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

Document Status	Final
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	19-03

Document Change History

Date	Release	Changed by	Description
2019-03-29	19-03	AUTOSAR Release Management	• 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.
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 to a corresponding 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] Specification of Execution Management
AUTOSAR_SWS_ExecutionManagement

[8] Specification of Core Types for Adaptive Platform
AUTOSAR_SWS_CoreTypes

[9] Methodology for Adaptive Platform
AUTOSAR_TR_AdaptiveMethodology

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

3.2 Related specification

See section 3.1.

4 Constraints and assumptions

4.1 Limitations

[SWS_PHM_00110]{DRAFT} ⌈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. ⌋*(RS_PHM_00108, RS_PHM_00109)*

[SWS_PHM_00111]{DRAFT} ⌈Platform Health Management configuration related to Supervision Modes is not fully supported in this document release. ⌋*(RS_PHM_00104, RS_HM_09253)*

[SWS_PHM_00112]{DRAFT} ⌈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. ⌋*(RS_HM_09237)*

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 [7]. 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 safe state. 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, Function Group or Application State and the State Manager can signal the Platform Helath Manager about a Machine, Function Group or Application 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_09237]	Health Monitoring shall provide an interface to Supervised Entities informing them about their Supervision State.	[SWS_PHM_00112] [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_09253]	Health Monitoring shall support mode-dependent behavior of Supervised Entities and it shall support the supervision on the transitions between Checkpoints belonging different Supervision Modes.	[SWS_PHM_00111]
[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 for 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_01002] [SWS_PHM_01013] [SWS_PHM_01020] [SWS_PHM_01101] [SWS_PHM_01114] [SWS_PHM_01115]
[RS_PHM_00002]	The service header files shall define the namespace for the respective service.	[SWS_PHM_01005] [SWS_PHM_01018] [SWS_PHM_01113]
[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]
[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_00104]	Platform Health Management shall realize the Supervision Mode as a tuple of Execution Management states.	[SWS_PHM_00111]

Requirement	Description	Satisfied by
[RS_PHM_00108]	Platform Health Management shall provide a standardized interface between Platform Health Management components used in a daisy chain.	[SWS_PHM_00110] [SWS_PHM_NA]
[RS_PHM_00109]	Platform Health Management shall provide the Daisy chaining interface over ara::com .	[SWS_PHM_00110] [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 Cluster are only informative and 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, application state>. The Platform Health Management uses the interface provided by the State Manager (StateChage() API) 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 Safe State (EnterSafeState Lib-API).
- Request the Execution Manager to restart a specified process (ProcessRestart Lib-API).

- 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 another PHM entity or an Application: not specified in this release.

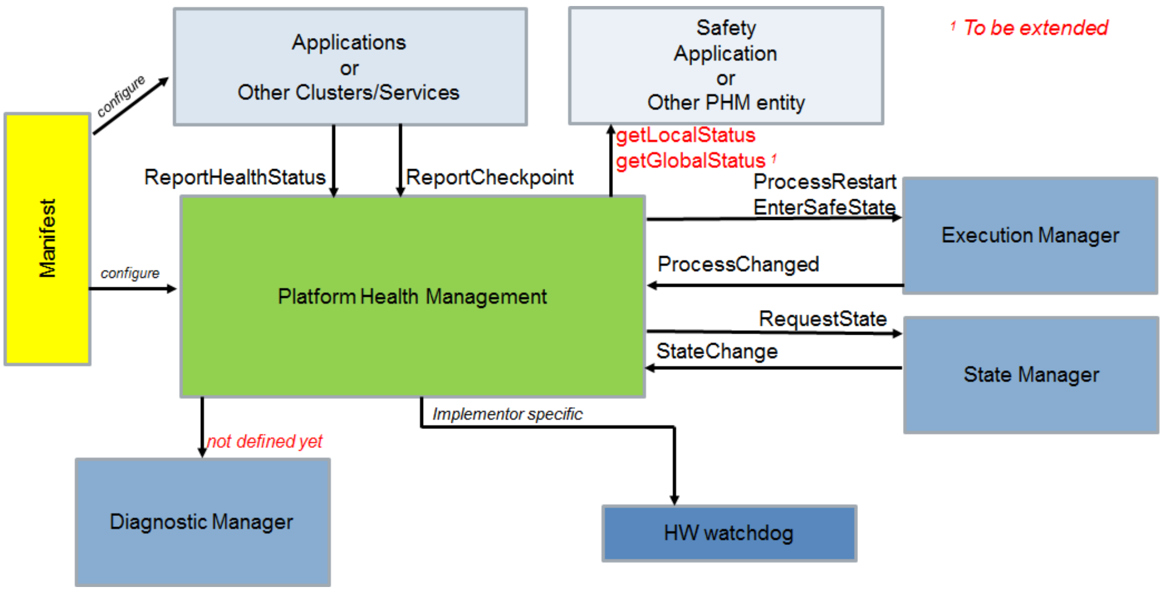

Figure 7.1: Platform Health Management and the environment

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 or the same process 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 shall be 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 [8].

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.

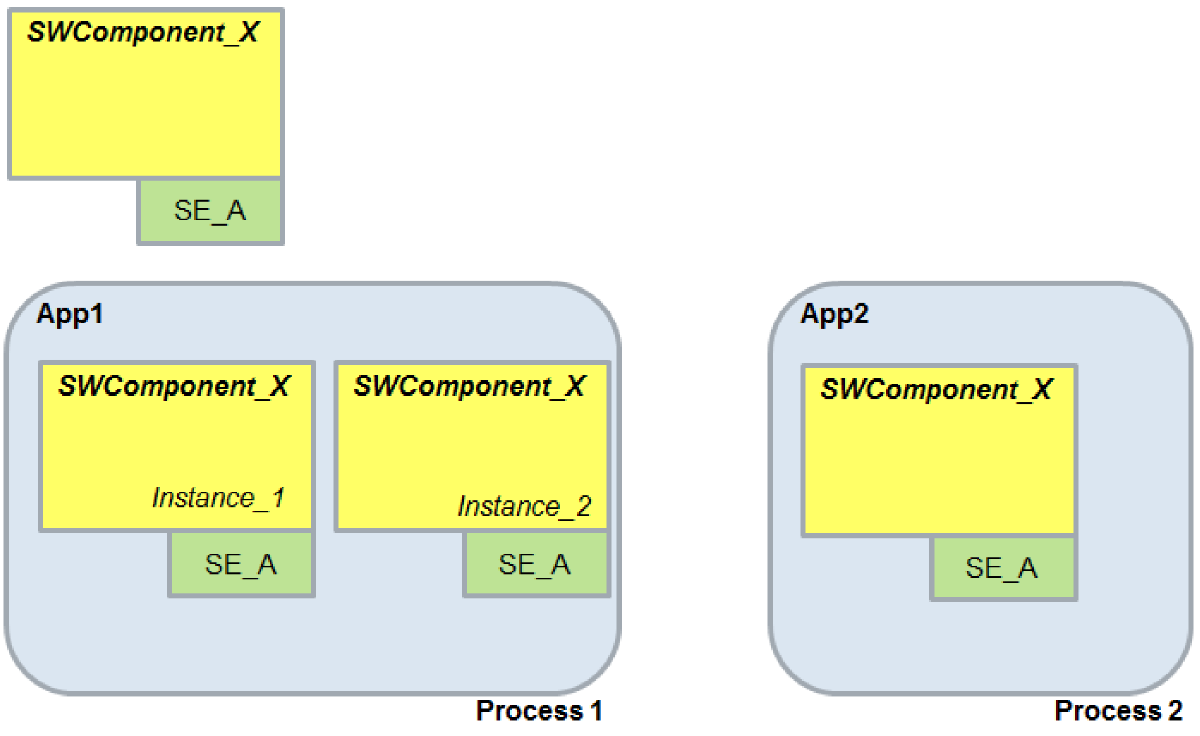

Figure 7.2: Example of multiple instance of the same Supervised Entity

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 explicitely 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 respresenting the Supervised Entity are created:

- <context>/.../App1/SW_Component_X/SE_A_Inst1
- <context>/.../App1/SW_Component_X/SE_A_Inst2
- <context>/.../App2/SW_Component_X/SE_A

These names shall be provided as Instance Specifiers during the creation of the Supervised Entities.

8 Platform Health Management API specification

8.1 C++ language binding

8.1.1 API Header files

This section describes the header files of the `ara::phm` API.

The input for the generated header files of `Platform Health Management` are the AUTOSAR metamodel classes within the `PlatformHealthManagementContribution` description, as defined in the AUTOSAR Adaptive Methodology Specification [9].

The header files include the types that provide the `ara::phm` API. Such type definitions are used in the standardized interfaces defined in chapter 8.1.3.

There are following classes:

1. `LocalSupervisionStatus` - an enum class representing `Local Supervision Status`.
2. `GlobalSupervisionStatus` - an enum class representing `Global Supervision Status`.
3. `SupervisedEntity` - a class to report `Checkpoints`.
4. `HealthChannel` - a class to report `Health Statuses`.

[SWS_PHM_01101]{DRAFT} **Folder structure for header files** ⌈ The *header file*s shall be located within the folder:

`<folder>/ara/phm/`

where:
`<folder>` is the start folder for the ara::phm header files specific for a project or platform vendor. ⌋*(RS_PHM_00001)*

[SWS_PHM_01018]{DRAFT} **Header file namespace** ⌈ The C++ namespace for the data type definitions included by the *header files* shall be:

```
namespace ara {
namespace phm {
...
} // namespace phm
} // namespace ara
```

⌋*(RS_PHM_00002)*

[SWS_PHM_01013]{DRAFT} **Header file existence** ⌈ The platform health management shall provide the following header files:

1. PHM.h
2. LocalSupervisionStatus.h

3. GlobalSupervisionStatus.h
4. SupervisedEntity.h
5. HealthChannel.h

⌋*(RS_PHM_00001)*

It is not mandatory that all data type definitions are located directly in the above mentioned *header file directory*. Health Management implementation can also distribute the definitions into different header files, but at least all those header files need to be included into the above mentioned *header file directory* .

8.1.2 API Types

This chapter describes the standardized types provided by the `ara::phm` API.

8.1.2.1 Checkpoint

[SWS_PHM_01138]{DRAFT} **Definition for `Checkpoint` data type** ⌈

```
using ara::phm::Checkpoint = typedef std::uint16_t
```

⌋*(RS_HM_09254, RS_PHM_00101)*

8.1.2.2 HealthStatus

[SWS_PHM_01139]{DRAFT} **Definition for `Health Status` data type** ⌈

```
using ara::phm::HealthStatus = typedef std::uint32_t
```

⌋*(RS_HM_09257, RS_PHM_00102)*

8.1.2.3 LocalSupervisionStatus

[SWS_PHM_01136]{DRAFT} **Definition of enumeration for `Local Supervision Statuss`** ⌈`Platform Health Management` shall provide a `LocalSupervisionStatus` enum class:

```
enum class LocalSupervisionStatus : uint8_t
{
        DEINIT,
        DEACTIVATED,
        OK,
        FAILED,
        EXPIRED
};
```

⌋*(RS_HM_09237)*

8.1.2.4 GlobalSupervisionStatus

[SWS_PHM_01137]{DRAFT} **Definition of enumeration for `Global Supervision Status`** ⌈`Platform Health Management` shall provide a `GlobalSupervisionStatus` enum class:

```
enum class GlobalSupervisionStatus : uint8_t
{
        DEINIT,
        DEACTIVATED,
        OK,
        FAILED,
        EXPIRED,
        STOPPED
};
```

⌋*(RS_HM_09237)*

8.1.2.5 SupervisedEntity

[SWS_PHM_01132]{DRAFT} **SupervisedEntity Class Template** ⌈ Platform Health Management shall provide a SupervisedEntity class template which shall provide a method to report Checkpoints and to retrieve the local and global supervision status corresponding to the related SupervisedEntity.

```
class SupervisedEntity
{
public:

   SupervisedEntity(
      ara::core::InstanceSpecifier const& instance_id);

   void ReportCheckpoint(Checkpoint checkpoint_id);

   LocalSupervisionStatus GetLocalSupervisionStatus();

   GlobalSupervisionStatus GetGlobalSupervisionStatus();
};
```

⌋*(RS_PHM_00003, RS_PHM_00101, RS_HM_09254)*

8.1.2.5.1 HealthChannel

[SWS_PHM_01122]{DRAFT} **HealthChannel Class Template** ⌈ Platform Health Management shall provide a HealthChannel class template which shall provide a method to report the Health Status.

```
class HealthChannel
{
public:

   HealthChannel(
      ara::core::InstanceSpecifier const& instance_id);

        void ReportHealthStatus(HealthStatus healthStatus_id);
};
```

⌋*(RS_PHM_00003, RS_PHM_00102, RS_HM_09257)*

8.1.2.6 Daisy Chaining Related Types (Non-generated)

Daisy chaining is not supported in this AUTOSAR release.

8.1.2.7 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 [10, 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.

8.1.2.8 E2E Related Data Types

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

8.1.3 API Reference

8.1.3.1 SupervisedEntity API

SupervisedEntity API can be used to report checkpoints or to query the status of a SupervisedEntity. It is possible to query a SupervisedEntity for which Checkpoints are reported, or not. So one can imagine a centralized error handler that queries all SupervisedEntitys by creating the SupervisedEntity objects and calling their getter methods.

8.1.3.1.1 Creation of a SupervisedEntity

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

```
SupervisedEntity(ara::core::InstanceSpecifier const& instance);
```

[SWS_PHM_01123]{DRAFT} ⌈

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

⌋*(RS_PHM_00101, RS_HM_09254, RS_HM_09240)*

8.1.3.1.2 ReportCheckpoint

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

```
void ReportCheckpoint(Checkpoint checkpointId);
```

[SWS_PHM_01127]{DRAFT} ⌈

Kind:	function
Symbol:	ara::phm::SupervisedEntity::ReportCheckpoint(Enum t)
Scope:	class ara::phm::SupervisedEntity

▽

△

Syntax:	`void ReportCheckpoint (Enum t);`	
Parameters (in):	t	checkpoint identifier.
Return value:	None	
Thread Safety:	tbd	
Header file:	#include "ara/phm/supervised_entity.h"	
Description:	Reports an occurrence of a Checkpoint.	

⌋*(RS_PHM_00101, RS_HM_09254)*

8.1.3.1.3 GetLocalSupervisionStatus

The Platform Health Management shall provide a method GetLocalSupervisionStatus, provided by `SupervisedEntity`.

```
LocalSupervisionStatus GetLocalSupervisionStatus();
```

Which shall return 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:	`LocalSupervisionStatus GetLocalSupervisionStatus ();`	
Return value:	LocalSupervisionStatus	the local supervision status.
Thread Safety:	tbd	
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)*

8.1.3.1.4 GetGlobalSupervisionStatus

The Platform Health Management shall provide a method GetGlobalSupervisionStatus, provided by `SupervisedEntity`.

```
GlobalSupervisionStatus GetGlobalSupervisionStatus();
```

Which shall return the current `Global Supervision Status` of this `SupervisedEntity`.

[SWS_PHM_01135]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::phm::SupervisedEntity::GetGlobalSupervisionStatus()	
Scope:	class ara::phm::SupervisedEntity	
Syntax:	`GlobalSupervisionStatus GetGlobalSupervisionStatus ();`	
Return value:	GlobalSupervisionStatus	the global supervision status.
Thread Safety:	tbd	
Header file:	#include "ara/phm/supervised_entity.h"	
Description:	returns the global supervision status that the supervised entity belongs to	

⌋*(RS_PHM_00101, RS_HM_09237)*

8.1.3.2 `HealthChannel` API

8.1.3.2.1 Creation of a `HealthChannel`

The Platform Health Management shall provide constructor for class `HealthChannel` indicating the instance specifier of the `HealthChannel`.

```
HealthChannel(ara::core::InstanceSpecifier const& instance);
```

[SWS_PHM_00457]{DRAFT} ⌈

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

⌋*(RS_PHM_00102, RS_HM_09257, RS_HM_09240)*

8.1.3.2.2 ReportHealthStatus

The Platform Health Management shall provide a method ReportHealthStatus, provided by `HealthChannel`.

```
void ReportHealthStatus(HealthStatus status);
```

Where Enum is defined by the class template `HealthChannel`

[SWS_PHM_01128]{DRAFT} ⌈

Kind:	function	
Symbol:	ara::phm::HealthChannel::ReportHealthStatus(Enum t)	
Scope:	class ara::phm::HealthChannel	
Syntax:	`void ReportHealthStatus (Enum t);`	
Parameters (in):	t	the Health Status.
Return value:	None	
Thread Safety:	tbd	
Header file:	#include "ara/phm/health_channel.h"	
Description:	Reports a Health Status.	

⌋*(RS_PHM_00102, RS_HM_09257)*

8.1.3.3 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)*

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

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.

Class	**PhmSupervisedEntityInterface**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface			
Note	This meta-class provides the ability to implement a PortInterface for interaction with the Platform Health Management Supervised Entity. **Tags:** atp.Status=draft atp.recommendedPackage=PlatformHealthManagementInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PlatformHealthManagementInterface*, *Port Interface*, *Referrable*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
checkpoint	PhmCheckpoint	*	aggr	Defines the set of checkpoints which can be reported on this supervised entity. **Tags:** atp.Status=draft

Table D.1: PhmSupervisedEntityInterface

Class	**PlatformHealthManagementContribution**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::PlatformHealthManagement			

Note	This element defines a contribution to the Platform Health Management.
	
	Tags: atp.ManifestKind=ExecutionManifest
	atp.Status=draft
	atp.recommendedPackage=PlatformHealthManagementContributions			

Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadablePackageElement*			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
action	PhmAction	*	aggr	Collection of Actions and ActionLists in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft xml.sequenceOffset=60
arbitration	PhmArbitration	*	aggr	Collection of Arbitrations in the context of a Platform HealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft xml.sequenceOffset=50
checkpoint	SupervisionCheckpoint	*	aggr	Collection of checkpoints in the context of a Platform HealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft xml.sequenceOffset=10
global Supervision	GlobalSupervision	*	aggr	Collection of GlobalSupervisions in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft xml.sequenceOffset=30
healthChannel	HealthChannel	*	aggr	Collection of HealthChannels in the context of a Platform HealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft xml.sequenceOffset=40
local Supervision	LocalSupervision	*	aggr	Collection of LocalSupervisions in the context of a PlatformHealthManagementContribution. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=shortName atp.Status=draft xml.sequenceOffset=20

Table D.2: PlatformHealthManagementContribution

Class	***PlatformHealthManagementInterface*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::ApplicationDesign::PortInterface
Note	This meta-class provides the abstract ability to define a PortInterface for the interaction with Platform Health Management. **Tags:** atp.Status=draft

▽

△

Class	***PlatformHealthManagementInterface*** (abstract)			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Subclasses	PhmHealthChannelInterface, PhmSupervisedEntityInterface			
Attribute	***Type***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table D.3: PlatformHealthManagementInterface