Document Title	Requirements on Execution Management
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	720

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	• Updated: RS_EM_00008 and RS_EM_00010
2018-10-31	18-10	AUTOSAR Release Management	• Removed: RS_EM_00003, RS_EM_00004, RS_EM_00110 and RS_EM_00111. • Added: [RS_EM_00014].
2018-03-29	18-03	AUTOSAR Release Management	• Removed: RS_EM_00006, RS_EM_00007 and RS_EM_00012 • Minor changes and document clean up
2017-10-27	17-10	AUTOSAR Release Management	• Minor changes, document clean up
2017-03-31	17-03	AUTOSAR Release Management	• Initial release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Scope of this document

This document specifies requirements of the AUTOSAR Adaptive Platform on the Execution Management. The motivation is to provide a standardized way to start, stop and police applications platform wide.

2 Conventions to be used

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template [1], chapter Support for Traceability.

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see Standardization Template [1], chapter Support for Traceability.

3 Acronyms and abbreviations

All technical terms used throughout this document – except the ones listed here – can be found in the official [2] AUTOSAR Glossary or [3] TPS Manifest Specification.

Term	Description
Process	A process is a loaded instance of an Executable to be executed on a Machine.
Self-terminating Process	A type of Process that self initiate termination procedure, please note that for a standard Process this procedure is initiated by Execution Management.
Execution Dependency	Dependencies between Executable instances can be configured to define a sequence for starting and terminating them.
Execution Management	The element of the AUTOSAR Adaptive Platform responsible for the ordered startup and shutdown of the AUTOSAR Adaptive Platform and the Applications.
State Management	The element of the Execution Management defining modes of operation for AUTOSAR Adaptive Platform. It allows flexible definition of functions which are active on the platform at any given time.
Identity and Access Management (IAM)	A Adaptive Platform Service within the AUTOSAR Adaptive Platform

Function Group	A Function Group is a set of coherent Processes, which need to be controlled consistently. Depending on the state of the Function Group, Processes are started or terminated. Processes can belong to more than one Function Group. "MachineState" is a Function Group with a predefined name, which is mainly used to control Machine lifecycle and Processes of platform level Applications. Other Function Groups are sort of general purpose tools used (for example) to control Processes of user level Applications.
Function Group State	The state of a Function Group (except "MachineState"). It defines a set of active Applications for any certain situation. The set of Function Groups and their states is machine specific and is deployed as part of the Machine Manifest.
Machine State	The state of Function Group "MachineState" with some predefined states (Startup/Shutdown/Restart).
Time Determinism	The results of a calculation are guaranteed to be available before a given deadline.
Data Determinism	The results of a calculation only depend on the input data and are reproducible, assuming a given initial internal state.
Full Determinism	Combination of Time and Data Determinism.
Communication Management	A Functional Cluster within the Adaptive Platform Foundation
Execution Manifest	Manifest file to configure execution of an Executable.
Machine Manifest	Manifest file to configure a Machine.
Operating System	Software responsible for managing Processes on a Machine and for providing an interface to hardware resources.
ResourceGroup	Configuration element to enable restrictions on resources uses by Adaptive Applications running in the group.
ExecutionClient	Adaptive Application interface to Execution Management.
DeterministicClient	Adaptive Application interface to Execution Management to support control of the process-internal cycle, a deterministic worker pool, activation time stamps and random numbers.
Platform Health Management	A Functional Cluster within the Adaptive Platform Foundation
Recovery Action	Actions defined by the integrator to control Adaptive Application error recovery.
Process State	Lifecycle state of a Process
Service Instance Manifest	Manifest file to configure Service usage of an Adaptive Application.
Trusted Platform	An execution platform supporting a continuous chain of trust from boot through to application supporting authentication (that all code executed is from the claimed source) and integrity validation (that prevents tampered code/data from being executed).

Table 3.1: Technical Terms

The following technical terms used throughout this document are defined in the official [2] AUTOSAR Glossary or [3] TPS Manifest Specification – they are repeated here for tracing purposes.

Term	**Description**

Adaptive Application	see [2] AUTOSAR Glossary
Application	see [2] AUTOSAR Glossary
AUTOSAR Adaptive Platform	see [2] AUTOSAR Glossary
Adaptive Platform Foundation	see [2] AUTOSAR Glossary
Manifest	see [2] AUTOSAR Glossary
Executable	see [2] AUTOSAR Glossary
Functional Cluster	see [2] AUTOSAR Glossary
Adaptive Platform Service	see [2] AUTOSAR Glossary
Machine	see [2] AUTOSAR Glossary
Service	see [2] AUTOSAR Glossary
Service Interface	see [2] AUTOSAR Glossary
Service Discovery	see [2] AUTOSAR Glossary

Table 3.2: Glossary-defined Technical Terms

3.1 Constraints and assumptions

3.1.1 Limitations

This chapter lists known limitations of Execution Management in terms of unimplemented requirements with the intent to provide an indication how Execution Management within the context of the AUTOSAR Adaptive Platform will evolve in future releases.

The following requirements are described within this document but not otherwise considered in this release:

- [RS_EM_00050] – System-wide coordination
- [RS_EM_00051] – External trigger conditions
- [RS_EM_00111] – Identification of Processes
- [RS_EM_00014] – Trusted Platform.

The functionality described above is subject to modification and will be considered for inclusion in a future release of this document.

3.1.2 Applicability to car domains

No restrictions to applicability.

4 Requirements Specification

4.1 Functional Overview

The AUTOSAR Adaptive Platform provides services to influence the lifecycle of Applications based on configuration. This document therefore includes requirements that determine the facilities provided by Execution Management to affect the machine-wide startup, shutdown and restart of an Application based on configuration.

Execution Management is responsible for all aspects of platform lifecycle management and application lifecycle management, including:

- Machine startup and shutdown.
  - Execution Management is the initial ("boot") process of the operating system.
- Required process hierarchy of started services, e.g., init and its child process.
  - after booting. The boot process in this case corresponds to machine init process.
- Provision of process isolation with each instance of an Executable managed as a single process.
- Startup and shutdown of Applications.
  - Loading Executable based on a defined Execution Dependency.
  - Specific requirements until starting an Executable main function (i.e. entry point)
- Privileges and use of access control
  - description and semantics of access control in manifest files
- State management
  - Conditions for the execution of Applications

4.2 Functional Requirements

This section describes all requirements driving the work to define Execution Management functionality.

4.2.1 Startup and Shutdown of Applications

[RS_EM_00002] Execution Management shall set-up one process for the execution of each Process. ⌈

Type:	draft
Description:	For each instance of an Executable, Execution Management shall allocate one POSIX process. Furthermore process specific properties (like priority, scheduling policy and access rights) shall be assigned based on the Execution Manifest.
Rationale:	Isolation of Executable instances from each other.
Dependencies:	–
Use Case:	Safety and security related Applications require isolation.
Supporting Material:	–

⌋*(RS_Main_00010, RS_Main_00049, RS_Main_00080, RS_Main_00320, RS_Main_00150, RS_Main_00420)*

[RS_EM_00014] Execution Management shall support a Trusted Platform. ⌈

Type:	draft
Description:	Execution Management shall ensure that integrity and authenticity are checked for all Executables and their corresponding Execution Management meta-data (i.e. processed Machine and Execution Manifests), and shall only allow starting Executables that passed validation check.
Rationale:	Execution Management takes over the responsibility from Operating System and/or boot loader for AUTOSAR Adaptive Platform startup and hence for keeping the platform trusted. After the boot process has finished, Execution Management is the only AUTOSAR Adaptive Platform entity allowed to start Executables and therefore responsible for the continuation of platform trust.
Dependencies:	–
Use Case:	Verify the integrity and authenticity of software deployed on AUTOSAR Adaptive Platform.
Supporting Material:	–

⌋*(RS_Main_00170, RS_Main_00514, RS_Main_00180)*

[RS_EM_00005] Execution Management shall support the configuration of OS resource budgets for Process and groups of Processes. ⌈

Type:	draft
Description:	Based on the `Execution Manifest`, `Execution Management` shall allocate OS resources to the `Process`. The allocation shall be possible for single `Process` and groups of `Processs`.
Rationale:	Real-time guarantees shall be defined
Dependencies:	–
Use Case:	Like `cgroups` (based on containers which contain one or more processes) and `ulimit`.
Supporting Material:	–

⌋(*RS_Main_00002*, *RS_Main_00010*, *RS_Main_00106*, *RS_Main_00340*, *RS_Main_00150*)

[RS_EM_00008] `Execution Management` shall support the binding of all threads of a given Process to a specified set of processor cores. ⌈

Type:	draft
Description:	`Execution Management` shall allow the binding of threads to specific set of processor cores based on configuration in the `Execution Manifest`. The binding granularity shall be at Process level.
Rationale:	Mechanism to influence load balancing, reaction times, and latencies.
Dependencies:	–
Use Case:	Assign two parallel threads to two processor cores to achieve true parallelism.
Supporting Material:	–

⌋(*RS_Main_00010*, *RS_Main_00050*, *RS_Main_00106*, *RS_Main_00320*, *RS_Main_00501*, *RS_Main_00150*)

[RS_EM_00009] Only `Execution Management` shall start `Processes`. ⌈

Type:	draft
Description:	`Execution Management` shall prevent `Processes` from directly starting other `Processes`.
Rationale:	`Execution Management` needs full control of starting applications to ensure required isolation of temporal and spatial properties. Only `Execution Management` shall start `Processes`.
Dependencies:	–
Use Case:	Segregation between applications with different safety and/or security properties.
Supporting Material:	–

⌋(*RS_Main_00010*, *RS_Main_00011*, *RS_Main_00049*, *RS_Main_00150*)

[RS_EM_00010] Execution Management shall support multiple instances of Executables. ⌈

Type:	draft
Description:	It shall be possible to start more than one Process from a single Executable. Instance specific information is described in Process startup configuration.
Rationale:	Avoid code duplication.
Dependencies:	–
Use Case:	Redundancy of an Executable by parallel execution of two instances.
Supporting Material:	–

⌋*(RS_Main_00002, RS_Main_00049, RS_Main_00106, RS_Main_00501)*

[RS_EM_00011] Execution Management shall support self-initiated graceful shutdown of Processes. ⌈

Type:	draft
Description:	Execution Management shall support self-initiated graceful shutdown of Processes.
Rationale:	Self-initiated graceful shutdown enables a Process to free allocated dedicated resources and inform other interacting entities about its shutdown (e.g. de-registering a service) to create a consistent state within the Machine/vehicle. Self-initiated Process shutdown is, by definition, only be initiated by the Process itself.
Dependencies:	–
Use Case:	The process of an Executable instance is finished and shuts down itself.
Supporting Material:	–

⌋*(RS_Main_00002, RS_Main_00049)*

[RS_EM_00013] Execution Management shall support configurable recovery actions. ⌈

Type:	draft
Description:	Execution Management shall support recovery actions (e.g. Application, Executable or Machine restart, degradation) in case an Executable deviates from normal behavior. The recovery action shall be configurable in the Execution Manifest.
Rationale:	The integrator defines how a Process should be recovered after deviation from normal behaviour. When such deviation is detected by Platform Health Monitor (PHM) that functional cluster requires support from Execution Managament to, for example, restart the Process.
Dependencies:	–

▽

△

Use Case:	Support PHM in recovering to a known good state when resource usage of a Process exceeds pre-defined limits.
Supporting Material:	–

⌋*(RS_Main_00010, RS_Main_00012)*

[RS_EM_00100] Execution Management shall support the ordered startup and shutdown of Processes. ⌈

Type:	draft
Description:	Execution Management shall support the ordered startup and shutdown of Executable instances.
Rationale:	Ensure that startup and shutdown dependencies between Executable instances are respected, if an execution dependency is specified in the Execution Manifest of an Executable instance. If no execution dependency is specified between Executable instances, they can be started and stopped in an arbitrary order.
Dependencies:	–
Use Case:	An Executable needs a specific functional cluster to be up and running before it can be started.
Supporting Material:	–

⌋*(RS_Main_00002, RS_Main_00049, RS_Main_00340, RS_Main_00460)*

4.2.2 Execution

[RS_EM_00050] Execution Management shall perform Machine-wide coordination of Processes. ⌈

Type:	draft
Description:	Execution Management shall provide an API for a Process to register its activities for being able to coordinate their execution.
Rationale:	Coordinated scheduling of activities across Executables.
Dependencies:	–
Use Case:	Usage of computation resources within the running Processes shall be managed in the Machine to ensure that activities can be coordinated across Processes. Registration enables Execution Management to form the necessary Machine-wide view for the coordination.
Supporting Material:	–

⌋*(RS_Main_00460)*

[RS_EM_00051] Execution Management shall provide APIs to the Process for configuring external trigger conditions for its activities. ⌈

Type:	draft
Description:	Execution Management shall provide an API for configuring the trigger conditions of registered activities.
Rationale:	Execution Management shall have the information when to schedule the activities.
Dependencies:	–
Use Case:	Execution on data receipt, sequencing of activity execution.
Supporting Material:	–

⌋*(RS_Main_00050, RS_Main_00060)*

[RS_EM_00052] Execution Management shall provide APIs to the Process for configuring cyclic triggering of its activities. ⌈

Type:	draft
Description:	Execution Management shall provide an API for configuring the cyclic triggering of registered activities.
Rationale:	Execution Management shall have the information when to schedule the activities.
Dependencies:	–
Use Case:	Cyclic execution of activities
Supporting Material:	–

⌋*(RS_Main_00050, RS_Main_00340)*

[RS_EM_00053] Execution Management shall provide APIs to the Process to support deterministic redundant execution of Processes. ⌈

Type:	draft
Description:	Execution Management shall provide APIs to support deterministic redundant execution of Processes.
Rationale:	High ASIL systems require safety mechanism like software lockstep to be implemented on non-automotive grade microprocessors. The redundant execution shall guarantee deterministic, i.e. reproducible results.
Dependencies:	–
Use Case:	Redundant execution of activities to implement software lockstep
Supporting Material:	–

⌋*(RS_Main_00010, RS_Main_00501)*

[RS_EM_00111] Execution Management shall assist identification of Processes during Machine runtime. ⌈

Type:	draft
Description:	Adaptive Applications shall be identifiable, for example by Identity and Access Management, during runtime so that access restrictions can be enforced. Execution Management spawns runtime processes based on Execution Manifest. Execution Management is qualified to assist AUTOSAR Adaptive Platform software, such as Identity and Access Management, by providing information about the link between runtime representation and modelled Process.
Rationale:	Adaptive Applications shall be identifiable by Identity and Access Management on the basis of their runtime representation as spawned by Execution Management.
Dependencies:	–
Use Case:	App A requests access on Service Interface. Identity and Access Management is able to retrieve runtime information of App A, e.g. POSIX pid or cryptographic token. Execution Management assists Identity and Access Management by resolving this runtime information to the Adaptive Application.
Supporting Material:	–

⌋*(RS_Main_00170, RS_Main_00514, RS_Main_00420)*

4.2.3 State Management

[RS_EM_00101] Execution Management shall support State Management functionality. ⌈

Type:	draft
Description:	Execution Management shall provide an interface to State Management to request a change in Function Group State.
Rationale:	To support the starting and stopping of Processes based on declared Function Group State dependencies, Execution Management provides an interface to request Function Group State (including Machine State) changes by the State Management fuctional cluster. In response to state change requests, Execution Management ensures that only the required set of Application Processes are running in any given operation conditions and therefore platform resources are saved for relevant Processes.
Dependencies:	–
Use Case:	Provide a mechanism to define modes of operation of the Machine.
Supporting Material:	–

⌋*(RS_Main_00460)*

[RS_EM_00103] Execution Management shall support Process lifecycle management. ⌈

Type:	draft
Description:	The lifecycle of a Process consists of its startup, running and terminating (shutdown) phases. As well as supporting transitions between these phases of the Process lifecycle, Execution Management should ensure that phases, e.g. the startup and shutdown, of Processes can be coordinated between groups of Processes which shall run in the same Machine State or Function Group State. Coordination and tracking of lifecycle phases enables Execution Management to ensure that Executable's Processes are fully established and running before other Processes which depend on their functionality can be started.
Rationale:	Coordination and tracking of lifecycle phases enables Execution Management to ensure that Executable Processes are fully established and running before other executable Processes which depend on their functionality can be started.
Dependencies:	–
Use Case:	
Supporting Material:	–

⌋*(RS_Main_00049, RS_Main_00050, RS_Main_00106, RS_Main_00460)*

4.2.4 Support for Diagnostics

Support for Diagnostics is handled by State Management and therefore the requirements are replaced by the ones from [4].

5 Requirements Tracing

The following tables reference the requirements specified in [5] 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. Likewise, an entry of [RS_EM_NA] indicates that the source requirement has been evaluated as "not applicable" to Execution Management.

Requirement	Description	Satisfied by
[RS_Main_00002]	AUTOSAR shall provide a software platform for high performance computing platforms	[RS_EM_00005] [RS_EM_00010] [RS_EM_00011] [RS_EM_00100]
[RS_Main_00010]	AUTOSAR shall support the development of safety related systems	[RS_EM_00002] [RS_EM_00005] [RS_EM_00008] [RS_EM_00009] [RS_EM_00013] [RS_EM_00053]
[RS_Main_00011]	AUTOSAR shall support the development of reliable systems	[RS_EM_00009]

[RS_Main_00012]	AUTOSAR shall provide a software platform to support the development of highly available systems	[RS_EM_00013]
[RS_Main_00026]	AUTOSAR shall support high speed and high bandwidth communication between executed SW	[RS_EM_NA]
[RS_Main_00030]	AUTOSAR shall support development processes for safety related systems	[RS_EM_NA]
[RS_Main_00049]	AUTOSAR shall provide an Execution Management for running multiple applications	[RS_EM_00002] [RS_EM_00009] [RS_EM_00010] [RS_EM_00011] [RS_EM_00100] [RS_EM_00103]
[RS_Main_00050]	AUTOSAR shall provide an Execution Framework towards applications to implement concurrent application internal control flows	[RS_EM_00008] [RS_EM_00051] [RS_EM_00052] [RS_EM_00103]
[RS_Main_00060]	AUTOSAR shall provide a standardized software interface for communication between Applications	[RS_EM_00051]
[RS_Main_00080]	AUTOSAR shall provide means to describe a component model for Application Software	[RS_EM_00002]
[RS_Main_00106]	AUTOSAR shall provide the possibility to extend the software with new SWCs without recompiling the platform foundation	[RS_EM_00005] [RS_EM_00008] [RS_EM_00010] [RS_EM_00103]
[RS_Main_00140]	AUTOSAR shall provide network independent communication mechanisms for applications	[RS_EM_NA]
[RS_Main_00150]	AUTOSAR shall support the deployment and reallocation of AUTOSAR Application Software	[RS_EM_00002] [RS_EM_00005] [RS_EM_00008] [RS_EM_00009]
[RS_Main_00160]	AUTOSAR shall provide means to describe interfaces of the entire system	[RS_EM_NA]
[RS_Main_00161]	AUTOSAR shall provide a unified way to describe software systems deployed to Adaptive and / or Classic platforms	[RS_EM_NA]
[RS_Main_00170]	AUTOSAR shall provide secure access to ECU data and services	[RS_EM_00014] [RS_EM_00111]
[RS_Main_00180]	AUTOSAR shall provide mechanisms to protect intellectual property in a shared development process	[RS_EM_00014]
[RS_Main_00190]	AUTOSAR shall support standardized interoperability with non-AUTOSAR software	[RS_EM_NA]

[RS_Main_00230]	AUTOSAR shall support network topologies including gateways	[RS_EM_NA]
[RS_Main_00250]	AUTOSAR methodology shall provide a predefinition of typical roles and activities	[RS_EM_NA]
[RS_Main_00261]	AUTOSAR shall provide means for calibration	[RS_EM_NA]
[RS_Main_00270]	AUTOSAR shall provide mitigation strategies towards new releases	[RS_EM_NA]
[RS_Main_00280]	AUTOSAR shall support standardized automotive communication protocols	[RS_EM_NA]
[RS_Main_00300]	AUTOSAR shall provide data exchange formats to support work-share in large inter and intra company development groups	[RS_EM_NA]
[RS_Main_00301]	AUTOSAR shall specify profiles for data exchange to support work-share in large inter- and intra-company development groups	[RS_EM_NA]
[RS_Main_00310]	AUTOSAR shall support hierarchical Application Software design methods	[RS_EM_NA]
[RS_Main_00320]	AUTOSAR shall provide formats to specify system development	[RS_EM_00002] [RS_EM_00008]
[RS_Main_00340]	AUTOSAR shall support the continuous timing requirement analysis	[RS_EM_00005] [RS_EM_00052] [RS_EM_00100]
[RS_Main_00350]	AUTOSAR specifications shall be analyzable and support according methods to demonstrate the achievement of safety related properties	[RS_EM_NA]
[RS_Main_00360]	AUTOSAR shall support variant management	[RS_EM_NA]
[RS_Main_00410]	AUTOSAR shall provide specifications for routines commonly used by Application Software to support sharing and optimization	[RS_EM_NA]
[RS_Main_00420]	AUTOSAR shall use established software standards and consolidate de-facto standards for basic software functionality	[RS_EM_00002] [RS_EM_00111]
[RS_Main_00440]	AUTOSAR shall standardize access to non-volatile memory	[RS_EM_NA]
[RS_Main_00445]	AUTOSAR shall standardize access to crypto-specific HW and SW	[RS_EM_NA]

[RS_Main_00460]	AUTOSAR shall standardize methods to organize mode management on Application, ECU and System level	[RS_EM_00050] [RS_EM_00100] [RS_EM_00101] [RS_EM_00103]
[RS_Main_00480]	AUTOSAR shall support the test of implementations	[RS_EM_NA]
[RS_Main_00490]	AUTOSAR processes shall be compliant to ISO26262	[RS_EM_NA]
[RS_Main_00491]	AUTOSAR shall provide means for logging	[RS_EM_NA]
[RS_Main_00500]	AUTOSAR shall provide naming conventions	[RS_EM_NA]
[RS_Main_00501]	AUTOSAR shall support redundancy concepts	[RS_EM_00008] [RS_EM_00010] [RS_EM_00053]
[RS_Main_00503]	AUTOSAR shall support change of communication and application software at runtime.	[RS_EM_NA]
[RS_Main_00507]	AUTOSAR shall reflect the stages of a software system development in a formal model description	[RS_EM_NA]
[RS_Main_00510]	AUTOSAR shall support secure onboard communication	[RS_EM_NA]
[RS_Main_00511]	AUTOSAR shall support virtualization	[RS_EM_NA]
[RS_Main_00513]	AUTOSAR shall support language bindings for different programming languages	[RS_EM_NA]
[RS_Main_00514]	AUTOSAR shall support the development of secure systems	[RS_EM_00014] [RS_EM_00111]
[RS_Main_00650]	AUTOSAR shall support up - and download of data and software	[RS_EM_NA]
[RS_Main_01001]	AUTOSAR shall support intra ECU communication	[RS_EM_NA]
[RS_Main_01002]	AUTOSAR shall support service-oriented communication	[RS_EM_NA]
[RS_Main_01003]	AUTOSAR shall support data-oriented communication	[RS_EM_NA]
[RS_Main_01004]	AUTOSAR shall support standards for wireless off-board communication	[RS_EM_NA]
[RS_Main_01005]	AUTOSAR shall establish communication paths dynamically	[RS_EM_NA]
[RS_Main_01007]	AUTOSAR communication shall assure quality of service on communication	[RS_EM_NA]
[RS_Main_01008]	AUTOSAR shall provide secure communication with off-board entities	[RS_EM_NA]
[RS_Main_01025]	AUTOSAR shall support debugging of software on the target and onboard	[RS_EM_NA]

[RS_Main_01026]	AUTOSAR shall support tracing and profiling on the target and onboard	[RS_EM_NA]

5.1 Not applicable requirements

[RS_EM_NA] ⌈ These requirements are not applicable as they are not within the scope of this release. ⌋*(RS_Main_01026, RS_Main_01025, RS_Main_00650, RS_Main_00026, RS_Main_00030, RS_Main_00140, RS_Main_00160, RS_Main_00161, RS_Main_00190, RS_Main_00230, RS_Main_00250, RS_Main_00261, RS_Main_00270, RS_Main_00280, RS_Main_00300, RS_Main_00301, RS_Main_00310, RS_Main_00350, RS_Main_00360, RS_Main_00410, RS_Main_00440, RS_Main_00445, RS_Main_00480, RS_Main_00490, RS_Main_00491, RS_Main_00500, RS_Main_00503, RS_Main_00507, RS_Main_00510, RS_Main_00511, RS_Main_00513, RS_Main_01001, RS_Main_01002, RS_Main_01003, RS_Main_01004, RS_Main_01005, RS_Main_01007, RS_Main_01008)*

6 References

[1] Standardization Template
AUTOSAR_TPS_StandardizationTemplate

[2] Glossary
AUTOSAR_TR_Glossary

[3] Specification of Manifest
AUTOSAR_TPS_ManifestSpecification

[4] Requirements of State Management
AUTOSAR_RS_StateManagement

[5] Main Requirements
AUTOSAR_RS_Main