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

Document Status	Final
Part of AUTOSAR Standard	Foundation
Part of Standard Release	1.3.0

Document Change History

Date	Release	Changed by	Description
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 4

2 How to read this document 5

2.1 Conventions to be used 5

3 Acronyms and abbreviations 6

4 Functional overview 8

5 Requirements traceability 9

6 Requirements specification 12

6.1 Functional requirements 12

6.1.1 Supervision functions 12

6.1.2 Interface to supervised entities 13

6.1.3 Features related to supervision functions 15

6.1.4 Features related to support for watchdogs 17

6.1.5 Supported error handling mechanisms 20

6.2 Non functional requirements 22

7 References 23

1 Scope of this document

This document specifies requirements on the Health Monitoring. Health monitoring has the following error detection functions:

1. Alive supervision
2. Deadline supervision
3. Logical supervision
4. Health/status supervision - fixme let's find the name for this.

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

The Health Supervision itself is specified in PRS 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 [1].

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 [1].

3 Acronyms and abbreviations

The glossary below includes acronyms and abbreviations relevant to AP_RS_WatchdogSupervision that are not included in the AUTOSAR Glossary [2].

Abbreviation / 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 provided by a Checkpoint of a Supervised Entity to signal its aliveness.
Alive Supervision	Kind of supervision that checks if a Supervised Entity executed in a correct frequency.
Checkpoint	A point in the control flow of a Supervised Entity where the activity is reported.
Deadline Supervision	Kind of supervision that checks if the execution time between two Checkpoints is within minimum/maximum time limit.
Deadline Start Checkpoint	A Checkpoint for which Deadline Supervision is configured and which is a starting point for a particular Deadline Supervision.
Deadline End Checkpoint	A Checkpoint for which Deadline Supervision is configured and which is a ending point for a particular Deadline Supervision. It is possible that a Checkpoint is both a Deadline Start Check-point and Deadline End Checkpoint - if Deadline Supervision is chained.
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 al-lowed amount of failed reference cycles).
Global Supervision Status	Status that summarizes the Local Supervision Status of all Su-pervised Entities.
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.
External Graph	Graph that may involve more than one Supervised Entity.
External Transition	An External Transition is a transition between two Checkpoints, where the Checkpoints belong to different Supervised Entities.
Failed Supervision Reference Cycle	A Supervision Reference Cycle that ends with a detected devi-ation (including tolerances) between the Alive Counter and the expected amount of Alive Indications.

Local Supervision Status	Status that represents the current result of Alive Supervision, Deadline Supervision and Logical Supervision of a single Supervised Entity.
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).
Internal Graph	Graph that may not span over several Supervised Entities.
Internal Transition	An Internal Transition is a transition between two Checkpoints of a Supervised Entity.
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.
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 vittual 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).
SE	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.

The following features are provided by the Health Monitoring:

1. Supervision of multiple individual supervised entities located on the microcontroller, 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 [3] and links to the fulfillments of these.

Feature	Description	Satisfied by
[RS_Main_00001]	AUTOSAR shall provide a software platform for deeply embedded 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_09256] [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_09256] [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_09256] [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_09256] [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] Watchdog supervision shall provide a logical supervision ⌈

Type:	draft
Description:	Watchdog supervision shall provide a logical supervision, which means a 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 errors in scheduling, to detect interference or residual errors in supervised software. In particular to ensure that the sequence in the execution is the same as specified/designed.
Dependencies:	–
Applies to:	FO, 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] Watchdog supervision shall provide an alive supervision ⌈

Type:	draft
Description:	Watchdog supervision shall provide a alive supervision, by checking if the frequency of reaching a given checkpoint in a supervised entity matches specified limits.
Rationale:	To detect errors in scheduling, to detect interference or residual errors in supervised software. In particular, to check if a periodic function is executed periodically according to specification/design.
Dependencies:	–
Applies to:	FO, AP
Use Case:	–
Supporting Material:	–

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

[RS_HM_09235] Watchdog supervision shall provide a deadline supervision ⌈

Type:	draft
Description:	Watchdog supervision shall provide a deadline supervision, by checking 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 errors in scheduling, to detect interference or residual errors in supervised software. In particular, to detect timeouts or loss of deadlines.
Dependencies:	–
Applies to:	FO, AP
Use Case:	–
Supporting Material:	–

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

[RS_HM_09255] Watchdog supervision shall provide a health supervision ⌈

Type:	draft
Description:	Watchdog supervision shall provide a health supervision, by checking 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:	–
Applies to:	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] Watchdog supervision shall provide an interface to supervised entities to report the currently reached checkpoint. ⌈

Type:	draft
Description:	Watchdog supervision 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. The interface shall be restricted to authorized software.
Rationale:	This is the only way how an application can report its progress.
Dependencies:	–
Applies to:	FO, AP
Use Case:	–

Supporting Material:	–

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

[RS_HM_09256] Watchdog supervision shall provide an interface to supervised entities to report the a list of values together with a checkpoint. ⌈

Type:	draft
Description:	Watchdog supervision shall provide an interface to supervised entities to report the currently reached checkpoint by a supervised entity, together with a state that the supervised entity has at that checkpoint, that shall be represented as a list of 8-bit unsigned integers associated with each checkpoint.
Rationale:	The Watchdog supervision can use the states associated with the checkpoints to monitor them or to compare them with a parallel software instance.
Dependencies:	–
Applies to:	AP
Use Case:	–
Supporting Material:	–

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

[RS_HM_09257] Watchdog supervision shall provide an interface to supervised entities to report their health. ⌈

Type:	draft
Description:	Watchdog supervision shall provide an interface to supervised entities to report their health.
Rationale:	
Dependencies:	–
Applies to:	AP
Use Case:	–
Supporting Material:	–

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

[RS_HM_09237] Watchdog supervision shall provide an interface to supervised entities informing them about their supervision state. ⌈

Type:	draft

Description:	Watchdog supervision shall provide an interface informing about supervision state, including: • which supervised entity failed, i.e. which supervised entity violated its specification of logical, alive or deadline supervision. • current 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:	–
Applies to:	FO, 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] Watchdog supervision shall support mode-dependent behavior of Supervised Entities. ⌈

Type:	draft
Description:	Watchdog supervision shall support supervision modes of Supervised entities, where - a Supervised entity has a different behavior in each supervision mode - a Supervision mode is shared across all Supervised Entities. - Supervision mode is defined as a hierarchical state machine. Watchdog supervision shall support the supervision on the transition between Checkpoints belonging to the same or to different Supervision Modes.
Rationale:	In different modes, a Supervised Entity can have a different behavior, e.g. other execution path, other timing.
Dependencies:	–
Applies to:	FO, AP
Use Case:	In init mode the init() function is supervised with its checkpoints in run mode, the run() function with other checkpoints of the same Supervised Entity is supervised.
Supporting Material:	–

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

[RS_HM_09240] Watchdog supervision shall support multiple occurrences of the same supervised entity. ⌈

Type:	draft
Description:	Watchdog supervision shall support multiple occurrences of the same supervised entity. Watchdog supervision shall support a variable number of supervised entity occurrences at runtime.
Rationale:	–
Dependencies:	–
Applies to:	FO, AP
Use Case:	Multiple occurrences of the same software component launched multiple times, as separate processes.
Supporting Material:	–

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

[RS_HM_09241] Watchdog supervision shall support multiple instances of checkpoints in a supervised entity occurrence. ⌈

Type:	draft
Description:	Watchdog supervision shall support multiple instances of checkpoints in a supervised entity occurrence, where the number of checkpoint instances at runtime may be variable.
Rationale:	–
Dependencies:	–
Applies to:	FO, 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] Watchdog supervision shall support the supervision within and across supervised entities. ⌈

Type:	draft
Description:	Watchdog supervision shall support the supervision (logical, alive and deadline) within one supervised entity and across different supervised entities.
Rationale:	–
Dependencies:	–
Applies to:	FO, 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] Watchdog supervision shall support the supervision in concurrent and parallel execution of supervised entities ⌈

Type:	draft
Description:	Watchdog supervision shall support the supervision in concurrent and parallel execution, executed on one or multiple cores or CPUs. It shall support: • preemption of supervised entities • priorities of supervised entities, including dynamic prioritization
Rationale:	–
Dependencies:	–
Applies to:	FO, AP
Use Case:	Preemptive scheduling (where each process gets one time slice), scheduling with priorities (e.g. for high-prio tasks), scheduling covering priority ceiling protocol.
Supporting Material:	–

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

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

Type:	draft
Description:	Watchdog supervision 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 reset.
Dependencies:	–
Applies to:	FO, 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] Watchdog supervision shall support timeout watchdogs. ⌈

Type:	draft
Description:	Watchdog supervision shall support simple timeout watchdogs, i.e. watchdogs that require that a value (possibly an alternating value) is written within a defined timeout value.
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:	–
Applies to:	FO, AP
Use Case:	–
Supporting Material:	–

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

[RS_HM_09245] Watchdog supervision shall support window watchdogs. ⌈

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

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

[RS_HM_09246] Watchdog supervision shall support question-answer watchdogs. ⌈

Type:	draft
Description:	Watchdog supervision shall support question-answer watchdogs, i.e. where the response provided to the watchdog depends on question from the watchdog and from the current Watchdog supervision results.
Rationale:	–
Dependencies:	–
Applies to:	FO, AP
Use Case:	The question-answer watchdog provides a random value as question, which is used as a seed to the Watchdog supervision. 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] Watchdog supervision shall support modes of the hardware watchdogs. ⌈

Type:	draft

Description:	Watchdog supervision shall support hardware watchdog modes, where by hardware watchdog mode it is meant the set of defined hardware options like current timeout value.
Rationale:	–
Dependencies:	–
Applies to:	FO, AP
Use Case:	A watchdog can provide modes like: normal, low, off, sleep.
Supporting Material:	–

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

[RS_HM_09248] Watchdog supervision shall support different watchdog realizations. ⌈

Type:	draft
Description:	Watchdog supervision shall support different watchdog realizations: • 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. Each of the watchdogs has it advantages and disadvantages.
Dependencies:	–
Applies to:	FO, AP
Use Case:	–
Supporting Material:	–

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

[RS_HM_09028] Watchdog supervision shall support multiple watchdogs ⌈

Type:	draft
Description:	Watchdog supervision shall support multiple watchdogs, of the same or different type, with the same or different configuration.
Rationale:	There are microcontrollers including both an internal and an external watchdog for monitoring the system, as a redundancy mechanism.
Dependencies:	–
Applies to:	FO, 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] Watchdog supervision shall report supervision errors. ⌈

Type:	draft
Description:	As a possible error reaction, watchdog supervision 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:	–
Applies to:	FO, 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 Watchdog supervision by others.
Supporting Material:	–

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

[RS_HM_09226] Watchdog supervision shall be able to wrongly trigger the serviced watchdogs. ⌈

Type:	draft
Description:	As a possible error reaction, watchdog supervision shall wrongly trigger the serviced watchdogs.
Rationale:	–
Dependencies:	–
Applies to:	FO, AP
Use Case:	Typical error reaction provided by hardware watchdogs is a quick reset of the microcontroller. 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] Watchdog supervision shall be able to perform microcontroller reset. ⌈

Type:	draft
Description:	As a possible error reaction, watchdog supervision shall perform microcontroller reset, including: • Clean microcontroller reset (e.g. with closing all services, closing sockets) • Quick microcontroller reset for which the worst-case time can be determined
Rationale:	Apart from wrong triggering of watchdog, this is the second main reaction that Watchdog supervision can perform to recover from the faulty system state.
Dependencies:	–
Applies to:	FO, 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] Watchdog supervision shall be able to request a change in the virtual machine or microcontroller state. ⌈

Type:	draft
Description:	As a possible error reaction, Watchdog supervision shall request a change in the virtual machine or microcontroller state.
Rationale:	–
Dependencies:	–
Applies to:	FO, AP
Use Case:	Reset the microcontroller as a way to recover the error.
Supporting Material:	–

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

[RS_HM_09251] Watchdog supervision shall be able to request a restart a Supervised entity. ⌈

Type:	draft
Description:	As a possible error reaction, Watchdog supervision shall be able to request a restart a faulty supervised entity occurrence.
Rationale:	–
Dependencies:	–
Applies to:	FO, AP
Use Case:	In Adaptive Platform, watchdog 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] Watchdog supervision shall support building safety-related systems. ⌈

Type:	draft
Description:	Watchdog supervision shall support building safety-related systems compliant to ISO 26262.
Rationale:	–
Dependencies:	–
Applies to:	FO, AP
Use Case:	Building driving assistance systems.
Supporting Material:	ISO 26262

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

7 References

[1] Standardization Template
AUTOSAR_TPS_StandardizationTemplate

[2] Glossary
AUTOSAR_TR_Glossary

[3] Requirements on AUTOSAR Features
AUTOSAR_RS_Features