuring synchronous behavior of `ara::per::KeyValueStorage::SyncToStorage` or `ara::per::ReadWriteAccessor::SyncToFile`.

Supporting Material: Documentation of POSIX `fsync()`.⌋

B.2 Direct Access to Storage Hardware

In this scenario, `Persistency` has very tight control of the operations of the hardware, and can take care of synchronous behavior of the methods `ara::per::KeyValueStorage::SyncToStorage` or `ara::per::ReadWriteAccessor::SyncToFile`. Any additional necessities for the integration in the system are expected to be provided with the documentation of the specific implementation.

C Platform Extension Interfaces (normative)

This functional cluster does not specify any Platform Extension Interfaces.

D Not Implemented Requirements

This chapter lists all functional requirements specified in the corresponding requirement specifications that are not implemented or violated by this specification and provides a rationale.

[SWS_PER_NA_00001] Requirements Not Applicable to this Specification

Upstream requirements: RS_AP_00111, RS_AP_00114, RS_AP_00116, RS_AP_00124, RS_AP_00130, RS_AP_00138, RS_AP_00142, RS_AP_00148, RS_AP_00150, RS_AP_00151, RS_AP_00154, RS_AP_00155, RS_AP_00156

⌈These requirements are not applicable to this specification.⌋

[SWS_PER_NA_00002] Invalid Specifier Requirements Not Applicable to this Specification

Upstream requirements: RS_AP_00170, RS_AP_00171, RS_AP_00172

⌈Having Persistency return an error allows for flexible handling of not configured storages, and reduces the dependency of the code on a specific configuration.⌋

[SWS_PER_NA_00003] Access Sharing Requirements Not Applicable to this Specification

Upstream requirements: RS_AP_00173

⌈Persistency allows opening the same storage twice, the access is handled through a SharedHandle.⌋

E Change History of AUTOSAR Traceable Items

This chapter provides an overview of the history of constraints and specification items. Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These constraints and specification items do not appear as hyperlinks in the document.

E.1 Traceable Item History of this Document According to AUTOSAR Release R24-11

E.1.1 Added Specification Items in R24-11

[SWS_PER_00574] [SWS_PER_00575] [SWS_PER_00576] [SWS_PER_00577]
[SWS_PER_00578] [SWS_PER_00579] [SWS_PER_00580] [SWS_PER_00581]
[SWS_PER_00582] [SWS_PER_10001] [SWS_PER_10002] [SWS_PER_10003]
[SWS_PER_20001] [SWS_PER_20002] [SWS_PER_20003] [SWS_PER_20004]
[SWS_PER_20005] [SWS_PER_20006] [SWS_PER_20007] [SWS_PER_20008]
[SWS_PER_20009] [SWS_PER_20010] [SWS_PER_20011] [SWS_PER_20012]
[SWS_PER_20013] [SWS_PER_20014] [SWS_PER_20015] [SWS_PER_20016]
[SWS_PER_20017] [SWS_PER_20018] [SWS_PER_20019] [SWS_PER_20020]
[SWS_PER_20021] [SWS_PER_20022] [SWS_PER_20023] [SWS_PER_20024]
[SWS_PER_20025] [SWS_PER_20026] [SWS_PER_20027] [SWS_PER_20031]
[SWS_PER_20032] [SWS_PER_20033] [SWS_PER_20034] [SWS_PER_20035]
[SWS_PER_20036] [SWS_PER_20037] [SWS_PER_20038] [SWS_PER_20039]
[SWS_PER_20040] [SWS_PER_20041] [SWS_PER_20042] [SWS_PER_20043]
[SWS_PER_20044] [SWS_PER_20045] [SWS_PER_20046] [SWS_PER_20047]
[SWS_PER_20048] [SWS_PER_20049] [SWS_PER_20050] [SWS_PER_20051]
[SWS_PER_20052] [SWS_PER_20053] [SWS_PER_20054] [SWS_PER_20055]
[SWS_PER_20056] [SWS_PER_20057] [SWS_PER_20058] [SWS_PER_20059]
[SWS_PER_20060] [SWS_PER_20061] [SWS_PER_20062] [SWS_PER_20063]
[SWS_PER_20064] [SWS_PER_20065] [SWS_PER_20066] [SWS_PER_20067]
[SWS_PER_20068] [SWS_PER_20069] [SWS_PER_20070] [SWS_PER_20071]
[SWS_PER_20072]

E.1.2 Changed Specification Items in R24-11

[SWS_PER_00050] [SWS_PER_00052] [SWS_PER_00116] [SWS_PER_00119]
[SWS_PER_00165] [SWS_PER_00167] [SWS_PER_00168] [SWS_PER_00311]
[SWS_PER_00313] [SWS_PER_00315] [SWS_PER_00322] [SWS_PER_00323]
[SWS_PER_00326] [SWS_PER_00327] [SWS_PER_00330] [SWS_PER_00343]
[SWS_PER_00350] [SWS_PER_00355] [SWS_PER_00367] [SWS_PER_00368]
[SWS_PER_00369] [SWS_PER_00370] [SWS_PER_00371] [SWS_PER_00372]
[SWS_PER_00383] [SWS_PER_00385] [SWS_PER_00388] [SWS_PER_00389]

[SWS_PER_00392] [SWS_PER_00393] [SWS_PER_00413] [SWS_PER_00414] [SWS_PER_00417] [SWS_PER_00418] [SWS_PER_00419] [SWS_PER_00420] [SWS_PER_00421] [SWS_PER_00422] [SWS_PER_00537] [SWS_PER_00541] [SWS_PER_00568] [SWS_PER_00569]

E.1.3 Deleted Specification Items in R24-11

[SWS_PER_00410]

E.1.4 Added Constraints in R24-11

[SWS_PER_CONSTR_00007]

E.1.5 Changed Constraints in R24-11

none

E.1.6 Deleted Constraints in R24-11

none

E.2 Traceable Item History of this Document According to AUTOSAR Release R23-11

E.2.1 Added Specification Items in R23-11

[SWS_PER_00567] [SWS_PER_00568] [SWS_PER_00569] [SWS_PER_00570] [SWS_PER_00571] [SWS_PER_00572] [SWS_PER_00573]

E.2.2 Changed Specification Items in R23-11

[SWS_PER_00049] [SWS_PER_00050] [SWS_PER_00052] [SWS_PER_00116] [SWS_PER_00122] [SWS_PER_00146] [SWS_PER_00147] [SWS_PER_00311] [SWS_PER_00312] [SWS_PER_00323] [SWS_PER_00327] [SWS_PER_00330] [SWS_PER_00333] [SWS_PER_00334] [SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00339] [SWS_PER_00340] [SWS_PER_00342] [SWS_PER_00343] [SWS_PER_00354] [SWS_PER_00356] [SWS_PER_00357] [SWS_PER_00358] [SWS_PER_00359] [SWS_PER_00362] [SWS_PER_00378] [SWS_PER_00386] [SWS_PER_00387] [SWS_PER_00396] [SWS_PER_00407] [SWS_PER_00409]

[SWS_PER_00411] [SWS_PER_00412] [SWS_PER_00414] [SWS_PER_00417] [SWS_PER_00432] [SWS_PER_00435] [SWS_PER_00436] [SWS_PER_00437] [SWS_PER_00438] [SWS_PER_00440] [SWS_PER_00441] [SWS_PER_00442] [SWS_PER_00443] [SWS_PER_00444] [SWS_PER_00445] [SWS_PER_00446] [SWS_PER_00447] [SWS_PER_00448] [SWS_PER_00463] [SWS_PER_00477] [SWS_PER_00479] [SWS_PER_00518] [SWS_PER_00533]

E.2.3 Deleted Specification Items in R23-11

[SWS_PER_00320]

E.2.4 Added Constraints in R23-11

none

E.2.5 Changed Constraints in R23-11

[SWS_PER_CONSTR_00001] [SWS_PER_CONSTR_00002] [SWS_PER_CONSTR_00005]

E.2.6 Deleted Constraints in R23-11

none

E.3 Traceable Item History of this Document According to AUTOSAR Release R22-11

E.3.1 Added Specification Items in R22-11

[SWS_PER_00044] [SWS_PER_00536] [SWS_PER_00537] [SWS_PER_00538] [SWS_PER_00539] [SWS_PER_00540] [SWS_PER_00541] [SWS_PER_00542] [SWS_PER_00543] [SWS_PER_00544] [SWS_PER_00545] [SWS_PER_00546] [SWS_PER_00547] [SWS_PER_00548] [SWS_PER_00549] [SWS_PER_00550] [SWS_PER_00551] [SWS_PER_00552] [SWS_PER_00553] [SWS_PER_00554] [SWS_PER_00555] [SWS_PER_00556] [SWS_PER_00557] [SWS_PER_00558] [SWS_PER_00559] [SWS_PER_00560] [SWS_PER_00561] [SWS_PER_00562] [SWS_PER_00563] [SWS_PER_00564] [SWS_PER_00565] [SWS_PER_00566] [SWS_PER_CONSTR_00005] [SWS_PER_CONSTR_00006]

E.3.2 Changed Specification Items in R22-11

[SWS_PER_00042] [SWS_PER_00043] [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_00165] [SWS_PER_00166] [SWS_PER_00167] [SWS_PER_00168]
[SWS_PER_00210] [SWS_PER_00211] [SWS_PER_00303] [SWS_PER_00311]
[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_00370] [SWS_PER_00375]
[SWS_PER_00376] [SWS_PER_00377] [SWS_PER_00390] [SWS_PER_00394]
[SWS_PER_00405] [SWS_PER_00406] [SWS_PER_00407] [SWS_PER_00418]
[SWS_PER_00419] [SWS_PER_00420] [SWS_PER_00421] [SWS_PER_00422]
[SWS_PER_00423] [SWS_PER_00426] [SWS_PER_00427] [SWS_PER_00428]
[SWS_PER_00429] [SWS_PER_00430] [SWS_PER_00431] [SWS_PER_00438]
[SWS_PER_00440] [SWS_PER_00449] [SWS_PER_00450] [SWS_PER_00451]
[SWS_PER_00453] [SWS_PER_00455] [SWS_PER_00464] [SWS_PER_00465]
[SWS_PER_00466] [SWS_PER_00467] [SWS_PER_00468] [SWS_PER_00491]
[SWS_PER_00492] [SWS_PER_00493] [SWS_PER_00512] [SWS_PER_00513]
[SWS_PER_00529] [SWS_PER_00530] [SWS_PER_00531] [SWS_PER_NA]

E.3.3 Deleted Specification Items in R22-11

none

E.4 Traceable Item History of this Document According to AUTOSAR Release R21-11

E.4.1 Added Specification Items in R21-11

[SWS_PER_00452] [SWS_PER_00453] [SWS_PER_00454] [SWS_PER_00455]
[SWS_PER_00456] [SWS_PER_00457] [SWS_PER_00458] [SWS_PER_00459]
[SWS_PER_00460] [SWS_PER_00461] [SWS_PER_00462] [SWS_PER_00463]
[SWS_PER_00464] [SWS_PER_00465] [SWS_PER_00466] [SWS_PER_00467]
[SWS_PER_00468] [SWS_PER_00469] [SWS_PER_00470] [SWS_PER_00471]
[SWS_PER_00472] [SWS_PER_00473] [SWS_PER_00474] [SWS_PER_00475]
[SWS_PER_00476] [SWS_PER_00477] [SWS_PER_00478] [SWS_PER_00479]
[SWS_PER_00480] [SWS_PER_00481] [SWS_PER_00482] [SWS_PER_00483]
[SWS_PER_00484] [SWS_PER_00485] [SWS_PER_00486] [SWS_PER_00487]
[SWS_PER_00488] [SWS_PER_00489] [SWS_PER_00490] [SWS_PER_00491]
[SWS_PER_00492] [SWS_PER_00493] [SWS_PER_00494] [SWS_PER_00495]
[SWS_PER_00496] [SWS_PER_00497] [SWS_PER_00498] [SWS_PER_00499]

[SWS_PER_00501] [SWS_PER_00502] [SWS_PER_00503] [SWS_PER_00504] [SWS_PER_00505] [SWS_PER_00506] [SWS_PER_00507] [SWS_PER_00508] [SWS_PER_00509] [SWS_PER_00510] [SWS_PER_00511] [SWS_PER_00512] [SWS_PER_00513] [SWS_PER_00514] [SWS_PER_00515] [SWS_PER_00516] [SWS_PER_00517] [SWS_PER_00518] [SWS_PER_00519] [SWS_PER_00520] [SWS_PER_00521] [SWS_PER_00522] [SWS_PER_00523] [SWS_PER_00524] [SWS_PER_00525] [SWS_PER_00526] [SWS_PER_00527] [SWS_PER_00528] [SWS_PER_00529] [SWS_PER_00530] [SWS_PER_00531] [SWS_PER_00532] [SWS_PER_00533] [SWS_PER_00534] [SWS_PER_00535] [SWS_PER_CONSTR_00001] [SWS_PER_CONSTR_00002]

E.4.2 Changed Specification Items in R21-11

[SWS_PER_00042] [SWS_PER_00043] [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_00163] [SWS_PER_00164] [SWS_PER_00165] [SWS_PER_00166] [SWS_PER_00167] [SWS_PER_00168] [SWS_PER_00210] [SWS_PER_00211] [SWS_PER_00221] [SWS_PER_00251] [SWS_PER_00252] [SWS_PER_00265] [SWS_PER_00275] [SWS_PER_00277] [SWS_PER_00281] [SWS_PER_00283] [SWS_PER_00311] [SWS_PER_00317] [SWS_PER_00318] [SWS_PER_00319] [SWS_PER_00320] [SWS_PER_00321] [SWS_PER_00322] [SWS_PER_00323] [SWS_PER_00326] [SWS_PER_00327] [SWS_PER_00331] [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] [SWS_PER_00378] [SWS_PER_00379] [SWS_PER_00380] [SWS_PER_00382] [SWS_PER_00383] [SWS_PER_00385] [SWS_PER_00386] [SWS_PER_00387] [SWS_PER_00391] [SWS_PER_00395] [SWS_PER_00396] [SWS_PER_00405] [SWS_PER_00406] [SWS_PER_00407] [SWS_PER_00410] [SWS_PER_00413] [SWS_PER_00414] [SWS_PER_00418] [SWS_PER_00419] [SWS_PER_00420] [SWS_PER_00421] [SWS_PER_00422] [SWS_PER_00423] [SWS_PER_00426] [SWS_PER_00427] [SWS_PER_00428] [SWS_PER_00429] [SWS_PER_00430] [SWS_PER_00431] [SWS_PER_00432] [SWS_PER_00433] [SWS_PER_00438] [SWS_PER_00446] [SWS_PER_00447] [SWS_PER_00449] [SWS_PER_00450] [SWS_PER_00451]

E.4.3 Deleted Specification Items in R21-11

[SWS_PER_00222] [SWS_PER_00397]

E.5 Traceable Item History of this Document According to AUTOSAR Release R20-11

E.5.1 Added Specification Items in R20-11

[SWS_PER_00411] [SWS_PER_00412] [SWS_PER_00413] [SWS_PER_00414]
[SWS_PER_00415] [SWS_PER_00416] [SWS_PER_00417] [SWS_PER_00418]
[SWS_PER_00419] [SWS_PER_00420] [SWS_PER_00421] [SWS_PER_00422]
[SWS_PER_00423] [SWS_PER_00424] [SWS_PER_00425] [SWS_PER_00426]
[SWS_PER_00427] [SWS_PER_00428] [SWS_PER_00429] [SWS_PER_00430]
[SWS_PER_00431] [SWS_PER_00432] [SWS_PER_00433] [SWS_PER_00434]
[SWS_PER_00435] [SWS_PER_00436] [SWS_PER_00437] [SWS_PER_00438]
[SWS_PER_00439] [SWS_PER_00440] [SWS_PER_00441] [SWS_PER_00442]
[SWS_PER_00443] [SWS_PER_00444] [SWS_PER_00445] [SWS_PER_00446]
[SWS_PER_00447] [SWS_PER_00448] [SWS_PER_00449] [SWS_PER_00450]
[SWS_PER_00451]

E.5.2 Changed Specification Items in R20-11

[SWS_PER_00042] [SWS_PER_00043] [SWS_PER_00046] [SWS_PER_00047]
[SWS_PER_00048] [SWS_PER_00049] [SWS_PER_00052] [SWS_PER_00107]
[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_00144] [SWS_PER_00146]
[SWS_PER_00147] [SWS_PER_00162] [SWS_PER_00163] [SWS_PER_00164]
[SWS_PER_00165] [SWS_PER_00166] [SWS_PER_00167] [SWS_PER_00168]
[SWS_PER_00210] [SWS_PER_00211] [SWS_PER_00251] [SWS_PER_00252]
[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_00311] [SWS_PER_00312] [SWS_PER_00317] [SWS_PER_00318]
[SWS_PER_00319] [SWS_PER_00332] [SWS_PER_00333] [SWS_PER_00334]
[SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00337] [SWS_PER_00338]
[SWS_PER_00339] [SWS_PER_00340] [SWS_PER_00342] [SWS_PER_00343]
[SWS_PER_00356] [SWS_PER_00357] [SWS_PER_00358] [SWS_PER_00365]
[SWS_PER_00375] [SWS_PER_00376] [SWS_PER_00377] [SWS_PER_00378]
[SWS_PER_00379] [SWS_PER_00380] [SWS_PER_00383] [SWS_PER_00385]
[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_00396] [SWS_PER_00405] [SWS_PER_00406] [SWS_PER_00407]
[SWS_PER_00409] [SWS_PER_CONSTR_00004]

E.5.3 Deleted Specification Items in R20-11

[SWS_PER_00106] [SWS_PER_00108] [SWS_PER_00124] [SWS_PER_00126] [SWS_PER_00127] [SWS_PER_00128] [SWS_PER_00140] [SWS_PER_00141] [SWS_PER_00142] [SWS_PER_00143] [SWS_PER_00145] [SWS_PER_00180] [SWS_PER_00181] [SWS_PER_00182] [SWS_PER_00341] [SWS_PER_00344] [SWS_PER_00345] [SWS_PER_00346] [SWS_PER_00347] [SWS_PER_00348] [SWS_PER_00349] [SWS_PER_00366] [SWS_PER_00381] [SWS_PER_00404] [SWS_PER_CONSTR_00002]

E.6 Traceable Item History of this Document According to AUTOSAR Release R19-11

E.6.1 Added Specification Items in R19-11

[SWS_PER_00398] [SWS_PER_00399] [SWS_PER_00400] [SWS_PER_00401] [SWS_PER_00402] [SWS_PER_00403] [SWS_PER_00404] [SWS_PER_00405] [SWS_PER_00406] [SWS_PER_00407] [SWS_PER_00408] [SWS_PER_00409] [SWS_PER_00410]

E.6.2 Changed Specification Items in R19-11

[SWS_PER_00049] [SWS_PER_00113] [SWS_PER_00114] [SWS_PER_00115] [SWS_PER_00144] [SWS_PER_00145] [SWS_PER_00146] [SWS_PER_00147] [SWS_PER_00163] [SWS_PER_00164] [SWS_PER_00303] [SWS_PER_00317] [SWS_PER_00318] [SWS_PER_00319] [SWS_PER_00323] [SWS_PER_00327] [SWS_PER_00345] [SWS_PER_00351] [SWS_PER_00365] [SWS_PER_00368] [SWS_PER_00370] [SWS_PER_00372]

E.6.3 Deleted Specification Items in R19-11

[SWS_PER_00044] [SWS_PER_CONSTR_00001]

E.7 Traceable Item History of this Document According to AUTOSAR Release 19-03

E.7.1 Added Specification Items in 19-03

[SWS_PER_00349] [SWS_PER_00350] [SWS_PER_00351] [SWS_PER_00352] [SWS_PER_00353] [SWS_PER_00354] [SWS_PER_00355] [SWS_PER_00356]

[SWS_PER_00357] [SWS_PER_00358] [SWS_PER_00359] [SWS_PER_00360] [SWS_PER_00361] [SWS_PER_00362] [SWS_PER_00363] [SWS_PER_00364] [SWS_PER_00365] [SWS_PER_00366] [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] [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_00396] [SWS_PER_00397] [SWS_PER_CONSTR_00001] [SWS_PER_CONSTR_00002] [SWS_PER_CONSTR_00003] [SWS_PER_CONSTR_00004]

E.7.2 Changed Specification Items in 19-03

[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_00127] [SWS_PER_00128] [SWS_PER_00144] [SWS_PER_00145] [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_00311] [SWS_PER_00312] [SWS_PER_00313] [SWS_PER_00314] [SWS_PER_00315] [SWS_PER_00322] [SWS_PER_00323] [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_00340]

E.7.3 Deleted Specification Items in 19-03

[SWS_PER_00160] [SWS_PER_00161] [SWS_PER_00255] [SWS_PER_00256] [SWS_PER_00257] [SWS_PER_00258] [SWS_PER_00259] [SWS_PER_00260] [SWS_PER_00261] [SWS_PER_00262] [SWS_PER_00264] [SWS_PER_00268] [SWS_PER_00269] [SWS_PER_00270] [SWS_PER_00271] [SWS_PER_00272] [SWS_PER_00273] [SWS_PER_00274] [SWS_PER_00276] [SWS_PER_00278] [SWS_PER_00279] [SWS_PER_00280] [SWS_PER_00282] [SWS_PER_00284] [SWS_PER_00285] [SWS_PER_00300] [SWS_PER_00301] [SWS_PER_00316]

E.8 Traceable Item History of this Document According to AUTOSAR Release 18-10

E.8.1 Added Specification Items in 18-10

[SWS_PER_00309] [SWS_PER_00311] [SWS_PER_00312] [SWS_PER_00313]
[SWS_PER_00314] [SWS_PER_00315] [SWS_PER_00316] [SWS_PER_00317]
[SWS_PER_00318] [SWS_PER_00319] [SWS_PER_00320] [SWS_PER_00321]
[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_00331] [SWS_PER_00332] [SWS_PER_00333]
[SWS_PER_00334] [SWS_PER_00335] [SWS_PER_00336] [SWS_PER_00337]
[SWS_PER_00338] [SWS_PER_00339] [SWS_PER_00340] [SWS_PER_00341]
[SWS_PER_00342] [SWS_PER_00343] [SWS_PER_00344] [SWS_PER_00345]
[SWS_PER_00346] [SWS_PER_00347] [SWS_PER_00348] [SWS_PER_NA]

E.8.2 Changed Specification Items in 18-10

[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_00147] [SWS_PER_00160] [SWS_PER_00161] [SWS_PER_00163]
[SWS_PER_00164] [SWS_PER_00165] [SWS_PER_00166] [SWS_PER_00180]
[SWS_PER_00181] [SWS_PER_00182] [SWS_PER_00210] [SWS_PER_00211]

E.8.3 Deleted Specification Items in 18-10

[SWS_PER_00004] [SWS_PER_00041] [SWS_PER_00080] [SWS_PER_00129]
[SWS_PER_00130] [SWS_PER_00131] [SWS_PER_00132] [SWS_PER_00133]
[SWS_PER_00134] [SWS_PER_00148] [SWS_PER_00169] [SWS_PER_00170]
[SWS_PER_00171] [SWS_PER_00172] [SWS_PER_00173] [SWS_PER_00174]
[SWS_PER_00175] [SWS_PER_00176] [SWS_PER_00200] [SWS_PER_00201]
[SWS_PER_00220] [SWS_PER_00250] [SWS_PER_00500] [SWS_PER_UNUSED]

E.9 Traceable Item History of this Document According to AUTOSAR Release 18-03

E.9.1 Added Specification Items in 18-03

[SWS_PER_00080] [SWS_PER_00146] [SWS_PER_00147] [SWS_PER_00148]
[SWS_PER_00162] [SWS_PER_00163] [SWS_PER_00164] [SWS_PER_00165]
[SWS_PER_00166] [SWS_PER_00167] [SWS_PER_00168] [SWS_PER_00169]
[SWS_PER_00170] [SWS_PER_00171] [SWS_PER_00172] [SWS_PER_00173]
[SWS_PER_00174] [SWS_PER_00175] [SWS_PER_00176] [SWS_PER_00180]
[SWS_PER_00181] [SWS_PER_00182] [SWS_PER_00250] [SWS_PER_00251]
[SWS_PER_00252] [SWS_PER_00253] [SWS_PER_00254] [SWS_PER_00255]
[SWS_PER_00256] [SWS_PER_00257] [SWS_PER_00258] [SWS_PER_00259]
[SWS_PER_00260] [SWS_PER_00261] [SWS_PER_00262] [SWS_PER_00264]
[SWS_PER_00265] [SWS_PER_00266] [SWS_PER_00267] [SWS_PER_00268]
[SWS_PER_00269] [SWS_PER_00270] [SWS_PER_00271] [SWS_PER_00272]
[SWS_PER_00273] [SWS_PER_00274] [SWS_PER_00275] [SWS_PER_00276]
[SWS_PER_00277] [SWS_PER_00278] [SWS_PER_00279] [SWS_PER_00280]
[SWS_PER_00281] [SWS_PER_00282] [SWS_PER_00283] [SWS_PER_00284]
[SWS_PER_00285] [SWS_PER_00300] [SWS_PER_00301] [SWS_PER_00302]
[SWS_PER_00303] [SWS_PER_00304] [SWS_PER_UNUSED]

E.9.2 Changed Specification Items in 18-03

[SWS_PER_00004] [SWS_PER_00113] [SWS_PER_00114] [SWS_PER_00115]
[SWS_PER_00132] [SWS_PER_00133] [SWS_PER_00134] [SWS_PER_00201]
[SWS_PER_00220] [SWS_PER_00500]

E.9.3 Deleted Specification Items in 18-03

[SWS_PER_00003] [SWS_PER_00005] [SWS_PER_00006] [SWS_PER_00007]
[SWS_PER_00008] [SWS_PER_00010] [SWS_PER_00012] [SWS_PER_00013]
[SWS_PER_00014] [SWS_PER_00015] [SWS_PER_00016] [SWS_PER_00017]
[SWS_PER_00018] [SWS_PER_00019] [SWS_PER_00020] [SWS_PER_00051]
[SWS_PER_00060] [SWS_PER_00061] [SWS_PER_00076] [SWS_PER_00100]
[SWS_PER_00101] [SWS_PER_00102] [SWS_PER_00103] [SWS_PER_00104]
[SWS_PER_00105] [SWS_PER_00109] [SWS_PER_00117] [SWS_PER_00118]
[SWS_PER_00120] [SWS_PER_00121] [SWS_PER_00123] [SWS_PER_00150]
[SWS_PER_00151] [SWS_PER_00152] [SWS_PER_00153] [SWS_PER_00154]
[SWS_PER_00155] [SWS_PER_00156] [SWS_PER_00157]

E.10 Traceable Item History of this Document According to AUTOSAR Release 17-10

E.10.1 Added Specification Items in 17-10

[SWS_PER_00008] [SWS_PER_00100] [SWS_PER_00101] [SWS_PER_00102]
[SWS_PER_00103] [SWS_PER_00104] [SWS_PER_00105] [SWS_PER_00106]
[SWS_PER_00107] [SWS_PER_00108] [SWS_PER_00109] [SWS_PER_00110]
[SWS_PER_00111] [SWS_PER_00112] [SWS_PER_00113] [SWS_PER_00114]
[SWS_PER_00115] [SWS_PER_00116] [SWS_PER_00117] [SWS_PER_00118]
[SWS_PER_00119] [SWS_PER_00120] [SWS_PER_00121] [SWS_PER_00122]
[SWS_PER_00123] [SWS_PER_00124] [SWS_PER_00125] [SWS_PER_00126]
[SWS_PER_00127] [SWS_PER_00128] [SWS_PER_00129] [SWS_PER_00130]
[SWS_PER_00131] [SWS_PER_00132] [SWS_PER_00133] [SWS_PER_00134]
[SWS_PER_00140] [SWS_PER_00141] [SWS_PER_00142] [SWS_PER_00143]
[SWS_PER_00144] [SWS_PER_00145] [SWS_PER_00150] [SWS_PER_00151]
[SWS_PER_00152] [SWS_PER_00153] [SWS_PER_00154] [SWS_PER_00155]
[SWS_PER_00156] [SWS_PER_00157] [SWS_PER_00160] [SWS_PER_00161]
[SWS_PER_00200] [SWS_PER_00201] [SWS_PER_00210] [SWS_PER_00211]
[SWS_PER_00220] [SWS_PER_00221] [SWS_PER_00222] [SWS_PER_00500]

E.10.2 Changed Specification Items in 17-10

[SWS_PER_00003] [SWS_PER_00004] [SWS_PER_00010] [SWS_PER_00013]
[SWS_PER_00014] [SWS_PER_00016] [SWS_PER_00017] [SWS_PER_00041]
[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_00051] [SWS_PER_00060] [SWS_PER_00061] [SWS_PER_00076]

E.10.3 Deleted Specification Items in 17-10

[SWS_PER_00011] [SWS_PER_00021] [SWS_PER_00022] [SWS_PER_00023]
[SWS_PER_00024] [SWS_PER_00025] [SWS_PER_00026] [SWS_PER_00027]
[SWS_PER_00028] [SWS_PER_00029] [SWS_PER_00040] [SWS_PER_00045]
[SWS_PER_00053] [SWS_PER_00054] [SWS_PER_00055] [SWS_PER_00056]
[SWS_PER_00057] [SWS_PER_00058] [SWS_PER_00059] [SWS_PER_00062]
[SWS_PER_00066] [SWS_PER_00069] [SWS_PER_00070] [SWS_PER_00071]
[SWS_PER_00072] [SWS_PER_00073] [SWS_PER_00074] [SWS_PER_00075]
[SWS_PER_00077] [SWS_PER_00078]

E.11 Traceable Item History of this Document According to AUTOSAR Release 17-03

E.11.1 Added Specification Items in 17-03

[SWS_PER_00002] [SWS_PER_00003] [SWS_PER_00004] [SWS_PER_00005]
[SWS_PER_00006] [SWS_PER_00007] [SWS_PER_00010] [SWS_PER_00011]
[SWS_PER_00012] [SWS_PER_00013] [SWS_PER_00014] [SWS_PER_00015]
[SWS_PER_00016] [SWS_PER_00017] [SWS_PER_00018] [SWS_PER_00019]
[SWS_PER_00020] [SWS_PER_00021] [SWS_PER_00022] [SWS_PER_00023]
[SWS_PER_00024] [SWS_PER_00025] [SWS_PER_00026] [SWS_PER_00027]
[SWS_PER_00028] [SWS_PER_00029] [SWS_PER_00040] [SWS_PER_00041]
[SWS_PER_00042] [SWS_PER_00043] [SWS_PER_00044] [SWS_PER_00045]
[SWS_PER_00046] [SWS_PER_00047] [SWS_PER_00048] [SWS_PER_00049]
[SWS_PER_00050] [SWS_PER_00051] [SWS_PER_00052] [SWS_PER_00053]
[SWS_PER_00054] [SWS_PER_00055] [SWS_PER_00056] [SWS_PER_00057]
[SWS_PER_00058] [SWS_PER_00059] [SWS_PER_00060] [SWS_PER_00061]
[SWS_PER_00062] [SWS_PER_00066] [SWS_PER_00069] [SWS_PER_00070]
[SWS_PER_00071] [SWS_PER_00072] [SWS_PER_00073] [SWS_PER_00074]
[SWS_PER_00075] [SWS_PER_00076] [SWS_PER_00077] [SWS_PER_00078]

E.11.2 Changed Specification Items in 17-03

none

E.11.3 Deleted Specification Items in 17-03

none