Document Title	Requirements on Health Monitoring
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	878

Document Status	published
Part of AUTOSAR Standard	Foundation
Part of Standard Release	R19-11

Document Change History

Date	Release	Changed by	Description
2019-11-28	R19-11	AUTOSAR Release Management	• Editorial changes • Changed Document Status from Final to published
2019-03-29	1.5.1	AUTOSAR Release Management	• Editorial changes
2018-10-31	1.5.0	AUTOSAR Release Management	• Editorial changes
2018-03-29	1.4.0	AUTOSAR Release Management	• Editorial changes
2017-12-08	1.3.0	AUTOSAR Release Management	• No content changes
2017-10-27	1.2.0	AUTOSAR Release Management	• Initial Release

Disclaimer

Table of Contents

1 Scope of this document

This document specifies requirements on the Health Monitoring.

For this release, this document applies to Adaptive Platform only: the alignment with Classic Platform will be done in a subsequent release. The "Applies to" fields in chapter 6 should be ignored. The alignment with Classic Platform will be done in a subsequent release."

Health Monitoring is required by [1] (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.

Health monitoring has the following error detection functions:

1. Alive supervision - checking if Checkpoints happens with a correct frequency
2. Deadline supervision - checking the delta time between two Checkpoints
3. Logical supervision - checking for correct sequence of execution of Checkpoints
4. Health status supervision - checking if Health Status information is valid

Health monitoring provides also a configurable error handling mechanism in order to recover from errors detected by the previous supervision functions.

The Health Supervision is supposed to be implemented by AUTOSAR classic platform and AUTOSAR adaptive platform. It may be implemented by other platforms as well.

The Health Supervision itself is specified in [2, SWS Health Monitoring], which specifies the implementation-independent behavior/algorithm of the four supervision functions.

2 How to read this document

2.1 Conventions to be used

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template, chapter Support for Traceability [3].

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see Standardization Template, chapter Support for Traceability [3].

3 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 [4, AUTOSAR glossary].

Abbreviation:	Description:
CM	AUTOSAR Adaptive Communication Management
DM	AUTOSAR Adaptive Diagnostic Management
PHM	Platform Health Management
SE	Supervised Entity

Acronym:	Description:
Alive Counter	An independent data resource in context of a Checkpoint to track and handle its amount of Alive Indications.
Alive Indication	An indication of a Supervised Entity to signal its aliveness by calling a checkpoint used for Alive Supervision.
Alive Supervision	Mechanism to check the timing constraints of cyclic Supervised Entitys to be within the configured min and max limits.
Checkpoint	A point in the control flow of a Supervised Entity where the activity is reported.
Deadline End Checkpoint	A Checkpoint for which Deadline Supervision is configured and which is a ending point for a particular Transition. It is possible that a Checkpoint is both a Deadline Start Checkpoint and Deadline End Checkpoint - if Deadline Supervision is chained.
Deadline Start Checkpoint	A Checkpoint for which Deadline Supervision is configured and which is a starting point for a particular Transition.
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.
Expired Supervision Cycle	A Supervision Cycle where the Alive Supervision has failed its two escalation steps (Alive Counter fails the expected amount of Alive Indications (including tolerances) more often than the allowed amount of failed reference cycles).
Failed Supervision Reference Cycle	A Supervision Reference Cycle that ends with a detected deviation (including tolerances) between the Alive Counter and the expected amount of Alive Indications.
Global Supervision Status	Status that summarizes the Local Supervision Status of all Supervised Entities 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 Channel Supervision	Kind of supervision that checks if the health indicators registered by the supervised software are within the tolerances/limits.
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 an application. 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.
Supervised Entity Identifier	An Identifier that identifies uniquely a Supervised Entity within an Application.
Supervision Counter	An independent data resource in context of a Supervised Entity which is updated during each supervision cycle and which is used by the `Alive Supervision` algorithm to perform the check against counted Alive Indications.
Supervision Cycle	The time period in which the cyclic `Alive Supervision` is performed.

Supervision Mode	An overall state of a microcontroller or virtual machine. Modes are mutually exclusive and all Supervised Entities are in the same Supervision Mode. A mode can be e.g. Startup, Shutdown, Low power.
Supervision Reference Cycle	The amount of Supervision Cycles to be used as reference by the `Alive Supervision` to perform the check of counted Alive Indications (individually for each Supervised Entity).

Table 3.1: Acronyms

4 Functional overview

The Health Monitoring is intended to supervise the execution of supervised entities with respect to timing constraints (alive and deadline supervision) and with respect to the required sequence of execution (logical supervision) and with respect to their health (health supervision).

The Health Monitoring can be performed on supervised entities, which can be any software components or groups of software components or Adaptive Applications.

The following features are provided by the Health Monitoring:

1. Supervision of multiple individual supervised entities located on the microprocessor or virtual machine, having independent supervision constraints.
2. Support for parallel and concurrent execution of supervised entities and for multiple instantiation.
3. Support for different modes of operation, with different behavior of software components depending on mode.
4. Support for multiple hardware watchdogs.
5. Support for several error handling mechanisms.

5 Requirements traceability

The following table references the features specified in [5] and links to the fulfillments of these.

Feature	Description	Satisfied by
[RS_Main_00001]	AUTOSAR shall provide a software platform for embedded real-time systems	[RS_HM_09028] [RS_HM_09125] [RS_HM_09159] [RS_HM_09163] [RS_HM_09169] [RS_HM_09222] [RS_HM_09226] [RS_HM_09235] [RS_HM_09237] [RS_HM_09240] [RS_HM_09241] [RS_HM_09242] [RS_HM_09243] [RS_HM_09244] [RS_HM_09245] [RS_HM_09246] [RS_HM_09247] [RS_HM_09248] [RS_HM_09249] [RS_HM_09250] [RS_HM_09251] [RS_HM_09253] [RS_HM_09254] [RS_HM_09255] [RS_HM_09257]
[RS_Main_00010]	AUTOSAR shall support the development of safety related systems	[RS_HM_09028] [RS_HM_09125] [RS_HM_09159] [RS_HM_09163] [RS_HM_09169] [RS_HM_09222] [RS_HM_09226] [RS_HM_09235] [RS_HM_09237] [RS_HM_09240] [RS_HM_09241] [RS_HM_09242] [RS_HM_09243] [RS_HM_09244] [RS_HM_09245] [RS_HM_09246] [RS_HM_09247] [RS_HM_09248] [RS_HM_09249] [RS_HM_09250] [RS_HM_09251] [RS_HM_09253] [RS_HM_09254] [RS_HM_09255]

		[RS_HM_09257]
[RS_Main_00011]	AUTOSAR shall support the development of reliable systems	[RS_HM_09028] [RS_HM_09125] [RS_HM_09159] [RS_HM_09163] [RS_HM_09169] [RS_HM_09222] [RS_HM_09226] [RS_HM_09235] [RS_HM_09237] [RS_HM_09240] [RS_HM_09241] [RS_HM_09242] [RS_HM_09243] [RS_HM_09244] [RS_HM_09245] [RS_HM_09246] [RS_HM_09247] [RS_HM_09248] [RS_HM_09249] [RS_HM_09250] [RS_HM_09251] [RS_HM_09253] [RS_HM_09254] [RS_HM_09255] [RS_HM_09257]
[RS_Main_00340]	AUTOSAR shall support the continuous timing requirement analysis	[RS_HM_09028] [RS_HM_09125] [RS_HM_09159] [RS_HM_09163] [RS_HM_09169] [RS_HM_09222] [RS_HM_09226] [RS_HM_09235] [RS_HM_09237] [RS_HM_09240] [RS_HM_09241] [RS_HM_09242] [RS_HM_09243] [RS_HM_09244] [RS_HM_09245] [RS_HM_09246] [RS_HM_09247] [RS_HM_09248] [RS_HM_09249] [RS_HM_09250] [RS_HM_09251] [RS_HM_09253] [RS_HM_09254] [RS_HM_09255] [RS_HM_09257]

[RS_Main_00435]	AUTOSAR shall support automotive microcontrollers	[RS_HM_09028] [RS_HM_09169] [RS_HM_09226] [RS_HM_09244] [RS_HM_09245] [RS_HM_09246] [RS_HM_09247] [RS_HM_09248] [RS_HM_09250]

6 Requirements specification

6.1 Functional requirements

6.1.1 Supervision functions

[RS_HM_09222] Health Monitoring shall provide a Logical Supervision ⌈

Type:	draft
Description:	Health Monitoring shall check if the sequence of Checkpoints in a Supervised Entity at runtime is the same as the one that is specified. This shall include: • start of if/else branch (decision node): exactly one of the code branches shall be entered, the choice is runtime-specific depending on logical condition • end of if/else branch (merge node): exactly one of the branches shall be reached so that the join is performed • fork of the flow into concurrent execution (fork node): all concurrent branches shall be entered • join of the flow of concurrent execution (join node): all concurrent branches shall be reached so that the join is performed.
Rationale:	To detect if the sequence in the execution is the same as specified/designed.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	Supervision of any software components: application software components or platform components (e.g. execution manager, state manager).
Supporting Material:	–

⌋*(RS_Main_00001, RS_Main_00010, RS_Main_00011, RS_Main_00340)*

[RS_HM_09125] Health Monitoring shall provide an Alive Supervision ⌈

Type:	draft
Description:	Health Monitoring shall check if the frequency of reaching a given Checkpoint in a Supervised Entity matches specified limits.
Rationale:	To detect if a periodic function is executed periodically according to specification/design.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	–

▽

△

Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*)

[RS_HM_09235] Health Monitoring shall provide a Deadline Supervision ⌈

Type:	draft
Description:	Health Monitoring shall check if the elapsed time between two Checkpoints is within the specified min and max limits, including the detection if the second Checkpoint never arrives.
Rationale:	To detect timeouts or loss of deadlines.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	–
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*)

[RS_HM_09255] Health Monitoring shall provide a Health Channel Supervision ⌈

Type:	draft
Description:	Health Monitoring shall check if the health indicators registered by the supervised software are within the tolerances/limits.
Rationale:	To detect errors like: over-temperature, high bus load, low memory.
Dependencies:	–
AppliesTo:	AP
Use Case:	–
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*)

6.1.2 Interface to Supervised Entities

[RS_HM_09254] Health Monitoring shall provide an interface to Supervised Entities to report the currently reached Checkpoint. ⌈

Type:	draft
Description:	Health Monitoring shall provide an interface to Supervised Entities to report the currently reached Checkpoint by a Supervised Entity, taking into account that a given code location can be achieved from different processes, threads or executed on different cores.
Rationale:	This is the only way how an application can report its progress.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	–
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*)

[RS_HM_09257] Health Monitoring shall provide an interface to Supervised Entities to report their health status. ⌈

Type:	draft
Description:	Health Monitoring shall provide an interface to Supervised Entities to report their health.
Rationale:	Health Status information can provide useful information on the correct behavior of the system
Dependencies:	–
AppliesTo:	AP
Use Case:	Health Monitoring can verify the Health Status of the Supervised Entities and take the appropriate actions.
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*)

[RS_HM_09237] Health Monitoring shall provide an interface to Supervised Entities informing them about their Supervision State. ⌈

Type:	draft
Description:	Health Monitoring shall provide an interface informing about Supervision State, including: • which Supervised Entity failed • current Local Supervision Status of each Supervised Entity • current Global Supervision Status of microcontroller or virtual machine ▽

▽

△

	△ • reason why the last error reactions were performed • upcoming microcontroller or virtual machine reset This shall be available by notification and by polling.
Rationale:	Some applications need to know their health/state.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	Reporting of OK/Failed to Supervised Entities.
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*)

6.1.3 Features related to supervision functions

[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. ⌈

Type:	draft
Description:	Health Monitoring shall support supervision modes of Supervised entities, where • a Supervised Entity has possibly a different behavior in each Supervision Mode • a Supervision Mode is shared across all Supervised Entities • a Supervision Mode is defined as a flat or hierarchical state machine.
Rationale:	In different modes, a Supervised Entity can have a different behavior, e.g. other execution path, other timing.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	In "init" mode, the function init() is supervised with its Checkpoints related to the "init" mode. In "run" mode, the run() function is supervised with its Checkpoints related to the "run" mode. The Supervision Modes are realized in AP as states of Execution Management.
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*)

[RS_HM_09240] Health Monitoring shall support multiple occurrences of the same Supervised Entity. ⌈

Type:	draft
Description:	Health Monitoring shall support multiple occurrences of the same Supervised Entity. Health Monitoring shall support a variable number of Supervised Entity occurrences at runtime.
Rationale:	An application or component can be instantiated multiple times
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	Multiple occurrences of the same software component or application launched multiple times, as separate processes or threads.
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*)

[RS_HM_09241] Health Monitoring shall support multiple instances of Checkpoints in a Supervised Entity occurrence. ⌈

Type:	draft
Description:	Health Monitoring shall support multiple instances of Checkpoints in a Supervised Entity occurrence, where the number of Checkpoint instances at runtime may be variable.
Rationale:	An application or component containing a checkpoint can be instantiated multiple times
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	Parallel/concurrent execution of the same worker threads that execute the same code.
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*)

[RS_HM_09242] Health Monitoring shall support the supervision within and across Supervised Entities. ⌈

Type:	draft
Description:	Health Monitoring shall support the supervision (logical, alive and deadline) within one Supervised Entity and across different Supervised Entities.
Rationale:	–
Dependencies:	–
AppliesTo:	CP, AP

▽

△

Use Case:	Activity chains across several activities, where different activities belong to one or to different POSIX processes.
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*)

[RS_HM_09243] Health Monitoring shall support the supervision of concurrent and parallel Supervised Entities. ⌈

Type:	draft
Description:	Health Monitoring shall support the supervision of Supervised Entities: • with parallel/concurrent execution • preempted by other Supervised Entities or by any other software • executed on multiple cores or CPUs.
Rationale:	Health Monitoring shall work also for systems with parallel and concurrent execution
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	Systems with parallel execution on multi-core processors.
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*)

[RS_HM_09163] Health Monitoring shall provide configurable tolerances for detected errors and configurable delays of error reactions. ⌈

Type:	draft
Description:	Health Monitoring shall provide configurable tolerances for detected errors and configurable delays of error reactions.
Rationale:	Giving the time to the whole software to prepare properly to the upcoming recovery actions, e.g. to the reset.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	–
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*)

6.1.4 Features related to support for watchdogs

This section specifies requirements for support of watchdogs. A watchdog is typically a simple hardware entity that expects a simple certain information within a defined time period. It can also be realized by a more complex system, e.g. by another microcontroller.

[RS_HM_09244] Health Monitoring shall support timeout watchdogs. ⌈

Type:	draft
Description:	Health Monitoring shall support simple timeout watchdogs, i.e. watchdogs that require that specific value(s) are written within a defined timeout.
Rationale:	Such hardware watchdogs are broadly available. Moreover, systems exist that apply several watchdogs as a redundancy measure (with a simple timeout watchdog and a complex question-answer watchdog).
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	–
Supporting Material:	–

⌋*(RS_Main_00001, RS_Main_00010, RS_Main_00011, RS_Main_00340, RS_Main_00435)*

[RS_HM_09245] Health Monitoring shall support window watchdogs. ⌈

Type:	draft
Description:	Health Monitoring shall support window watchdogs, i.e. where the watchdog requires a correct value to be written within a defined min/max time window.
Rationale:	Window watchdogs are broadly used in automotive systems.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	System using a window watchdog
Supporting Material:	–

⌋*(RS_Main_00001, RS_Main_00010, RS_Main_00011, RS_Main_00340, RS_Main_00435)*

[RS_HM_09246] Health Monitoring shall support question-answer watchdogs. ⌈

Type:	draft
Description:	Health Monitoring shall support question-answer watchdogs, i.e. where the response provided to the watchdog depends on question from the watchdog and from the current Health Monitoring results.
Rationale:	Using systems with such a watchdog.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	The question-answer watchdog provides a random value as question, which is used as a seed to the Health Monitoring. The result of the supervision - the signature - is returned to the external watchdog as answer. Only if the answer is sent in time and matches the expected response, the external watchdog is serviced correctly and sends out the next question.
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*, *RS_Main_-00435*)

[RS_HM_09247] Health Monitoring shall support modes of the hardware watchdogs. ⌈

Type:	draft
Description:	Health Monitoring shall support hardware watchdog modes, where by hardware watchdog mode it is meant the set of defined hardware options like current timeout value.
Rationale:	A watchdog can provide modes like: normal, low, off, sleep.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	–
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*, *RS_Main_-00435*)

[RS_HM_09248] Health Monitoring shall support different watchdog realizations. ⌈

Type:	draft

▽

△

Description:	Health Monitoring shall support different watchdog realizations, including, but not limited to: • internal hardware watchdog (in the microcontroller) • external hardware watchdog • separate dedicated chip (ASIC) • an application on a separate microcontroller
Rationale:	Different watchdog realizations already exist on the market.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	–
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*, *RS_Main_-00435*)

[RS_HM_09028] Health Monitoring shall support multiple watchdogs ⌈

Type:	draft
Description:	Health Monitoring shall support multiple watchdogs, of the same or different type, with the same or different configuration.
Rationale:	There are microprocesors including both an internal and an external watchdog for monitoring the system, as a redundancy mechanism.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	In case the internal watchdog uses the same clock as the CPU, then due to the usage of the same clock, the internal watchdog doesn't recognize the "hang-up" of a system. To achieve a higher robustness an external watchdog is used too.
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*, *RS_Main_-00435*)

6.1.5 Supported error handling mechanisms

[RS_HM_09159] Health Monitoring shall be able to report supervision errors. ⌈

Type:	draft
Description:	As a possible error reaction, Health Monitoring shall report supervision errors, providing information on what kind of error was detected.
Rationale:	Reporting of errors is needed so that they can be logged and analyzed or so that a centralized error reaction can take place.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	Reporting that a Supervised Entity violated its Alive Supervision, but still within limits. Reporting that the entire microcontroller is in such a bad state that it needs to be reset. Handling of the error reported by Health Monitoring by others.
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*)

[RS_HM_09226] Health Monitoring shall be able to wrongly trigger the serviced watchdogs. ⌈

Type:	draft
Description:	As a possible error reaction, Health Monitoring shall be able to wrongly trigger the serviced watchdogs.
Rationale:	In order to provide a quick reset of the microprocessor.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	Typical error reaction provided by hardware watchdogs is a quick reset of the microprocessor. A typical wrong triggering of watchdogs includes: • Immediate generation of a answer to a question (in case of a question-answer watchdog) • Immediate generation of a wrong trigger/notification to the watchdog (timeout watchdog and window watchdog) • Generation of no answer (timeout watchdog and window watchdog)
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*, *RS_Main_-00435*)

[RS_HM_09169] Health Monitoring shall be able to trigger microcontroller reset. ⌈

Type:	draft
Description:	As a possible error reaction, Health Monitoring shall trigger microcontroller reset, including, but not limited to: • Clean microcontroller reset (e.g. with closing all services, closing sockets) • Quick microcontroller reset.
Rationale:	Apart from wrong triggering of watchdog, this is the second main reaction that Health Monitoring can perform to recover from the faulty system state.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	Health manager requesting machine state manager to perform the reset.
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*, *RS_Main_00435*)

[RS_HM_09250] Health Monitoring shall be able to request a change of Supervision Mode of the virtual machine or microcontroller. ⌈

Type:	draft
Description:	As a possible error reaction, Health Monitoring shall request a change of the Supervision Mode of the virtual machine or microcontroller.
Rationale:	An error reaction which is less drastic than a reset may be another change of state.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	Switch from "low power mode" to "normal" on error detected.
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*, *RS_Main_00435*)

[RS_HM_09251] Health Monitoring shall be able to request a restart a Supervised entity. ⌈

Type:	draft
Description:	As a possible error reaction, Health Monitoring shall be able to request a restart a faulty Supervised Entity occurrence.

▽

△

Rationale:	A restart of faulty supervised entity may bring it in working condition once again.
Dependencies:	–
AppliesTo:	AP
Use Case:	In Adaptive Platform, Health manager requesting to restart the failed software.
Supporting Material:	–

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*)

6.2 Non functional requirements

[RS_HM_09249] Health Monitoring shall support building safety-related systems.
⌈

Type:	draft
Description:	Health Monitoring shall support building safety-related systems compliant to ISO 26262.
Rationale:	Health Monitoring shall not prevent but facilitate the implemantation of safe systems compliant with ISO 26262.
Dependencies:	–
AppliesTo:	CP, AP
Use Case:	Building driving assistance systems.
Supporting Material:	[1, ISO 26262]

⌋(*RS_Main_00001*, *RS_Main_00010*, *RS_Main_00011*, *RS_Main_00340*)

7 References

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

[2] Specification of Health Monitoring
AUTOSAR_SWS_HealthMonitoring

[3] Standardization Template
AUTOSAR_TPS_StandardizationTemplate

[4] Glossary
AUTOSAR_TR_Glossary

[5] Main Requirements
AUTOSAR_RS_Main