Document Title	Specification of Safe Hardware Acceleration
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	1088

Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R25-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2025-11-27	R25-11	AUTOSAR Release Management	• Initial release

Disclaimer

Table of Contents

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the Functional Cluster Safe Hardware Acceleration.

The Safe Hardware Acceleration Functional Cluster will be referenced as Safe Hardware Acceleration in the remainder of this document.

Safe Hardware Acceleration offers mechanisms to Adaptive Applications and other Functional Clusters to send tasks for parallel execution to the hardware accelerators availalbe in the system.

The Safe Hardware Acceleration 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 Safe Hardware Acceleration will be called Safe Hardware Acceleration 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.

The explanation of how to use the functional cluster API and its use-cases are described in [1, EXP SafeHardwareAccelerationAPI]

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 [2].

Abbreviation / Acronym	Description
SHWA	Safe Hardware Acceleration

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

Term	Description
Accessor	Refers to the Accessor the only way to access to Buffer underlying data
Adaptive Application	Refers to the Adaptive Application defined in [2], or to a Functional Cluster implemented as a Daemon.
Adaptive Platform	Refers to the AUTOSAR Adaptive Platform defined in [2].
Adaptive Platform Foundation	Refers to the Adaptive Platform Foundation defined in [2].
Buffer	Refers to a Buffer memory area allocated for data storage.
Daemon	Refers to a Functional Cluster implemented as a Daemon.
Device	A device represents a physical computing unit, where parallel computations can be executed. This could be a CPU or GPU that is accessed directly by Safe Hardware Acceleration.
Execution Manifest	Refers to the Execution Manifest defined in [2].
Functional Cluster	Refers to the Functional Cluster defined in [2].
Host	The host refers to the general-purpose CPU and its associated memory that runs the main application code and controls the overall program execution. Host term is equal to the system that runs your C++ program (the application). It is typically the CPU + host memory (RAM).
Kernel	Refers to a Kernel. A kernel is a function that executes in parallel on a device (like a GPU or other accelerator). It represents the computational work that is distributed and performed concurrently by multiple work-items on the device.
Queue	Refers to the object which holds all tasks for execution on hardware accelerators.
Platform	Provides high-level representations of platform underlying hardware and software environment.
Safe Hardware Acceleration	The functional cluster described in this document, which handles the tasks of AUTOSAR Adaptive Applications and other functional clusters.
Safe Hardware Acceleration User	An Adaptive Application or another Functional Cluster that uses Safe Hardware Acceleration.
Service Interface	Refers to the Service Interface defined in [2].
SHWA Object	The object created by Safe Hardware Acceleration classes.
Software Package	Refers to the Software Package defined in [2].

Table 2.2: Terms used in the Scope of this Document

3 Related documentation

3.1 Input documents & related standards and norms

[1] Explanation of Safe API for hardware accelerators
AUTOSAR_AP_EXP_SafeHardwareAccelerationAPI

[2] Glossary
AUTOSAR_FO_TR_Glossary

[3] Specification of Adaptive Platform Core
AUTOSAR_AP_SWS_Core

[4] Specification of Execution Management
AUTOSAR_AP_SWS_ExecutionManagement

[5] Requirements on Safe Hardware Acceleration
AUTOSAR_AP_RS_SafeHardwareAcceleration

[6] General Requirements specific to Adaptive Platform
AUTOSAR_AP_RS_General

[7] Explanation of Adaptive Platform Design
AUTOSAR_AP_EXP_PlatformDesign

[8] Specification of Manifest
AUTOSAR_AP_TPS_ManifestSpecification

3.2 Further applicable specification

AUTOSAR provides a core specification [3] which is also applicable for this functional cluster. The chapter [3] 7.1 *"General requirements for all Functional Clusters"* shall be considered an additional and required specification for implementing this functional cluster.

4 Constraints and assumptions

4.1 Known limitations

The interaction with hardware acceleration devices could be done only within one ECU.

4.2 Assumptions about the Implementation of Safe Hardware Acceleration

Safe Hardware Acceleration can be implemented with different internal architectures, and at the same time, Safe Hardware Acceleration users expect a well defined behavior that is independent of the actual implementation of Safe Hardware Acceleration. 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 Safe Hardware Acceleration.

The following subsections list some of the possible implementations of Safe Hardware Acceleration.

4.2.1 Library Only

In this scenario, Safe Hardware Acceleration 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 Safe Hardware Acceleration at run-time (deployment parts).

4.2.2 Using a Central Daemon

In this scenario, the Safe Hardware Acceleration 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 underlying hardware acceleration framework. 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 Direct Access to Hardware Acceleration Devices

Independently of whether a daemon is used or not, Safe Hardware Acceleration can be implemented to directly access hardware devices like CPU, GPU or other hardware accelerator.

Most operating systems include drivers that take care of communication with the underlying hardware. But this approach is not recommended to be followed due to complexity and potential risks. This use case is therefore not described in any detail in this specification.

5 Dependencies to other Functional Clusters

Safe Hardware Acceleration(SHWA) is a new AUTOSAR Functional Cluster (FC). At the moment of writing there are no dependencies to other FCs.

5.1 Safe Hardware Acceleration(SHWA) Specifics

The Safe Hardware Acceleration is 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]).

5.2 Provided Interfaces

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

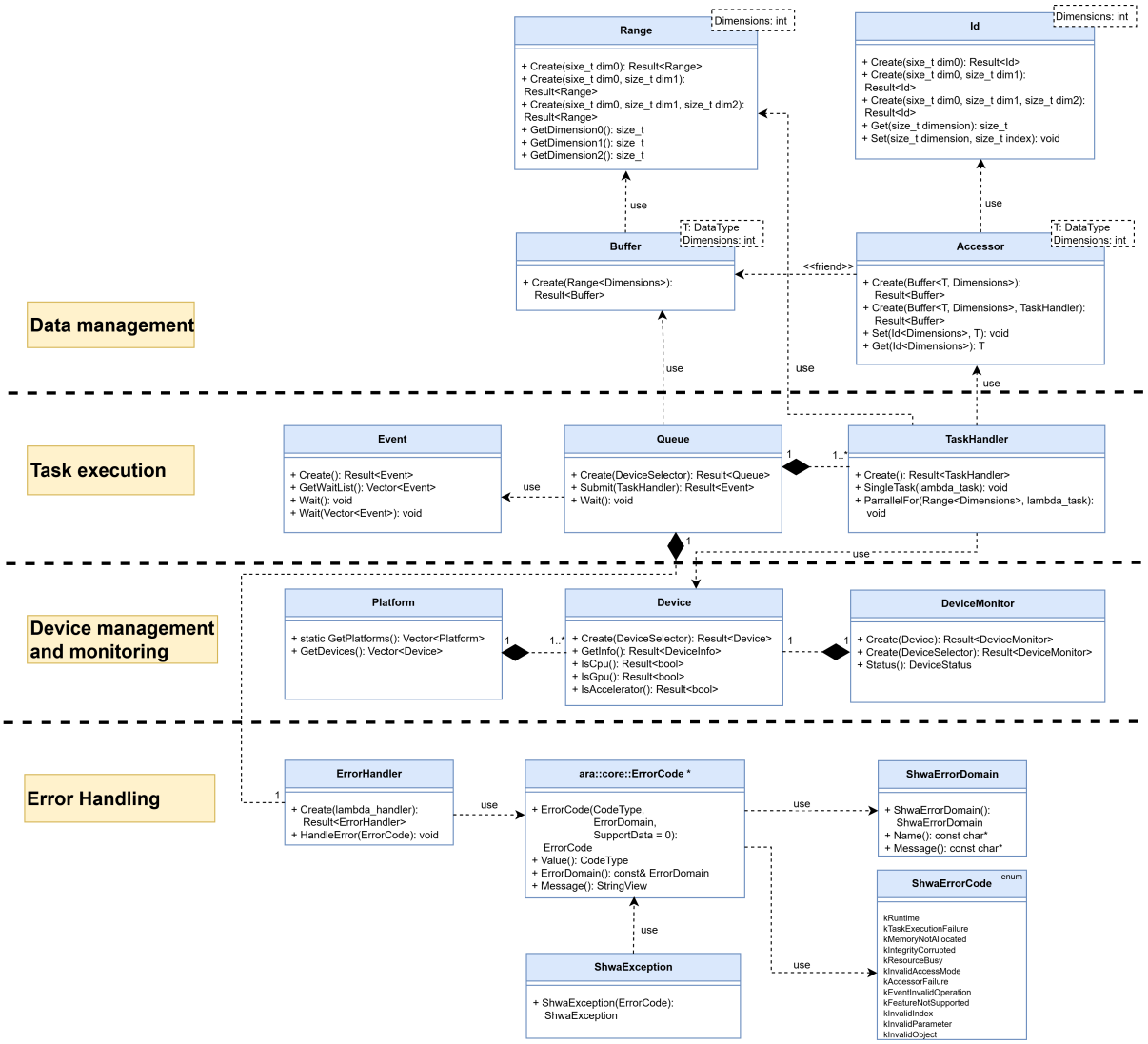

Figure 5.1: Interfaces Provided by Safe Hardware Acceleration to Other Functional Clusters

*Details can be found in [3, SWS Core]

Figure 5.1 shows interfaces provided by `Safe Hardware Acceleration` to other `Functional Clusters` within the AUTOSAR `Adaptive Platform`.

Interfaces provided by `Safe Hardware Acceleration` can be logically split into three section, as described on Figure 5.1

- Storage and management of data which will be used for operations on `HWA`
- Task execution on `HWA`
- `HWA` Device management and monitoring

6 Requirements Tracing

The following tables reference the requirements specified in the [5, RS SafeHardwareAcceleration] 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	[AP_SWS_SHWA_00013]
[RS_AP_00119]	Return values / application errors	[AP_SWS_SHWA_01307] [AP_SWS_SHWA_01308] [AP_SWS_SHWA_01309] [AP_SWS_SHWA_01311] [AP_SWS_SHWA_01312]
[RS_AP_00120]	Method and Function names	[AP_SWS_SHWA_01303] [AP_SWS_SHWA_01307] [AP_SWS_SHWA_01308] [AP_SWS_SHWA_01309] [AP_SWS_SHWA_01310] [AP_SWS_SHWA_01311] [AP_SWS_SHWA_01312]
[RS_AP_00121]	Parameter names	[AP_SWS_SHWA_01303] [AP_SWS_SHWA_01309] [AP_SWS_SHWA_01310] [AP_SWS_SHWA_01312]
[RS_AP_00122]	Type names	[AP_SWS_SHWA_01300] [AP_SWS_SHWA_01301] [AP_SWS_SHWA_01302] [AP_SWS_SHWA_01304] [AP_SWS_SHWA_01305] [AP_SWS_SHWA_01306]
[RS_AP_00125]	Enumerator and constant names	[AP_SWS_SHWA_01300] [AP_SWS_SHWA_01301]
[RS_AP_00127]	Usage of ara::core types	[AP_SWS_SHWA_01300] [AP_SWS_SHWA_01301] [AP_SWS_SHWA_01302] [AP_SWS_SHWA_01304]
[RS_AP_00128]	Error reporting	[AP_SWS_SHWA_00470] [AP_SWS_SHWA_00471] [AP_SWS_SHWA_00472] [AP_SWS_SHWA_00474] [AP_SWS_SHWA_00475] [AP_SWS_SHWA_00477] [AP_SWS_SHWA_00478] [AP_SWS_SHWA_00479] [AP_SWS_SHWA_00480] [AP_SWS_SHWA_00481] [AP_SWS_SHWA_00482] [AP_SWS_SHWA_00483] [AP_SWS_SHWA_00484] [AP_SWS_SHWA_00485] [AP_SWS_SHWA_00486] [AP_SWS_SHWA_00487] [AP_SWS_SHWA_00488] [AP_SWS_SHWA_00551]
[RS_AP_00135]	Avoidance of shared ownership	[AP_SWS_SHWA_01303] [AP_SWS_SHWA_01312]
[RS_AP_00140]	Usage of "final specifier"	[AP_SWS_SHWA_01304]
[RS_AP_00142]	Handling of unsuccessful operations	[SWS_SHWA_00576]
[RS_AP_00149]	Error handling for non-initialized Functional Cluster	[AP_SWS_SHWA_00476] [AP_SWS_SHWA_01300] [AP_SWS_SHWA_01301]
[RS_AP_00159]	usage of "noexcept" specifier	[AP_SWS_SHWA_01303] [AP_SWS_SHWA_01307] [AP_SWS_SHWA_01308] [AP_SWS_SHWA_01309] [AP_SWS_SHWA_01311] [AP_SWS_SHWA_01312]

△

Requirement	Description	Satisfied by
[RS_SHWA_00001]	SHWA Operations Execution on Hardware Accelerator	[AP_SWS_SHWA_00015] [AP_SWS_SHWA_00016] [AP_SWS_SHWA_00102] [AP_SWS_SHWA_00215] [AP_SWS_SHWA_00216] [AP_SWS_SHWA_00301] [AP_SWS_SHWA_00800] [AP_SWS_SHWA_00801] [AP_SWS_SHWA_00802] [AP_SWS_SHWA_00900] [AP_SWS_SHWA_00901] [AP_SWS_SHWA_00902] [AP_SWS_SHWA_00903] [AP_SWS_SHWA_00904] [AP_SWS_SHWA_00905] [AP_SWS_SHWA_00906] [AP_SWS_SHWA_00908] [AP_SWS_SHWA_00909] [AP_SWS_SHWA_00910] [AP_SWS_SHWA_00911] [AP_SWS_SHWA_00912] [AP_SWS_SHWA_00913] [AP_SWS_SHWA_00914] [AP_SWS_SHWA_00915] [AP_SWS_SHWA_00916] [AP_SWS_SHWA_00917] [AP_SWS_SHWA_01200] [AP_SWS_SHWA_01201] [AP_SWS_SHWA_01202] [AP_SWS_SHWA_01203] [AP_SWS_SHWA_01204]
[RS_SHWA_00002]	SHWA Data Exchange with Hardware Accelerators	[AP_SWS_SHWA_00201] [AP_SWS_SHWA_00202] [AP_SWS_SHWA_00203] [AP_SWS_SHWA_00204] [AP_SWS_SHWA_00205] [AP_SWS_SHWA_00206] [AP_SWS_SHWA_00209] [AP_SWS_SHWA_00210] [AP_SWS_SHWA_00211] [AP_SWS_SHWA_00212] [AP_SWS_SHWA_00213] [AP_SWS_SHWA_00214]
[RS_SHWA_00004]	Hardware Accelerator Runtime State Monitoring	[AP_SWS_SHWA_00103] [AP_SWS_SHWA_01400] [AP_SWS_SHWA_01401] [AP_SWS_SHWA_01402] [AP_SWS_SHWA_01403] [AP_SWS_SHWA_01404] [AP_SWS_SHWA_01405] [AP_SWS_SHWA_01406] [AP_SWS_SHWA_01407] [AP_SWS_SHWA_01408] [AP_SWS_SHWA_01409]
[RS_SHWA_00005]	Timeout-based Blocking Operations Cancellation Mechanism	[AP_SWS_SHWA_00606] [AP_SWS_SHWA_00917]
[RS_SHWA_00006]	Hardware Accelerator Type Selection	[AP_SWS_SHWA_00309] [AP_SWS_SHWA_00400] [AP_SWS_SHWA_00401] [AP_SWS_SHWA_00402] [AP_SWS_SHWA_00403] [AP_SWS_SHWA_00404] [AP_SWS_SHWA_00405] [AP_SWS_SHWA_00406] [AP_SWS_SHWA_00407] [AP_SWS_SHWA_00408] [AP_SWS_SHWA_00409] [AP_SWS_SHWA_00410] [AP_SWS_SHWA_00411] [AP_SWS_SHWA_00412] [AP_SWS_SHWA_00413]
[RS_SHWA_00007]	Synchronous Errors Handling Mechanism	[AP_SWS_SHWA_00007] [AP_SWS_SHWA_00008] [AP_SWS_SHWA_00011] [AP_SWS_SHWA_00012] [AP_SWS_SHWA_00015] [AP_SWS_SHWA_00016] [AP_SWS_SHWA_00209] [AP_SWS_SHWA_00210] [AP_SWS_SHWA_00211] [AP_SWS_SHWA_00212] [AP_SWS_SHWA_00213] [AP_SWS_SHWA_00214] [AP_SWS_SHWA_00215] [AP_SWS_SHWA_00216] [AP_SWS_SHWA_00308] [AP_SWS_SHWA_00309] [AP_SWS_SHWA_00310] [AP_SWS_SHWA_00311] [AP_SWS_SHWA_00312] [AP_SWS_SHWA_00313] [AP_SWS_SHWA_00403] [AP_SWS_SHWA_00404] [AP_SWS_SHWA_00406] [AP_SWS_SHWA_00407] [AP_SWS_SHWA_00409] [AP_SWS_SHWA_00410] [AP_SWS_SHWA_00412] [AP_SWS_SHWA_00413] [AP_SWS_SHWA_00507] [AP_SWS_SHWA_00602] [AP_SWS_SHWA_00603] [AP_SWS_SHWA_00604] [AP_SWS_SHWA_00605] [AP_SWS_SHWA_00606] [AP_SWS_SHWA_00707] [AP_SWS_SHWA_00708] [AP_SWS_SHWA_00709] [AP_SWS_SHWA_00802] [AP_SWS_SHWA_00908] [AP_SWS_SHWA_00909] [AP_SWS_SHWA_00910] [AP_SWS_SHWA_00911] [AP_SWS_SHWA_00912] [AP_SWS_SHWA_00913] [AP_SWS_SHWA_00914] [AP_SWS_SHWA_00915] ▽

▽

△

Requirement	Description	Satisfied by
		△ [AP_SWS_SHWA_00916] [AP_SWS_SHWA_00917] [AP_SWS_SHWA_01007] [AP_SWS_SHWA_01008] [AP_SWS_SHWA_01009] [AP_SWS_SHWA_01107] [AP_SWS_SHWA_01108] [AP_SWS_SHWA_01202] [AP_SWS_SHWA_01300] [AP_SWS_SHWA_01301] [AP_SWS_SHWA_01304] [AP_SWS_SHWA_01407] [AP_SWS_SHWA_01408]
[RS_SHWA_00008]	Asynchronous Error Handling Mechanism	[AP_SWS_SHWA_00500] [AP_SWS_SHWA_00501] [AP_SWS_SHWA_00502] [AP_SWS_SHWA_00503] [AP_SWS_SHWA_00504] [AP_SWS_SHWA_00505] [AP_SWS_SHWA_00506] [AP_SWS_SHWA_00507] [AP_SWS_SHWA_00508] [AP_SWS_SHWA_00913]
[RS_SHWA_00009]	Data Integrity Support	[AP_SWS_SHWA_00001] [AP_SWS_SHWA_00912] [AP_SWS_SHWA_00914] [AP_SWS_SHWA_00917]
[RS_SHWA_00010]	Allocated Resources Clean Up	[AP_SWS_SHWA_00575]
[RS_SHWA_00011]	SHWA multi-processing	[AP_SWS_SHWA_00550]
[RS_SHWA_00012]	Initialization and Shutdown	[AP_SWS_SHWA_00574]
[RS_SHWA_00013]	Memory Allocation	[AP_SWS_SHWA_00200] [AP_SWS_SHWA_00201] [AP_SWS_SHWA_00202] [AP_SWS_SHWA_00203] [AP_SWS_SHWA_00204] [AP_SWS_SHWA_00205] [AP_SWS_SHWA_00206] [AP_SWS_SHWA_00209] [AP_SWS_SHWA_00210] [AP_SWS_SHWA_00211] [AP_SWS_SHWA_00212] [AP_SWS_SHWA_00213] [AP_SWS_SHWA_00214]
[RS_SHWA_00014]	Device Properties	[AP_SWS_SHWA_00310] [AP_SWS_SHWA_00311] [AP_SWS_SHWA_00312] [AP_SWS_SHWA_00313] [AP_SWS_SHWA_00800] [AP_SWS_SHWA_00801] [AP_SWS_SHWA_00802]
[RS_SHWA_00015]	Device Abstraction	[AP_SWS_SHWA_00300] [AP_SWS_SHWA_00301] [AP_SWS_SHWA_00302] [AP_SWS_SHWA_00303] [AP_SWS_SHWA_00304] [AP_SWS_SHWA_00305] [AP_SWS_SHWA_00306] [AP_SWS_SHWA_00308] [AP_SWS_SHWA_00309] [AP_SWS_SHWA_00310] [AP_SWS_SHWA_00311] [AP_SWS_SHWA_00312] [AP_SWS_SHWA_00313]
[RS_SHWA_00016]	Event-based Synchronization	[AP_SWS_SHWA_00600] [AP_SWS_SHWA_00601] [AP_SWS_SHWA_00602] [AP_SWS_SHWA_00603] [AP_SWS_SHWA_00604] [AP_SWS_SHWA_00605] [AP_SWS_SHWA_00606]
[RS_SHWA_00017]	Data Access	[AP_SWS_SHWA_00000] [AP_SWS_SHWA_00001] [AP_SWS_SHWA_00002] [AP_SWS_SHWA_00003] [AP_SWS_SHWA_00004] [AP_SWS_SHWA_00005] [AP_SWS_SHWA_00006] [AP_SWS_SHWA_00007] [AP_SWS_SHWA_00008] [AP_SWS_SHWA_00009] [AP_SWS_SHWA_00010] [AP_SWS_SHWA_00011] [AP_SWS_SHWA_00012] [AP_SWS_SHWA_00014] [AP_SWS_SHWA_00100] [AP_SWS_SHWA_00101] [AP_SWS_SHWA_00102] [AP_SWS_SHWA_00700] [AP_SWS_SHWA_00701] [AP_SWS_SHWA_00702] [AP_SWS_SHWA_00703] [AP_SWS_SHWA_00704] [AP_SWS_SHWA_00705] [AP_SWS_SHWA_00706] [AP_SWS_SHWA_00707] [AP_SWS_SHWA_00708] [AP_SWS_SHWA_00709] [AP_SWS_SHWA_00710] [AP_SWS_SHWA_00711] [AP_SWS_SHWA_01000] [AP_SWS_SHWA_01001] [AP_SWS_SHWA_01002] [AP_SWS_SHWA_01003] [AP_SWS_SHWA_01004] ▽

▽

△

Requirement	Description	Satisfied by
		△ [AP_SWS_SHWA_01005] [AP_SWS_SHWA_01006] [AP_SWS_SHWA_01007] [AP_SWS_SHWA_01008] [AP_SWS_SHWA_01009] [AP_SWS_SHWA_01010] [AP_SWS_SHWA_01011] [AP_SWS_SHWA_01012]
[RS_SHWA_00018]	Environment Information	[AP_SWS_SHWA_01100] [AP_SWS_SHWA_01101] [AP_SWS_SHWA_01102] [AP_SWS_SHWA_01103] [AP_SWS_SHWA_01104] [AP_SWS_SHWA_01105] [AP_SWS_SHWA_01106] [AP_SWS_SHWA_01107] [AP_SWS_SHWA_01108] [AP_SWS_SHWA_01405]

Table 6.1: Requirements Tracing

7 Functional Specification

7.1 The Architecture of Safe Hardware Acceleration

The Safe Hardware Acceleration offers a mechanism to utilize hardware acceleration for different tasks which can be executed in parallel.

The high level design of the Safe Hardware Acceleration within an Adaptive Platform is depicted in Figure 7.1. As shown there, an Adaptive Application should use Safe Hardware Acceleration APIs for parallel tasks execution with simultaneous usage of other Functional Clusters like PHM to achieve required level of safe execution.

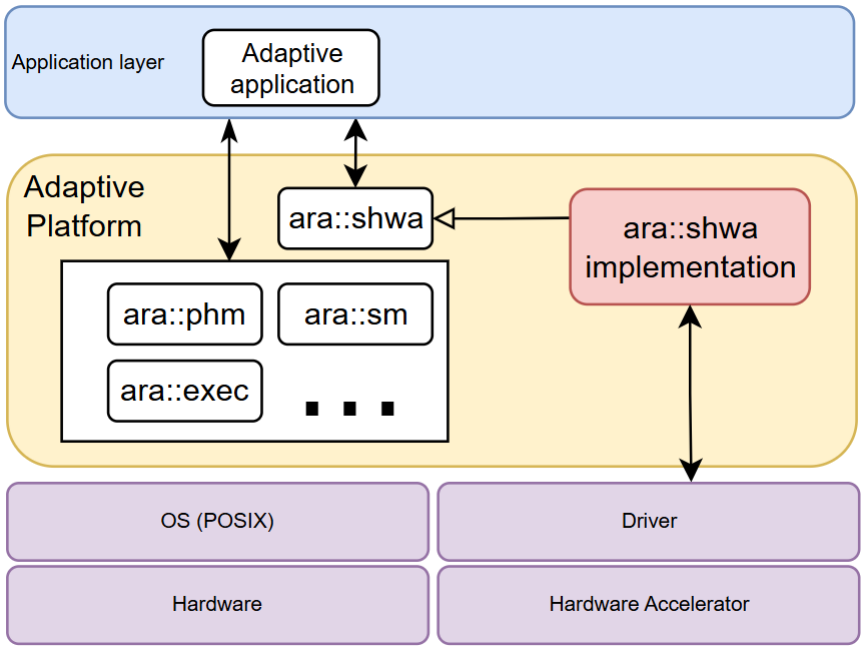

Figure 7.1: High Level Design of Safe Hardware Acceleration in an Adaptive Platform

The typical usage of the Safe Hardware Acceleration within an Adaptive Application is depicted in Figure 7.2. As shown there, an Adaptive Application can use a combination of multiple Queues, Devices and Buffers. Of course, the same applies to other Functional Clusters using Safe Hardware Acceleration.

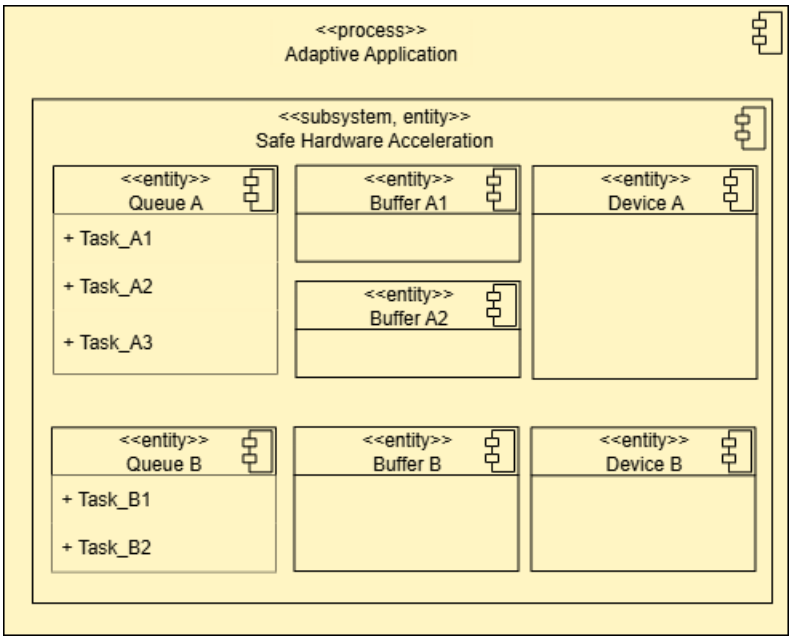

Figure 7.2: Typical usage of `Safe Hardware Acceleration` within an `Adaptive Application`

`Safe Hardware Acceleration` 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 `Safe Hardware Acceleration`, while `Daemons` of `Functional Clusters` can be modeled similarly to `Adaptive Applications`.

7.2 General Features of Safe Hardware Acceleration

[AP_SWS_SHWA_00013] SHWA Namespaces

Status: DRAFT
Upstream requirements: RS_AP_00115

⌈All specified classes within the Safe Hardware Acceleration shall reside within the C++ namespace ara::shwa.⌋

7.2.1 Error Handling

Error handling in Safe Hardware Acceleration is aligned with the guidelines described in [3]. To this end, the Safe Hardware Acceleration has to implement a set of standard classes and APIs, which are described in this section.

[AP_SWS_SHWA_00470] Safe Hardware Acceleration Error handling approach

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈Safe Hardware Acceleration shall use the error codes defined in ara::shwa::ShwaErrorCode to report problems to the calling Safe Hardware Acceleration user via ara::core::Result. Vendors of Safe Hardware Acceleration should add their own errors to a separate vendor-specific error-domain.⌋

ara::shwa::ShwaErrorCode belongs to the ara::shwa::ShwaErrorDomain, which can be used by a Safe Hardware Acceleration user to classify returned errors.

[AP_SWS_SHWA_00471] Safe Hardware Acceleration Error Domain

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈ara::shwa::GetShwaDomain shall return the global ara::shwa::ShwaErrorDomain object.⌋

[AP_SWS_SHWA_00472] Safe Hardware Acceleration Error Code

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈ara::shwa::MakeErrorCode shall return an ara::core::ErrorCode object when called with an error code enum from ara::shwa::ShwaErrorCode.⌋

[AP_SWS_SHWA_00474] Safe Hardware Acceleration Error message

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈ara::shwa::ShwaErrorDomain::Message shall return the error message associated with the passed ara::core::ErrorCode.⌋

The whole Safe Hardware Acceleration API has been designed to be exceptionless. If a Safe Hardware Acceleration user prefers to use exceptions, it may use ara::shwa::ShwaErrorDomain::ThrowAsException, or simply ara::core::ErrorDomain::ThrowAsException or ara::core::ErrorCode::ThrowAsException.

[AP_SWS_SHWA_00475] Safe Hardware Acceleration exceptions capabilities

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈ara::shwa::ShwaErrorDomain::ThrowAsException shall throw an ara::shwa::ShwaException that is created from the passed error code.⌋

[AP_SWS_SHWA_00476] Safe Hardware Acceleration Abort usage

Status: DRAFT
Upstream requirements: RS_AP_00149

⌈When Safe Hardware Acceleration detects an inconsistency in the manifest including invalid or non-existing URIs, it shall treat this situation as a violation as defined in [3] and call ara::core::Abort.⌋

7.2.1.1 Handling of General Errors

[AP_SWS_SHWA_00477] kRuntime General Error

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈When a function or method of Safe Hardware Acceleration encounters a runtime issues with memory management, device management or task execution, Safe Hardware Acceleration shall return the error kRuntime.⌋

When kRuntime occurs, the Safe Hardware Acceleration user shell perform additional measures to estimate if system is in appropriate state for safe performing of further operations.

[AP_SWS_SHWA_00478] kTaskExecutionFailure General Error

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈When a method of Safe Hardware Acceleration encounters problems during task execution on selected device, Safe Hardware Acceleration shall return the error kTaskExecutionFailure.⌋

When kTaskExecutionFailure occurs, Safe Hardware Acceleration user shall estimate health state of actual device, check data integrity of involved buffers (if there was any) and repeat needed task on current or alternative device.

[AP_SWS_SHWA_00479] kMemoryNotAllocated General Error

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈When a method of Safe Hardware Acceleration have issues to allocate memory needed for particular buffer, Safe Hardware Acceleration shall return the error kMemoryNotAllocated.⌋

[AP_SWS_SHWA_00480] kIntegrityCorrupted General Error

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈When usage of method of Safe Hardware Acceleration leads to integrity issues for allocated data, Safe Hardware Acceleration shall return the error kIntegrityCorrupted.⌋

In case of kIntegrityCorrupted, it's unsafe to use appropriate Buffer anymore.

[AP_SWS_SHWA_00481] kResourceBusy General Error

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈When method of Safe Hardware Acceleration is trying to use resource which is currently occupied by another Adaptive Platform Application (Applications) therefore can't perform requested operation, Safe Hardware Acceleration shall return the error kResourceBusy.⌋

[AP_SWS_SHWA_00482] kInvalidAccessMode General Error

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈When method of Safe Hardware Acceleration is trying to modify data inside of a Buffer without having appropriate access rights, Safe Hardware Acceleration shall return the error kInvalidAccessMode.⌋

[AP_SWS_SHWA_00483] kAccessorFailure General Error

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈When constraction of an Accessor with appropriate parameters is not possible, Safe Hardware Acceleration shall return the error kAccessorFailure.⌋

[AP_SWS_SHWA_00484] kEventInvalidOperation General Error

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈When Safe Hardware Acceleration user is performing not alloved operation with Event, like calling Wait() for ordered Queue, Safe Hardware Acceleration shall return the error kEventInvalidOperation.⌋

[AP_SWS_SHWA_00485] kFeatureNotSupported General Error

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈When method of `Safe Hardware Acceleration` is trying to use feature which is not supported in current implementation, `Safe Hardware Acceleration` shall return the error `kFeatureNotSupported`.⌋

[AP_SWS_SHWA_00486] kInvalidIndex General Error

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈When method of `Safe Hardware Acceleration` is trying to use which exceeds capability of current `Id`, `Range` or `Accessor`, `Safe Hardware Acceleration` shall return the error `kInvalidIndex`.⌋

[AP_SWS_SHWA_00487] kInvalidParameter General Error

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈When `Safe Hardware Acceleration` user is calling `Safe Hardware Acceleration` method with invalid parameters, `Safe Hardware Acceleration` shall return the error `kInvalidParameter`.⌋

[AP_SWS_SHWA_00488] kInvalidObject General Error

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈When `Safe Hardware Acceleration` user is trying to use destroyed or invalid object (like a `Buffer` or `Queue`), `Safe Hardware Acceleration` shall return the error `kInvalidObject`.⌋

7.2.2 Parallel Access to Hardware Acceleration Objects

According to [7], the SHWA Objects like Buffer, Queue or Accessor are local to one Process. Therefore, Safe Hardware Acceleration will never share SHWA Objects between two (or more) Processes, even of the same Executable. The background of this decision is that Safe Hardware Acceleration 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).

[AP_SWS_SHWA_00550] Objects sharing between processes

Status: DRAFT
Upstream requirements: RS_SHWA_00011

⌈SHWA Objects shall always be local to one Process.⌋

If SHWA Objects 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 and guarantee safety and security measures.

Safe Hardware Acceleration provides move semantics only for SHWA Objects. Having move semantics, SHWA Objects are prepared for handling by multiple threads of the same Adaptive Application, running in the context of the same Process.

To create shared(concurrent) access to SHWA Objects, it is the duty of the application designer to implement appropriate mechanisms, which allow to pass (copy) to another thread.

[AP_SWS_SHWA_00551] Objects sharing between threads

Status: DRAFT
Upstream requirements: RS_AP_00128

⌈When a function of Safe Hardware Acceleration that modifies a Buffer via Accessor's Set is called while another Set function that modifies the same Buffer is being executed, Safe Hardware Acceleration shall return the error kResourceBusy.⌋

Modifying Access to the same Buffers via ara::shwa::Accessor::Set is thread-safe.

Reading Access to the same Buffers via ara::shwa::Accessor::Get is thread-safe.

Access to other SHWA Objects can be shared between multiple threads, but it may require addional synchronization of read and write accesses.

All ***Create**** functions of Safe Hardware Acceleration are thread-safe.

7.2.3 Security Concepts

The Safe Hardware Acceleration does not implement own security mechanisms and relies on existed security mechanisms within Adaptive Platform.

7.2.4 Resource Management Concepts

The Safe Hardware Acceleration does not implement hardware resources management due to the following:

Unified hardware accelerator resource management is challenging because accelerators like GPUs, FPGAs, and TPUs differ significantly in architecture, memory models, and how they execute workloads. Each vendor provides its own software stack and APIs such as CUDA, ROCm, or oneAPI, which are not compatible with each other. Unlike CPUs, there is no standard abstraction layer or unified way to represent and schedule these devices.

Security and isolation requirements also vary widely, especially in multi-tenant or safety-critical environments. These differences make it impractical to manage all accelerators through a single, unified software layer without sacrificing performance, flexibility, or reliability.

Therefore, hardware acceleration resource management can be implemented as extension for specific hardware by platform vendor.

7.2.5 Safety Concepts

The Safe Hardware Acceleration FC was designed with safety in consideration. It provides the following concepts related to the safety:

7.2.5.1 Exceptionless API

All methods in SHWA are marked as noexcept. In case if method can return an error, its return value is wrapped into Result class in order to provide an error without usage of exceptions.

7.2.5.2 Factory methods instead of constructors

Classic object creation constructors are excluded in SHWA design. Naturally during object construction exception situation can be occured, which is not expected for SHWA since it's exceptionless. It's possible to mark constructor with noexcept, but in this case there will be no way to communicate an error to the Adaptive Application developer. Therefore factory methods with return value wrapped into Result class are used instead of constructors in SHWA.

7.2.5.3 Mandatory asynchronous error handler

During task submission and execution, two types of errors can be occured

- Synchronous - wrapped in Result return value from ara::shwa::Queue::Submit method.
- Asynchronous - can be occurred on the Device during task execution.

Asynchronous errors can be handled only with asynchronous error handler (callback function), which is mandatory to be defined in SHWA to guarantee explicit error handling for each error case.

7.2.5.4 Device status monitoring

SHWA provides specific methods for monitoring of actual status of Device to give possibility to develop reliable applications with appropriate safety measures in case of inappropriate status of Device.

7.2.5.5 RAII idiom

RAII idiom is used for design of all SHWA classes for safe resource management via object lifetime.

7.2.5.6 Controllable (timeout-based) locking mechanism

In some cases Adaptive Application logic requires locking mechanism to synchronize main thread with task execution on the Device. But main thread locking with dependency on the task execution on the Device can affect determinism of an Adaptive Application. Therefore SHWA provides additional (timeout-based) locking mechanism which can establish threshold for task execution and unlock main thread if this threshold was exceeded.

7.3 Data Access Specifics

In SHWA FC Accessors are the main mechanism for safely and efficiently accessing data on a Device, whether from Buffers or host memory, while maintaining synchronization between host and Device code.

7.3.1 Purpose of Accessors

- Provide typed, safe access to memory in Kernels.
- Abstract the complexity of data movement between host and Device.
- Specify access modes (Read, Write, Read/Write) so the runtime can schedule operations efficiently and avoid race conditions.

7.3.2 Accessors Key Concepts

- Binding to a Data Source

  An accessor is always associated with a data source, such as buffer (most common)

- Access Modes

  The access mode defines how the kernel interacts with the data:

  Read - Only read data.

  Write - Only write data (initial content ignored).

  Read/Write - Read existing data and write back updates.

  This approach lets the runtime optimize memory transfers and avoid unnecessary copies.

- Memory Spaces

  Global Memory is default for Buffer Accessors (accessible by all work-items).

- Synchronization

  Accessors are tied to the command group they are created in.

  When the Kernel finishes, writes are synchronized back to the host Buffer if the access mode allows it.

7.3.3 Data Access Specifics Summary

`Accessors` in `SHWA` are a bridge between `Kernels` and data, letting us specify what we want to access, how we want to access it, and where it resides - while the appropriate runtime transparently manages data movement and synchronization.

7.4 Device Specifics

The Device abstraction represents the actual compute hardware where kernels run - such as a CPU, GPU, FPGA, or other accelerator. The runtime part lets you choose, query, and manage devices in a flexible way so your code can target heterogeneous systems.

7.4.1 Device Abstraction

A Device object wraps:

- The type of Device (CPU, GPU, accelerator, custom)
- Its capabilities (extensions, max work-group size, supported memory types)
- Its vendor-specific properties

Example:

```
ara::shwa::Device dev = ara::shwa::Device(ara::shwa::GpuSelector);
std::cout << "Running on: "
          << dev.GetInfo<ara::shwa::info::Device::Name>() << "\n";
```

7.4.2 Device Types

Devices are typically grouped as:

- CPU devices - Host processor
- GPU devices - Discrete or integrated graphics
- Accelerator devices - FPGA, AI chips, DSPs
- Custom devices - Vendor-defined

7.4.3 Device Selection Mechanisms

SHWA provides Device selectors to pick a device at runtime:

Built-in selectors:

- CpuSelector - Chooses the best available CPU device.
- GpuSelector - Chooses the best available GPU device.
- AcceleratorSelector - Chooses an accelerator device.

Custom selectors:

You can write your own logic:

```

```

```
struct MySelector {
    int operator()(const ara::shwa::Device &dev) const {
        if (dev.IsGpu()) return 100; // high score for GPU
        return -1; // reject other devices
    }
};
ara::shwa::Device myDev = ara::shwa::Device(MySelector{});
```

7.4.4 Querying Device Properties

It is possible to inspect a device to decide if it meets requirements:

```
if (dev.GetInfo<ara::shwa::info::Device::local_mem_type>()
    != ara::shwa::info::local_mem_type::none) {
    // Device supports local memory
}
```

Common queries:

- name
- vendor
- max compute units
- max work group size
- global mem size
- extensions

7.4.5 Device and Queue Relationship

Device: The actual hardware unit.

Queue: Associated with a device; used to submit kernels.

Typical flow:

```
ara::shwa::Queue q(ara::shwa::GpuSelector);
ara::shwa::Device dev = q.GetDevice();
```

7.4.6 Device Choice Strategy

Performance-tuned -> pick explicitly (`GpuSelector` or custom scoring)

Specialized hardware -> check vendor name and extensions

7.4.7 Device Specifics Summary

`Device` lets you discover and choose the most suitable compute hardware at runtime, while device selectors automate or customize that choice based on performance or capability needs.

7.5 Memory Allocation Specifics

7.5.1 Memory Allocation Model

Memory management is a core part of SHWA, because heterogeneous computing often requires moving data between host (CPU) and device (GPU/FPGA/etc.). SHWA provides the following memory allocation model based on automatic movement.

Buffer-Accessor Model

Buffer manages memory automatically.

Data is copied between host and device when needed.

Accessed inside Kernels via Accessors.

Example:

```
1
2 auto range = ara::shwa::Range<bufferDimension>::Create(bufferSize);
3 auto buf = ara::shwa::Buffer::Create<int, 1>(range);
4
5 q.submit([&](ara::shwa::TaskHandler& h) {
6     auto acc = ara::shwa::Accessor::Create<int, 1, ara::shwa::AccessMode::
    kWrite>(buf);
7     h.ParallelFor(N, [=](ara::shwa::Id<1> i) {
8         acc[i] = i * 2;
9     });
10 });
```

Pros: Simple, portable, runtime handles transfers.

Cons: Less explicit control, might introduce synchronization overhead.

7.6 Task Queuing And Ordering Specifics

7.6.1 Task Queuing Approach

SHWA provides mechanism for task queuing for execution on the device using Queue class. ara::shwa::Queue::Submit method provides possibility to submit kernel function for execution on the device.

7.6.2 Task Ordering Approach

Task Queuing can be ordered or unordered, dependent on created Queue type. For unordered Queue it's possible to syncronyze tasks execution using Event object and its ara::shwa::Event::Wait method.

NOTE: SHWA can guarantee only task submition order, but not order of tasks execution on device

Example:

```
1
2 auto event = q.submit([&](ara::shwa::TaskHandler& handler) {
3     // task code
4 });
5
6 event.Wait();
```

ara::shwa::Event::Wait method allows to block main thread till task execution finish. There are also alternative methods ara::shwa::Event::Wait and ara::shwa::Event::WaitFor allowing to block main thread with dependency to other Events and perform time based block accordingly.

7.7 Functional Cluster Life-Cycle

Using `ara::core::Initialize` and `ara::core::Deinitialize` defined in [3, SWS Core], the Adaptive Application initializes and de-initializes all Functional Clusters with direct ARA interfaces (i.e. the Adaptive Platform Foundation).

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

[AP_SWS_SHWA_00574] Check for Initialization

Status: DRAFT
Upstream requirements: RS_SHWA_00012

⌈`ara::shwa::Buffer::Create`, `ara::shwa::Accessor::Create`, `ara::shwa::Queue::Create` 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, Safe Hardware Acceleration user shall handle this situation as a violation according to [SWS_CORE_00021].⌋

7.7.1 Startup

There is no specific information about start up.

7.7.2 Shutdown

[AP_SWS_SHWA_00575] Shutdown

Status: DRAFT
Upstream requirements: RS_SHWA_00010

⌈When `ara::core::Deinitialize` is called, the Safe Hardware Acceleration shall implicitly ensure that all Buffers, Queues and Accessors were destructed. Afterwards, Safe Hardware Acceleration shall free remaining resources that are not exposed by objects held by the Safe Hardware Acceleration user.⌋

7.8 Reporting

7.8.1 Security Events

This functional cluster does not define any security events.

7.8.2 Log Messages

This functional cluster does not define any non-verbose log messages (i.e., modelled DLT messages).

7.8.3 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.
Please note that concrete implementations might additionally implement Non-Standardized Violations (see also [SWS_CORE_00003]).

7.8.3.1 CalledWhileUnititialized

[SWS_SHWA_00576] ViolationMessage CalledWhileUnititializedViolation

Status: DRAFT
Upstream requirements: RS_AP_00142

⌈

Dlt-Message	CalledWhileUnititializedViolation		
Description	A named constructor or global function of the SHWA API was called before ara::core::Initialize.		
MessageId	0x80012fff		
MessageType Info	DLT_LOG_FATAL		
Dlt-Argument	***ArgumentDescription***	***ArgumentType***	***ArgumentUnit***
modeledProcessId	Meta-model identifier of the process that caused the violation, i.e., its short name path with '/' as a separator	uint8 [encoding UTF-8]	
location	An implementation-defined identifier of the location where the violation was detected, for example \\\\{filename}:\\\\{linenumber}.	uint8 [encoding UTF-8]	
functionName	Affected function or method of the SHWA API.	uint8 [encoding UTF-8]	

⌋

7.9 Platform Specifics

7.9.1 Platform Management Approach

In SHWA `Platform` provides high-level representation of the underlying hardware and software environment. There could be more than one `Platform` available in the vehicle, so `Platform` provides method `ara::shwa::Platform::GetPlatforms` to get all available platforms. `Platform` has one or more `Device`s bind to it, but one `Device` can be bind on;y to one `Platform`.

`Platform`-`Device` can be described using next example:

```
1       |
2       +-- Platform (Intel OpenCL)
3       |       +-- Device: Intel CPU
4       |       +-- Device: Intel GPU
5       |
6       +-- Platform (NVIDIA CUDA)
7       |       +-- Device: GeForce RTX 3080
8       |
9       +-- Platform (AMD ROCm)
10              +-- Device: Radeon RX 6800
```

7.10 Properties Management Specifics

7.10.1 Properties Management Approach

Since SHWA implementation is widely dependent on underlying hardware and software environment, capabilities and specifics of actual hardware accelerators, SHWA PropertyList as opportunity of SHWA functionality adjustment and extension.

Some SHWA classes can get PropertyList as an input parameter to Create method, which will allow to adapt SHWA Object behavior according to vendor specific capabilities.

Particular Properties should be defined by vendor.

```
namespace property {
namespace queue {

     class EnableProfiling {
     public:
       EnableProfiling() = default;
     };

}; // device
}; // info
```

Class using PropertyList is obliged to have ara::shwa::Queue::HasProperty and ara::shwa::Queue::GetProperty methods defined. Than it's possible to check if property available for current object and get it.

```
     auto q = ara::shwa::Queue::Create{
               device,
               errorHandler,
               PropertyList{
                    ara::shwa::property::queue::EnableProfiling{}
               }
          };

     if (q.HasProperty<ara::shwa::property::queue::EnableProfiling>()) {
          auto prop =
               buf.GetProperty<
                    ara::shwa::property::queue::EnableProfiling>();
     }
```

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)	
Port Interfaces:	States that the C++ API `class` is the related C++ API binding for the given modeled sub-class of `PortInterface`	
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 C++ `namespace` (in case of a `class` or non-member function) or a `class` declaration (in case of a member)	
Symbol:	C++ symbol 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 parameterize 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 C++ `Exception`s that may be thrown by the function	
Violations:	List of violations that may raised by the function	
Errors:	Error type (0..*)	List of defined `ara::core::ErrorCode`s 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 standardized by AUTOSAR
Description:	Brief description of the function	

Table 8.1: Explanation of an API table

The API of `Safe Hardware Acceleration` is designed around the `ara::core::Result`, which is returned by factory functions like `ara::shwa::Buffer::Create`. The classes defined in this chapter cannot be constructed directly by the `Safe Hardware Acceleration user`, and consequently the default constructors are considered to be not publicly accessible (i.e. to be deleted, private, or protected).

Classes that own resources (`Queue`, `Buffer` and `Accessor`) are designed as not copyable and intended to be managed exclusively by `std::unique_ptr` in order to implement single ownership paradigm.

8.1 Header: ara/shwa/accessor.h

[AP_SWS_SHWA_00000] Definition of Header File ara/shwa/accessor.h

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	Header File
Syntax:	`ara/shwa/accessor.h`
Description:	File contains classes and methods for accessing the data in the underlying buffers.

⌋

8.1.1 Class: Accessor

[AP_SWS_SHWA_00001] Definition of API class ara::shwa::Accessor

Status: DRAFT
Upstream requirements: RS_SHWA_00017, RS_SHWA_00009

⌈

Kind:	class	
Header file:	#include "ara/shwa/accessor.h"	
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"	
Scope:	namespace ara::shwa	
Symbol:	Accessor	
Syntax:	`template <typename DataT, int Dimensions = 1, AccessMode AccessMode Type = AccessMode::kRead, Target AccessTargetType = Target::kDevice, typename AllocatorT = BufferAllocator<std::remove_const_t<DataT>>> class Accessor final {...};`	
Template param:	typename DataT	The type of the value that shall be accessed.
	int Dimensions = 1	Dimensions of the underlying data.
	AccessMode Access ModeType = Access Mode::kRead	Type of the access to the underlying data.
	Target AccessTargetType = Target::kDevice	Target type of the access to the underlying hardware.
	typename AllocatorT = Buffer Allocator<std::remove_ const_t<DataT>>	Buffer memory allocator.
Description:	Accessor manages read and write access to allocated memory by Buffer. Designed as not copyable, since it's owning resources. Intended to be managed exclusively by std::unique_ptr, which may be moved.	

⌋

8.1.1.1 Public Member Types

8.1.1.1.1 Type Alias: ReferenceT

[AP_SWS_SHWA_00014] Definition of API type ara::shwa::Accessor::ReferenceT

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	type alias
Header file:	#include "ara/shwa/accessor.h"
Scope:	`class ara::shwa::Accessor`
Symbol:	ReferenceT
Syntax:	`using ReferenceT = std::conditional_t<AccessModeType == AccessMode::kRead, const DataT&, DataT&>;`
Description:	Reference type is const DataT& for read-only accessors and DataT& for other types of accessors.

⌋

8.1.1.2 Public Member Functions

8.1.1.2.1 Special Member Functions

8.1.1.2.1.1 Default Constructor

[AP_SWS_SHWA_00002] Definition of API function ara::shwa::Accessor::Accessor

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

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

⌋

8.1.1.2.1.2 Move Constructor

[AP_SWS_SHWA_00005] Definition of API function ara::shwa::Accessor::Accessor

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function
Header file:	#include "ara/shwa/accessor.h"
Scope:	`class ara::shwa::Accessor`
Syntax:	`Accessor (Accessor &&)=delete;`
Description:	The default move constructor for Accessor shall not be used.

⌋

8.1.1.2.1.3 Copy Assignment Operator

[AP_SWS_SHWA_00004] Definition of API function ara::shwa::Accessor::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

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

⌋

8.1.1.2.1.4 Move Assignment Operator

[AP_SWS_SHWA_00006] Definition of API function ara::shwa::Accessor::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function
Header file:	#include "ara/shwa/accessor.h"
Scope:	`class ara::shwa::Accessor`
Syntax:	`Accessor & operator= (Accessor &&)=delete;`

▽

△

Description:	The default move assignment operator for Accessor shall not be used.

⌋

8.1.1.2.2 Constructors

8.1.1.2.2.1 Accessor

[AP_SWS_SHWA_00003] Definition of API function ara::shwa::Accessor::Accessor

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

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

⌋

8.1.1.2.3 Member Functions

8.1.1.2.3.1 Create

[AP_SWS_SHWA_00008] Definition of API function ara::shwa::Accessor::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00017, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/accessor.h"	
Scope:	`class ara::shwa::Accessor`	
Syntax:	`static ara::core::Result< std::unique_ptr< Accessor > > Create (const std::unique_ptr< ara::shwa::Buffer< DataT, Dimensions, AllocatorT > > &bufferRef, TaskHandler &handler, const PropertyList &propList={}) noexcept;`	
Parameters (in):	bufferRef	Reference to accessed Buffer
	handler	Task Handler within which current Accessor can act
	propList	List of properties of the created accessor
Return value:	ara::core::Result< std::unique_ptr< Accessor > >	Unique pointer to created Accessor or an Error
Exception Safety:	exception safe	

▽

△

Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::kAccessorFailure	rollback_semantics
		Invalid use or construction of accessors
	ShwaErrorCode::kIntegrityCorrupted	no_rollback_semantics
		The structural integrity of the memory allocated by Buffer could not be established
Description:	Creates Accessor with specified parameters.	

⌋

8.1.1.2.3.2 Create

[AP_SWS_SHWA_00007] Definition of API function ara::shwa::Accessor::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00017, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/accessor.h"	
Scope:	`class ara::shwa::Accessor`	
Syntax:	`static ara::core::Result< std::unique_ptr< Accessor > > Create (const std::unique_ptr< ara::shwa::Buffer< DataT, Dimensions, AllocatorT > > &bufferRef, const PropertyList &propList={}) noexcept;`	
Parameters (in):	bufferRef	Reference to accessed Buffer
	propList	List of properties of the created accessor
Return value:	ara::core::Result< std::unique_ptr< Accessor > >	Unique pointer to created Accessor or an Error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::kAccessorFailure	rollback_semantics
		Invalid use or construction of accessors
	ShwaErrorCode::kIntegrityCorrupted	no_rollback_semantics
		The structural integrity of the memory allocated by Buffer could not be established
Description:	Creates Accessor with specified parameters.	

⌋

8.1.1.2.3.3 Get

[AP_SWS_SHWA_00012] Definition of API function ara::shwa::Accessor::Get

Status: DRAFT
Upstream requirements: RS_SHWA_00017, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/accessor.h"	
Scope:	`class ara::shwa::Accessor`	
Syntax:	`ara::core::Result< DataT > Get (Id< Dimensions > index);`	
Parameters (in):	index	Index of requested element
Return value:	ara::core::Result< DataT >	Value of requested element of a Buffer
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::kInvalid Index	rollback_semantics
		The operation could not be performed because no initial value is available
	ShwaErrorCode::k ResourceBusy	rollback_semantics
		The operation could not be performed because the resource is currently busy.
Description:	Provides value of an element with appropriate index.	

⌋

8.1.1.2.3.4 GetProperty

[AP_SWS_SHWA_00016] Definition of API function ara::shwa::Accessor::Get Property

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/accessor.h"	
Scope:	`class ara::shwa::Accessor`	
Syntax:	`template <typename PropertyT>` `ara::core::Result< PropertyT > GetProperty () const noexcept;`	
Template param:	PropertyT	Property type
Return value:	ara::core::Result< PropertyT >	Property object or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k FeatureNotSupported	rollback_semantics
		Used when a requested feature isn't supported

▽

△

Description:	Getting property from current Accessor object.

⌋

8.1.1.2.3.5 HasProperty

[AP_SWS_SHWA_00015] Definition of API function ara::shwa::Accessor::HasProperty

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/accessor.h"	
Scope:	`class ara::shwa::Accessor`	
Syntax:	`template <typename PropertyT>` `bool HasProperty () const noexcept;`	
Template param:	PropertyT	Property type
Return value:	bool	TRUE if property is available and FALSE otherwise
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Checking if property is available for current Accessor object.	

⌋

8.1.1.2.3.6 Set

[AP_SWS_SHWA_00011] Definition of API function ara::shwa::Accessor::Set

Status: DRAFT
Upstream requirements: RS_SHWA_00017, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/accessor.h"	
Scope:	`class ara::shwa::Accessor`	
Syntax:	`ara::core::Result< void > Set (Id< Dimensions > index, DataT value);`	
Parameters (in):	index	Index of requested element
	value	Value to set
Return value:	ara::core::Result< void >	Void or an Error
Exception Safety:	not exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::kInvalidIndex	no_rollback_semantics

▽

△

		The operation could not be performed because no initial value is available
	ShwaErrorCode::kInvalid AccessMode	no_rollback_semantics
		Opening a file failed because the requested combination of Open Modes is invalid
	ShwaErrorCode::k IntegrityCorrupted	no_rollback_semantics
		The structural integrity of the memory allocated by Buffer could not be established
	ShwaErrorCode::k ResourceBusy	rollback_semantics
		The operation could not be performed because the resource is currently busy.
Description:	Set a value of selected element of Buffer.	

⌋

8.1.1.2.3.7 operator[]

[AP_SWS_SHWA_00009] Definition of API function ara::shwa::Accessor::operator[]

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function	
Header file:	#include "ara/shwa/accessor.h"	
Scope:	`class ara::shwa::Accessor`	
Syntax:	`ReferenceT operator[] (Id< Dimensions > index) const;`	
Parameters (in):	index	Index of requested element
Return value:	ReferenceT	Reference to requested element of a Buffer
Exception Safety:	not exception safe	
Thread Safety:	not thread-safe	
Description:	Provides access to the element with appropriate index. Available only when: (Dimensions > 0)	

⌋

8.1.1.2.3.8 operator[]

[AP_SWS_SHWA_00010] Definition of API function ara::shwa::Accessor::operator[]

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function	
Header file:	#include "ara/shwa/accessor.h"	
Scope:	`class ara::shwa::Accessor`	
Syntax:	`ReferenceT operator[] (std::size_t index) const;`	
Parameters (in):	index	Index of requested element
Return value:	ReferenceT	Reference to requested element of a Buffer
Exception Safety:	not exception safe	
Thread Safety:	not thread-safe	
Description:	Provides access to the element with appropriate index. Available only when: (AccessMode != access_mode::atomic && Dimensions == 1)	

⌋

8.2 Header: ara/shwa/buffer.h

[AP_SWS_SHWA_00200] Definition of Header File ara/shwa/buffer.h

Status: DRAFT
Upstream requirements: RS_SHWA_00013

⌈

Kind:	Header File
Syntax:	`ara/shwa/buffer.h`
Description:	File contains classes and methods for data storage in underlying buffers.

⌋

8.2.1 Class: Buffer

[AP_SWS_SHWA_00201] Definition of API class ara::shwa::Buffer

Status: DRAFT
Upstream requirements: RS_SHWA_00002, RS_SHWA_00013

⌈

Kind:	class
Header file:	#include "ara/shwa/buffer.h"
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"

▽

△

Scope:	namespace ara::shwa	
Symbol:	Buffer	
Syntax:	`template <typename T, int Dimensions = 1, typename AllocatorT = BufferAllocator<std::remove_const_t<T>>> class Buffer final {...};`	
Template param:	typename T	The type of the underlying data to be sotred.
	int Dimensions = 1	Dimensions of the underlying data.
	typename AllocatorT = BufferAllocator<std::remove_const_t<T>>	Buffer memory allocator
Description:	Buffer manages memory allocation for operations on the host or/and HWA. Designed as not copyable, since it's owning resources. Intended to be managed exclusively by std::unique_ptr, which may be moved.	

⌋

8.2.1.1 Public Member Functions

8.2.1.1.1 Special Member Functions

8.2.1.1.1.1 Default Constructor

[AP_SWS_SHWA_00202] Definition of API function ara::shwa::Buffer::Buffer

Status: DRAFT
Upstream requirements: RS_SHWA_00002, RS_SHWA_00013

⌈

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

⌋

8.2.1.1.1.2 Move Constructor

[AP_SWS_SHWA_00205] Definition of API function ara::shwa::Buffer::Buffer

Status: DRAFT
Upstream requirements: RS_SHWA_00002, RS_SHWA_00013

⌈

Kind:	function
Header file:	#include "ara/shwa/buffer.h"
Scope:	`class ara::shwa::Buffer`
Syntax:	`Buffer (Buffer &&)=delete;`
Description:	The default move constructor for Buffer shall not be used.

⌋

8.2.1.1.1.3 Move Assignment Operator

[AP_SWS_SHWA_00206] Definition of API function ara::shwa::Buffer::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00002, RS_SHWA_00013

⌈

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

⌋

8.2.1.1.1.4 Copy Assignment Operator

[AP_SWS_SHWA_00204] Definition of API function ara::shwa::Buffer::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00002, RS_SHWA_00013

⌈

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

⌋

8.2.1.1.2 Constructors

8.2.1.1.2.1 Buffer

[AP_SWS_SHWA_00203] Definition of API function ara::shwa::Buffer::Buffer

Status: DRAFT
Upstream requirements: RS_SHWA_00002, RS_SHWA_00013

⌈

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

⌋

8.2.1.1.3 Member Functions

8.2.1.1.3.1 Create

[AP_SWS_SHWA_00212] Definition of API function ara::shwa::Buffer::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00002, RS_SHWA_00013, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/buffer.h"	
Scope:	`class ara::shwa::Buffer`	
Syntax:	`static ara::core::Result< std::unique_ptr< Buffer > > Create (AllocatorT allocator, const PropertyList &propList={}) noexcept;`	
Parameters (in):	allocator	Specific memory allocator
	propList	Array with required properties
Return value:	ara::core::Result< std::unique_ptr< Buffer > >	Unique pointer to Buffer
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::kRuntime	rollback_semantics
		General runtime error
	ShwaErrorCode::kMemoryNotAllocated	rollback_semantics
		Failed memory allocation on host/device.
Description:	Creates buffer with specified parameters.	

⌋

8.2.1.1.3.2 Create

[AP_SWS_SHWA_00211] Definition of API function ara::shwa::Buffer::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00002, RS_SHWA_00013, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/buffer.h"	
Scope:	`class ara::shwa::Buffer`	
Syntax:	`static ara::core::Result< std::unique_ptr< Buffer > > Create (T *hostData, const Range< Dimensions > &bufferRange, const PropertyList &propList={}) noexcept;`	
Parameters (in):	hostData	The type of the value that shall be retrieved
	bufferRange	Range with required dimensions
	propList	Array with required properties
Return value:	ara::core::Result< std::unique_ptr< Buffer > >	unique pointer to created Buffer
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::kRuntime	rollback_semantics
		General runtime error
	ShwaErrorCode::kMemoryNotAllocated	rollback_semantics
		Failed memory allocation on host/device.
Description:	Creates buffer with specified parameters.	

⌋

8.2.1.1.3.3 Create

[AP_SWS_SHWA_00214] Definition of API function ara::shwa::Buffer::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00013, RS_SHWA_00002, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/buffer.h"	
Scope:	`class ara::shwa::Buffer`	
Syntax:	`static ara::core::Result< std::unique_ptr< Buffer > > Create (const T *hostData, const Range< Dimensions > &bufferRange, AllocatorT allocator, const PropertyList &propList={}) noexcept;`	
Parameters (in):	hostData	The type of the const value that shall be retrieved
	bufferRange	Range with required dimensions
	allocator	Specific memory allocator
	propList	Array with required properties

▽

△

Return value:	ara::core::Result< std::unique_ptr< Buffer > >	Unique pointer to created Buffer
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::kRuntime	rollback_semantics
		General runtime error
	ShwaErrorCode::kMemoryNotAllocated	rollback_semantics
		Failed memory allocation on host/device.
Description:	Creates buffer with specified parameters.	

⌋

8.2.1.1.3.4 Create

[AP_SWS_SHWA_00209] Definition of API function ara::shwa::Buffer::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00002, RS_SHWA_00013, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/buffer.h"	
Scope:	`class ara::shwa::Buffer`	
Syntax:	`static ara::core::Result< std::unique_ptr< Buffer > > Create (const Range< Dimensions > &bufferRange, const PropertyList &propList={}) noexcept;`	
Parameters (in):	bufferRange	Range with required dimensions
	propList	Array with required properties
Return value:	ara::core::Result< std::unique_ptr< Buffer > >	Unique pointer to created Buffer
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::kRuntime	rollback_semantics
		General runtime error
	ShwaErrorCode::kMemoryNotAllocated	rollback_semantics
		Failed memory allocation on host/device.
Description:	Creates buffer with specified parameters.	

⌋

8.2.1.1.3.5 Create

[AP_SWS_SHWA_00210] Definition of API function ara::shwa::Buffer::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00002, RS_SHWA_00013, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/buffer.h"	
Scope:	`class ara::shwa::Buffer`	
Syntax:	`static ara::core::Result< std::unique_ptr< Buffer > > Create (const Range< Dimensions > &bufferRange, AllocatorT allocator, const PropertyList &propList={}) noexcept;`	
Parameters (in):	bufferRange	Range with required dimensions
	allocator	Specific memory allocator
	propList	Array with required properties
Return value:	ara::core::Result< std::unique_ptr< Buffer > >	Unique pointer to created Buffer
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::kRuntime	rollback_semantics
		General runtime error
	ShwaErrorCode::kMemoryNotAllocated	rollback_semantics
		Failed memory allocation on host/device.
Description:	Creates buffer with specified parameters.	

⌋

8.2.1.1.3.6 Create

[AP_SWS_SHWA_00213] Definition of API function ara::shwa::Buffer::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00002, RS_SHWA_00013, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/buffer.h"	
Scope:	`class ara::shwa::Buffer`	
Syntax:	`static ara::core::Result< std::unique_ptr< Buffer > > Create (const T *hostData, const Range< Dimensions > &bufferRange, const PropertyList &propList={}) noexcept;`	
Parameters (in):	hostData	The type of the const value that shall be retrieved
	bufferRange	Range with required dimensions
	propList	Array with required properties

▽

△

Return value:	ara::core::Result< std::unique_ptr< Buffer > >	Unique pointer to created Buffer
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::k Runtime	rollback_semantics
		General runtime error
	ShwaErrorCode::k MemoryNotAllocated	rollback_semantics
		Failed memory allocation on host/device.
Description:	Creates buffer with specified parameters.	

⌋

8.2.1.1.3.7 GetProperty

[AP_SWS_SHWA_00216] Definition of API function ara::shwa::Buffer::GetProperty

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/buffer.h"	
Scope:	`class ara::shwa::Buffer`	
Syntax:	`template <typename PropertyT>` `ara::core::Result< PropertyT > GetProperty () const noexcept;`	
Template param:	PropertyT	Property type
Return value:	ara::core::Result< PropertyT >	Property object or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k FeatureNotSupported	rollback_semantics
		Used when a requested feature isn't supported
Description:	Getting property from current Buffer object.	

⌋

8.2.1.1.3.8 HasProperty

[AP_SWS_SHWA_00215] Definition of API function ara::shwa::Buffer::HasProperty

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/buffer.h"	
Scope:	`class ara::shwa::Buffer`	
Syntax:	`template <typename PropertyT>` `bool HasProperty () const noexcept;`	
Template param:	PropertyT	Property type
Return value:	bool	TRUE if property is available and FALSE otherwise
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Checking if property is available for current Buffer object.	

⌋

8.3 Header: ara/shwa/device.h

[AP_SWS_SHWA_00300] Definition of Header File ara/shwa/device.h

Status: DRAFT
Upstream requirements: RS_SHWA_00015

⌈

Kind:	Header File
Syntax:	`ara/shwa/device.h`
Description:	File contains classes and methods for handling devices.

⌋

8.3.1 Class: Device

[AP_SWS_SHWA_00301] Definition of API class ara::shwa::Device

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00015

⌈

Kind:	class
Header file:	#include "ara/shwa/device.h"
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"

▽

△

Scope:	namespace ara::shwa
Symbol:	Device
Syntax:	`class Device final {...};`
Description:	Device class represents HWA device. Provides access to the device but not controlling it. Designed as copyable since not owning resources.

⌋

8.3.1.1 Public Member Functions

8.3.1.1.1 Special Member Functions

8.3.1.1.1.1 Default Constructor

[AP_SWS_SHWA_00302] Definition of API function ara::shwa::Device::Device

Status: DRAFT
Upstream requirements: RS_SHWA_00015

⌈

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

⌋

8.3.1.1.1.2 Move Constructor

[AP_SWS_SHWA_00305] Definition of API function ara::shwa::Device::Device

Status: DRAFT
Upstream requirements: RS_SHWA_00015

⌈

Kind:	function	
Header file:	#include "ara/shwa/device.h"	
Scope:	`class ara::shwa::Device`	
Syntax:	`Device (Device &&dev) noexcept=default;`	
Parameters (in):	dev	Rvalue reference to Device instance to move
Exception Safety:	exception safe	
Description:	The default Device move constructor.	

⌋

8.3.1.1.1.3 Move Assignment Operator

[AP_SWS_SHWA_00306] Definition of API function ara::shwa::Device::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00015

⌈

Kind:	function	
Header file:	#include "ara/shwa/device.h"	
Scope:	`class ara::shwa::Device`	
Syntax:	`Device & operator= (Device &&dev) noexcept=default;`	
Parameters (in):	dev	Rvalue reference to Device instance to perform move assignment
Return value:	Device &	returns reference to a Device instance
Exception Safety:	exception safe	
Description:	The default Device move assignment operator.	

⌋

8.3.1.1.1.4 Copy Assignment Operator

[AP_SWS_SHWA_00304] Definition of API function ara::shwa::Device::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00015

⌈

Kind:	function	
Header file:	#include "ara/shwa/device.h"	
Scope:	`class ara::shwa::Device`	
Syntax:	`Device & operator= (const Device &dev) noexcept=default;`	
Parameters (in):	dev	Reference to Device instance to assign
Return value:	Device &	returns reference to a Device instance
Exception Safety:	exception safe	
Description:	The default Device assignment operator.	

⌋

8.3.1.1.2 Constructors

8.3.1.1.2.1 Device

[AP_SWS_SHWA_00303] Definition of API function ara::shwa::Device::Device

Status: DRAFT
Upstream requirements: RS_SHWA_00015

⌈

Kind:	function	
Header file:	#include "ara/shwa/device.h"	
Scope:	`class ara::shwa::Device`	
Syntax:	`Device (Device const &dev) noexcept=default;`	
Parameters (in):	dev	Reference to Device instance to copy
Exception Safety:	exception safe	
Description:	The default Device copy constructor.	

⌋

8.3.1.1.3 Member Functions

8.3.1.1.3.1 Create

[AP_SWS_SHWA_00309] Definition of API function ara::shwa::Device::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00015, RS_SHWA_00006, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/device.h"	
Scope:	`class ara::shwa::Device`	
Syntax:	`template <typename DeviceSelector>` `static ara::core::Result< Device > Create (const DeviceSelector &deviceSelector) noexcept;`	
Template param:	deviceSelector	type of the device to be used for device creation.
DIRECTION NOT DEFINED	deviceSelector	--
Return value:	ara::core::Result< Device >	A pointer to the created device. In case of the error returns vendor-specific errors
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k ResourceBusy	rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::k FeatureNotSupported	rollback_semantics
		Used when a requested feature isn't supported
Description:	Creates a device based on specified device selector.	

⌋

8.3.1.1.3.2 Create

[AP_SWS_SHWA_00308] Definition of API function ara::shwa::Device::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00015, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/device.h"	
Scope:	`class ara::shwa::Device`	
Syntax:	`static ara::core::Result< Device > Create () noexcept;`	
Return value:	ara::core::Result< Device >	A pointer to the created device. In case of the error returns vendor-specific errors
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k ResourceBusy	rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::k FeatureNotSupported	rollback_semantics
		Used when a requested feature isn't supported
Description:	Creates a device based on default type, implementation defined.	

⌋

8.3.1.1.3.3 GetInfo

[AP_SWS_SHWA_00313] Definition of API function ara::shwa::Device::GetInfo

Status: DRAFT
Upstream requirements: RS_SHWA_00014, RS_SHWA_00015, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/device.h"	
Scope:	`class ara::shwa::Device`	
Syntax:	`template <typename InfoType> ara::core::Result< typename InfoType::ReturnType > GetInfo () const noexcept;`	
Return value:	ara::core::Result< typename Info Type::ReturnType >	Returns requested Device information using ReturnType declared in InfoType struct
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k FeatureNotSupported	rollback_semantics
		Used when a requested feature isn't supported
Description:	Provides requested information regurding Device.	

⌋

8.3.1.1.3.4 IsAccelerator

[AP_SWS_SHWA_00312] Definition of API function ara::shwa::Device::IsAccelerator

Status: DRAFT
Upstream requirements: RS_SHWA_00014, RS_SHWA_00015, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/device.h"	
Scope:	`class ara::shwa::Device`	
Syntax:	`ara::core::Result< bool > IsAccelerator () const noexcept;`	
Return value:	ara::core::Result< bool >	A Result containing the TRUE value if the device has type accelerator. In case of the error returns vendor-specific errors
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k ResourceBusy	rollback_semantics
		The operation could not be performed because the resource is currently busy
Description:	Checks if the device type is accelerator.	

⌋

8.3.1.1.3.5 IsCpu

[AP_SWS_SHWA_00310] Definition of API function ara::shwa::Device::IsCpu

Status: DRAFT
Upstream requirements: RS_SHWA_00014, RS_SHWA_00015, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/device.h"	
Scope:	`class ara::shwa::Device`	
Syntax:	`ara::core::Result< bool > IsCpu () const noexcept;`	
Return value:	ara::core::Result< bool >	A Result containing the TRUE value if the device has type CPU. In case of the error returns vendor-specific errors
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k ResourceBusy	rollback_semantics
		The operation could not be performed because the resource is currently busy
Description:	Checks if the device type is CPU.	

⌋

8.3.1.1.3.6 IsGpu

[AP_SWS_SHWA_00311] Definition of API function ara::shwa::Device::IsGpu

Status: DRAFT
Upstream requirements: RS_SHWA_00014, RS_SHWA_00015, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/device.h"	
Scope:	`class ara::shwa::Device`	
Syntax:	`ara::core::Result< bool > IsGpu () const noexcept;`	
Return value:	ara::core::Result< bool >	A Result containing the TRUE value if the device has type GPU. In case of the error returns vendor-specific errors
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k ResourceBusy	rollback_semantics
		The operation could not be performed because the resource is currently busy
Description:	Checks if the device type is GPU.	

⌋

8.4 Header: ara/shwa/device_monitor.h

[AP_SWS_SHWA_01400] Definition of Header File ara/shwa/device_monitor.h

Status: DRAFT
Upstream requirements: RS_SHWA_00004

⌈

Kind:	Header File
Syntax:	`ara/shwa/device_monitor.h`
Description:	File contains classes and methods for devices monitoring.

⌋

8.4.1 Class: DeviceMonitor

[AP_SWS_SHWA_01401] Definition of API class ara::shwa::DeviceMonitor

Status: DRAFT
Upstream requirements: RS_SHWA_00004

⌈

Kind:	class
Header file:	#include "ara/shwa/device_monitor.h"
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"
Scope:	namespace ara::shwa
Symbol:	DeviceMonitor
Syntax:	`class DeviceMonitor final {...};`
Description:	DeviceMonitor API class. Designed as copyable since not owning resources.

⌋

8.4.1.1 Private Member Functions

8.4.1.1.1 Special Member Functions

8.4.1.1.1.1 Default Constructor

[AP_SWS_SHWA_01402] Definition of API function ara::shwa::DeviceMonitor::DeviceMonitor

Status: DRAFT
Upstream requirements: RS_SHWA_00004

⌈

Kind:	function
Header file:	#include "ara/shwa/device_monitor.h"
Scope:	`class ara::shwa::DeviceMonitor`
Syntax:	`DeviceMonitor ()=delete;`
Description:	The default constructor for DeviceMonitor shall not be used.
Visibility:	private

⌋

8.4.1.1.1.2 Move Constructor

[AP_SWS_SHWA_01405] Definition of API function ara::shwa::DeviceMonitor::DeviceMonitor

Status: DRAFT
Upstream requirements: RS_SHWA_00004, RS_SHWA_00018

⌈

Kind:	function	
Header file:	#include "ara/shwa/device_monitor.h"	
Scope:	`class ara::shwa::DeviceMonitor`	
Syntax:	`DeviceMonitor (DeviceMonitor &&devMonitor) noexcept=default;`	
Parameters (in):	devMonitor	Rvalue reference to DeviceMonitor instance to move
Exception Safety:	exception safe	
Description:	The default DeviceMonitor move constructor.	
Visibility:	private	

⌋

8.4.1.1.1.3 Copy Assignment Operator

[AP_SWS_SHWA_01404] Definition of API function ara::shwa::DeviceMonitor::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00004

⌈

Kind:	function	
Header file:	#include "ara/shwa/device_monitor.h"	
Scope:	`class ara::shwa::DeviceMonitor`	
Syntax:	`DeviceMonitor & operator= (const DeviceMonitor &devMonitor) noexcept=default;`	
Parameters (in):	devMonitor	Reference to DeviceMonitor instance to assign
Return value:	DeviceMonitor &	A reference to a DeviceMonitor instance
Exception Safety:	exception safe	
Description:	The default DeviceMonitor assignment operator.	
Visibility:	private	

⌋

8.4.1.1.1.4 Move Assignment Operator

[AP_SWS_SHWA_01406] Definition of API function ara::shwa::DeviceMonitor::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00004

⌈

Kind:	function	
Header file:	#include "ara/shwa/device_monitor.h"	
Scope:	`class ara::shwa::DeviceMonitor`	
Syntax:	`DeviceMonitor & operator= (DeviceMonitor &&devMonitor) noexcept=default;`	
Parameters (in):	devMonitor	Rvalue reference to DeviceMonitor instance to perform move assignment
Return value:	DeviceMonitor &	A reference to a DeviceMonitor instance
Exception Safety:	exception safe	
Description:	The default DeviceMonitor move assignment operator.	
Visibility:	private	

⌋

8.4.1.1.2 Constructors

8.4.1.1.2.1 DeviceMonitor

[AP_SWS_SHWA_01403] Definition of API function ara::shwa::DeviceMonitor::DeviceMonitor

Status: DRAFT
Upstream requirements: RS_SHWA_00004

⌈

Kind:	function	
Header file:	#include "ara/shwa/device_monitor.h"	
Scope:	`class ara::shwa::DeviceMonitor`	
Syntax:	`DeviceMonitor (DeviceMonitor const &devMonitor) noexcept=default;`	
Parameters (in):	devMonitor	Reference to DeviceMonitor instance to copy
Exception Safety:	exception safe	
Description:	The default DeviceMonitor copy constructor.	
Visibility:	private	

⌋

8.4.1.1.3 Member Functions

8.4.1.1.3.1 Create

[AP_SWS_SHWA_01407] Definition of API function ara::shwa::DeviceMonitor::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00004, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/device_monitor.h"	
Scope:	`class ara::shwa::DeviceMonitor`	
Syntax:	`static ara::core::Result< DeviceMonitor > Create (const Device &device) noexcept;`	
Parameters (in):	device	Device specifying particular Device for tasks execution
Return value:	ara::core::Result< DeviceMonitor >	DeviceMonitor instance or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::kRuntime	rollback_semantics
		General runtime error
	ShwaErrorCode::kResourceBusy	rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::kInvalidParameter	rollback_semantics
		Invalid argument to an API function
	ShwaErrorCode::kFeatureNotSupported	rollback_semantics
		Used when a requested feature isn't supported
Description:	Factory method for creation of DeviceMonitor instance particular Device.	
Visibility:	private	

⌋

8.4.1.1.3.2 Create

[AP_SWS_SHWA_01408] Definition of API function ara::shwa::DeviceMonitor::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00004, RS_SHWA_00007

⌈

Kind:	function
Header file:	#include "ara/shwa/device_monitor.h"
Scope:	`class ara::shwa::DeviceMonitor`

▽

△

Syntax:	`template <typename DeviceSelector> static ara::core::Result< DeviceMonitor > Create (const DeviceSelector &deviceSelector) noexcept;`	
Template param:	deviceSelector	Device selector specifying type of Device for monitoring
DIRECTION NOT DEFINED	deviceSelector	--
Return value:	ara::core::Result< DeviceMonitor >	DeviceMonitor instance or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::k Runtime	rollback_semantics
		General runtime error
	ShwaErrorCode::k ResourceBusy	rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::kInvalid Parameter	rollback_semantics
		Invalid argument to an API function
	ShwaErrorCode::k FeatureNotSupported	rollback_semantics
		Used when a requested feature isn't supported
Description:	Factory method for creation of DeviceMonitor instance for Device using DeviceSelector.	
Visibility:	private	

⌋

8.4.1.1.3.3 Status

[AP_SWS_SHWA_01409] Definition of API function ara::shwa::DeviceMonitor::Status

Status: DRAFT

Upstream requirements: RS_SHWA_00004

⌈

Kind:	function	
Header file:	#include "ara/shwa/device_monitor.h"	
Scope:	`class ara::shwa::DeviceMonitor`	
Syntax:	`DeviceStatus Status () const noexcept;`	
Return value:	DeviceStatus	Device status
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Method to get actual Device status.	
Visibility:	private	

⌋

8.5 Header: ara/shwa/device_selector.h

[AP_SWS_SHWA_00400] Definition of Header File ara/shwa/device_selector.h

Status: DRAFT
Upstream requirements: RS_SHWA_00006

⌈

Kind:	Header File
Syntax:	`ara/shwa/device_selector.h`
Description:	File contains classes and methods for device selection.

⌋

8.5.1 Class: AcceleratorSelector

[AP_SWS_SHWA_00411] Definition of API class ara::shwa::AcceleratorSelector

Status: DRAFT
Upstream requirements: RS_SHWA_00006

⌈

Kind:	class
Header file:	#include "ara/shwa/device_selector.h"
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"
Scope:	namespace ara::shwa
Symbol:	AcceleratorSelector
Base class:	DeviceSelector
Syntax:	`class AcceleratorSelector : public DeviceSelector {...};`
Description:	AcceleratorSelector manages accelerator selection for operations on HWA if available.

⌋

8.5.1.1 Public Member Functions

8.5.1.1.1 Member Functions

8.5.1.1.1.1 SelectDevice

[AP_SWS_SHWA_00413] Definition of API function ara::shwa::AcceleratorSelector::SelectDevice

Status: DRAFT
Upstream requirements: RS_SHWA_00006, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/device_selector.h"	
Scope:	`class ara::shwa::AcceleratorSelector`	
Syntax:	`virtual ara::core::Result< Device > SelectDevice () const noexcept override;`	
Return value:	ara::core::Result< Device >	The device with the highest rank of the accelerator type.
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k ResourceBusy	no_rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::k FeatureNotSupported	no_rollback_semantics
		Used when a requested feature isn't supported
Description:	Method returns device created as an instance of accelerator device. The device with the highest non-negative value is selected.	

⌋

8.5.1.1.1.2 operator()

[AP_SWS_SHWA_00412] Definition of API function ara::shwa::AcceleratorSelector::operator()

Status: DRAFT
Upstream requirements: RS_SHWA_00006, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/device_selector.h"	
Scope:	`class ara::shwa::AcceleratorSelector`	
Syntax:	`ara::core::Result< int > operator() (const Device &device) const noexcept override;`	
Parameters (in):	device	A device reference to get the rank from
Return value:	ara::core::Result< int >	A Result containing the highest ranked device of the available of the type accelerator. In case of the error returns vendor-specific errors

▽

△

Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k ResourceBusy	no_rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::k FeatureNotSupported	no_rollback_semantics
		Used when a requested feature isn't supported
Description:	Get the highest device rank based on the specified device.	

⌋

8.5.2 Class: CpuSelector

[AP_SWS_SHWA_00408] Definition of API class ara::shwa::CpuSelector

Status: DRAFT
Upstream requirements: RS_SHWA_00006

⌈

Kind:	class
Header file:	#include "ara/shwa/device_selector.h"
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"
Scope:	namespace ara::shwa
Symbol:	CpuSelector
Base class:	DeviceSelector
Syntax:	`class CpuSelector : public DeviceSelector {...};`
Description:	CpuSelector manages CPU selection for operations on HWA if available Allows to get a device from any supported backends for which the device type is CPU.

⌋

8.5.2.1 Public Member Functions

8.5.2.1.1 Member Functions

8.5.2.1.1.1 SelectDevice

[AP_SWS_SHWA_00410] Definition of API function ara::shwa::CpuSelector::SelectDevice

Status: DRAFT
Upstream requirements: RS_SHWA_00006, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/device_selector.h"	
Scope:	`class ara::shwa::CpuSelector`	
Syntax:	`virtual ara::core::Result< Device > SelectDevice () const noexcept override;`	
Return value:	ara::core::Result< Device >	The device with the highest rank of the CPU type
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::kResourceBusy	no_rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::kFeatureNotSupported	no_rollback_semantics
		Used when a requested feature isn't supported
Description:	Method returns device created as an instance of CPU device. The device with the highest non-negative value is selected.	

⌋

8.5.2.1.1.2 operator()

[AP_SWS_SHWA_00409] Definition of API function ara::shwa::CpuSelector::operator()

Status: DRAFT
Upstream requirements: RS_SHWA_00006, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/device_selector.h"	
Scope:	`class ara::shwa::CpuSelector`	
Syntax:	`ara::core::Result< int > operator() (const Device &device) const noexcept override;`	
Parameters (in):	device	A device reference
Return value:	ara::core::Result< int >	A Result containing the highest ranked device of the available of the type CPU. In case of the error returns vendor-specific errors

▽

△

Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k ResourceBusy	no_rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::k FeatureNotSupported	no_rollback_semantics
		Used when a requested feature isn't supported
Description:	Get the highest device rank based on the specified device.	

⌋

8.5.3 Class: DeviceSelector

[AP_SWS_SHWA_00401] Definition of API class ara::shwa::DeviceSelector

Status: DRAFT
Upstream requirements: RS_SHWA_00006

⌈

Kind:	class
Header file:	#include "ara/shwa/device_selector.h"
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"
Scope:	namespace ara::shwa
Symbol:	DeviceSelector
Syntax:	`class DeviceSelector {...};`
Description:	Class allowing to choose the most appropriate device of the available ones. This is a callable object taking a device reference and returning an integer rank. One of the device with the highest non-negative value is selected.

⌋

8.5.3.1 Public Member Functions

8.5.3.1.1 Special Member Functions

8.5.3.1.1.1 Destructor

[AP_SWS_SHWA_00402] Definition of API function ara::shwa::DeviceSelector::~DeviceSelector

Status: DRAFT
Upstream requirements: RS_SHWA_00006

⌈

Kind:	function
Header file:	#include "ara/shwa/device_selector.h"
Scope:	`class ara::shwa::DeviceSelector`
Syntax:	`virtual ~DeviceSelector () noexcept=default;`
Exception Safety:	exception safe
Description:	Virtual Destructor of DeviceSelector class.

⌋

8.5.3.1.2 Member Functions

8.5.3.1.2.1 SelectDevice

[AP_SWS_SHWA_00403] Definition of API function ara::shwa::DeviceSelector::SelectDevice

Status: DRAFT
Upstream requirements: RS_SHWA_00006, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/device_selector.h"	
Scope:	`class ara::shwa::DeviceSelector`	
Syntax:	`virtual ara::core::Result< Device > SelectDevice () const noexcept=0;`	
Return value:	ara::core::Result< Device >	The default device with the highest rank
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k ResourceBusy	no_rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::k FeatureNotSupported	no_rollback_semantics
		Used when a requested feature isn't supported
Description:	Method returns device created as an instance that is a copy of the default device. The device with the highest non-negative value is selected.	

⌋

8.5.3.1.2.2 operator()

[AP_SWS_SHWA_00404] Definition of API function ara::shwa::DeviceSelector::operator()

Status: DRAFT
Upstream requirements: RS_SHWA_00006, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/device_selector.h"	
Scope:	`class ara::shwa::DeviceSelector`	
Syntax:	`virtual ara::core::Result< int > operator() (const Device &device) const noexcept=0;`	
Parameters (in):	device	A device reference
Return value:	ara::core::Result< int >	The highest rank of the specified device
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k ResourceBusy	no_rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::k FeatureNotSupported	no_rollback_semantics
		Used when a requested feature isn't supported
Description:	Method allows to get the highest rank of the provided device.	

⌋

8.5.4 Class: GpuSelector

[AP_SWS_SHWA_00405] Definition of API class ara::shwa::GpuSelector

Status: DRAFT
Upstream requirements: RS_SHWA_00006

⌈

Kind:	class
Header file:	#include "ara/shwa/device_selector.h"
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"
Scope:	namespace ara::shwa
Symbol:	GpuSelector
Base class:	DeviceSelector
Syntax:	`class GpuSelector : public DeviceSelector {...};`
Description:	GpuSelector manages GPU selection for operations on HWA if available Select a device from any supported backends for which the device type is GPU.

⌋

8.5.4.1 Public Member Functions

8.5.4.1.1 Member Functions

8.5.4.1.1.1 SelectDevice

[AP_SWS_SHWA_00407] Definition of API function ara::shwa::GpuSelector::SelectDevice

Status: DRAFT
Upstream requirements: RS_SHWA_00006, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/device_selector.h"	
Scope:	`class ara::shwa::GpuSelector`	
Syntax:	`virtual ara::core::Result< Device > SelectDevice () const noexcept override;`	
Return value:	ara::core::Result< Device >	The device with the highest rank of the GPU type
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::kResourceBusy	no_rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::kFeatureNotSupported	no_rollback_semantics
		Used when a requested feature isn't supported
Description:	Method returns device created as an instance of GPU device. The device with the highest non-negative value is selected.	

⌋

8.5.4.1.1.2 operator()

[AP_SWS_SHWA_00406] Definition of API function ara::shwa::GpuSelector::operator()

Status: DRAFT
Upstream requirements: RS_SHWA_00006, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/device_selector.h"	
Scope:	`class ara::shwa::GpuSelector`	
Syntax:	`ara::core::Result< int > operator() (const Device &device) const noexcept override;`	
Parameters (in):	device	A device reference
Return value:	ara::core::Result< int >	A Result containing the highest ranked device of the available of the type GPU. In case of the error returns vendor-specific errors

▽

△

Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k ResourceBusy	no_rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::k FeatureNotSupported	no_rollback_semantics
		Used when a requested feature isn't supported
Description:	Method allows to get the highest rank fo the specified device.	

⌋

8.6 Header: ara/shwa/error_handler.h

[AP_SWS_SHWA_00500] Definition of Header File ara/shwa/error_handler.h

Status: DRAFT
Upstream requirements: RS_SHWA_00008

⌈

Kind:	Header File
Syntax:	`ara/shwa/error_handler.h`
Description:	File contains classes and methods for error handling.

⌋

8.6.1 Class: ErrorHandler

[AP_SWS_SHWA_00501] Definition of API class ara::shwa::ErrorHandler

Status: DRAFT
Upstream requirements: RS_SHWA_00008

⌈

Kind:	class
Header file:	#include "ara/shwa/error_handler.h"
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"
Scope:	namespace ara::shwa
Symbol:	ErrorHandler
Syntax:	`class ErrorHandler final {...};`
Description:	ErrorHandler manages asynchronous error handling for operations on HWA.

⌋

8.6.1.1 Public Member Functions

8.6.1.1.1 Special Member Functions

8.6.1.1.1.1 Default Constructor

[AP_SWS_SHWA_00502] Definition of API function ara::shwa::ErrorHandler::ErrorHandler

Status: DRAFT
Upstream requirements: RS_SHWA_00008

⌈

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

⌋

8.6.1.1.1.2 Move Constructor

[AP_SWS_SHWA_00505] Definition of API function ara::shwa::ErrorHandler::ErrorHandler

Status: DRAFT
Upstream requirements: RS_SHWA_00008

⌈

Kind:	function
Header file:	#include "ara/shwa/error_handler.h"
Scope:	`class ara::shwa::ErrorHandler`
Syntax:	`ErrorHandler (ErrorHandler &&)=delete;`
Description:	The default assignment operator for ErrorHandler shall not be used.

⌋

8.6.1.1.1.3 Copy Assignment Operator

[AP_SWS_SHWA_00504] Definition of API function ara::shwa::ErrorHandler::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00008

⌈

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

⌋

8.6.1.1.1.4 Move Assignment Operator

[AP_SWS_SHWA_00506] Definition of API function ara::shwa::ErrorHandler::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00008

⌈

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

⌋

8.6.1.1.2 Constructors

8.6.1.1.2.1 ErrorHandler

[AP_SWS_SHWA_00503] Definition of API function ara::shwa::ErrorHandler::ErrorHandler

Status: DRAFT
Upstream requirements: RS_SHWA_00008

⌈

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

⌋

8.6.1.1.3 Member Functions

8.6.1.1.3.1 Create

[AP_SWS_SHWA_00507] Definition of API function ara::shwa::ErrorHandler::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00008, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/error_handler.h"	
Scope:	`class ara::shwa::ErrorHandler`	
Syntax:	`static ara::core::Result< std::unique_ptr< ErrorHandler > > Create (std::function< void(ara::core::ErrorCode)> handler) noexcept;`	
Parameters (in):	handler	a handler to be used for handling the errors
Return value:	ara::core::Result< std::unique_ptr< Error Handler > >	ErrorHandler instance or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	This function does not specify any standardized errors.	
Description:	Factory method for creation of ErrorHandler instance.	

⌋

8.6.1.1.3.2 HandleError

[AP_SWS_SHWA_00508] Definition of API function ara::shwa::ErrorHandler::HandleError

Status: DRAFT
Upstream requirements: RS_SHWA_00008

⌈

Kind:	function	
Header file:	#include "ara/shwa/error_handler.h"	
Scope:	`class ara::shwa::ErrorHandler`	
Syntax:	`void HandleError (ara::shwa::ShwaErrorCode errorCode);`	
Parameters (in):	errorCode	code of the specific error to be handled
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	not thread-safe	
Description:	Mehod for the specific handling of the error specified as parameter.	

⌋

8.7 Header: ara/shwa/event.h

[AP_SWS_SHWA_00600] Definition of Header File ara/shwa/event.h

Status: DRAFT
Upstream requirements: RS_SHWA_00016

⌈

Kind:	Header File
Syntax:	`ara/shwa/event.h`
Description:	File contains classes and methods for Events handling

⌋

8.7.1 Class: Event

[AP_SWS_SHWA_00601] Definition of API class ara::shwa::Event

Status: DRAFT
Upstream requirements: RS_SHWA_00016

⌈

Kind:	class
Header file:	#include "ara/shwa/event.h"
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"

▽

△

Scope:	namespace ara::shwa
Symbol:	Event
Syntax:	`class Event final {...};`
Description:	Event class helps to manage task execution synchronization between HWA and Host application. Can be created only inside of the Queue otherwise it's meaningless.

⌋

8.7.1.1 Public Member Functions

8.7.1.1.1 Member Functions

8.7.1.1.1.1 Create

[AP_SWS_SHWA_00602] Definition of API function ara::shwa::Event::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00007, RS_SHWA_00016

⌈

Kind:	function	
Header file:	#include "ara/shwa/event.h"	
Scope:	`class ara::shwa::Event`	
Syntax:	`static ara::core::Result< Event > Create () noexcept;`	
Return value:	ara::core::Result< Event >	Event instance or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::kRuntime	rollback_semantics
		General runtime error
Description:	Factory method for creation of Event instance.	

⌋

8.7.1.1.1.2 GetWaitList

[AP_SWS_SHWA_00603] Definition of API function ara::shwa::Event::GetWaitList

Status: DRAFT
Upstream requirements: RS_SHWA_00007, RS_SHWA_00016

⌈

Kind:	function
Header file:	#include "ara/shwa/event.h"
Scope:	`class ara::shwa::Event`

▽

△

Syntax:	`ara::core::Result< ara::core::Vector< Event > > GetWaitList () noexcept;`	
Return value:	ara::core::Result< ara::core::Vector< Event > >	Event wait-list vector or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::kRuntime	rollback_semantics
		General runtime error
	ShwaErrorCode::kEventInvalidOperation	rollback_semantics
		Error related to events (e.g., invalid wait on event)
Description:	Gets vector of events that current event waits for in the dependence graph.	

⌋

8.7.1.1.1.3 Wait

[AP_SWS_SHWA_00605] Definition of API function ara::shwa::Event::Wait

Status: DRAFT
Upstream requirements: RS_SHWA_00007, RS_SHWA_00016

⌈

Kind:	function	
Header file:	#include "ara/shwa/event.h"	
Scope:	`class ara::shwa::Event`	
Syntax:	`static ara::core::Result< void > Wait (const ara::core::Vector< Event > &eventList) noexcept;`	
Parameters (in):	eventList	Event wait-list vector
Return value:	ara::core::Result< void >	Void or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::kRuntime	no_rollback_semantics
		General runtime error. The data used within failed task execution should be considered as changed in unintended way
	ShwaErrorCode::kTaskExecutionFailure	no_rollback_semantics
		Error during task execution. The data used within failed task execution should be considered as changed in unintended way
	ShwaErrorCode::kEventInvalidOperation	no_rollback_semantics
		Error related to events (e.g., invalid wait on event)
Description:	Behaves as if calling Wait method on each event in eventList.	

⌋

8.7.1.1.1.4 Wait

[AP_SWS_SHWA_00604] Definition of API function ara::shwa::Event::Wait

Status: DRAFT
Upstream requirements: RS_SHWA_00007, RS_SHWA_00016

⌈

Kind:	function	
Header file:	#include "ara/shwa/event.h"	
Scope:	`class ara::shwa::Event`	
Syntax:	`ara::core::Result< void > Wait () noexcept;`	
Return value:	ara::core::Result< void >	Void or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k Runtime	no_rollback_semantics
		General runtime error. The data used within failed task execution should be considered as changed in unintended way
	ShwaErrorCode::kTask ExecutionFailure	no_rollback_semantics
		Error during task execution. The data used within failed task execution should be considered as changed in unintended way
	ShwaErrorCode::kEvent InvalidOperation	no_rollback_semantics
		Error related to events (e.g., invalid wait on event)
Description:	Blocks until all commands associated with this event and any dependent events have completed.	

⌋

8.7.1.1.1.5 WaitFor

[AP_SWS_SHWA_00606] Definition of API function ara::shwa::Event::WaitFor

Status: DRAFT
Upstream requirements: RS_SHWA_00005, RS_SHWA_00007, RS_SHWA_00016

⌈

Kind:	function	
Header file:	#include "ara/shwa/event.h"	
Scope:	`class ara::shwa::Event`	
Syntax:	`static ara::core::Result< void > WaitFor (const size_t ms) noexcept;`	
Parameters (in):	ms	Duration of the wait in milliseconds
Return value:	ara::core::Result< void >	Void or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k Runtime	rollback_semantics
		General runtime error. The data used within failed task execution should be considered as changed in unintended way
	ShwaErrorCode::kEvent InvalidOperation	no_rollback_semantics
		Error related to events (e.g., invalid wait on event)

▽

△

	ShwaErrorCode::kTask ExecutionFailure	no_rollback_semantics
		Error during task execution. The data used within failed task execution should be considered as changed in unintended way
	ShwaErrorCode::kTimed Out	no_rollback_semantics
		Task execution timed-out. The data used within failed task execution should be considered as changed in unintended way
Description:	Behaves as if calling time-constrained Wait method on each event in eventList.	

⌋

8.8 Header: ara/shwa/id.h

[AP_SWS_SHWA_00700] Definition of Header File ara/shwa/id.h

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	Header File
Syntax:	`ara/shwa/id.h`
Description:	File contains classes and methods for identifiers handling

⌋

8.8.1 Class: Id

[AP_SWS_SHWA_00701] Definition of API class ara::shwa::Id

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	class	
Header file:	#include "ara/shwa/id.h"	
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"	
Scope:	namespace ara::shwa	
Symbol:	Id	
Syntax:	`template <int Dimensions = 1>` `class Id final {...};`	
Template param:	int Dimensions = 1	--
Description:	Id API class. Designed as copyable since not owning resources.	

⌋

8.8.1.1 Public Member Functions

8.8.1.1.1 Special Member Functions

8.8.1.1.1.1 Default Constructor

[AP_SWS_SHWA_00702] Definition of API function ara::shwa::Id::Id

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

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

⌋

8.8.1.1.1.2 Move Constructor

[AP_SWS_SHWA_00705] Definition of API function ara::shwa::Id::Id

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function	
Header file:	#include "ara/shwa/id.h"	
Scope:	`class ara::shwa::Id`	
Syntax:	`Id (Id &&other) noexcept=default;`	
Parameters (in):	other	Rvalue reference to Id instance to move
Exception Safety:	exception safe	
Description:	The default Id move constructor.	

⌋

8.8.1.1.1.3 Move Assignment Operator

[AP_SWS_SHWA_00706] Definition of API function ara::shwa::Id::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function	
Header file:	#include "ara/shwa/id.h"	
Scope:	`class ara::shwa::Id`	
Syntax:	`Id & operator= (Id &&other) noexcept=default;`	
Parameters (in):	other	Rvalue reference to Id instance to perform move assignment
Return value:	Id &	A reference to a Id instance
Exception Safety:	exception safe	
Description:	The default Id move assignment operator.	

⌋

8.8.1.1.1.4 Copy Assignment Operator

[AP_SWS_SHWA_00704] Definition of API function ara::shwa::Id::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function	
Header file:	#include "ara/shwa/id.h"	
Scope:	`class ara::shwa::Id`	
Syntax:	`Id & operator= (const Id &other) noexcept=default;`	
Parameters (in):	other	Reference to Id instance to assign
Return value:	Id &	A reference to a Id instance
Exception Safety:	exception safe	
Description:	The default Id assignment operator.	

⌋

8.8.1.1.2 Constructors

8.8.1.1.2.1 Id

[AP_SWS_SHWA_00703] Definition of API function ara::shwa::Id::Id

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function	
Header file:	#include "ara/shwa/id.h"	
Scope:	`class ara::shwa::Id`	
Syntax:	`Id (Id const &other) noexcept=default;`	
Parameters (in):	other	Reference to Id instance to copy
Exception Safety:	exception safe	
Description:	The default Id copy constructor.	

⌋

8.8.1.1.3 Member Functions

8.8.1.1.3.1 Create

[AP_SWS_SHWA_00707] Definition of API function ara::shwa::Id::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00017, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/id.h"	
Scope:	`class ara::shwa::Id`	
Syntax:	`static ara::core::Result< Id > Create (const std::size_t dim0) noexcept;`	
Parameters (in):	dim0	Size of first dimension
Return value:	ara::core::Result< Id >	Id instance or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::k Runtime	rollback_semantics
		General runtime error
Description:	Factory method for creation of one-dimentional Id.	

⌋

8.8.1.1.3.2 Create

[AP_SWS_SHWA_00709] Definition of API function ara::shwa::Id::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00017, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/id.h"	
Scope:	`class ara::shwa::Id`	
Syntax:	`static ara::core::Result< Id > Create (std::size_t dim0, std::size_t dim1, std::size_t dim2) noexcept;`	
Parameters (in):	dim0	Size of first dimension
	dim1	Size of second dimension
	dim2	Size of third dimension
Return value:	ara::core::Result< Id >	Id instance or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k Runtime	rollback_semantics
		General runtime error
Description:	Factory method for creation of three-dimentional Id.	

⌋

8.8.1.1.3.3 Create

[AP_SWS_SHWA_00708] Definition of API function ara::shwa::Id::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00017, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/id.h"	
Scope:	`class ara::shwa::Id`	
Syntax:	`static ara::core::Result< Id > Create (std::size_t dim0, std::size_t dim1) noexcept;`	
Parameters (in):	dim0	Size of first dimension
	dim1	Size of second dimension
Return value:	ara::core::Result< Id >	Id instance or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k Runtime	rollback_semantics
		General runtime error
Description:	Factory method for creation of two-dimentional Id.	

⌋

8.8.1.1.3.4 Get

[AP_SWS_SHWA_00710] Definition of API function ara::shwa::Id::Get

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function	
Header file:	#include "ara/shwa/id.h"	
Scope:	`class ara::shwa::Id`	
Syntax:	`size_t Get (int dimension) const;`	
Parameters (in):	dimension	A Dimension value
Return value:	size_t	Current index value for selected dimension
Exception Safety:	not exception safe	
Thread Safety:	not thread-safe	
Description:	Method for getting current Id dimension.	

⌋

8.8.1.1.3.5 Set

[AP_SWS_SHWA_00711] Definition of API function ara::shwa::Id::Set

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function	
Header file:	#include "ara/shwa/id.h"	
Scope:	`class ara::shwa::Id`	
Syntax:	`void Set (int dimension, size_t index);`	
Parameters (in):	dimension	A dimension value
	index	Index value for selected dimension
Return value:	None	
Exception Safety:	not exception safe	
Thread Safety:	not thread-safe	
Description:	Method for setting current Id dimension.	

⌋

8.9 Header: ara/shwa/platform.h

[AP_SWS_SHWA_01100] Definition of Header File ara/shwa/platform.h

Status: DRAFT
Upstream requirements: RS_SHWA_00018

⌈

Kind:	Header File
Syntax:	`ara/shwa/platform.h`
Description:	File contains classes and methods for handling platforms, high-level representations of the underlying hardware and software environment.

⌋

8.9.1 Class: Platform

[AP_SWS_SHWA_01101] Definition of API class ara::shwa::Platform

Status: DRAFT
Upstream requirements: RS_SHWA_00018

⌈

Kind:	class
Header file:	#include "ara/shwa/platform.h"
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"
Scope:	namespace ara::shwa
Symbol:	Platform
Syntax:	`class Platform final {...};`
Description:	Platform class is responsible for extracting different HWA platforms available for current application and listing Devices bind to each platform. Designed as copyable since not owning resources.

⌋

8.9.1.1 Private Member Functions

8.9.1.1.1 Special Member Functions

8.9.1.1.1.1 Default Constructor

[AP_SWS_SHWA_01102] Definition of API function ara::shwa::Platform::Platform

Status: DRAFT
Upstream requirements: RS_SHWA_00018

⌈

Kind:	function
Header file:	#include "ara/shwa/platform.h"
Scope:	`class ara::shwa::Platform`
Syntax:	`Platform ()=delete;`
Description:	The default constructor for Platform shall not be used.
Visibility:	private

⌋

8.9.1.1.1.2 Move Constructor

[AP_SWS_SHWA_01105] Definition of API function ara::shwa::Platform::Platform

Status: DRAFT
Upstream requirements: RS_SHWA_00018

⌈

Kind:	function	
Header file:	#include "ara/shwa/platform.h"	
Scope:	`class ara::shwa::Platform`	
Syntax:	`Platform (Platform &&ptfm) noexcept=default;`	
Parameters (in):	ptfm	Rvalue reference to Platform instance to move
Exception Safety:	exception safe	
Description:	The default Platform move constructor.	
Visibility:	private	

⌋

8.9.1.1.1.3 Copy Assignment Operator

[AP_SWS_SHWA_01104] Definition of API function ara::shwa::Platform::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00018

⌈

Kind:	function	
Header file:	#include "ara/shwa/platform.h"	
Scope:	`class ara::shwa::Platform`	
Syntax:	`Platform & operator= (const Platform &ptfm) noexcept=default;`	
Parameters (in):	ptfm	Reference to Platform instance to assign
Return value:	Platform &	A reference to a Platform instance
Exception Safety:	exception safe	
Description:	The default Platform assignment operator.	
Visibility:	private	

⌋

8.9.1.1.1.4 Move Assignment Operator

[AP_SWS_SHWA_01106] Definition of API function ara::shwa::Platform::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00018

⌈

Kind:	function	
Header file:	#include "ara/shwa/platform.h"	
Scope:	`class ara::shwa::Platform`	
Syntax:	`Platform & operator= (Platform &&ptfm) noexcept=default;`	
Parameters (in):	ptfm	Rvalue reference to Platform instance to perform move assignment
Return value:	Platform &	A reference to a Platform instance
Exception Safety:	exception safe	
Description:	The default Platform move assignment operator.	
Visibility:	private	

⌋

8.9.1.1.2 Constructors

8.9.1.1.2.1 Platform

[AP_SWS_SHWA_01103] Definition of API function ara::shwa::Platform::Platform

Status: DRAFT
Upstream requirements: RS_SHWA_00018

⌈

Kind:	function	
Header file:	#include "ara/shwa/platform.h"	
Scope:	`class ara::shwa::Platform`	
Syntax:	`Platform (Platform const &ptfm) noexcept=default;`	
Parameters (in):	ptfm	Reference to Platform instance to copy
Exception Safety:	exception safe	
Description:	The default Platform copy constructor.	
Visibility:	private	

⌋

8.9.1.1.3 Member Functions

8.9.1.1.3.1 GetDevices

[AP_SWS_SHWA_01108] Definition of API function ara::shwa::Platform::GetDevices

Status: DRAFT
Upstream requirements: RS_SHWA_00018, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/platform.h"	
Scope:	`class ara::shwa::Platform`	
Syntax:	`ara::core::Result< ara::core::Vector< Device > > GetDevices () noexcept;`	
Return value:	ara::core::Result< ara::core::Vector< Device > >	Vector of unique pointers to Device objects or an Error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::kRuntime	rollback_semantics
		General runtime error
	ShwaErrorCode::kResourceBusy	no_rollback_semantics
		The operation could not be performed because the resource is currently busy
Description:	Creates vector of unique pointers to Device objects.	

▽

△

Visibility:	private

⌋

8.9.1.1.3.2 GetPlatforms

[AP_SWS_SHWA_01107] Definition of API function ara::shwa::Platform::GetPlatforms

Status: DRAFT
Upstream requirements: RS_SHWA_00018, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/platform.h"	
Scope:	`class ara::shwa::Platform`	
Syntax:	`static ara::core::Result< ara::core::Vector< Platform > > GetPlatforms () noexcept;`	
Return value:	ara::core::Result< ara::core::Vector< Platform > >	Vector of Platform objects or an Error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::kRuntime	rollback_semantics
		General runtime error
	ShwaErrorCode::kResourceBusy	no_rollback_semantics
		The operation could not be performed because the resource is currently busy
Description:	Creates vector of Platform objects.	
Visibility:	private	

⌋

8.10 Header: ara/shwa/property_list.h

[AP_SWS_SHWA_00800] Definition of Header File ara/shwa/property_list.h

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00014

⌈

Kind:	Header File
Syntax:	`ara/shwa/property_list.h`
Description:	File contains classes and methods for handling properties list for objects creation

⌋

8.10.1 Class: PropertyList

[AP_SWS_SHWA_00801] Definition of API class ara::shwa::PropertyList

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00014

⌈

Kind:	class
Header file:	#include "ara/shwa/property_list.h"
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"
Scope:	namespace ara::shwa
Symbol:	PropertyList
Syntax:	`class PropertyList final {...};`
Description:	Property List class is designed to enable vendor specific SHWA functionality extension and adjustment. Properties are not standardized and can be defined by Vendor. Class using Property list is obliged to have HasProperty() and GetProperty() methods defined.

⌋

8.10.1.1 Public Member Functions

8.10.1.1.1 Constructors

8.10.1.1.1.1 PropertyList

[AP_SWS_SHWA_00802] Definition of API function ara::shwa::PropertyList::PropertyList

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00007, RS_SHWA_00014

⌈

Kind:	function	
Header file:	#include "ara/shwa/property_list.h"	
Scope:	`class ara::shwa::PropertyList`	
Syntax:	`template <typename... Properties>` `PropertyList (Properties... props) noexcept;`	
Parameters (in):	props	List of property objects
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Property list constructor, takes various property objects as input.	

⌋

8.11 Header: ara/shwa/queue.h

[AP_SWS_SHWA_00900] Definition of Header File ara/shwa/queue.h

Status: DRAFT
Upstream requirements: RS_SHWA_00001

⌈

Kind:	Header File
Syntax:	`ara/shwa/queue.h`
Description:	File contains classes and methods for handling queues of the tasks for execution

⌋

8.11.1 Class: Queue

[AP_SWS_SHWA_00901] Definition of API class ara::shwa::Queue

Status: DRAFT
Upstream requirements: RS_SHWA_00001

⌈

Kind:	class
Header file:	#include "ara/shwa/queue.h"
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"
Scope:	namespace ara::shwa
Symbol:	Queue
Syntax:	`class Queue final {...};`
Description:	Queue API provides the functionality for parallel processing on selected Device. Designed as not copyable, since it's owning resources. Intended to be managed exclusively by std::unique_ptr, which may be moved.

⌋

8.11.1.1 Public Member Functions

8.11.1.1.1 Special Member Functions

8.11.1.1.1.1 Move Constructor

[AP_SWS_SHWA_00905] Definition of API function ara::shwa::Queue::Queue

Status: DRAFT
Upstream requirements: RS_SHWA_00001

⌈

Kind:	function
Header file:	#include "ara/shwa/queue.h"
Scope:	`class ara::shwa::Queue`
Syntax:	`Queue (Queue &&)=delete;`
Description:	The default move constructor for Queue shall not be used.

⌋

8.11.1.1.1.2 Default Constructor

[AP_SWS_SHWA_00902] Definition of API function ara::shwa::Queue::Queue

Status: DRAFT
Upstream requirements: RS_SHWA_00001

⌈

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

⌋

8.11.1.1.1.3 Move Assignment Operator

[AP_SWS_SHWA_00906] Definition of API function ara::shwa::Queue::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00001

⌈

Kind:	function
Header file:	#include "ara/shwa/queue.h"
Scope:	`class ara::shwa::Queue`

▽

△

Syntax:	`Queue & operator= (Queue &&)=delete;`
Description:	The default move assignment operator for Queue shall not be used.

⌋

8.11.1.1.1.4 Copy Assignment Operator

[AP_SWS_SHWA_00904] Definition of API function ara::shwa::Queue::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00001

⌈

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

⌋

8.11.1.1.2 Constructors

8.11.1.1.2.1 Queue

[AP_SWS_SHWA_00903] Definition of API function ara::shwa::Queue::Queue

Status: DRAFT
Upstream requirements: RS_SHWA_00001

⌈

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

⌋

8.11.1.1.3 Member Functions

8.11.1.1.3.1 Create

[AP_SWS_SHWA_00909] Definition of API function ara::shwa::Queue::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/queue.h"	
Scope:	`class ara::shwa::Queue`	
Syntax:	`static ara::core::Result< std::unique_ptr< Queue > > Create (const Device &device, const std::unique_ptr< ErrorHandler > &errorHandler, const PropertyList &propList={}) noexcept;`	
Parameters (in):	device	Device specifying particular Device for tasks execution
	errorHandler	Functional object to handle asynchronous errors in runtime
	propList	Optional property list
Return value:	ara::core::Result< std::unique_ptr< Queue > >	Queue instance or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::kRuntime	rollback_semantics
		General runtime error
	ShwaErrorCode::kResourceBusy	rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::kInvalidParameter	rollback_semantics
		Invalid argument to an API function
	ShwaErrorCode::kFeatureNotSupported	rollback_semantics
		Used when a requested feature isn't supported
Description:	Factory method for creation of Queue instance.	

⌋

8.11.1.1.3.2 Create

[AP_SWS_SHWA_00908] Definition of API function ara::shwa::Queue::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00007

⌈

Kind:	function
Header file:	#include "ara/shwa/queue.h"
Scope:	`class ara::shwa::Queue`

▽

△

Syntax:	`template <typename DeviceSelector> static ara::core::Result< std::unique_ptr< Queue > > Create (const DeviceSelector &deviceSelector, const std::unique_ptr< ErrorHandler > &errorHandler, const PropertyList &propList={}) noexcept;`	
Parameters (in):	deviceSelector	Device selector specifying type of Device for tasks execution
	errorHandler	Functional object to handle asynchronous errors in runtime
	propList	Optional property list
Return value:	ara::core::Result< std::unique_ptr< Queue > >	Queue instance or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::kRuntime	rollback_semantics
		General runtime error
	ShwaErrorCode::kResourceBusy	rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::kInvalidParameter	rollback_semantics
		Invalid argument to an API function
	ShwaErrorCode::kFeatureNotSupported	rollback_semantics
		Used when a requested feature isn't supported
Description:	Factory method for creation of Queue instance.	

⌋

8.11.1.1.3.3 GetDevice

[AP_SWS_SHWA_00910] Definition of API function ara::shwa::Queue::GetDevice

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/queue.h"	
Scope:	`class ara::shwa::Queue`	
Syntax:	`ara::core::Result< Device > GetDevice () const noexcept;`	
Return value:	ara::core::Result< Device >	Device instance or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::kRuntime	rollback_semantics
		General runtime error
	ShwaErrorCode::kResourceBusy	rollback_semantics
		The operation could not be performed because the resource is currently busy
Description:	Method to get the Device binded to current Queue.	

⌋

8.11.1.1.3.4 GetProperty

[AP_SWS_SHWA_00916] Definition of API function ara::shwa::Queue::GetProperty

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/queue.h"	
Scope:	`class ara::shwa::Queue`	
Syntax:	`template <typename PropertyT>` `ara::core::Result< PropertyT > GetProperty () const noexcept;`	
Template param:	Property	Property type
Return value:	ara::core::Result< PropertyT >	Property object or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k FeatureNotSupported	rollback_semantics
		Used when a requested feature isn't supported
Description:	Getting property from current Queue object.	

⌋

8.11.1.1.3.5 HasProperty

[AP_SWS_SHWA_00915] Definition of API function ara::shwa::Queue::HasProperty

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/queue.h"	
Scope:	`class ara::shwa::Queue`	
Syntax:	`template <typename PropertyT>` `bool HasProperty () const noexcept;`	
Template param:	Property	Property type
Return value:	bool	TRUE if property is available and FALSE otherwise
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Checking if property is available for current Queue object.	

⌋

8.11.1.1.3.6 IsInOrder

[AP_SWS_SHWA_00911] Definition of API function ara::shwa::Queue::IsInOrder

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/queue.h"	
Scope:	`class ara::shwa::Queue`	
Syntax:	`ara::core::Result< bool > IsInOrder () const noexcept;`	
Return value:	ara::core::Result< bool >	Boolean value instance or an error. TRUE if Queue is ordered and FALSE if not
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k Runtime	rollback_semantics
		General runtime error
Description:	Method to get an information if Queue tasks execution is ordered or will be managed using Events.	

⌋

8.11.1.1.3.7 Submit

[AP_SWS_SHWA_00913] Definition of API function ara::shwa::Queue::Submit

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00007, RS_SHWA_00008

⌈

Kind:	function	
Header file:	#include "ara/shwa/queue.h"	
Scope:	`class ara::shwa::Queue`	
Syntax:	`template <typename T>` `ara::core::Result< Event > Submit (T cfg, const Queue &secondaryQueue) noexcept;`	
Parameters (in):	cfg	Lambda expression with task to execute
	secondaryQueue	Secondary Queue to execute task if main one is not able to execute it
Return value:	ara::core::Result< Event >	Event instance corresponding to current task or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k Runtime	no_rollback_semantics
		General runtime error. The data used within failed task execution should be considered as changed in unintended way
	ShwaErrorCode::k ResourceBusy	no_rollback_semantics

▽

△

		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::kInvalid Parameter	no_rollback_semantics
		Invalid argument to an API function
	ShwaErrorCode::kInvalid Object	no_rollback_semantics
		Using a destroyed or invalid object (like a buffer or queue)
	ShwaErrorCode::kTask ExecutionFailure	no_rollback_semantics
		Error during task execution. The data used within failed task execution should be considered as changed in unintended way
Description:	Submits a task to the Device for execution. If there will be an errors occured during task submition to the Queue, current task will be submitted to the secondary Queue.	

⌋

8.11.1.1.3.8 Submit

[AP_SWS_SHWA_00912] Definition of API function ara::shwa::Queue::Submit

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00007, RS_SHWA_00009

⌈

Kind:	function	
Header file:	#include "ara/shwa/queue.h"	
Scope:	`class ara::shwa::Queue`	
Syntax:	`template <typename T> ara::core::Result< Event > Submit (T cfg, ara::shwa::ErrorHandler &&handler) noexcept;`	
Parameters (in):	cfg	Lambda expression with task to execute
	handler	Error handler to be used
Return value:	ara::core::Result< Event >	Event instance or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::k Runtime	no_rollback_semantics
		General runtime error. The data used within failed task execution should be considered as changed in unintended way
	ShwaErrorCode::k ResourceBusy	no_rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::kInvalid Parameter	no_rollback_semantics
		Invalid argument to an API function
	ShwaErrorCode::kInvalid Object	no_rollback_semantics
		Using a destroyed or invalid object (like a buffer or queue)
	ShwaErrorCode::kTask ExecutionFailure	no_rollback_semantics
		Error during task execution. The data used within failed task execution should be considered as changed in unintended way

▽

△

Description:	Submits a task to the Device for execution.

⌋

8.11.1.1.3.9 Wait

[AP_SWS_SHWA_00914] Definition of API function ara::shwa::Queue::Wait

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00007, RS_SHWA_00009

⌈

Kind:	function	
Header file:	#include "ara/shwa/queue.h"	
Scope:	`class ara::shwa::Queue`	
Syntax:	`ara::core::Result< void > Wait () noexcept;`	
Return value:	ara::core::Result< void >	Void or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::kRuntime	no_rollback_semantics
		General runtime error. The data used within failed task execution should be considered as changed in unintended way
	ShwaErrorCode::kResourceBusy	rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::kInvalidObject	rollback_semantics
		Using a destroyed or invalid object (like a buffer or queue)
	ShwaErrorCode::kFeatureNotSupported	rollback_semantics
		Used when a requested feature isn't supported
	ShwaErrorCode::kTaskExecutionFailure	no_rollback_semantics
		Error during task execution. The data used within failed task execution should be considered as changed in unintended way
Description:	Performs a blocking wait for the completion of all enqueued tasks in the Queue.	

⌋

8.11.1.1.3.10 WaitFor

[AP_SWS_SHWA_00917] Definition of API function ara::shwa::Queue::WaitFor

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00005, RS_SHWA_00007, RS_SHWA_00009

⌈

Kind:	function	
Header file:	#include "ara/shwa/queue.h"	
Scope:	`class ara::shwa::Queue`	
Syntax:	`ara::core::Result< void > WaitFor (const size_t ms) noexcept;`	
DIRECTION NOT DEFINED	ms	--
Return value:	ara::core::Result< void >	Void or an error
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Errors:	ShwaErrorCode::kRuntime	no_rollback_semantics
		General runtime error. The data used within failed task execution should be considered as changed in unintended way
	ShwaErrorCode::kResourceBusy	rollback_semantics
		The operation could not be performed because the resource is currently busy
	ShwaErrorCode::kInvalidObject	rollback_semantics
		Using a destroyed or invalid object (like a buffer or queue)
	ShwaErrorCode::kFeatureNotSupported	rollback_semantics
		Used when a requested feature isn't supported
	ShwaErrorCode::kTaskExecutionFailure	no_rollback_semantics
		Error during task execution. The data used within failed task execution should be considered as changed in unintended way
	ShwaErrorCode::kTimedOut	no_rollback_semantics
		Task execution timed-out. The data used within failed task execution should be considered as changed in unintended way
Description:	Performs a time-constrained wait for the completion of all enqueued tasks in the Queue.	

⌋

8.12 Header: ara/shwa/range.h

[AP_SWS_SHWA_01000] Definition of Header File ara/shwa/range.h

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	Header File
Syntax:	`ara/shwa/range.h`

▽

△

Description:	File contains classes and methods for handling ranges of the memory buffers

⌋

8.12.1 Class: Range

[AP_SWS_SHWA_01001] Definition of API class ara::shwa::Range

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	class	
Header file:	#include "ara/shwa/range.h"	
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"	
Scope:	namespace ara::shwa	
Symbol:	Range	
Syntax:	`template <int Dimensions = 1>` `class Range final {...};`	
Template param:	int Dimensions = 1	Dimensions of the underlying data.
Description:	Defines size of Buffer for one, two or three dimensions. Designed as copyable since not owning resources.	

⌋

8.12.1.1 Public Member Functions

8.12.1.1.1 Special Member Functions

8.12.1.1.1.1 Default Constructor

[AP_SWS_SHWA_01002] Definition of API function ara::shwa::Range::Range

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

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

⌋

8.12.1.1.1.2 Move Constructor

[AP_SWS_SHWA_01005] Definition of API function ara::shwa::Range::Range

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function	
Header file:	#include "ara/shwa/range.h"	
Scope:	`class ara::shwa::Range`	
Syntax:	`Range (Range &&other) noexcept=default;`	
Parameters (in):	other	Rvalue reference to Range instance to move
Exception Safety:	exception safe	
Description:	The default Range move constructor.	

⌋

8.12.1.1.1.3 Move Assignment Operator

[AP_SWS_SHWA_01006] Definition of API function ara::shwa::Range::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function	
Header file:	#include "ara/shwa/range.h"	
Scope:	`class ara::shwa::Range`	
Syntax:	`Range & operator= (Range &&other) noexcept=default;`	
Parameters (in):	other	Rvalue reference to Range instance to perform move assignment
Return value:	Range &	returns reference to a Range instance
Exception Safety:	exception safe	
Description:	The default Range move assignment operator.	

⌋

8.12.1.1.1.4 Copy Assignment Operator

[AP_SWS_SHWA_01004] Definition of API function ara::shwa::Range::operator=

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function	
Header file:	#include "ara/shwa/range.h"	
Scope:	`class ara::shwa::Range`	
Syntax:	`Range & operator= (const Range &other) noexcept=default;`	
Parameters (in):	other	Reference to Range instance to assign
Return value:	Range &	returns reference to a Range instance
Exception Safety:	exception safe	
Description:	The default Range assignment operator.	

⌋

8.12.1.1.2 Constructors

8.12.1.1.2.1 Range

[AP_SWS_SHWA_01003] Definition of API function ara::shwa::Range::Range

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function	
Header file:	#include "ara/shwa/range.h"	
Scope:	`class ara::shwa::Range`	
Syntax:	`Range (Range const &other) noexcept=default;`	
Parameters (in):	other	Reference to Range instance to copy
Exception Safety:	exception safe	
Description:	The default Range copy constructor.	

⌋

8.12.1.1.3 Member Functions

8.12.1.1.3.1 Create

[AP_SWS_SHWA_01007] Definition of API function ara::shwa::Range::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00017, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/range.h"	
Scope:	`class ara::shwa::Range`	
Syntax:	`static ara::core::Result< Range > Create (const std::size_t dim0) noexcept;`	
Parameters (in):	dim0	Size of first dimension
Return value:	ara::core::Result< Range >	Range instance or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::k Runtime	rollback_semantics
		General runtime error
Description:	Factory method for creation of one-dimensional Range.	

⌋

8.12.1.1.3.2 Create

[AP_SWS_SHWA_01009] Definition of API function ara::shwa::Range::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00017, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/range.h"	
Scope:	`class ara::shwa::Range`	
Syntax:	`static ara::core::Result< Range > Create (std::size_t dim0, std::size_t dim1, std::size_t dim2) noexcept;`	
Parameters (in):	dim0	Size of first dimension
	dim1	Size of second dimension
	dim2	Size of third dimension
Return value:	ara::core::Result< Range >	Range instance or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::k Runtime	rollback_semantics
		General runtime error

▽

△

Description:	Factory method for creation of three-dimentional Range.

⌋

8.12.1.1.3.3 Create

[AP_SWS_SHWA_01008] Definition of API function ara::shwa::Range::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00017, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/range.h"	
Scope:	`class ara::shwa::Range`	
Syntax:	`static ara::core::Result< Range > Create (std::size_t dim0, std::size_t dim1) noexcept;`	
Parameters (in):	dim0	Size of first dimension
	dim1	Size of second dimension
Return value:	ara::core::Result< Range >	Range instance or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::kRuntime	rollback_semantics
		General runtime error
Description:	Factory method for creation of two-dimensional Range.	

⌋

8.12.1.1.3.4 GetDimension0

[AP_SWS_SHWA_01010] Definition of API function ara::shwa::Range::GetDimension0

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function	
Header file:	#include "ara/shwa/range.h"	
Scope:	`class ara::shwa::Range`	
Syntax:	`std::size_t GetDimension0 () const;`	
Return value:	std::size_t	First Range dimension value
Exception Safety:	not exception safe	
Thread Safety:	not thread-safe	

▽

△

Description:	Method for getting first Range dimension value.

⌋

8.12.1.1.3.5 GetDimension1

[AP_SWS_SHWA_01011] Definition of API function ara::shwa::Range::GetDimension1

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function	
Header file:	#include "ara/shwa/range.h"	
Scope:	`class ara::shwa::Range`	
Syntax:	`std::size_t GetDimension1 () const;`	
Return value:	std::size_t	Second Range dimension value
Exception Safety:	not exception safe	
Thread Safety:	not thread-safe	
Description:	Method for getting second Range dimension value.	

⌋

8.12.1.1.3.6 GetDimension2

[AP_SWS_SHWA_01012] Definition of API function ara::shwa::Range::GetDimension2

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	function	
Header file:	#include "ara/shwa/range.h"	
Scope:	`class ara::shwa::Range`	
Syntax:	`std::size_t GetDimension2 () const;`	
Return value:	std::size_t	Third Range dimension value
Exception Safety:	not exception safe	
Thread Safety:	not thread-safe	
Description:	Method for getting third Range dimension value.	

⌋

8.13 Header: ara/shwa/shwa_enums.h

[AP_SWS_SHWA_00100] Definition of Header File ara/shwa/shwa_enums.h

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	Header File
Syntax:	`ara/shwa/shwa_enums.h`
Description:	File contains data types used in SHWA classes

⌋

8.13.1 Non-Member Types

8.13.1.1 Enumeration: AccessMode

[AP_SWS_SHWA_00101] Definition of API enum ara::shwa::AccessMode

Status: DRAFT
Upstream requirements: RS_SHWA_00017

⌈

Kind:	enumeration	
Header file:	#include "ara/shwa/shwa_enums.h"	
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"	
Scope:	namespace ara::shwa	
Symbol:	AccessMode	
Underlying type:	std::uint32_t	
Syntax:	`enum class AccessMode : std::uint32_t {...};`	
Values:	kRead	Read ONLY mode.
	kWrite	Write ONLY mode.
	kReadWrite	Read-Write mode.
Description:	This enumeration describes data access modes.	

⌋

8.13.1.2 Enumeration: DeviceStatus

[AP_SWS_SHWA_00103] Definition of API enum ara::shwa::DeviceStatus

Status: DRAFT
Upstream requirements: RS_SHWA_00004

⌈

Kind:	enumeration	
Header file:	#include "ara/shwa/shwa_enums.h"	
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"	
Scope:	namespace ara::shwa	
Symbol:	DeviceStatus	
Underlying type:	std::uint32_t	
Syntax:	`enum class DeviceStatus : std::uint32_t {...};`	
Values:	kReady	= 0
		Device is in proper state for task execution.
	kNotAccesible	= 1
		Device can not be accessed.
	kFailure	= 2
		Device in failure state.
	kUnknown	= 255
		Status can't be defined.
Description:	This enumeration describes the device status. The enumeration values 0 - 255 are reserved for AUTOSAR assigned statuses, the stack provider is free to define additional statuses starting from 256.	

⌋

8.13.1.3 Enumeration: Target

[AP_SWS_SHWA_00102] Definition of API enum ara::shwa::Target

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00017

⌈

Kind:	enumeration	
Header file:	#include "ara/shwa/shwa_enums.h"	
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"	
Scope:	namespace ara::shwa	
Symbol:	Target	
Underlying type:	std::uint32_t	
Syntax:	`enum class Target : std::uint32_t {...};`	
Values:	kDevice	Task will be executed on the device (HW accelerator like GPU, FPGA etc...)
	kHostTask	Task will be executed on the host (CPU)
Description:	This enumeration describes the target where the task will be executed.	

⌋

8.14 Header: ara/shwa/shwa_error_domain.h

[AP_SWS_SHWA_01300] Definition of Header File ara/shwa/shwa_error_domain.h

Status: DRAFT
Upstream requirements: RS_AP_00122, RS_AP_00125, RS_AP_00127, RS_AP_00149, RS_SHWA_00007

⌈

Kind:	Header File
Syntax:	`ara/shwa/shwa_error_domain.h`
Description:	File contains classes defining error domain specifics

⌋

8.14.1 Non-Member Types

8.14.1.1 Enumeration: ShwaErrorCode

[AP_SWS_SHWA_01301] Definition of API enum ara::shwa::ShwaErrorCode

Status: DRAFT
Upstream requirements: RS_AP_00122, RS_AP_00125, RS_AP_00127, RS_AP_00149, RS_SHWA_00007

⌈

Kind:	enumeration	
Header file:	#include "ara/shwa/shwa_error_domain.h"	
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"	
Scope:	namespace ara::shwa	
Symbol:	ShwaErrorCode	
Underlying type:	ara::core::ErrorDomain::CodeType	
Syntax:	`enum class ShwaErrorCode : ara::core::ErrorDomain::CodeType {...};`	
Values:	kRuntime	= 1
		General runtime error.
	kTaskExecutionFailure	= 2
		Error during task execution.
	kMemoryNotAllocated	= 3
		Failed memory allocation on host/device.
	kIntegrityCorrupted	= 4
		The structural integrity of the memory allocated by Buffer could not be established.
	kResourceBusy	= 5
		The operation could not be performed because the resource is currently busy.
	kInvalidAccessMode	= 6

▽

△

		Opening a file failed because the requested combination of Open Modes is invalid.
	kAccessorFailure	= 7
		Invalid use or construction of accessors.
	kEventInvalidOperation	= 8
		Error related to events (e.g., invalid wait on event).
	kFeatureNotSupported	= 9
		Used when a requested feature isn't supported.
	kInvalidIndex	= 10
		The operation could not be performed because no initial value is available.
	kInvalidParameter	= 11
		Invalid argument to an API function.
	kInvalidObject	= 12
		Using a destroyed or invalid object (like a buffer or queue).
	kTimedOut	= 13
		Operation timed-out.
Description:	Defines the errors for Safe Hardware Acceleration. The enumeration values 0 - 255 are reserved for AUTOSAR assigned errors, the stack provider is free to define additional errors starting from 256.	

⌋

8.14.2 Non-Member Functions

8.14.2.1 Other

8.14.2.1.1 GetShwaDomain

[AP_SWS_SHWA_01311] Definition of API function ara::shwa::GetShwaDomain

Status: DRAFT
Upstream requirements: RS_AP_00119, RS_AP_00120, RS_AP_00159

⌈

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

⌋

8.14.2.1.2 MakeErrorCode

[AP_SWS_SHWA_01312] Definition of API function ara::shwa::MakeErrorCode

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

⌈

Kind:	function	
Header file:	#include "ara/shwa/shwa_error_domain.h"	
Scope:	namespace ara::shwa	
Syntax:	`constexpr ara::core::ErrorCode MakeErrorCode (ShwaErrorCode 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.	

⌋

8.14.3 Class: ShwaErrorDomain

[AP_SWS_SHWA_01304] Definition of API class ara::shwa::ShwaErrorDomain

Status: DRAFT
Upstream requirements: RS_AP_00122, RS_AP_00127, RS_AP_00140, RS_SHWA_00007

⌈

Kind:	class
Header file:	#include "ara/shwa/shwa_error_domain.h"
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"
Scope:	namespace ara::shwa
Symbol:	ShwaErrorDomain
Base class:	ara::core::ErrorDomain
Syntax:	`class ShwaErrorDomain final : public ara::core::ErrorDomain {...};`
Unique ID:	As per `ara::shwa::ShwaErrorDomain` in [SWS_CORE_90023]
Description:	Defines the error domain for Safe Hardware Acceleration.

⌋

8.14.3.1 Public Member Types

8.14.3.1.1 Type Alias: Errc

[AP_SWS_SHWA_01305] Definition of API type ara::shwa::ShwaErrorDomain::Errc

Status: DRAFT
Upstream requirements: RS_AP_00122

⌈

Kind:	type alias
Header file:	#include "ara/shwa/shwa_error_domain.h"
Scope:	`class ara::shwa::ShwaErrorDomain`
Symbol:	Errc
Syntax:	`using Errc = ShwaErrorCode;`
Description:	Alias for the error code value enumeration.

⌋

8.14.3.1.2 Type Alias: Exception

[AP_SWS_SHWA_01306] Definition of API type ara::shwa::ShwaErrorDomain::Exception

Status: DRAFT
Upstream requirements: RS_AP_00122

⌈

Kind:	type alias
Header file:	#include "ara/shwa/shwa_error_domain.h"
Scope:	`class ara::shwa::ShwaErrorDomain`
Symbol:	Exception
Syntax:	`using Exception = ShwaException;`
Description:	Alias for the exception base class.

⌋

8.14.3.2 Public Member Functions

8.14.3.2.1 Special Member Functions

8.14.3.2.1.1 Default Constructor

[AP_SWS_SHWA_01307] Definition of API function ara::shwa::ShwaErrorDomain::ShwaErrorDomain

Status: DRAFT
Upstream requirements: RS_AP_00119, RS_AP_00120, RS_AP_00159

⌈

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

⌋

8.14.3.2.2 Member Functions

8.14.3.2.2.1 Message

[AP_SWS_SHWA_01309] Definition of API function ara::shwa::ShwaErrorDomain::Message

Status: DRAFT
Upstream requirements: RS_AP_00119, RS_AP_00120, RS_AP_00121, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/shwa/shwa_error_domain.h"	
Scope:	`class ara::shwa::ShwaErrorDomain`	
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.	

⌋

8.14.3.2.2.2 Name

[AP_SWS_SHWA_01308] Definition of API function ara::shwa::ShwaErrorDomain::Name

Status: DRAFT
Upstream requirements: RS_AP_00119, RS_AP_00120, RS_AP_00159

⌈

Kind:	function	
Header file:	#include "ara/shwa/shwa_error_domain.h"	
Scope:	`class ara::shwa::ShwaErrorDomain`	
Syntax:	`const char * Name () const noexcept override;`	
Return value:	const char *	As per `ara::shwa::ShwaErrorDomain` in [SWS_CORE_90023]
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Description:	Returns the name of the error domain.	

⌋

8.14.3.2.2.3 ThrowAsException

[AP_SWS_SHWA_01310] Definition of API function ara::shwa::ShwaErrorDomain::ThrowAsException

Status: DRAFT
Upstream requirements: RS_AP_00120, RS_AP_00121

⌈

Kind:	function	
Header file:	#include "ara/shwa/shwa_error_domain.h"	
Scope:	`class ara::shwa::ShwaErrorDomain`	
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. In shwa this function does not participate in overload resolution when C++ exceptions are disabled in the compiler toolchain.	

⌋

8.14.4 Class: ShwaException

[AP_SWS_SHWA_01302] Definition of API class ara::shwa::ShwaException

Status: DRAFT
Upstream requirements: RS_AP_00122, RS_AP_00127

⌈

Kind:	class
Header file:	#include "ara/shwa/shwa_error_domain.h"
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"
Scope:	namespace ara::shwa
Symbol:	ShwaException
Base class:	ara::core::Exception
Syntax:	`class ShwaException : public ara::core::Exception {...};`
Description:	Exception type could be thrown by application or other FC using Safe Hardware Acceleration. Safe Hardware Acceleration API is exception free and can only return an ErrorCode, but user code can transform this error to exception and throw if it's alligned with system error handling architecture.

⌋

8.14.4.1 Public Member Functions

8.14.4.1.1 Constructors

8.14.4.1.1.1 ShwaException

[AP_SWS_SHWA_01303] Definition of API function ara::shwa::ShwaException::ShwaException

Status: DRAFT
Upstream requirements: RS_AP_00120, RS_AP_00121, RS_AP_00135, RS_AP_00159

⌈

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

⌋

8.15 Header: ara/shwa/task_handler.h

[AP_SWS_SHWA_01200] Definition of Header File ara/shwa/task_handler.h

Status: DRAFT
Upstream requirements: RS_SHWA_00001

⌈

Kind:	Header File
Syntax:	`ara/shwa/task_handler.h`
Description:	File contains classes and methods for tasks handling

⌋

8.15.1 Class: TaskHandler

[AP_SWS_SHWA_01201] Definition of API class ara::shwa::TaskHandler

Status: DRAFT
Upstream requirements: RS_SHWA_00001

⌈

Kind:	class
Header file:	#include "ara/shwa/task_handler.h"
Forwarding header file:	#include "ara/shwa/shwa_fwd.h"
Scope:	namespace ara::shwa
Symbol:	TaskHandler
Syntax:	`class TaskHandler final {...};`
Description:	Handles tasks execution on the selected Device Can be created only inside of the Queue otherwise it's meaningless.

⌋

8.15.1.1 Public Member Functions

8.15.1.1.1 Member Functions

8.15.1.1.1.1 Create

[AP_SWS_SHWA_01202] Definition of API function ara::shwa::TaskHandler::Create

Status: DRAFT
Upstream requirements: RS_SHWA_00001, RS_SHWA_00007

⌈

Kind:	function	
Header file:	#include "ara/shwa/task_handler.h"	
Scope:	`class ara::shwa::TaskHandler`	
Syntax:	`static ara::core::Result< TaskHandler > Create () noexcept;`	
Return value:	ara::core::Result< Task Handler >	TaskHandler instance or an error
Exception Safety:	exception safe	
Thread Safety:	thread-safe	
Errors:	ShwaErrorCode::k Runtime	rollback_semantics
		General runtime error
Description:	Factory method to create TaskHandler instance.	

⌋

8.15.1.1.1.2 ParallelFor

[AP_SWS_SHWA_01204] Definition of API function ara::shwa::TaskHandler::ParallelFor

Status: DRAFT
Upstream requirements: RS_SHWA_00001

⌈

Kind:	function	
Header file:	#include "ara/shwa/task_handler.h"	
Scope:	`class ara::shwa::TaskHandler`	
Syntax:	`template <typename T>` `void ParallelFor (std::size_t bufferSize, T cfg) noexcept;`	
Parameters (in):	bufferSize	Size of buffer to iterate
	cfg	command group handler to be used. e.g. Lambda expression with a task to execute
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Executes in parallel with one work-item per array index on the selected Device. Can be executed only within submitted task to Queue.	

⌋

8.15.1.1.1.3 SingleTask

[AP_SWS_SHWA_01203] Definition of API function ara::shwa::TaskHandler::SingleTask

Status: DRAFT
Upstream requirements: RS_SHWA_00001

⌈

Kind:	function	
Header file:	#include "ara/shwa/task_handler.h"	
Scope:	`class ara::shwa::TaskHandler`	
Syntax:	`template <typename T>` `void SingleTask (T cfg) noexcept;`	
Template param:	cfg	command group handler to be used. e.g. Lambda expression with a task to execute
DIRECTION NOT DEFINED	cfg	--
Return value:	None	
Exception Safety:	exception safe	
Thread Safety:	not thread-safe	
Description:	Executes once on the selected Device. Can be executed only within submitted task to Queue.	

⌋

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 functional cluster does not define any semantic constraints for elements specified in [8].

A Mentioned Manifest Elements

This chapter contains the remaining set of meta-class tables which are not shown directly in the main body of this document.

This chapter is generated.

Class	**Executable**			
Note	This meta-class represents an executable program. **Tags:** atp.recommendedPackage=Executables This Class is only used by the AUTOSAR Adaptive Platform.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
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.
suspendToRam Awareness	SuspendToRam AwarenessEnum	0..1	attr	This attribute describes the type of awareness of the enclosing Executable to suspend-to-RAM functionality. **Tags:** atp.Status=candidate
version	StrongRevisionLabel String	0..1	attr	Version of the executable.

Table A.1: Executable

Class	***PortInterface*** (abstract)			
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*, *AbstractSuspendToRamInterface*, *AbstractSynchronizedTimeBase Interface*, ClientServerInterface, *CryptoInterface*, *DataInterface*, *DiagnosticPortInterface*, FirewallState SwitchInterface, *IdsmAbstractPortInterface*, LogAndTraceInterface, ModeSwitchInterface, Network ManagementPortInterface, *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 This Attribute is only used by the AUTOSAR Adaptive Platform.

Table A.2: PortInterface

Class	**Process**			
Note	This meta-class provides information required to execute the referenced `Executable`. **Tags:** atp.recommendedPackage=Processes This Class is only used by the AUTOSAR Adaptive Platform.			
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.3: Process

B Demands and constraints on Base Software (normative)

This functional cluster defines no demands or constraints for the Base Software on which the AUTOSAR Adaptive Platform is running on (usually a POSIX-compatible operating system).

C Platform Extension Interfaces (normative)

This functional cluster does not specify any Platform Extension Interfaces.

D 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.

D.1 Constraint and Specification Item History of this document according to AUTOSAR Release yy-mm

D.2 Change History of this document according to AUTOSAR Release R25-11

D.2.1 Added Specification Items in R25-11

Number	Heading
[AP_SWS_SHWA_00000]	Definition of Header File ara/shwa/accessor.h
[AP_SWS_SHWA_00001]	Definition of API class ara::shwa::Accessor
[AP_SWS_SHWA_00002]	Definition of API function ara::shwa::Accessor::Accessor
[AP_SWS_SHWA_00003]	Definition of API function ara::shwa::Accessor::Accessor
[AP_SWS_SHWA_00004]	Definition of API function ara::shwa::Accessor::operator=
[AP_SWS_SHWA_00005]	Definition of API function ara::shwa::Accessor::Accessor
[AP_SWS_SHWA_00006]	Definition of API function ara::shwa::Accessor::operator=
[AP_SWS_SHWA_00007]	Definition of API function ara::shwa::Accessor::Create
[AP_SWS_SHWA_00008]	Definition of API function ara::shwa::Accessor::Create
[AP_SWS_SHWA_00009]	Definition of API function ara::shwa::Accessor::operator[]
[AP_SWS_SHWA_00010]	Definition of API function ara::shwa::Accessor::operator[]
[AP_SWS_SHWA_00011]	Definition of API function ara::shwa::Accessor::Set
[AP_SWS_SHWA_00012]	Definition of API function ara::shwa::Accessor::Get
[AP_SWS_SHWA_00013]	SHWA Namespaces

▽

△

Number	Heading
[AP_SWS_SHWA_00014]	Definition of API type ara::shwa::Accessor::ReferenceT
[AP_SWS_SHWA_00015]	Definition of API function ara::shwa::Accessor::HasProperty
[AP_SWS_SHWA_00016]	Definition of API function ara::shwa::Accessor::GetProperty
[AP_SWS_SHWA_00100]	Definition of Header File ara/shwa/shwa_enums.h
[AP_SWS_SHWA_00101]	Definition of API enum ara::shwa::AccessMode
[AP_SWS_SHWA_00102]	Definition of API enum ara::shwa::Target
[AP_SWS_SHWA_00103]	Definition of API enum ara::shwa::DeviceStatus
[AP_SWS_SHWA_00200]	Definition of Header File ara/shwa/buffer.h
[AP_SWS_SHWA_00201]	Definition of API class ara::shwa::Buffer
[AP_SWS_SHWA_00202]	Definition of API function ara::shwa::Buffer::Buffer
[AP_SWS_SHWA_00203]	Definition of API function ara::shwa::Buffer::Buffer
[AP_SWS_SHWA_00204]	Definition of API function ara::shwa::Buffer::operator=
[AP_SWS_SHWA_00205]	Definition of API function ara::shwa::Buffer::Buffer
[AP_SWS_SHWA_00206]	Definition of API function ara::shwa::Buffer::operator=
[AP_SWS_SHWA_00209]	Definition of API function ara::shwa::Buffer::Create
[AP_SWS_SHWA_00210]	Definition of API function ara::shwa::Buffer::Create
[AP_SWS_SHWA_00211]	Definition of API function ara::shwa::Buffer::Create
[AP_SWS_SHWA_00212]	Definition of API function ara::shwa::Buffer::Create
[AP_SWS_SHWA_00213]	Definition of API function ara::shwa::Buffer::Create
[AP_SWS_SHWA_00214]	Definition of API function ara::shwa::Buffer::Create
[AP_SWS_SHWA_00215]	Definition of API function ara::shwa::Buffer::HasProperty

▽

△

Number	Heading
[AP_SWS_SHWA_00216]	Definition of API function ara::shwa::Buffer::GetProperty
[AP_SWS_SHWA_00300]	Definition of Header File ara/shwa/device.h
[AP_SWS_SHWA_00301]	Definition of API class ara::shwa::Device
[AP_SWS_SHWA_00302]	Definition of API function ara::shwa::Device::Device
[AP_SWS_SHWA_00303]	Definition of API function ara::shwa::Device::Device
[AP_SWS_SHWA_00304]	Definition of API function ara::shwa::Device::operator=
[AP_SWS_SHWA_00305]	Definition of API function ara::shwa::Device::Device
[AP_SWS_SHWA_00306]	Definition of API function ara::shwa::Device::operator=
[AP_SWS_SHWA_00308]	Definition of API function ara::shwa::Device::Create
[AP_SWS_SHWA_00309]	Definition of API function ara::shwa::Device::Create
[AP_SWS_SHWA_00310]	Definition of API function ara::shwa::Device::IsCpu
[AP_SWS_SHWA_00311]	Definition of API function ara::shwa::Device::IsGpu
[AP_SWS_SHWA_00312]	Definition of API function ara::shwa::Device::IsAccelerator
[AP_SWS_SHWA_00313]	Definition of API function ara::shwa::Device::GetInfo
[AP_SWS_SHWA_00400]	Definition of Header File ara/shwa/device_selector.h
[AP_SWS_SHWA_00401]	Definition of API class ara::shwa::DeviceSelector
[AP_SWS_SHWA_00402]	Definition of API function ara::shwa::DeviceSelector::~DeviceSelector
[AP_SWS_SHWA_00403]	Definition of API function ara::shwa::DeviceSelector::SelectDevice
[AP_SWS_SHWA_00404]	Definition of API function ara::shwa::DeviceSelector::operator()
[AP_SWS_SHWA_00405]	Definition of API class ara::shwa::GpuSelector
[AP_SWS_SHWA_00406]	Definition of API function ara::shwa::GpuSelector::operator()

▽

△

Number	Heading
[AP_SWS_SHWA_00407]	Definition of API function ara::shwa::GpuSelector::SelectDevice
[AP_SWS_SHWA_00408]	Definition of API class ara::shwa::CpuSelector
[AP_SWS_SHWA_00409]	Definition of API function ara::shwa::CpuSelector::operator()
[AP_SWS_SHWA_00410]	Definition of API function ara::shwa::CpuSelector::SelectDevice
[AP_SWS_SHWA_00411]	Definition of API class ara::shwa::AcceleratorSelector
[AP_SWS_SHWA_00412]	Definition of API function ara::shwa::AcceleratorSelector::operator()
[AP_SWS_SHWA_00413]	Definition of API function ara::shwa::AcceleratorSelector::SelectDevice
[AP_SWS_SHWA_00470]	Safe Hardware Acceleration Error handling approach
[AP_SWS_SHWA_00471]	Safe Hardware Acceleration Error Domain
[AP_SWS_SHWA_00472]	Safe Hardware Acceleration Error Code
[AP_SWS_SHWA_00474]	Safe Hardware Acceleration Error message
[AP_SWS_SHWA_00475]	Safe Hardware Acceleration exceptions capabilities
[AP_SWS_SHWA_00476]	Safe Hardware Acceleration Abort usage
[AP_SWS_SHWA_00477]	kRuntime General Error
[AP_SWS_SHWA_00478]	kTaskExecutionFailure General Error
[AP_SWS_SHWA_00479]	kMemoryNotAllocated General Error
[AP_SWS_SHWA_00480]	kIntegrityCorrupted General Error
[AP_SWS_SHWA_00481]	kResourceBusy General Error
[AP_SWS_SHWA_00482]	kInvalidAccessMode General Error
[AP_SWS_SHWA_00483]	kAccessorFailure General Error
[AP_SWS_SHWA_00484]	kEventInvalidOperation General Error

▽

△

Number	Heading
[AP_SWS_SHWA_00485]	kFeatureNotSupported General Error
[AP_SWS_SHWA_00486]	kInvalidIndex General Error
[AP_SWS_SHWA_00487]	kInvalidParameter General Error
[AP_SWS_SHWA_00488]	kInvalidObject General Error
[AP_SWS_SHWA_00500]	Definition of Header File ara/shwa/error_handler.h
[AP_SWS_SHWA_00501]	Definition of API class ara::shwa::ErrorHandler
[AP_SWS_SHWA_00502]	Definition of API function ara::shwa::ErrorHandler::ErrorHandler
[AP_SWS_SHWA_00503]	Definition of API function ara::shwa::ErrorHandler::ErrorHandler
[AP_SWS_SHWA_00504]	Definition of API function ara::shwa::ErrorHandler::operator=
[AP_SWS_SHWA_00505]	Definition of API function ara::shwa::ErrorHandler::ErrorHandler
[AP_SWS_SHWA_00506]	Definition of API function ara::shwa::ErrorHandler::operator=
[AP_SWS_SHWA_00507]	Definition of API function ara::shwa::ErrorHandler::Create
[AP_SWS_SHWA_00508]	Definition of API function ara::shwa::ErrorHandler::HandleError
[AP_SWS_SHWA_00550]	Objects sharing between processes
[AP_SWS_SHWA_00551]	Objects sharing between threads
[AP_SWS_SHWA_00574]	Check for Initialization
[AP_SWS_SHWA_00575]	Shutdown
[AP_SWS_SHWA_00600]	Definition of Header File ara/shwa/event.h
[AP_SWS_SHWA_00601]	Definition of API class ara::shwa::Event
[AP_SWS_SHWA_00602]	Definition of API function ara::shwa::Event::Create
[AP_SWS_SHWA_00603]	Definition of API function ara::shwa::Event::GetWaitList

▽

△

Number	Heading
[AP_SWS_SHWA_00604]	Definition of API function ara::shwa::Event::Wait
[AP_SWS_SHWA_00605]	Definition of API function ara::shwa::Event::Wait
[AP_SWS_SHWA_00606]	Definition of API function ara::shwa::Event::WaitFor
[AP_SWS_SHWA_00700]	Definition of Header File ara/shwa/id.h
[AP_SWS_SHWA_00701]	Definition of API class ara::shwa::Id
[AP_SWS_SHWA_00702]	Definition of API function ara::shwa::Id::Id
[AP_SWS_SHWA_00703]	Definition of API function ara::shwa::Id::Id
[AP_SWS_SHWA_00704]	Definition of API function ara::shwa::Id::operator=
[AP_SWS_SHWA_00705]	Definition of API function ara::shwa::Id::Id
[AP_SWS_SHWA_00706]	Definition of API function ara::shwa::Id::operator=
[AP_SWS_SHWA_00707]	Definition of API function ara::shwa::Id::Create
[AP_SWS_SHWA_00708]	Definition of API function ara::shwa::Id::Create
[AP_SWS_SHWA_00709]	Definition of API function ara::shwa::Id::Create
[AP_SWS_SHWA_00710]	Definition of API function ara::shwa::Id::Get
[AP_SWS_SHWA_00711]	Definition of API function ara::shwa::Id::Set
[AP_SWS_SHWA_00800]	Definition of Header File ara/shwa/property_list.h
[AP_SWS_SHWA_00801]	Definition of API class ara::shwa::PropertyList
[AP_SWS_SHWA_00802]	Definition of API function ara::shwa::PropertyList::PropertyList
[AP_SWS_SHWA_00900]	Definition of Header File ara/shwa/queue.h
[AP_SWS_SHWA_00901]	Definition of API class ara::shwa::Queue
[AP_SWS_SHWA_00902]	Definition of API function ara::shwa::Queue::Queue

▽

△

Number	Heading
[AP_SWS_SHWA_00903]	Definition of API function ara::shwa::Queue::Queue
[AP_SWS_SHWA_00904]	Definition of API function ara::shwa::Queue::operator=
[AP_SWS_SHWA_00905]	Definition of API function ara::shwa::Queue::Queue
[AP_SWS_SHWA_00906]	Definition of API function ara::shwa::Queue::operator=
[AP_SWS_SHWA_00908]	Definition of API function ara::shwa::Queue::Create
[AP_SWS_SHWA_00909]	Definition of API function ara::shwa::Queue::Create
[AP_SWS_SHWA_00910]	Definition of API function ara::shwa::Queue::GetDevice
[AP_SWS_SHWA_00911]	Definition of API function ara::shwa::Queue::IsInOrder
[AP_SWS_SHWA_00912]	Definition of API function ara::shwa::Queue::Submit
[AP_SWS_SHWA_00913]	Definition of API function ara::shwa::Queue::Submit
[AP_SWS_SHWA_00914]	Definition of API function ara::shwa::Queue::Wait
[AP_SWS_SHWA_00915]	Definition of API function ara::shwa::Queue::HasProperty
[AP_SWS_SHWA_00916]	Definition of API function ara::shwa::Queue::GetProperty
[AP_SWS_SHWA_00917]	Definition of API function ara::shwa::Queue::WaitFor
[AP_SWS_SHWA_01000]	Definition of Header File ara/shwa/range.h
[AP_SWS_SHWA_01001]	Definition of API class ara::shwa::Range
[AP_SWS_SHWA_01002]	Definition of API function ara::shwa::Range::Range
[AP_SWS_SHWA_01003]	Definition of API function ara::shwa::Range::Range
[AP_SWS_SHWA_01004]	Definition of API function ara::shwa::Range::operator=
[AP_SWS_SHWA_01005]	Definition of API function ara::shwa::Range::Range
[AP_SWS_SHWA_01006]	Definition of API function ara::shwa::Range::operator=

▽

△

Number	Heading
[AP_SWS_SHWA_01007]	Definition of API function ara::shwa::Range::Create
[AP_SWS_SHWA_01008]	Definition of API function ara::shwa::Range::Create
[AP_SWS_SHWA_01009]	Definition of API function ara::shwa::Range::Create
[AP_SWS_SHWA_01010]	Definition of API function ara::shwa::Range::GetDimension0
[AP_SWS_SHWA_01011]	Definition of API function ara::shwa::Range::GetDimension1
[AP_SWS_SHWA_01012]	Definition of API function ara::shwa::Range::GetDimension2
[AP_SWS_SHWA_01100]	Definition of Header File ara/shwa/platform.h
[AP_SWS_SHWA_01101]	Definition of API class ara::shwa::Platform
[AP_SWS_SHWA_01102]	Definition of API function ara::shwa::Platform::Platform
[AP_SWS_SHWA_01103]	Definition of API function ara::shwa::Platform::Platform
[AP_SWS_SHWA_01104]	Definition of API function ara::shwa::Platform::operator=
[AP_SWS_SHWA_01105]	Definition of API function ara::shwa::Platform::Platform
[AP_SWS_SHWA_01106]	Definition of API function ara::shwa::Platform::operator=
[AP_SWS_SHWA_01107]	Definition of API function ara::shwa::Platform::GetPlatforms
[AP_SWS_SHWA_01108]	Definition of API function ara::shwa::Platform::GetDevices
[AP_SWS_SHWA_01200]	Definition of Header File ara/shwa/task_handler.h
[AP_SWS_SHWA_01201]	Definition of API class ara::shwa::TaskHandler
[AP_SWS_SHWA_01202]	Definition of API function ara::shwa::TaskHandler::Create
[AP_SWS_SHWA_01203]	Definition of API function ara::shwa::TaskHandler::SingleTask
[AP_SWS_SHWA_01204]	Definition of API function ara::shwa::TaskHandler::ParallelFor
[AP_SWS_SHWA_01300]	Definition of Header File ara/shwa/shwa_error_domain.h

▽

△

Number	Heading
[AP_SWS_SHWA_01301]	Definition of API enum ara::shwa::ShwaErrorCode
[AP_SWS_SHWA_01302]	Definition of API class ara::shwa::ShwaException
[AP_SWS_SHWA_01303]	Definition of API function ara::shwa::ShwaException::ShwaException
[AP_SWS_SHWA_01304]	Definition of API class ara::shwa::ShwaErrorDomain
[AP_SWS_SHWA_01305]	Definition of API type ara::shwa::ShwaErrorDomain::Errc
[AP_SWS_SHWA_01306]	Definition of API type ara::shwa::ShwaErrorDomain::Exception
[AP_SWS_SHWA_01307]	Definition of API function ara::shwa::ShwaErrorDomain::ShwaErrorDomain
[AP_SWS_SHWA_01308]	Definition of API function ara::shwa::ShwaErrorDomain::Name
[AP_SWS_SHWA_01309]	Definition of API function ara::shwa::ShwaErrorDomain::Message
[AP_SWS_SHWA_01310]	Definition of API function ara::shwa::ShwaErrorDomain::ThrowAsException
[AP_SWS_SHWA_01311]	Definition of API function ara::shwa::GetShwaDomain
[AP_SWS_SHWA_01312]	Definition of API function ara::shwa::MakeErrorCode
[AP_SWS_SHWA_01400]	Definition of Header File ara/shwa/device_monitor.h
[AP_SWS_SHWA_01401]	Definition of API class ara::shwa::DeviceMonitor
[AP_SWS_SHWA_01402]	Definition of API function ara::shwa::DeviceMonitor::DeviceMonitor
[AP_SWS_SHWA_01403]	Definition of API function ara::shwa::DeviceMonitor::DeviceMonitor
[AP_SWS_SHWA_01404]	Definition of API function ara::shwa::DeviceMonitor::operator=
[AP_SWS_SHWA_01405]	Definition of API function ara::shwa::DeviceMonitor::DeviceMonitor
[AP_SWS_SHWA_01406]	Definition of API function ara::shwa::DeviceMonitor::operator=
[AP_SWS_SHWA_01407]	Definition of API function ara::shwa::DeviceMonitor::Create
[AP_SWS_SHWA_01408]	Definition of API function ara::shwa::DeviceMonitor::Create

▽

△

Number	Heading
[AP_SWS_SHWA_01409]	Definition of API function ara::shwa::DeviceMonitor::Status
[SWS_SHWA_00576]	ViolationMessage CalledWhileUnititializedViolation

Table D.1: Added Specification Items in R25-11

D.2.2 Changed Specification Items in R25-11

none

D.2.3 Deleted Specification Items in R25-11

none

E 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.

[AP_SWS_SHWA_NA_01500] Hardware-specific Requirements

Status: DRAFT
Upstream requirements: RS_SHWA_00003

⌈These requirements are not implemented as they are hardware-specific and should be implemented by the vendor based on the available hardware on the platform.⌋

[AP_SWS_SHWA_NA_01501] Instance Specifier Standard Voilation Not Applicable to this Specification

Status: DRAFT
Upstream requirements: RS_AP_00170, RS_AP_00171, RS_AP_00172, RS_AP_00173, RS_AP_00177

⌈These requirements are not implemented as they are refering to Instance Specifier of named constructors when Safe Hardware Acceleration uses named constructors without Instance Specifier.⌋