Document Title	Specification of Platform Health Management for Adaptive Platform
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	851

Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R19-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2019-11-28	R19-11	AUTOSAR Release Management	• Added recovery action via application • Usage of `ara::core` types in PHM APIs • Set data types to uint32_t by default • Editorial rework of chapters 7 and 8 • Changed Document Status from Final to published
2019-03-29	19-03	AUTOSAR Release Management	• Modified the API for Supervised Entity and Health Channel • Modified the interface with the Execution Manager
2018-10-31	18-10	AUTOSAR Release Management	• Described the interfaces with functional clusters execution management and state management
2018-03-29	18-03	AUTOSAR Release Management	• Initial release

Table of Contents

1 Introduction and functional overview

This document is the software specification of the Platform Health Management functional cluster within the Adaptive Platform [1].

The specification implements the requirements specified in [2, RS Platform Health Management].

It also implements the general functionality described in the Foundation documents [3, RS Health Monitoring] and [4, SWS Health Monitoring].

`Health Monitoring` is required by [5, ISO 26262] (under the terms control flow monitoring, external monitoring facility, watchdog, logical monitoring, temporal monitoring, program sequence monitoring) and this specification is supposed to address all relevant requirements from this standard.

2 Acronyms and abbreviations

The glossary below includes acronyms and abbreviations relevant to the specification or implementation of Health Monitoring that are not included in the [6, AUTOSAR glossary].

Abbreviation:	Description:
CM	AUTOSAR Adaptive Communication Management
DM	AUTOSAR Adaptive Diagnostic Management
E2E	AUTOSAR End to End communication protection mechanism
PHM	Platform Health Management
SE	Supervised Entity

Acronym:	Description:
Alive Supervision	Mechanism to check the timing constraints of cyclic Supervised Entitys to be within the configured min and max limits.
Application Recovery Action	Callback for notifying applications of detected failures.
ara::com	Communication middleware for the AUTOSAR Adaptive Platform
AUTOSAR Adaptive Platform	see [6] AUTOSAR Glossary
Checkpoint	A point in the control flow of a Supervised Entity where the activity is reported.
Daisy chaining	Chaining multiple instances of Health Monitoring
Deadline Supervision	Mechanism to check that the timing constraints for execution of the transition from a Deadline Start Checkpoint to a corresponding Deadline End Checkpoint are within the configured min and max limits.
Global Supervision Status	Status that summarizes the Local Supervision Status of all Supervised Entitys of a software subsystem.
Graph	A set of Checkpoints connected through Transitions, where at least one of Checkpoints is an Initial Checkpoint and there is a path (through Transitions) between any two Checkpoints of the Graph.
Health Channel	Channel providing information about the health status of a (sub)system. This might be the Global Supervision Status of an application, the result any test routine or the status reported by a (sub)system (e.g. voltage monitoring, OS kernel, ECU status, ...).
Health Monitoring	Supervision of the software behaviour for correct timing and sequence.

Health Status	A set of states that are relevant to the supervised software (e.g. the Global Supervision Status of an application, a Voltage State, an application state, the result of a RAM monitoring algorithm).
Logical Supervision	Kind of online supervision of software that checks if the software (Supervised Entity or set of Supervised Entities) is executed in the sequence defined by the programmer (by the developed code).
Local Supervision Status	Status that represents the current result of Alive Supervision, Deadline Supervision and Logical Supervision of a single Supervised Entity.
Platform Health Management	Health Monitoring for the Adaptive Platform
Supervised Entity	A software entity which is included in the supervision. A Supervised Entity denotes a collection of Checkpoints within a software component. There may be zero, one or more Supervised Entities in a Software Component. A Supervised Entity may be instantiated multiple times, in which case each instance is independently supervised.

Table 2.1: Acronyms

3 Related documentation

3.1 Input documents & related standards and norms

[1] Explanation of Adaptive Platform Design
AUTOSAR_EXP_PlatformDesign

[2] Requirements on Platform Health Management for Adaptive Platform
AUTOSAR_RS_PlatformHealthManagement

[3] Requirements on Health Monitoring
AUTOSAR_RS_HealthMonitoring

[4] Specification of Health Monitoring
AUTOSAR_SWS_HealthMonitoring

[5] ISO 26262 (Part 1-10) – Road vehicles – Functional Safety, First edition
http://www.iso.org

[6] Glossary
AUTOSAR_TR_Glossary

[7] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral

[8] Specification of Execution Management
AUTOSAR_SWS_ExecutionManagement

[9] Specification of Core Types for Adaptive Platform
AUTOSAR_SWS_CoreTypes

[10] Guidelines for using Adaptive Platform interfaces
AUTOSAR_EXP_AdaptivePlatformInterfacesGuidelines

[11] Guidelines for the use of the C++14 language in critical and safety-related systems
AUTOSAR_RS_CPP14Guidelines

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [7, SWS_BSWGeneral], which is also valid for SWS_PlatformHealthManagement.

Thus, the specification SWS_BSWGeneral shall be considered as additional and required specification for SWS_PlatformHealthManagement.

4 Constraints and assumptions

4.1 Limitations

- Daisy chaining (i.e. forwarding Supervision Status, Checkpoint or Health channel information to an entity external to PHM or another PHM instance) is currently not supported in this document release.
- Platform Health Management configuration related to Supervision Modes is not fully supported in this document release.
- An API to inform Supervised Entities about the Supervision states is available only in polling mode. No API using notification mode is available in this release.
- Interface with the Diagnostic Manager is not specified in this release.

4.2 Applicability to car domains

No restriction

5 Dependencies to other modules

5.1 Platform dependencies

The interfaces within AUTOSAR Platform are not standardized.

5.1.1 Dependencies on Execution Management

The Platform Health Management functional cluster is dependent on the Execution Management Interface [8]. The Execution Management Interfaces are used by Platform Health Manager for error recovery to request restarting a Process associated to an Application or to force entering a predefined Unrecoverable State where all running processes are stopped, see [8] for details.

The Platform Health Manager can also request the Execution Manager to provide the state of all processes currently running on the Machine. The inter functional cluster interface between Platform Health Manager and the Execution Manager is also used for notifying a state change of a process.

5.1.2 Dependencies on State Management

The Platform Health Management functional cluster has an interface also with the State Management: the Platform Health Manager can request the State Manager to switch to a specific Machine or Function Group State and the State Manager can signal the Platform Health Manager about a Machine or Function Group State change. This interface is provided by the public API of the State Manager, using ara::com.

5.1.3 Dependencies on Watchdog Interface

The Platform Health Management functional cluster is dependent also on the Watchdog Interface.

5.1.4 Dependencies on other Functional Clusters

It is possible for all functional clusters to use the Supervision mechanisms provided by the Platform Health Management by using `Checkpoint`s and the `Health Channel`s as the other Applications.

6 Requirements Tracing

The following tables reference the requirements specified in [2] and links to the fulfillment 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_HM_09159]	Health Monitoring shall be able to report supervision errors.	[SWS_PHM_01140] [SWS_PHM_01141] [SWS_PHM_01142] [SWS_PHM_01143] [SWS_PHM_01144] [SWS_PHM_01146] [SWS_PHM_01147] [SWS_PHM_01148]
[RS_HM_09237]	Health Monitoring shall provide an interface to Supervised Entities informing them about their Supervision State.	[SWS_PHM_01134] [SWS_PHM_01135] [SWS_PHM_01136] [SWS_PHM_01137]
[RS_HM_09240]	Health Monitoring shall support multiple occurrences of the same Supervised Entity.	[SWS_PHM_00457] [SWS_PHM_01116] [SWS_PHM_01120] [SWS_PHM_01121] [SWS_PHM_01123] [SWS_PHM_01133]
[RS_HM_09241]	Health Monitoring shall support multiple instances of Checkpoints in a Supervised Entity occurrence.	[SWS_PHM_01116] [SWS_PHM_01120] [SWS_PHM_01121] [SWS_PHM_01133]
[RS_HM_09254]	Health Monitoring shall provide an interface to Supervised Entities to report the currently reached Checkpoint.	[SWS_PHM_00321] [SWS_PHM_00424] [SWS_PHM_00425] [SWS_PHM_00458] [SWS_PHM_01010] [SWS_PHM_01123] [SWS_PHM_01124] [SWS_PHM_01125] [SWS_PHM_01126] [SWS_PHM_01127] [SWS_PHM_01131] [SWS_PHM_01132] [SWS_PHM_01138]
[RS_HM_09257]	Health Monitoring shall provide an interface to Supervised Entities to report their health status.	[SWS_PHM_00321] [SWS_PHM_00457] [SWS_PHM_00458] [SWS_PHM_01010] [SWS_PHM_01118] [SWS_PHM_01119] [SWS_PHM_01122] [SWS_PHM_01124] [SWS_PHM_01128] [SWS_PHM_01131] [SWS_PHM_01139]

Requirement	Description	Satisfied by
[RS_PHM_00001]	The Platform Health Management shall provide a standardized header file structure for each service.	[SWS_PHM_00457] [SWS_PHM_01002] [SWS_PHM_01013] [SWS_PHM_01020] [SWS_PHM_01101] [SWS_PHM_01114] [SWS_PHM_01115] [SWS_PHM_01122] [SWS_PHM_01123] [SWS_PHM_01127] [SWS_PHM_01128] [SWS_PHM_01132] [SWS_PHM_01134] [SWS_PHM_01135] [SWS_PHM_01138] [SWS_PHM_01139]
[RS_PHM_00002]	The service header files shall define the namespace for the respective service.	[SWS_PHM_00457] [SWS_PHM_01005] [SWS_PHM_01018] [SWS_PHM_01113] [SWS_PHM_01122] [SWS_PHM_01123] [SWS_PHM_01127] [SWS_PHM_01128] [SWS_PHM_01132] [SWS_PHM_01134] [SWS_PHM_01135] [SWS_PHM_01138] [SWS_PHM_01139]
[RS_PHM_00003]	The Platform Health Management shall define how language specific data types are derived from modeled data types.	[SWS_PHM_00424] [SWS_PHM_00425] [SWS_PHM_01116] [SWS_PHM_01118] [SWS_PHM_01119] [SWS_PHM_01120] [SWS_PHM_01121] [SWS_PHM_01122] [SWS_PHM_01132] [SWS_PHM_01133] [SWS_PHM_01140] [SWS_PHM_01141] [SWS_PHM_01142] [SWS_PHM_01143] [SWS_PHM_01144] [SWS_PHM_01146] [SWS_PHM_01147] [SWS_PHM_01148]

Requirement	Description	Satisfied by
[RS_PHM_00101]	Platform Health Management shall provide a standardized C++ interface for the reporting of Checkpoints.	[SWS_PHM_00321] [SWS_PHM_00424] [SWS_PHM_00425] [SWS_PHM_00458] [SWS_PHM_01010] [SWS_PHM_01123] [SWS_PHM_01124] [SWS_PHM_01125] [SWS_PHM_01127] [SWS_PHM_01131] [SWS_PHM_01132] [SWS_PHM_01134] [SWS_PHM_01135] [SWS_PHM_01138]
[RS_PHM_00102]	Platform Health Management shall provide a standardized C++ interface for the reporting of Health Channel.	[SWS_PHM_00321] [SWS_PHM_00457] [SWS_PHM_00458] [SWS_PHM_01010] [SWS_PHM_01118] [SWS_PHM_01119] [SWS_PHM_01122] [SWS_PHM_01124] [SWS_PHM_01126] [SWS_PHM_01128] [SWS_PHM_01131] [SWS_PHM_01139]
[RS_PHM_00108]	Platform Health Management shall provide a standardized interface between Platform Health Management components used in a daisy chain.	[SWS_PHM_NA]
[RS_PHM_00109]	Platform Health Management shall provide the Daisy chaining interface over ara::com.	[SWS_PHM_NA]

7 Functional specification

7.1 General description

The Platform Health Management supervises the Applications and could trigger a Recovery Action in case any Supervised Entity fails. The Recovery Actions are defined by the integrator based on the software architecture requirements for the Platform Health Management and configured in the Manifests. The Execution Management is responsible for the state dependent management of Application start/stop. All the algorithms and the procedures for the Platform Health Management are described in the Autosar Foundation document [4] and are not specified here: only the Autosar Adaptive specificities, including the interfaces with the other functional clusters, are shown here below.

The interfaces of Health Management to other Functional Clusters are only informative and are not standardized.

7.2 Supervision of Supervised Entities

In order to determine if a Supervised Entity is activated or deactivated at the specific time, the Platform Health Management uses the interface with the Execution Manager: the Platform Health Management requests the state of all processes by invoking GetAllProcessState() and it is notified by the Execution Manager by a change in a process state by ProcessChanged() internal interface (for example when a process state has changed from running to terminating).

7.3 Supervision Modes

A Supervision Mode represents an overall state of a machine or a group of Applications. It is identified by a tuple <machine state, function group state>. The Platform Health Management uses the State Management interface field parameter FunctionGroupState to be notified when one of the states has changed.

7.4 Recovery actions

The following recovery actions are available for an Autosar Adaptive Platform:

- Request the State Manager to switch to a specified FunctionGroup state (RequestState ara::com API).
- Request the Execution Manager to force switching to a specified Unrecoverable State (Name proposal: *EnterUnrecoverableState*).

- Request the Execution Manager to restart a specified process (Name proposal: *ProcessRestart*).
- Request the Watchdog driver to perform a watchdog reset (implementor specific API).
- Report error information to the Diagnostic Manager: not specified in this release.
- Forward error information to an Application, see section 7.4.1.

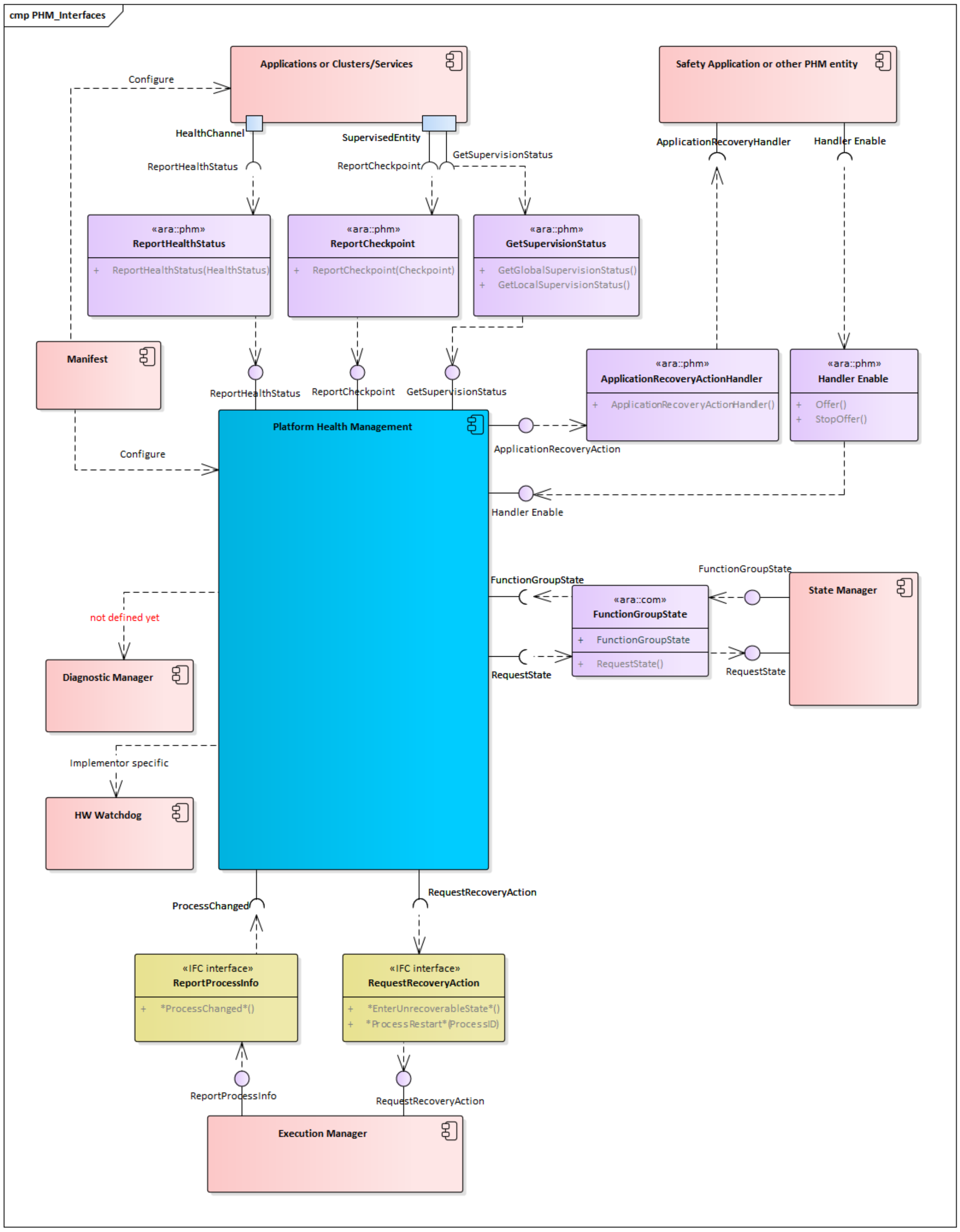
cmp PHM_Interfaces
Applications or Clusters/Services
Safety Application or other PHM entity
Configure
HealthChannel
SupervisedEntity
GetSupervisionStatus
ReportHealthStatus
ReportCheckpoint
ApplicationRecoveryHandler
Handler Enable
«ara::phm» ReportHealthStatus
+ ReportHealthStatus(HealthStatus)
«ara::phm» ReportCheckpoint
+ ReportCheckpoint(Checkpoint)
«ara::phm» GetSupervisionStatus
+ GetGlobalSupervisionStatus()
+ GetLocalSupervisionStatus()
Manifest
«ara::phm» ApplicationRecoveryActionHandler
+ ApplicationRecoveryActionHandler()
«ara::phm» Handler Enable
+ Offer()
+ StopOffer()
Platform Health Management
ApplicationRecoveryAction
Configure
Handler Enable
FunctionGroupState
«ara::com» FunctionGroupState
+ FunctionGroupState
+ RequestState()
State Manager
RequestState
not defined yet
Diagnostic Manager
Implementor specific
HW Watchdog
ProcessChanged
RequestRecoveryAction
«IFC interface» ReportProcessInfo
+ *ProcessChanged*()
«IFC interface» RequestRecoveryAction
+ *EnterUnrecoverableState*()
+ *ProcessRestart*(ProcessID)
ReportProcessInfo
Execution Manager

Figure 7.1: Platform Health Management and the environment

7.4.1 Recovery Action via Application

Platform Health Management can forward information on detected failures to safety applications to allow complex error reactions on application level. This can be done by registering a handler in the corresponding action list which is called within the application context. The mapping of handlers to PHM actions is supported at configuration time only.

[SWS_PHM_01146]{DRAFT} **`ApplicationRecoveryAction` Handler invocation** ⌈`Platform Health Management` shall invoke `ApplicationRecoveryAction::RecoveryHandler` if the `ApplicationRecoveryAction` is mapped to a PHM action and this action is triggered. The handler invocation must be enabled before by a call of `ApplicationRecoveryAction::Offer` (see [SWS_PHM_01143]).⌋ *(RS_PHM_00003, RS_HM_09159)*

Remark: Recognition of the notification and calling of the handler will be taken care by class ApplicationRecoveryAction.

[SWS_PHM_01147]{DRAFT} **Enable handler** ⌈`Platform Health Management` shall enable potential invocations of `ApplicationRecoveryAction::RecoveryHandler` when `ApplicationRecoveryAction::Offer` is called.⌋*(RS_PHM_00003, RS_HM_09159)*

[SWS_PHM_01148]{DRAFT} **Disable handler** ⌈`Platform Health Management` shall disable invocations of `ApplicationRecoveryAction::RecoveryHandler` when `ApplicationRecoveryAction::StopOffer` is called.⌋*(RS_PHM_00003, RS_HM_09159)*

7.5 Multiple processes and multiple instances

During the application deployment phase, a single `Supervised Entity` or a single `Health Channel` may be instanciated several times: this happens for example when the same C++ object class representing a `Supervised Entity` or a `Health Channel` is explicitly instanciated inside the code or when the same executable containing the `Supervised Entity` or the `Health Channel` is started/run multiple times. In such a case, each instance of the `Supervised Entity` is individually supervised, each of them generating an instance of the `Local Supervision Status`.

In order to identify the relevant instance of the `Supervised Entity` or `Health Channel`, the reference to meta-model name of the specific instance is used. This can be done by a call to the function

```
ara::core::InstanceSpecifier (StringView metaModelIdentifier);
```

that is defined in the Autosar Adaptive Platform Core Types specification [9].

The parameter `metaModelIdentifier` is actually a meta-model related string in the syntax `<shortname context1>/<shortname context2>/.../<shortname`

contextN>/<shortname of PortPrototype>, where the shortname of the Port Prototype corresponds to the Port representing the Supervised Entity or Health Channel in the meta-model.

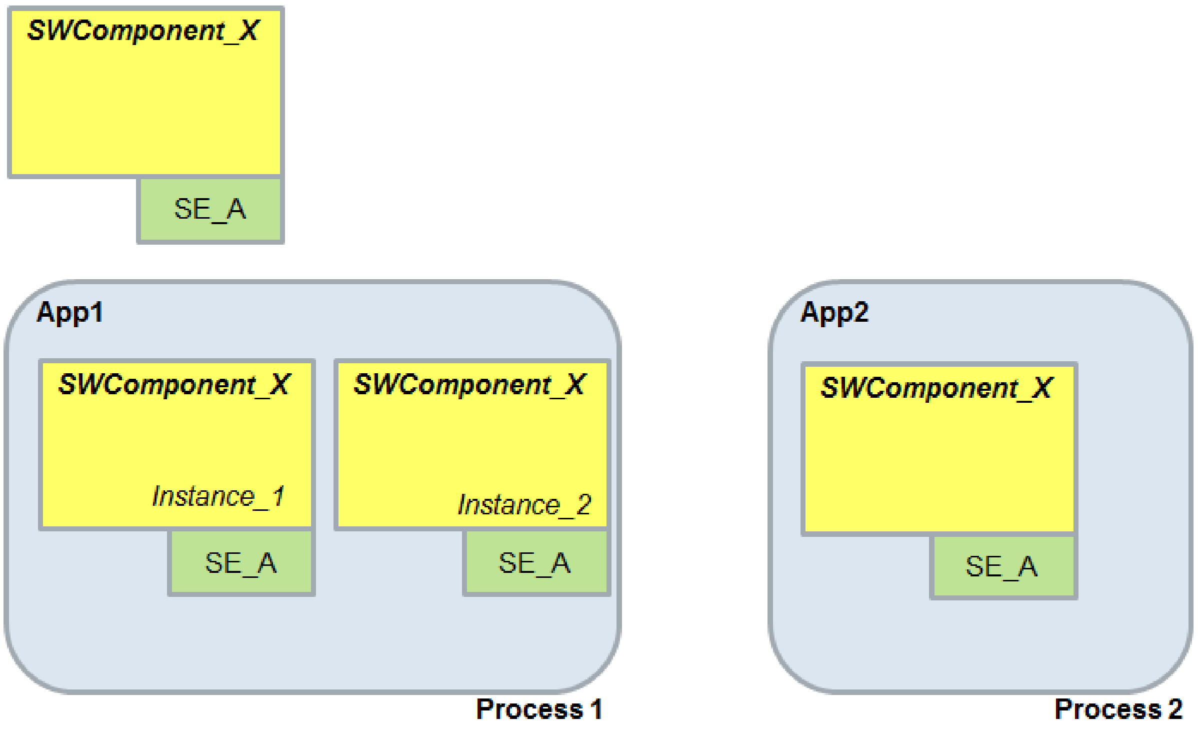

Figure 7.2: Example of multiple instance of the same Supervised Entity

Figure 7.2 shows an example of a single Supervised Entity (called SE_A) belonging to a unique SW Component (SWCompontent_X in the example). SWComponent_X is instanciated explicitly twice in the same process (Process 1) and another time in a different process/application (process 2). In such a case, three instances of the Port Prototype representing the Supervised Entity are created.

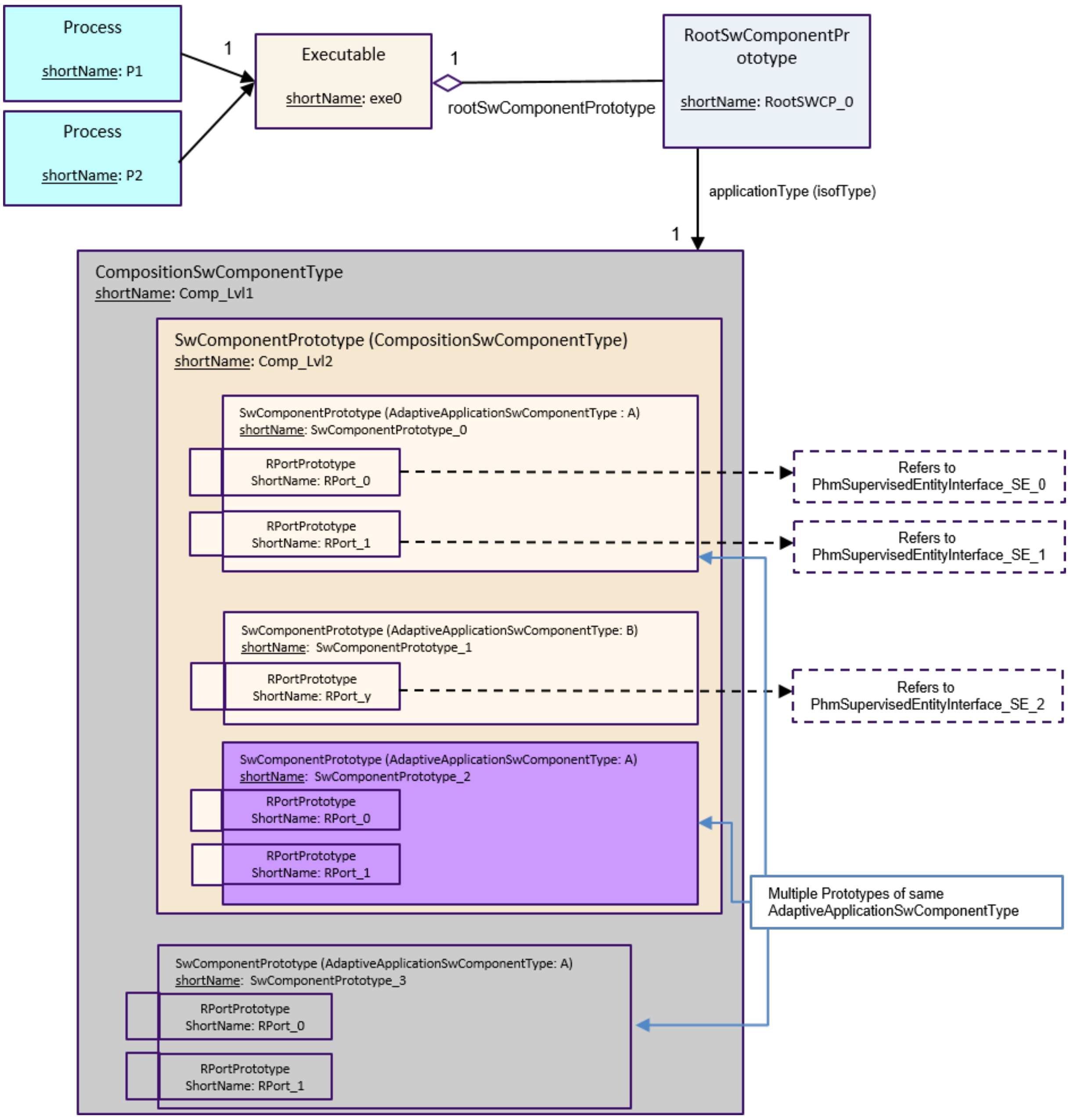
Process
shortName: P1
Process
shortName: P2
1
Executable
shortName: exe0
1
rootSwComponentPrototype
RootSwComponentPrototype
shortName: RootSWCP_0
applicationType (isofType)
1
CompositionSwComponentType
shortName: Comp_Lvl1
SwComponentPrototype (CompositionSwComponentType)
shortName: Comp_Lvl2
SwComponentPrototype (AdaptiveApplicationSwComponentType : A)
shortName: SwComponentPrototype_0
RPortPrototype
ShortName: RPort_0
RPortPrototype
ShortName: RPort_1
Refers to PhmSupervisedEntityInterface_SE_0
Refers to PhmSupervisedEntityInterface_SE_1
SwComponentPrototype (AdaptiveApplicationSwComponentType: B)
shortName: SwComponentPrototype_1
RPortPrototype
ShortName: RPort_y
Refers to PhmSupervisedEntityInterface_SE_2
SwComponentPrototype (AdaptiveApplicationSwComponentType: A)
shortName: SwComponentPrototype_2
RPortPrototype
ShortName: RPort_0
RPortPrototype
ShortName: RPort_1
Multiple Prototypes of same AdaptiveApplicationSwComponentType
SwComponentPrototype (AdaptiveApplicationSwComponentType: A)
shortName: SwComponentPrototype_3
RPortPrototype
ShortName: RPort_0
RPortPrototype
ShortName: RPort_1

Figure 7.3: Example for Instance Specifier and context

Figure 7.3 shows a more complex example to demonstrate multiple levels of context. In the figure, all RPortPrototypes shown are referring to PHM `Supervised Entity` interface. `<shortname context1>` is the short name of the executable. The other contexts, i.e. `<shortname context2>` to `<shortname contextN>`, represent the different levels of composition. See [9] for details on the syntax of Instance Specifier.

Using the naming and composition presented in figure 7.3, the following strings are possible instance specifiers:

- exe0/RootSWCP_0/Comp_Lvl1/Comp_Lvl2/SwComponentPrototype_0/RPort_0
- exe0/RootSWCP_0/Comp_Lvl1/Comp_Lvl2/SwComponentPrototype_0/RPort_1

- exe0/RootSWCP_0/Comp_Lvl1/Comp_Lvl2/SwComponentPrototype_1/Rport_y
- exe0/RootSWCP_0/Comp_Lvl1/Comp_Lvl2/SwComponentPrototype_2/RPort_0
- exe0/RootSWCP_0/Comp_Lvl1/Comp_Lvl2/SwComponentPrototype_2/RPort_1
- exe0/RootSWCP_0/Comp_Lvl1/SwComponentPrototype_3/RPort_0
- exe0/RootSWCP_0/Comp_Lvl1/SwComponentPrototype_3/RPort_1

7.6 Functional cluster lifecycle

7.6.1 Shutdown

Using `ara::core::Deinitialize`, all functional clusters with direct ARA interfaces can be shut down. When Platform Health Management functional cluster is shut down, all configured supervisions are terminated. It is the integrators responsibility to make correct use of the shutdown mechanism. Details for ensuring safe execution are given in [10].

8 Platform Health Management API specification

8.1 API Types

This chapter describes the standardized types provided by the `ara::phm` API. The `ara::phm` API is based on the `ara::core` types defined in [9].

8.1.1 Checkpoint

Definition for `Checkpoint` data type

[SWS_PHM_01138]{DRAFT} ⌈

Kind:	type alias
Symbol:	ara::phm::Checkpoint
Scope:	namespace ara::phm
Derived from:	typedef std::uint32_t
Syntax:	`using ara::phm::Checkpoint = std::uint32_t;`
Header file:	#include "ara/phm/supervised_entity.h"
Description:	Represents a Checkpoint.

⌋*(RS_HM_09254, RS_PHM_00101, RS_PHM_00001, RS_PHM_00002)*

8.1.2 LocalSupervisionStatus

`Platform Health Management` provides a `LocalSupervisionStatus` enum class:

[SWS_PHM_01136]{DRAFT} ⌈

Kind:	enumeration	
Symbol:	ara::phm::LocalSupervisionStatus	
Scope:	namespace ara::phm	
Underlying type:	uint32_t	
Syntax:	`enum class LocalSupervisionStatus : uint32_t {...};`	
Values:	kDeactivated	Supervision is not active.
	kOK	Supervision is active and no failure is present.
	kFailed	A failure was detected but still within tolerance/ debouncing.
	kExpired	A failure was detected and qualified.
Header file:	#include "ara/phm/supervised_entity.h"	

▽

△

Description:	Enumeration of local supervision status. Scoped Enumeration of uint32_t.

⌋*(RS_HM_09237)*

8.1.3 GlobalSupervisionStatus

Platform Health Management provides a GlobalSupervisionStatus enum class:

[SWS_PHM_01137]{DRAFT} ⌈

Kind:	enumeration	
Symbol:	ara::phm::GlobalSupervisionStatus	
Scope:	namespace ara::phm	
Underlying type:	uint32_t	
Syntax:	`enum class GlobalSupervisionStatus : uint32_t {...};`	
Values:	kDeactivated	Supervision is not active.
	kOK	All local supervisions are in status kOK or k Deactivated.
	kFailed	At least one local supervision is in status kFailed but none in status kExpired.
	kExpired	At least one local supervision is in status kExpired but the number of Supervision Cycle since reaching kExpired has not exceeded the tolerance.
	kStopped	At least one local supervision is in status kExpired and the number of Supervision Cycle since reaching kExpired has exceeded the tolerance.
Header file:	#include "ara/phm/supervised_entity.h"	
Description:	Enumeration of global supervision status. Scoped Enumeration of uint32_t.	

⌋*(RS_HM_09237)*

8.1.4 SupervisedEntity

Platform Health Management provides a SupervisedEntity class which provides methods to report Checkpoints and to retrieve the local and global supervision status corresponding to the related SupervisedEntity.

[SWS_PHM_01132]{DRAFT} ⌈

Kind:	class
Symbol:	ara::phm::SupervisedEntity
Scope:	namespace ara::phm
Syntax:	`class SupervisedEntity {...};`
Header file:	#include "ara/phm/supervised_entity.h"
Description:	SupervisedEntity Class.

⌋*(RS_PHM_00003, RS_PHM_00101, RS_HM_09254, RS_PHM_00001, RS_PHM_-00002)*

8.1.5 HealthChannel

`Platform Health Management` provides a `HealthChannel` class which provides a method to report the `Health Status`.

[SWS_PHM_01122]{DRAFT} ⌈

Kind:	class
Symbol:	ara::phm::HealthChannel
Scope:	namespace ara::phm
Syntax:	`class HealthChannel {...};`
Header file:	#include "ara/phm/health_channel.h"
Description:	HealthChannel Class.

⌋*(RS_PHM_00003, RS_PHM_00102, RS_HM_09257, RS_PHM_00001, RS_PHM_-00002)*

8.1.6 HealthStatus

Definition for `Health Status` data type

[SWS_PHM_01139]{DRAFT} ⌈

Kind:	type alias
Symbol:	ara::phm::HealthStatus
Scope:	namespace ara::phm
Derived from:	typedef std::uint32_t
Syntax:	`using ara::phm::HealthStatus = std::uint32_t;`
Header file:	#include "ara/phm/health_channel.h"
Description:	Represents a Health Status.

⌋*(RS_HM_09257, RS_PHM_00102, RS_PHM_00001, RS_PHM_00002)*

8.1.7 ApplicationRecoveryAction

Platform Health Management provides an ApplicationRecoveryAction abstract class which provides a virtual method to implement the recovery handler.

[SWS_PHM_01140]{DRAFT} ⌈

Kind:	class
Symbol:	ara::phm::ApplicationRecoveryAction
Scope:	namespace ara::phm
Syntax:	`class ApplicationRecoveryAction {...};`
Header file:	#include "ara/phm/application_recovery_action.h"
Description:	ApplicationRecoveryAction abstract class.

⌋*(RS_PHM_00003, RS_HM_09159)*

8.1.8 Daisy Chaining Related Types (Non-generated)

Daisy chaining is not supported in this AUTOSAR release.

8.1.9 Error and Exception Types

The ara::phm API does not explicitly make use of C++ exceptions. The AUTOSAR implementer is free to provide an exception-free implementation or an implementation that uses Unchecked Exceptions. The implementer is however not allowed to define Checked Exceptions.

ara::phm API does hereby strictly follow [11, AUTOSAR CPP14 guidelines] regarding exception usage. I.e. there is a clean separation of exception types into Unchecked Exceptions and Checked Exceptions, which ara::phm API builds upon.

The former ones (i.e., Unchecked Exceptions) can basically occur in *any* ara::phm API call, are not formally modeled in the Manifest, and are fully implementation specific.

The latter ones (i.e., Checked Exceptions) are not used by Health Management API.

8.1.10 E2E Related Data Types

The usage of E2E communication protection for Health Management is not standardized.

8.2 API Reference

8.2.1 SupervisedEntity API

SupervisedEntity API can be used to report checkpoints or to query the status of a SupervisedEntity.

8.2.1.1 SupervisedEntity::SupervisedEntity

The Platform Health Management provides constructor for class SupervisedEntity indicating the instance specifier of the SupervisedEntity.

[SWS_PHM_01123]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::phm::SupervisedEntity::SupervisedEntity(ara::core::InstanceSpecifier const &instance)	
Scope:	class ara::phm::SupervisedEntity	
Syntax:	`explicit SupervisedEntity (ara::core::InstanceSpecifier const &instance);`	
Parameters (in):	instance	instance specifier of the supervised entity.
Header file:	#include "ara/phm/supervised_entity.h"	
Description:	Creation of a SupervisedEntity.	

⌋*(RS_PHM_00101, RS_HM_09254, RS_HM_09240, RS_PHM_00001, RS_PHM_-00002)*

Note that additionally process Identification information is necessary. This has to be obtained by the constructor.

8.2.1.2 SupervisedEntity::ReportCheckpoint

The Platform Health Management provides a method ReportCheckpoint, provided by SupervisedEntity.

[SWS_PHM_01127]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::phm::SupervisedEntity::ReportCheckpoint(Checkpoint checkpointId)	
Scope:	class ara::phm::SupervisedEntity	
Syntax:	`ara::core::Result<void> ReportCheckpoint (Checkpoint checkpointId);`	
Parameters (in):	checkpointId	checkpoint identifier.
Return value:	ara::core::Result< void >	A Result, being either empty or containing an implementation specific error code.

▽

△

Header file:	#include "ara/phm/supervised_entity.h"
Description:	Reports an occurrence of a Checkpoint.

⌋(*RS_PHM_00101*, *RS_HM_09254*, *RS_PHM_00001*, *RS_PHM_00002*)

8.2.1.3 SupervisedEntity::GetLocalSupervisionStatus

The Platform Health Management provides a method GetLocalSupervisionStatus, provided by SupervisedEntity. This method returns the current Local Supervision Status of this SupervisedEntity.

[SWS_PHM_01134]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::phm::SupervisedEntity::GetLocalSupervisionStatus()	
Scope:	class ara::phm::SupervisedEntity	
Syntax:	`ara::core::Result<LocalSupervisionStatus> GetLocalSupervisionStatus () const;`	
Return value:	ara::core::Result< LocalSupervision Status >	the local supervision status.
Header file:	#include "ara/phm/supervised_entity.h"	
Description:	Returns the local supervision status that the supervised entity belongs to.	

⌋(*RS_PHM_00101*, *RS_HM_09237*, *RS_PHM_00001*, *RS_PHM_00002*)

8.2.1.4 SupervisedEntity::GetGlobalSupervisionStatus

The Platform Health Management provides a method GetGlobalSupervisionStatus, provided by SupervisedEntity. This method returns the current Global Supervision Status corresponding to this SupervisedEntity.

[SWS_PHM_01135]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::phm::SupervisedEntity::GetGlobalSupervisionStatus()	
Scope:	class ara::phm::SupervisedEntity	
Syntax:	`ara::core::Result<GlobalSupervisionStatus> GetGlobalSupervisionStatus () const;`	
Return value:	ara::core::Result< GlobalSupervision Status >	the global supervision status.
Header file:	#include "ara/phm/supervised_entity.h"	

▽

△

Description:	Returns the status of global supervision that the supervised entity belongs to.

⌋*(RS_PHM_00101, RS_HM_09237, RS_PHM_00001, RS_PHM_00002)*

8.2.2 HealthChannel API

8.2.2.1 HealthChannel::HealthChannel

The Platform Health Management provides constructor for class HealthChannel indicating the instance specifier of the HealthChannel.

[SWS_PHM_00457]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::phm::HealthChannel::HealthChannel(ara::core::InstanceSpecifier const &instance)	
Scope:	class ara::phm::HealthChannel	
Syntax:	`explicit HealthChannel (ara::core::InstanceSpecifier const &instance);`	
Parameters (in):	instance	instance specifier of the health channel
Header file:	#include "ara/phm/health_channel.h"	
Description:	Creation of a HealthChannel.	

⌋*(RS_PHM_00102, RS_HM_09257, RS_HM_09240, RS_PHM_00001, RS_PHM_00002)*

Note that additionally process Identification information is necessary. This has to be obtained by the constructor.

8.2.2.2 HealthChannel::ReportHealthStatus

The Platform Health Management provides a method ReportHealthStatus, provided by HealthChannel.

[SWS_PHM_01128]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::phm::HealthChannel::ReportHealthStatus(HealthStatus healthStatusId)	
Scope:	class ara::phm::HealthChannel	
Syntax:	`ara::core::Result<void> ReportHealthStatus (HealthStatus healthStatusId);`	
Parameters (in):	healthStatusId	The identifier representing the Health Status. The mapping is implementation specific.

▽

△

Return value:	ara::core::Result< void >	A Result, being either empty or containing an implementation specific error code.
Header file:	#include "ara/phm/health_channel.h"	
Description:	Reports a Health Status.	

⌋(*RS_PHM_00102*, *RS_HM_09257*, *RS_PHM_00001*, *RS_PHM_00002*)

8.2.3 `ApplicationRecoveryAction` API

8.2.3.1 ApplicationRecoveryAction::ApplicationRecoveryAction

The `Platform Health Management` provides constructor for class ApplicationRecoveryAction indicating the instance specifier of the `ApplicationRecoveryAction`.

[SWS_PHM_01141]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::phm::ApplicationRecoveryAction::ApplicationRecoveryAction(ara::core::InstanceSpecifier const &instance)	
Scope:	class ara::phm::ApplicationRecoveryAction	
Syntax:	`explicit ApplicationRecoveryAction (ara::core::InstanceSpecifier const &instance);`	
Parameters (in):	instance	instance specifier of the application recovery action.
Header file:	#include "ara/phm/application_recovery_action.h"	
Description:	Creation of an ApplicationRecoveryAction.	

⌋(*RS_PHM_00003*, *RS_HM_09159*)

Note that additionally process Identification information is necessary. This has to be obtained by the constructor.

8.2.3.2 ApplicationRecoveryAction::RecoveryHandler

[SWS_PHM_01142]{DRAFT} ⌈

Kind:	function
Symbol:	ara::phm::ApplicationRecoveryAction::RecoveryHandler()
Scope:	class ara::phm::ApplicationRecoveryAction
Syntax:	`virtual void RecoveryHandler ()=0;`
Return value:	None
Header file:	#include "ara/phm/application_recovery_action.h"

▽

△

Description:	RecoveryHandler to be invoked by PHM. The handler invocation must be enabled before by a call of ApplicationRecoveryAction::Offer.

⌋(*RS_PHM_00003*, *RS_HM_09159*)

8.2.3.3 ApplicationRecoveryAction::Offer

[SWS_PHM_01143]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::phm::ApplicationRecoveryAction::Offer()	
Scope:	class ara::phm::ApplicationRecoveryAction	
Syntax:	`ara::core::Result<void> Offer ();`	
Return value:	ara::core::Result< void >	A Result, being either empty or containing an implementation specific error code.
Header file:	#include "ara/phm/application_recovery_action.h"	
Description:	Enables potential invocations of RecoveryHandler.	

⌋(*RS_PHM_00003*, *RS_HM_09159*)

8.2.3.4 ApplicationRecoveryAction::StopOffer

[SWS_PHM_01144]{DRAFT} ⌈

Kind:	function
Symbol:	ara::phm::ApplicationRecoveryAction::StopOffer()
Scope:	class ara::phm::ApplicationRecoveryAction
Syntax:	`void StopOffer ();`
Return value:	None
Header file:	#include "ara/phm/application_recovery_action.h"
Description:	Disables invocations of RecoveryHandler.

⌋(*RS_PHM_00003*, *RS_HM_09159*)

8.2.4 Forward supervision state (daisy-chain)

This feature is not supported by this AUTOSAR release.

A Not applicable requirements

[SWS_PHM_NA]{DRAFT} ⌈These requirements are not applicable as they are not within the scope of this release.⌋*(RS_PHM_00108, RS_PHM_00109)*

B Removed requirements

[SWS_PHM_01005]{DRAFT} **Namespace of generated header files for a Supervised Entity** ⌈This requirement has been removed.⌋*(RS_PHM_00002)*

[SWS_PHM_01020]{DRAFT} **Folder structure for Supervised Entity files** ⌈This requirement has been removed.⌋*(RS_PHM_00001)*

[SWS_PHM_01002]{DRAFT} **Generated header files for Supervised Entities** ⌈This requirement has been removed.⌋*(RS_PHM_00001)*

[SWS_PHM_01113]{DRAFT} **Namespace of generated header files for a Health Channel** ⌈This requirement has been removed.⌋*(RS_PHM_00002)*

[SWS_PHM_01114]{DRAFT} **Folder structure for Supervised Entity files** ⌈This requirement has been removed.⌋*(RS_PHM_00001)*

[SWS_PHM_01115]{DRAFT} **Generated header files for Health Channels** ⌈This requirement has been removed.⌋*(RS_PHM_00001)*

[SWS_PHM_00424]{DRAFT} **Enumeration for Supervised Entity** ⌈This requirement has been removed.⌋*(RS_PHM_00003, RS_PHM_00101, RS_HM_09254)*

[SWS_PHM_00425]{DRAFT} **Definition of enumerators of Supervised Entitys** ⌈This requirement has been removed.⌋*(RS_PHM_00003, RS_PHM_00101, RS_HM_09254)*

[SWS_PHM_01116]{DRAFT} **Definition of an identifier for a Supervised Entitys** ⌈This requirement has been removed.⌋*(RS_PHM_00003, RS_HM_09240, RS_HM_09241)*

[SWS_PHM_01133]{DRAFT} **Definition of an identifier for a Supervised Entity Prototype** ⌈This requirement has been removed.⌋*(RS_PHM_00003, RS_HM_09240, RS_HM_09241)*

[SWS_PHM_01118]{DRAFT} **Enumeration for Health Channel** ⌈This requirement has been removed.⌋*(RS_PHM_00003, RS_PHM_00102, RS_HM_09257)*

[SWS_PHM_01119]{DRAFT} **Definition of enumerators of Health Channels** ⌈This requirement has been removed.⌋*(RS_PHM_00003, RS_PHM_00102, RS_HM_09257)*

[SWS_PHM_01120]{DRAFT} **Definition of an identifier for a `Health Channel`** ⌈This requirement has been removed.⌋*(RS_PHM_00003, RS_HM_09240, RS_HM_09241)*

[SWS_PHM_01121]{DRAFT} **Definition of an identifier for a Health Channel Prototype** ⌈This requirement has been removed.⌋*(RS_PHM_00003, RS_HM_09240, RS_HM_09241)*

[SWS_PHM_00321]{DRAFT} **Underlying data types** ⌈This requirement has been removed.⌋*(RS_PHM_00101, RS_PHM_00102, RS_HM_09254, RS_HM_09257)*

[SWS_PHM_01131]{DRAFT} **Identifier Class Template** ⌈This requirement has been removed.⌋*(RS_PHM_00101, RS_PHM_00102, RS_HM_09254, RS_HM_09257)*

[SWS_PHM_01010]{DRAFT} **PHM Class** ⌈This requirement has been removed.⌋*(RS_PHM_00101, RS_PHM_00102, RS_HM_09254, RS_HM_09257)*

[SWS_PHM_00458]{DRAFT} **Creation of PHM service interface** ⌈This requirement has been removed.⌋*(RS_PHM_00101, RS_PHM_00102, RS_HM_09254, RS_HM_09257)*

[SWS_PHM_01124]{DRAFT} **Copy constructor for the use by SupervisedEntity and by HealthChannel** ⌈This requirement has been removed.⌋*(RS_PHM_00101, RS_PHM_00102, RS_HM_09254, RS_HM_09257)*

[SWS_PHM_01125]{DRAFT} **The Platform Health Management shall provide a protected method ReportCheckpoint, provided by PHM** ⌈This requirement has been removed.⌋*(RS_PHM_00101, RS_HM_09254)*

[SWS_PHM_01126]{DRAFT} **The Platform Health Management shall provide a protected method ReportHealthStatus, provided by PHM** ⌈This requirement has been removed.⌋*(RS_PHM_00102, RS_HM_09254)*

[SWS_PHM_01101]{DRAFT} **Folder structure for header files** ⌈This requirement has been removed.⌋*(RS_PHM_00001)*

[SWS_PHM_01018]{DRAFT} **Header file namespace** ⌈This requirement has been removed.⌋*(RS_PHM_00002)*

[SWS_PHM_01013]{DRAFT} **Header file existence** ⌈This requirement has been removed.⌋*(RS_PHM_00001)*

C Interfaces to other Functional Clusters (informative)

AUTOSAR decided not to standardize interfaces which are exclusively used between Functional Clusters (on platform-level only), to allow efficient implementations, which might depend e.g. on the used Operating System.

This chapter provides informative guidelines how the interaction between Functional Clusters looks like, by clustering the relevant requirements of this document to describe Inter-Functional Cluster (IFC) interfaces. In addition, the standardized public interfaces which are accessible by user space applications can also be used for interaction between Functional Clusters.

The goal is to provide a clear understanding of Functional Cluster boundaries and interaction, without specifying syntactical details. This ensures compatibility between documents specifying different Functional Clusters and supports parallel implementation of different Functional Clusters. Details of the interfaces are up to the platform provider. Additional interfaces, parameters and return values can be added.

C.1 Interface Tables

C.1.1 Process State Transition Event

	Name	Description	Requirements
Intended users	Execution Management		
Name proposal	*ProcessChanged*		
Functionality	Notify a change of a Process State	The process state change notification can be used by the Platform Health Manager to detemine which Supervision Entity is activated or deactivated	
Parameters (in)	Process identifier	Unique named identifier of the Process that changed state.	
	State	New state of the specified process.	
Parameters (inout)	None		
Parameters (out)	None		
Return value	None		

Table C.1: Process State Transition Event

C.1.2 Get Process States

Execution Management provides an Interfunctional Cluster Interface to poll state information about currently running processes. If called, Execution Management provides a list of the currently running processes. With this information the Platform Health Management can identify, based on Manifest information, if and how each process should be monitored. Details on the API and the IFC table are given in [8].

This polling API could be called e.g. once after startup to get the status of processes started before PHM. The process status can be maintained by the information received via the reporting API specified in C.1.1.

D Mentioned Class Tables

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.