Document Title	Specification of State Management
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	908

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	• Removed components • RequestState and ReleaseRequest are now deprecated • State Managements internal states can now be influenced by "Trigger" and are distributed by "Notifier" fields
2018-10-31	18-10	AUTOSAR Release Management	• Initial release

Disclaimer

Table of Contents

1 Introduction and functional overview

This document is the software specification of the State Management functional cluster within the Adaptive Platform Services.

State Management is responsible for determination the state of any of its internal statemachines, based on information received from other AUTOSAR Adaptive Platform Application or Adaptive Application.

State Management controls State of (partial networks using provided fields (NetworkHandle) of Network Management.

State Management interacts with the Execution Management to request Function Groups and the Machine State to enter specific states that are determined by project requirements. Function Group States might additionally depend on Network Managements State.

State Management provides access to its internal state via ara::com services. A particular service implements one of standardized service interfaces. The service interfaces have fields for getting current state (field "Notifier" (see section 9.2.2)) and requesting new state (field "Trigger" (see section 9.2.1)). AUTOSAR Adaptive Platform Applications or Adaptive Applications can use the fields for reacting on the system state changes or for influencing the system state(when they are configured to have write permissions).

Chapter 7 describes how State Management concepts are realized within the AUTOSAR Adaptive Platform.

1.1 Interaction with AUTOSAR Runtime for Adaptive

The set of programming interfaces to the Adaptive Applications is called AUTOSAR Runtime for Adaptive (ARA). APIs accessed by State Management using the interfunctional cluster API is described in Appendix A which is not part of ARA.

The Adaptive AUTOSAR Services are provided via mechanisms provided by the Communication Management functional cluster [1] of the Adaptive Platform Foundation

2 Acronyms and Abbreviations

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

Terms:	Description:
State Management	The element defining modes of operation for AUTOSAR Adaptive Platform. It allows flexible definition of functions which are active on the platform at any given time.
Execution Management [3]	A Functional Cluster within the Adaptive Platform Foundation
Platform Health Management [4]	A Functional Cluster within the Adaptive Platform Foundation
Communication Management [1]	A Functional Cluster within the Adaptive Platform Foundation
Network Management [5]	A Functional Cluster within the Adaptive Platform Services. Part of Communication Management.
Adaptive Diagnostics [6]	A Functional Cluster within the Adaptive Platform Services
Update And Config Management [7]	A Functional Cluster within the Adaptive Platform Services
Network Handle	Network Handles are provided by Network Management. A handle represents a set of (partial) networks.
Process	A process is a loaded instance of an Executable to be executed on a Machine.
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.
Function Group State	The element of State Management that characterizes the current status of a set of (functionally coherent) user-level Applications. The set of Function Groups and their Function Group States is machine specific and are configured in the Machine Manifest [8].
Machine State	The state of Function Group "MachineState" with some predefined states (Startup/Shutdown/Restart).
Execution Manifest	Manifest file to configure execution of an Adaptive Application.
Machine Manifest	Manifest file to configure a Machine.

Table 2.1: Acronyms and Abbreviations

The following technical terms used throughout this document are defined in the official [2] AUTOSAR Glossary or [8] 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
Adaptive Platform Services	see [2] AUTOSAR Glossary
Manifest	see [2] AUTOSAR Glossary

Executable	see [2] AUTOSAR Glossary
Functional Cluster	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 2.2: Glossary-defined Technical Terms

3 Related documentation

3.1 Input documents & related standards and norms

The main documents that serve as input for the specification of the State Management are:

[1] Specification of Communication Management
AUTOSAR_SWS_CommunicationManagement

[2] Glossary
AUTOSAR_TR_Glossary

[3] Specification of Execution Management
AUTOSAR_SWS_ExecutionManagement

[4] Specification of Platform Health Management for Adaptive Platform
AUTOSAR_SWS_PlatformHealthManagement

[5] Specification for Network Management
AUTOSAR_SWS_NetworkManagement

[6] Specification of Diagnostics
AUTOSAR_SWS_Diagnostics

[7] Specification of Update and Configuration Management
AUTOSAR_SWS_UpdateAndConfigManagement

[8] Specification of Manifest
AUTOSAR_TPS_ManifestSpecification

[9] General Specification of Adaptive Platform
AUTOSAR_SWS_General

[10] Requirements of State Management
AUTOSAR_RS_StateManagement

3.2 Related specification

AUTOSAR provides a General Specification [9, SWS General], which is also valid for State Management.

The specification SWS General shall be considered as additional and required specification for implementation of State Management.

4 Constraints and assumptions

4.1 Limitations

This section lists known limitations of State Management and their relation to this release of the AUTOSAR Adaptive Platform with the intent to provide an indication how State Management within the context of the AUTOSAR Adaptive Platform will evolve in future releases.

The following functionality is mentioned within this document but is not (fully) specified in this release:

- Section 7.2 This document will show the basic principles of the intended functionality of State Management. To enable State Management to be portable, in future versions of this document standardized fields and values shall be introduced.
- Section 7.3 Communication Control for Diagnostic reasons this is not yet discussed with Adaptive Diagnostics.
- Section 7.3 RequestRestart for Diagnostic reasons this is discussed with Adaptive Diagnostics, but some interface details are not yet finalized.

4.2 Applicability to car domains

If a superior State Management instance to the one from the ECU is available in a hierarchical car context, the State Management of the ECU shall also evaluate events generated by the superior instance of State Management. Section 7.7 will give further details.

5 Dependencies to other modules

5.1 Platform dependencies

5.1.1 Operating System Interface

State Management has no direct interface to the Operating System. All OS dependencies are abstracted by the Execution Management.

5.1.2 Execution Manager Interface

State Management is dependent on Execution Management to start and stop processes - as part of defined Function Groups or Machine States. State Management therefore uses the API referenced in Appendix A and defined in [3]. State Management additionally uses the StateClient funtionality of Execution Management to inform Execution Management about State Managements Process State.

5.1.3 Adaptive Diagnostics

State Management is dependent on the Adaptive Diagnostics [6] functional cluster. Adaptive Diagnostics provides information about an ongoing diagnostics session. This information is evaluated by State Management to prevent shutdown of the system during an active diagnostics session.

5.1.4 Update And Config Management

State Management is dependent on the Update and Config Management [7] functional cluster. Update and Config Management provides information about an ongoing update session. This information is evaluated by State Management to prevent shutdown of the system during an active update session.

5.1.5 Network Management

State Management is dependent on the Network Management [5] functional cluster. Network Management provides multiple NetworkHandle fields which represents a set of (partial) networks. State Management evaluates the NetworkCurrentState field to set Function Groups to the corresponding Function Group State and set the NetworkRequestedState field in dependency of Function Groups and their Function Group State . Additionally State Management shall prevent network from shutting down during an active update or diagnostic session.

5.2 Other dependencies

Currently, there are no other library dependencies.

6 Requirements Tracing

The following tables reference the requirements specified in [10] 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_SM_00001]	State Management shall coordinate and control multiple sets of Applications .	[SWS_SM_00001] [SWS_SM_00002] [SWS_SM_00003] [SWS_SM_00004] [SWS_SM_00005] [SWS_SM_00006] [SWS_SM_00400] [SWS_SM_00401] [SWS_SM_00402]
[RS_SM_00004]	State Management shall provide standardized interfaces.	[SWS_SM_00002] [SWS_SM_00003] [SWS_SM_00020] [SWS_SM_00021]
[RS_SM_00005]	State Management internal states.	[SWS_SM_00020] [SWS_SM_00021]
[RS_SM_00100]	State Management shall support ECU reset	[SWS_SM_00100] [SWS_SM_00101] [SWS_SM_00103] [SWS_SM_00104] [SWS_SM_00105] [SWS_SM_00200] [SWS_SM_00201] [SWS_SM_00202]
[RS_SM_00101]	State Management shall support diagnostic reset cause	[SWS_SM_00103] [SWS_SM_00104] [SWS_SM_00105]
[RS_SM_00200]	State Management shall provide an interface between State Management instances.	[SWS_SM_00500] [SWS_SM_00501]
[RS_SM_00300]	State Management shall support variant handling based on calibration data.	[SWS_SM_00005] [SWS_SM_00006]
[RS_SM_00400]	State Management shall establish communication paths dynamically.	[SWS_SM_00300] [SWS_SM_00301] [SWS_SM_00303] [SWS_SM_00304]
[RS_SM_00401]	State Management shall control Applications depending on dynamic communication paths .	[SWS_SM_00302]

7 Functional specification

Please note that the semantics in the following chapter is not yet fully specified.

State Management is a functional cluster contained in the Adaptive Platform Services. State Management is responsible for all aspects of Operational State Management including handling of incoming events, prioritization of these events/requests setting the corresponding internal States. Incoming events are issued when AUTOSAR Adaptive Platform or Adaptive Applications which are configured to have write access permissions change the value of "Trigger" fields provided by State Management. State Management may consist of one or more state machines, which might be more or less loosely coupled depending on project needs.

Additionally the State Management takes care of not shutting down the system as long as any diagnostic or update session is active as part of State Managements internal State. State Management supervises the shutdown prevention with a project-specific timeout.

In dependency of the current internal States, State Management might decide to request Function Groups or Machine State to enter specific state by using interfaces of Execution Management.

State Management is responsible for en- and disabling (partial) networks by means of Network Management. Network Management provides ara::com fields (NetworkHandle) where each of the fields represents a set of (partial) networks. State Management can influence these fields in dependency of Function Groups states and - vice versa - can set Function Groups to a defined state depending on the value of Network Managements NetworkHandle fields.

Adaptive Applications and AUTOSAR Adaptive Platform Applications can register to the events of the "Notifier" fields provided by State Management. They can change their internal behavior based on the value provided in the fields. Adaptive Applications and AUTOSAR Adaptive Platform Applications can influence the internal States of State Management by writing to the "Trigger" fields provided by State Management.

This chapter describes the functional behavior of State Management and the relation to other AUTOSAR Adaptive Platform Applications State Management interacts with.

- Section 7.1 covers the core State Management run-time responsibilities including the start of Applications.
- Section 7.2 describes how Adaptive Applications and AUTOSAR Adaptive Platform Applications could be influenced in their behavior based on provided "Notifier" fields of State Management and how they can influence the internal states of State Management by using provided "Trigger" fields.
- Section 7.3 covers several topics related to Adaptive Diagnostics including shutdown prevention and executing of different reset types

- Section 7.4 describes how Update and Config Management interacts with State Management
- Section 7.5 documents support provided by Network Management to de-/activate (partial) networks in dependency of Function Group States and vice versa.
- Section 7.6 describes how Execution Management is used to change Function Group State or Machine State.
- Section 7.7 provides an introduction to how State Management will work within a virtualized/hierarchical environment.

7.1 State Management Responsibilities

State Management is the functional cluster which is responsible for determining the current internal States, and for initiating Function Group and Machine State transitions by requesting them from Execution Management.

State Management is the central point where any operation event is received that might have an influence to the internal States of State Management. The State Management is responsible to evaluate these events and decide based on

- Event type (defined in project specific implementation based on project specific requirements).
- Event priority (defined in project specific implementation based on project specific requirements).
- Application identifier (Application identifier is not supported in this release. It is under discussion with FT-SEC if such an identifier could be provided by Identity and Access Management).

If an State Managements internal State change is triggered then Execution Management may be requested to set Function Groups or Machine State into new States.

The state change request for Function Groups can be issued by several AUTOSAR Adaptive Platform Applicationss:

- Platform Health Management to trigger error recovery, e.g. to activate fallback Functionality.
- Adaptive Diagnostics, to switch the system into different diagnostic states and to issue resets of the system.
- Update and Config Management to switch the system into states where software or configuration can be updated and updates can be verified.
- Network Management to coordinate required functionality and network state. This is no active request by Network Management. Network Management provides several sets of NetworkHandle fields, where State Management registers to and reacts on changes of these fields issued by Network Management.

The final decision if any effect is performed is taken by State Managements internal logic based on project-specific requirements.

Adaptive Applications may provide their own property or event via an ara com interface, where the State Management is subscribing to, to trigger State Management internal events. Since State Management functionality is critical, access from other Adaptive Applications must be secured, e.g. by Identity and Access Management.

- State Management shall be monitored and supervised by Platform Health Management.

- State Management provides ara::com fields as interface to provide information about its current internal States

State Management is responsible for handling the following states:

- Machine State see 7.1.1
- Function Group State see 7.1.2

7.1.1 Machine State

A Machine State is a specific type of Function Group State (see 7.1.2). Machine States and all other Function Group States are determined and requested by the State Management functional cluster, see 7.1.3. The set of active States is significantly influenced by vehicle-wide events and modes which are evaluated into State Managements internal States.

The Function Group States, including the Machine State, define the current set of running Processes. Each Application can declare in its Execution Manifests in which Function Group States its Processes have to be running.

The start-up sequence from initial state Startup to the point where State Management, SM, requests the initial running machine state Driving is illustrated in Figure 7.1 as an example Driving State is no mandatory State.

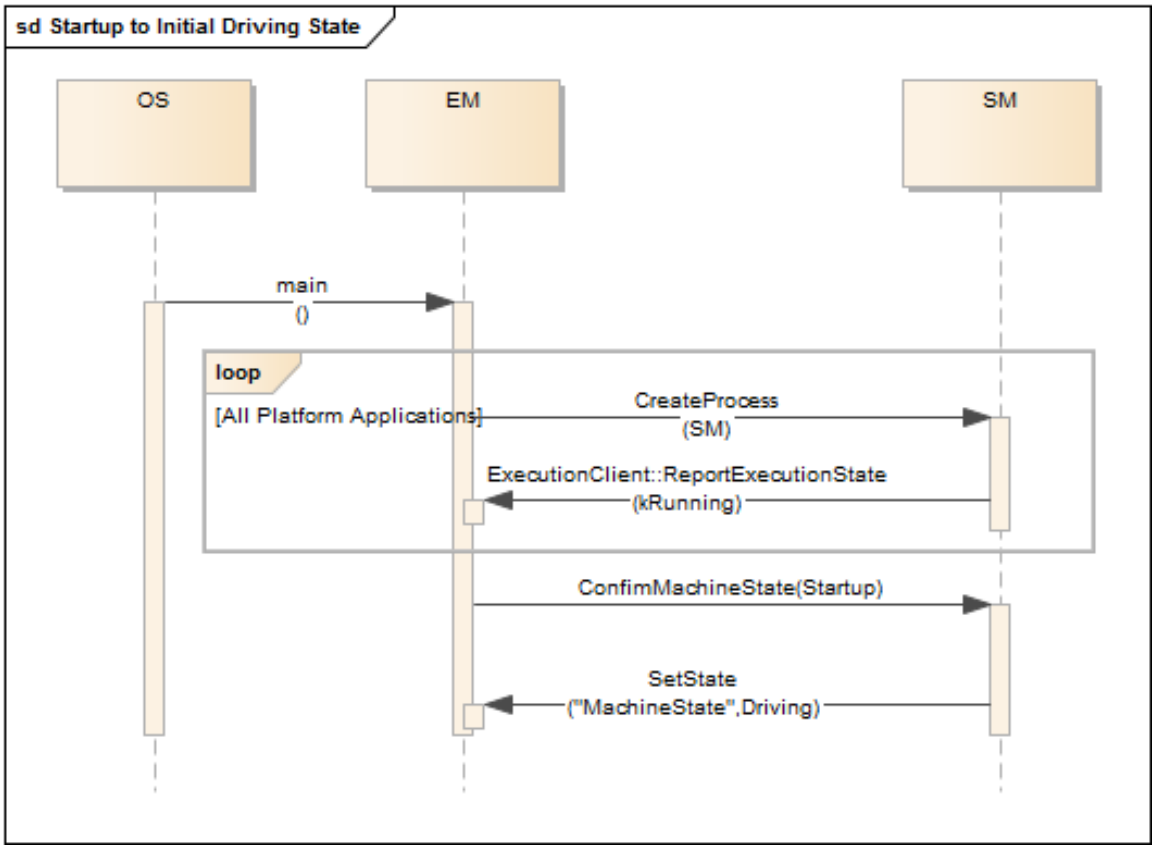

Figure 7.1: Start-up Sequence – from Startup to initial running state Driving

An arbitrary state change sequence to machine state `StateXYZ` is illustrated in Figure 7.2. Here, on receipt of the state change request, `Execution Management` terminates running `Processes` and then starts `Processes` active in the new state before confirming the state change to `State Management`.

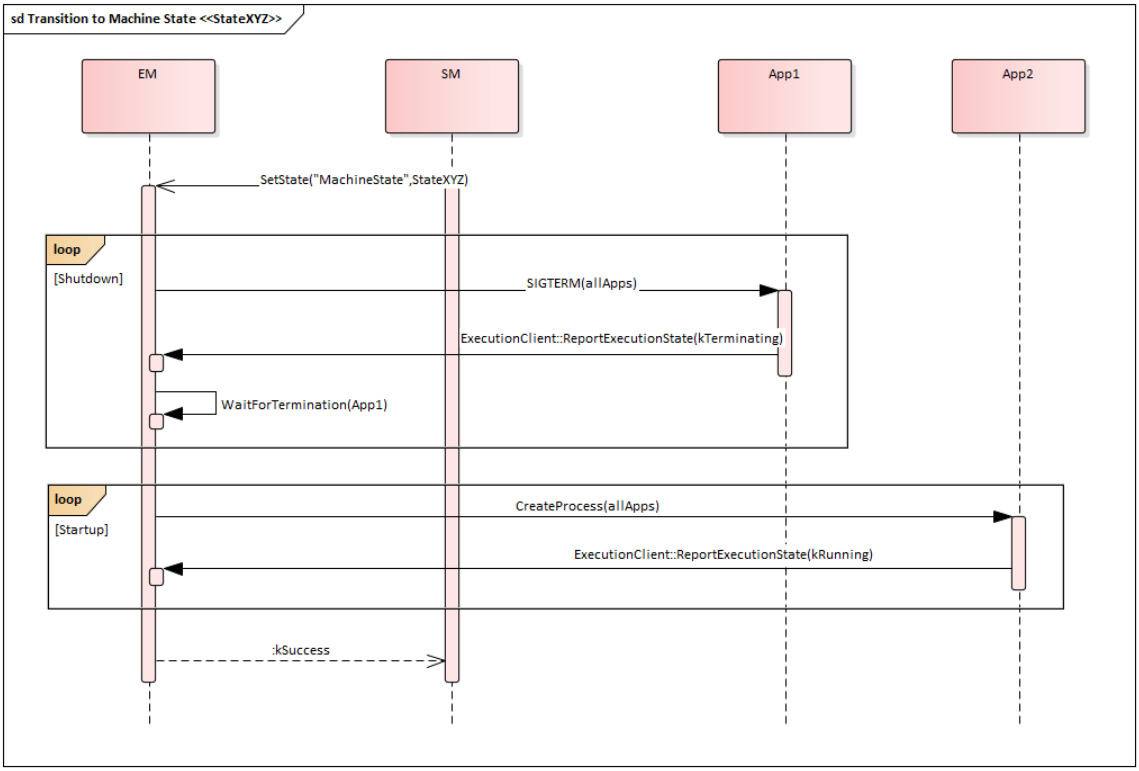

Figure 7.2: State Change Sequence – Transition to machine state `StateXYZ`

7.1.1.1 Startup

`Execution Management` will be controlled by `State Management` and therefore it should not execute any `Function Group State` changes on its own. This creates some expectations towards system configuration. The configuration shall be done in this way that `State Management` will run in every `Machine State` (this includes `Startup`, `Shutdown` and `Restart`). Above expectation is needed in order to ensure that there is always a software entity that can introduce changes in the current state of the `Machine`. If (for example) system integrator doesn't configure `State Management` to be started in `Startup` `Machine State`, then `Machine` will never be able transit to any other state and will be stuck forever in it. This also applies to any other `Machine State` state that doesn't have `State Management` configured.

7.1.1.2 Shutdown

As mentioned in 7.1.1.1 AUTOSAR assumes that State Management will be configured to run in Shutdown. State transition is not a trivial system change and it can fail for a number of reasons. When ever this happens you may want State Management to be still alive, so you can report an error and wait for further instructions. Please note that the very purpose of this state is to shutdown Machine (this includes State Management) in a clean manner. Unfortunately this means that at some point State Management will no longer be available and it will not be able to report errors anymore. Those errors will be handled in a implementation specific way.

7.1.1.3 Restart

As mentioned in 7.1.1.1 AUTOSAR assumes that State Management will be configured to run in Restart. The reasons for doing so are the same as for 7.1.1.2.

7.1.2 Function Group State

If more than one group of functionally coherent Applications is installed on the same machine, the Machine State mechanism is not flexible enough to control these functional clusters individually, in particular if they have to be started and terminated with interleaving lifecycles. Many different Machine States would be required in this case to cover all possible combinations of active functional clusters.

To support this use case, additional Function Groups and Function Group States can be configured. Other use cases where starting and terminating individual groups of Processes might be necessary including diagnostic and error recovery.

In general, Machine States are used to control machine lifecycle (startup/shutdown/restart) and Processes of platform level Applications while other Function Group States individually control Processes which belong to groups of functionally coherent user level Applications.

[SWS_SM_00001]{DRAFT} **Available Function Group (states)** ⌈ State Management shall obtain available Function Groups and their potential states from the Machine Manifest to set-up the Function Group specific state management. ⌋ *(RS_SM_00001)*

Processes reference in their Execution Manifest the states in which they want to be executed. A state can be any Function Group State, including a Machine State. For details see [8], especially "Mode-dependent Startup Configuration" chapter and "Function Groups" chapter.

The arbitrary state change sequence as shown in Figure 7.2 applies to state changes of any Function Group - just replace "MachineState" by the name of the Function Group. On receipt of the state change request, Execution Management terminates

not longer needed Processes and then starts Processes active in the new Function Group State before confirming the state change to State Management.

Please note that the following requirement is obsolete and will be removed in 19-11 release of this document!

[SWS_SM_00002]{DRAFT} **Function Group State Change Request** ⌈ State Management shall implement functionality to enable Adaptive Applications and AUTOSAR Adaptive Platform Applications to change the Function Group State of Function Groups ⌋(*RS_SM_00001, RS_SM_00004*)

It might be that State Management declines the request to change a Function Groups state, based on State Management internal State or another Adaptive Application or AUTOSAR Adaptive Platform Application with higher priority that has the ownership of a Function Group. As per current specification several Adaptive Applications or AUTOSAR Adaptive Platform Application use the service interface of State Management e.g. Update and Config Management and a superior Function Group Manager. To ensure that the decision to set Function Groups into a dedicated Function Group State of a ”more important” application is not ”undermined” by a ”less important” application, the application with a higher priority (project specific) get the ownership of the requested Function Groups as long as it does not release the request.

From the point of view of Execution Management, Function Groups are independent entities that doesn't influence each other. However from the point of view of State Management this may not always be the true. Let's consider a simple use case of Machine shutdown. From the point of view of Execution Management State Management (at some point in time) will request a Machine State transition to Shutdown state. One of the Processes configured to run in that particular state, will initiate OS / HW shutdown and the Machine will power off. However from the point of view of State Management you will need to asses, if it's valid to request a Machine State transition to Shutdown state. Even if the assessment was positive and the Machine can be powered off, project specific requirements may mandate to switch all available Function Groups to Off state before we start power off sequence. For this reason we are considering existence of dependencies between Function Groups. This kind of information will tell State Management if a single request from Adaptive Application, shall be translated to several Function Group state transitions and if those transitions can be requested in parallel, or a specific order should be applied. Please note that currently those dependencies are implementation specific.

[SWS_SM_00003]{DRAFT} **Function Group State Retrieval** ⌈ State Management shall implement functionality to enable Adaptive Applications and AUTOSAR Adaptive Platform Applications to retrieve the Function Group State of Function Groups and State of Machine State ⌋(*RS_SM_00001, RS_SM_00004*)

[SWS_SM_00004]{DRAFT} **Function Group State Change Request Result** ⌈ State Management shall return an appropriate result to the Adaptive Applications and AUTOSAR Adaptive Platform Applications which has requested a Function Group State change. ⌋*(RS_SM_00001)*

The system might contain calibration data for variant handling. This might include that some of the Function Groups configured in the Machine Manifest are not intended to be executed on this system. therefore State Management has to evaluate calibration data to gather information about Function Groups not configured for the system variant

[SWS_SM_00005]{DRAFT} **Function Group Calibration Support** ⌈ State Management shall receive information about deactivated Function Groups from calibration data. ⌋*(RS_SM_00001, RS_SM_00300)*

The storage and reception of calibration data is implementation specific.

[SWS_SM_00006]{DRAFT} **Function Group Calibration Support** ⌈ State Management shall decline the request of Adaptive Applications and AUTOSAR Adaptive Platform Applications to change the Function Group State of a Function Group which is not configured to run in this variant. ⌋*(RS_SM_00001, RS_SM_00300)*

7.1.3 State Management Architecture

State Management is the functional cluster which is responsible for determining the current set of active Function Group States, including the Machine State, and for initiating State transitions by requesting them from Execution Management. Execution Management performs the State transitions and controls the actual set of running Processes, depending on the current States.

State Management is the central point where new Function Group States can be requested and where the requests are arbitrated, including coordination of contradicting requests from different sources. Additional data and events might need to be considered for arbitration.

State Management functionality is highly project specific, and AUTOSAR decided against specifying functionality like the Classic Platforms BswM for the Adaptive Platform. It is planned to only specify set of basic service interfaces, and to encapsulate the actual arbitration logic into project specific code (e.g. a library), which can be plugged into the State Management framework and has standardized interfaces between framework and arbitration logic, so the code can be reused on different platforms.

The arbitration logic code might be individually developed or (partly) generated, based on standardized configuration parameters.

An overview of the interaction of State Management, AUTOSAR Adaptive Platform Applications and Adaptive Applications is shown in Figure 7.3.

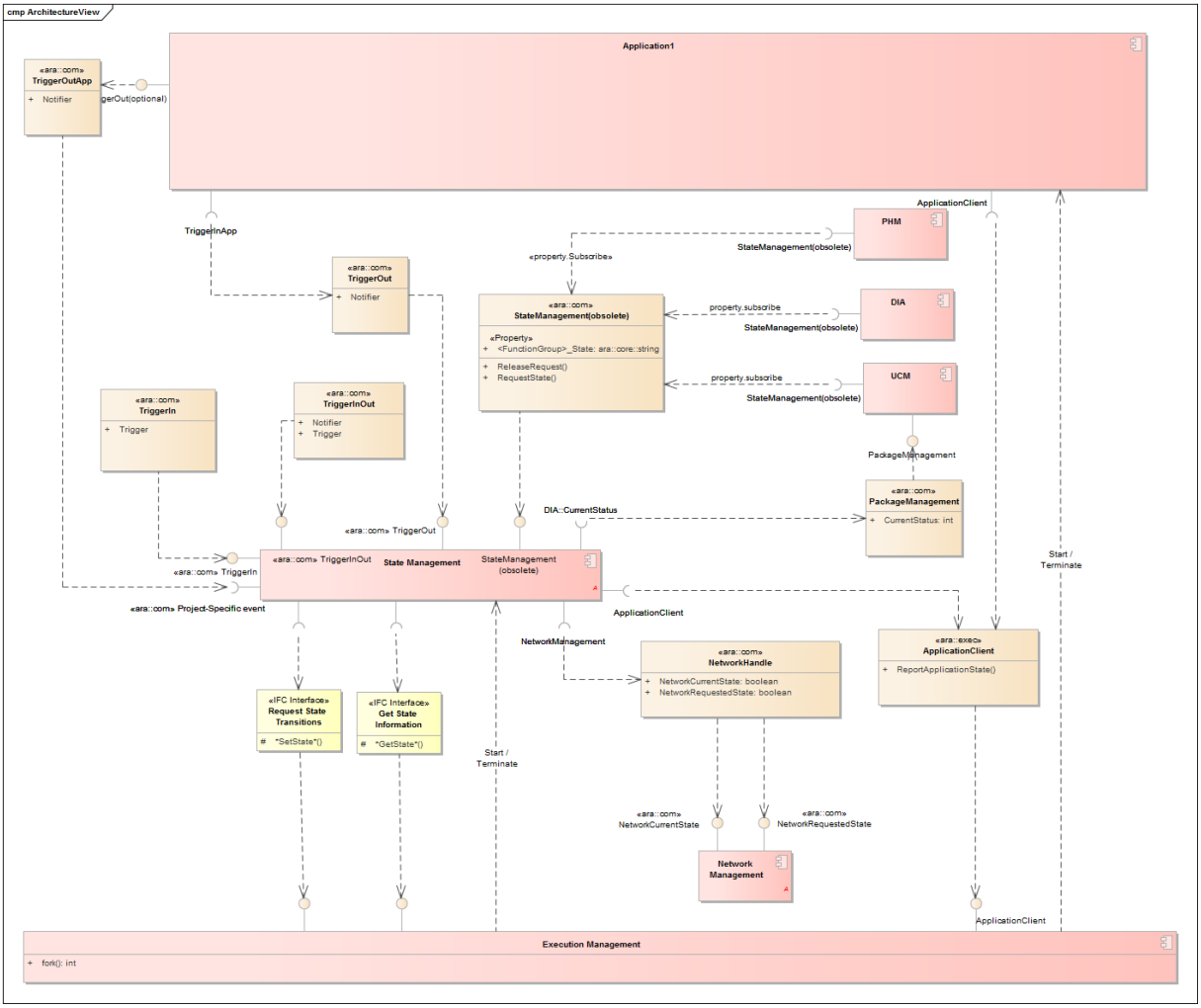

Figure 7.3: State Management Architecture

7.2 State Management and Adaptive (Platform) Applications

7.2.1 Interaction between the SM and Adaptive Applications

To fulfill the needs of a resource optimized system it is necessary to control Processes in a more fine-grained way than it is possible by Execution Management. When the internal behavior of a Process should be changed by Execution Management it is needed to unload Process from memory (including high latency due to persisting) and reload the Executable from filesystem to memory. This behavior is resource consuming with respect to (flash-)memory bandwidth, CPU load and execution time.

Therefor State Management provides a service interface with a "Notifier" (see section 9.2.2) field, where each Adaptive Application can subscribe to, thus it is informed whenever a State Managements internal State changes. When an Adaptive Application recognizes the change it can carry out the appropriate action.

In the opposite way each Adaptive Application can influence the behavior of StateManagement by writing to the "Trigger" fields provided by State Management. Therefore the Adaptive Application has to by configured in a way that write access to State Managements fields is granted.

State Management provides a third service interface, where both fields are available: "Trigger" and "Notifier"'. This combined field is provided with the intention that whenever the "Trigger" field changes the "Notifier" field changes as well after State Management has carried out its operation issued by the "Trigger" change.

An overview of the interaction of State Management and Adaptive Applications for a non-synchonized behavior is shown in Figure 7.4.

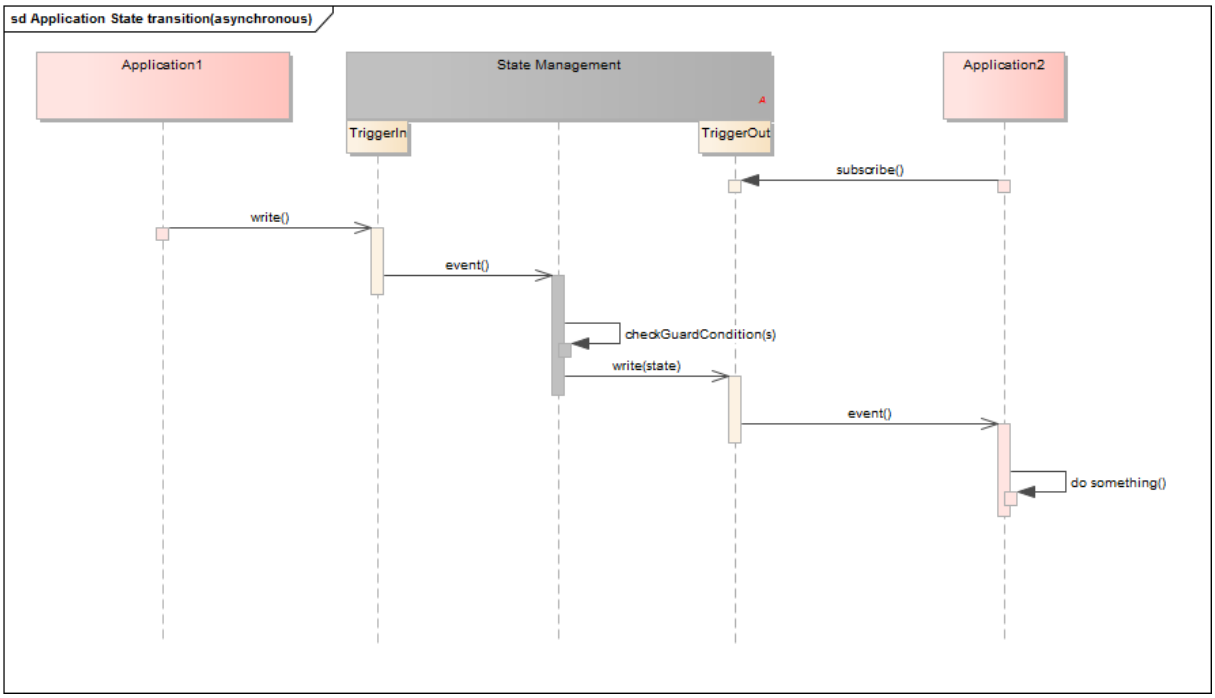

Figure 7.4: Non-Synchronized Application State handling

7.2.2 Synchronization across multiple Adaptive Applications

Some scenarios in AUTOSAR Adaptive Platform might require a more sophisticated handling, where a change in State Managements internal state could only be finally carried out, when related Processes have entered a dedicated 'State', which is triggered by State Management.

One important use-case is the 'late-wakeup', where a new wakeup reason is found during a running shutdown. With the current approach the shutdown can't be interrupted and all Processes have to be unloaded and newly loaded. This is one Example for having 'overall' system states. To enable Adaptive Applications to react on this system states they have to register to the corresponding "Notifier" fields of State Management.

An Adaptive Application shall provide means to State Management to be able to check if an Adaptive Application has carried out its actions (in time).

Therefore Adaptive Applications which require a synchronized behavior shall offer a service following the TriggerOut("Notifier") Interface pattern of State Management when they are spawned and shall stop offering this service when they are unloaded, thus State Management can find its peers dynamically by calling 'FindService'.

An overview of the interaction of State Management and Adaptive Applications for a synchronized behavior is shown in Figure 7.5.

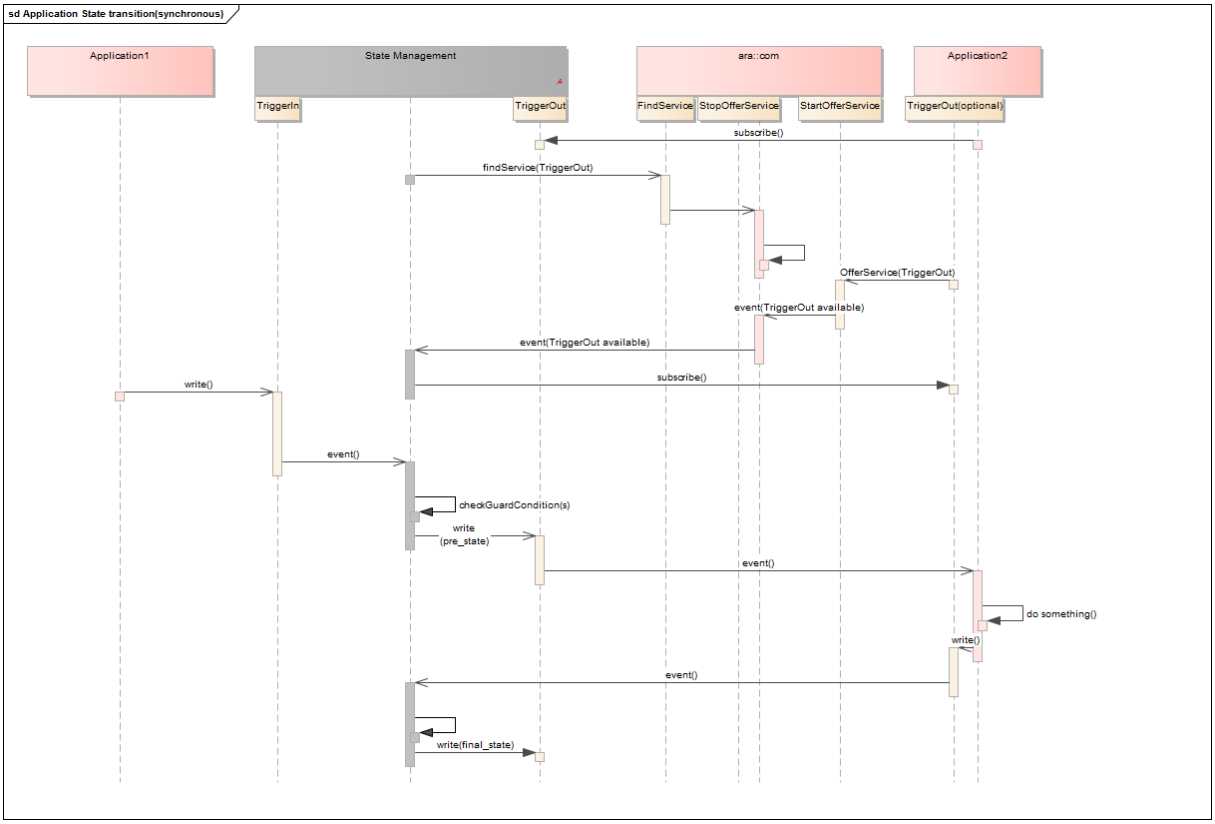

Figure 7.5: Synchronized Application State handling

Implementing the service into Adaptive Applications is optional and only recommended when a synchronized behavior is required.

[SWS_SM_00020]{DRAFT} **InternalState Propagation** ⌈ State Management shall have multiple instances of a "Notifier" field which reflect State Managements internal states thus Application can get State Managements states.⌋ *(RS_SM_00004, RS_SM_00005)*

[SWS_SM_00021]{DRAFT} **InternalState Influence** ⌈ State Management shall have multiple instances of a "Trigger" field which affect State Managements internal states thus Application can influence State Managements states.⌋ *(RS_SM_00004, RS_SM_00005)*

7.3 Interaction with Adaptive Diagnostics

Adaptive Diagnostics is responsible for diagnosing, configuring and resetting Function Groups. During any diagnostic session is executed it is necessary to prevent system from shutting down.

[SWS_SM_00100]{DRAFT} **Prevent Shutdown due to Diagnostic Session** ⌈ State Management shall not shutdown the system during an active diagnostic session. Therefore State Management has to register to Adaptive Diagnostics to receive information about active diagnostic session ⌋*(RS_SM_00100)*

From Adaptive Diagnostics point of view several different reset types have to be carried out to fulfill functionality of Adaptive Diagnostics. Because the interpretation of the reset types (defined in ISO 14229-1)

- hardReset
- keyOffOnReset
- softReset

is done differently by each OEM, parts of the reset functionality have to be delegated by State Management to Adaptive Applications and AUTOSAR Adaptive Platform Applications.

A 'keyOffOnReset' may be translated by State Managements internal logic to stop and start the provided Function Groups.

Please note that this behavior is currently under discussion and therefore subject of change!

[SWS_SM_00101]{DRAFT} **Diagnostic Reset** ⌈ State Management shall implement means to receive reset requests for Function Groups from Adaptive Diagnostics. State Management shall carry out the project specific actions for the specific reset type ⌋*(RS_SM_00100)*

The Function Group Machine State has to be handled in a different way when executing reset requests from Adaptive Diagnostics: A 'hardReset' could be interpreted e.g. that an Adaptive Application has to be launched (by requesting e.g. Function Group 'reset' from Execution Management) which carries out the OS or hardware specific reset. A 'softReset' could be interpreted by shutting down all Function Groups and requesting a Machine State 'restart' from Execution Management.

But this functionality is project-specific. So therefore the correct mapping has to be done by the project specific code.

State Management is the central point in the system, where a reset for the Machine could be requested. So State Management has to keep track of reset causes and has to reset the persistent reset cause when it is newly spawned.

[SWS_SM_00103]{DRAFT} **Diagnostic Reset Last Cause** ⌈ State Management shall provide functionality to persist reset type before Machine reset is carried. ⌋ *(RS_SM_00100, RS_SM_00101)*

[SWS_SM_00104]{DRAFT} **Diagnostic Reset Last Cause Retrieval** ⌈ State Management shall read out the last persisted reset cause when State Management is spawned. This reset cause has to be provided via its service interface ⌋ *(RS_SM_00100, RS_SM_00101)*

[SWS_SM_00105]{DRAFT} **Diagnostic Reset Last Cause Reset** ⌈ State Management shall reset the last persisted reset cause immediately after State Management has read out the current value. ⌋*(RS_SM_00100, RS_SM_00101)*

7.4 Interaction with Update and Config Management

Update and Config Management is responsible for updating Function Groups, Manifests (execution or machine manifest) or the whole AUTOSAR Adaptive Platform. During any update is executed it is necessary to prevent system from shutting down.

[SWS_SM_00200]{DRAFT} **Prevent Shutdown during to Update Session** ⌈ State Management shall not shutdown the system during an active update session. Therefore State Management has to register to Update and Config Management to receive information about active update session ⌋*(RS_SM_00100)*

[SWS_SM_00201]{DRAFT} **Supervision of Shutdown Prevention** ⌈ When State Management shall not shutdown the system during an active update session State Management shall supervise the duration of the update session with a project-specific timeout, thus the system does not run forever. ⌋*(RS_SM_00100)*

To enable Update and Config Management to fulfill its functionality an update and a verify state should be available in the Manifests for each Function Group and for Machine State. Update and Config Management has to request the corresponding Function Group State from State Management.

[SWS_SM_00202]{DRAFT} **Reset Execution** ⌈ State Management shall implement means to issue a Machine reset when Machine State changes from Function Group State 'update' to 'verify'. ⌋*(RS_SM_00100)*

In case of an update of an Adaptive Applications which does not imply an ECU/-machine reset (i.e. soft reset), Execution Management needs to be triggered to reparse the Manifests (execution or machine manifest) of this Adaptive Application prior to restarting it (in order to start it with the right (i.e updated) configuration). Otherwise Execution Management would only reparse the processed Manifests during the next ECU/machine reset (which is too late). For that purpose, an additional interface is needed between Update and Config Management Execution Management which has to be used before the update session is no longer active.

7.5 Interaction with Network Management

To be portable between different ECUs the Adaptive Applications should not have the need to know which networks are needed to fulfill its functionality, because on different ECUs the networks could be configured differently. To control the availability of networks for several Adaptive Applications State Management interacts with Network Management via a service interface.

Network Management provides multiple instances of NetworkHandles, where each represents a set of (partial) networks.

The NetworkHandles are defined in the Machine Manifest and are there assigned to a Function Group State.

An overview of the interaction of State Management, Network Management and Adaptive Applications is shown in Figure 7.7.

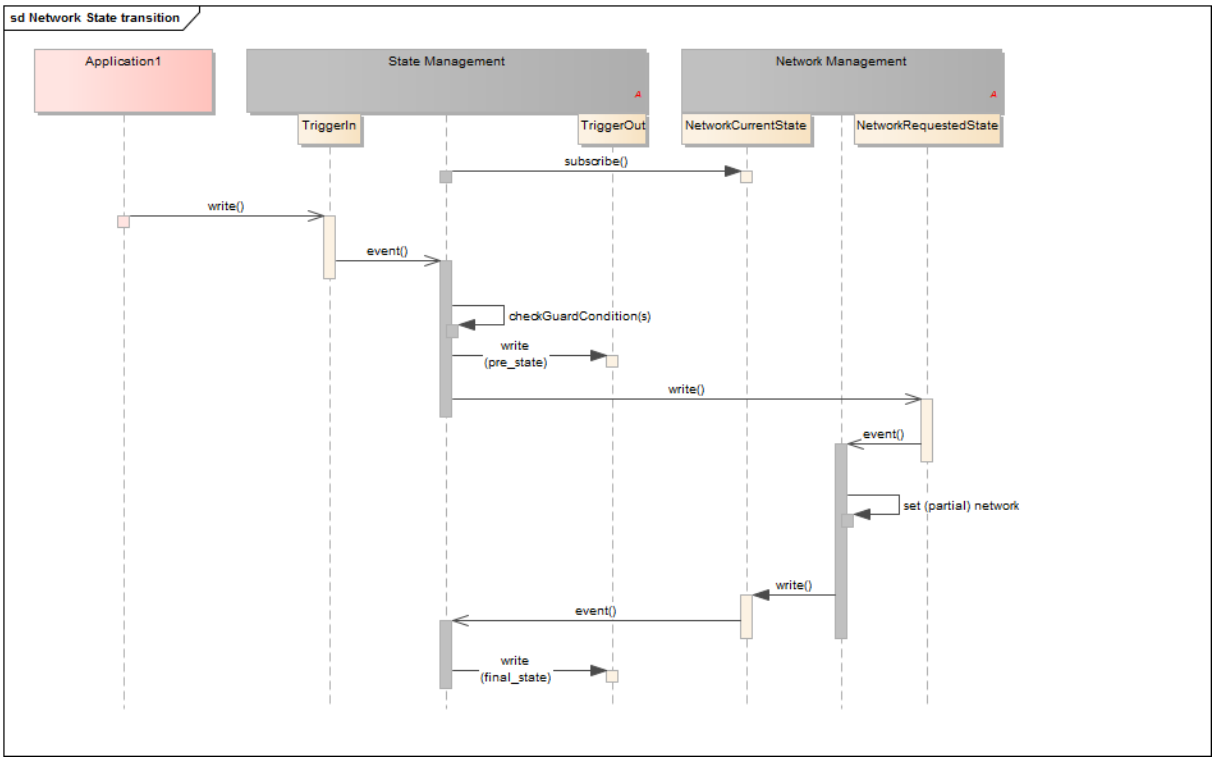

Figure 7.6: Switching Network State by "Trigger"

[SWS_SM_00300]{DRAFT} **NetworkHandle Configuration** ⌈ State Management shall receive information about NetworkHandles and their associated Function Group States from Machine Manifest. ⌋*(RS_SM_00400)*

Whenever (partial) networks are activated or deactivated from outside request and this set of (partial) networks is represented by a NetworkHandle in Machine Manifest Network Management will change the value of the corresponding NetworkHandle. State Management is notified about the change, because it has registered to all availabe NetworkHandle fields. When State Management recognizes a change in

a fields value it sets the corresponding Function Group in the Function Group State where the NetworkHandle is configured for in the Machine Manifest.

[SWS_SM_00301]{DRAFT} **NetworkHandle Registration** ⌈ State Management shall register for all NetworkHandles provided by Network Managements which are available from Machine Manifest. ⌋*(RS_SM_00400)*

[SWS_SM_00302]{DRAFT} **NetworkHandle to FunctionGroupState** ⌈ State Management shall set Function Groups to the corresponding Function Group State which is configured in the Machine Manifest for the NetworkHandle when it recognizes a change in NetworkHandle value. ⌋*(RS_SM_00401)*

Vice versa State Managements shall change the value of the NetworkHandle when a Function Group has to change its Function Group State and an association between this Function Group State and the Network handle is available in Machine Manifest. Network Management will recognize this change and will change the state of the (partial) networks accordingly to the NetworkHandle.

[SWS_SM_00303]{DRAFT} **FunctionGroupState to NetworkHandle** ⌈ State Management shall change the value of NetworkHandle when Function Groups changes its Function Group State and a NetworkHandle is associated to this Function Group State in the Machine Manifest. ⌋*(RS_SM_00400)*

It might be needed that a Function Group stays longer in its Function Group State when the causing (partial) network set has been switched off or a (partial) network is longer available than the causing Function Group has been switched to Function Group State 'Off'. This is called 'afterrun'. The corresponding timeout-value has to be configured in Machine Manifest

[SWS_SM_00304]{DRAFT} **Network Afterrun** ⌈ State Management shall support means to support 'afterrun' to switch off related Function Groups or (partial) networks. The timeout value for this 'afterrun' has to be read from e.g. Machine Manifest. ⌋*(RS_SM_00400)*

7.6 Interaction with Execution Management

Execution Management is used to execute the Function Group State changes. The decision to change the State of Machine State or the Function Group State of Function Groups might come from inside of State Management based on State Managements States (or other project specific requirements) or might be requested at State Management from an external Adaptive Application.

An overview of the interaction of State Management, Execution Management and Adaptive Applications is shown in Figure 7.7.

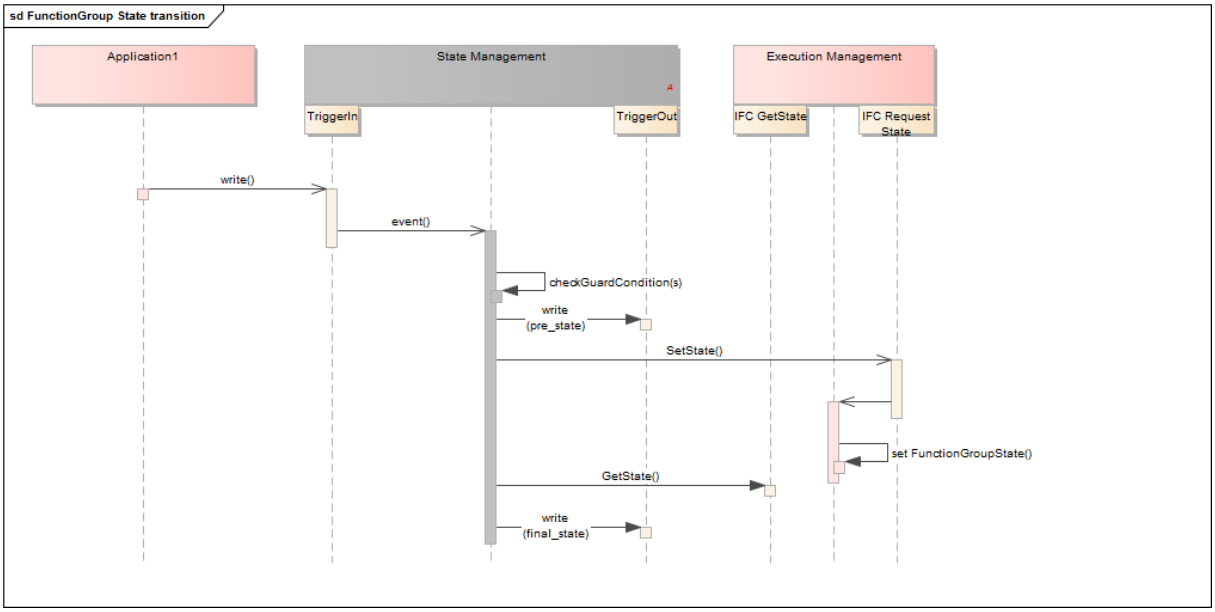

Figure 7.7: Switching FunctionGroup State by "Trigger"

[SWS_SM_00400]{DRAFT} **Execution Management** ⌈ State Management shall use API of Execution Management to change the State of Machine State or Function Group State of Function Groups. ⌋(*RS_SM_00001*)

Execution Management might not be able to carry out the requested Function Group State change due to several reasons (e.g. corrupted binary). Execution Management returns the result of the request.

[SWS_SM_00401]{DRAFT} **Execution Management Results** ⌈ State Management shall evaluate the results of request to Execution Management. Based on the results State Management may do project-specific actions ⌋(*RS_SM_00001*)

[SWS_SM_00402]{DRAFT} **Function Group State Change Results** ⌈ State Management shall provide Function Group States based on the results of Function Group State change requests to Execution Management via its service interface ⌋(*RS_SM_00001*)

7.7 State Management in a virtualized/hierarchical environment

On an ECU several machines might run in a virtualized environment. Each of the virtual machines might contain an AUTOSAR Adaptive platform. So therefore each of the virtual machines contain State Management. To have coordinated control over the several virtual machines there has to be virtual machine which supervises the whole ECU state. This is not only valid for a virtualized environment, but for a hierarchical environment, too.

[SWS_SM_00500]{DRAFT} **Virtualized/hierarchical State Management** ⌈ State Management shall be able to register to the "Trigger" fields of a supervising

State Management instance to receive information about the whole ECU state. ⌋ *(RS_SM_00200)*

[SWS_SM_00501]{DRAFT} **Virtualized/hierarchical State Management internal State** ⌈ State Management shall implement means to calculate its internal States based on information from a supervising State Management instance. ⌋ *(RS_SM_00200)*

8 API specification

`State Management` does not provide any API. All functional interfaces will be found in Chapter 9 Service Interfaces.

9 Service Interfaces

9.1 Type definitions

Name	FunctionGroupStates	
Kind	STRING	
Derived from	-	
Description	Default FunctionGroup states	
Range / Symbol	***Limit***	***Description***
kOff	'kOff'	FunctionGroup is in Off state.
kRunning	'kRunning'	FunctionGroup is in running state.
kUpdate	'kUpdate'	FunctionGroup is in update state.
kVerify	'kVerify'	FunctionGroup is in verify state.

Table 9.1: Implementation Data Type - FunctionGroupStates

9.2 State Management Provided Interfaces

9.2.1 State Management TriggerIn

Port

Name	TriggerIn_{State}		
Kind	ProvidedPort	***Interface***	`TriggerIn`
Description	To be used by Adaptive (Platform) Applications to tigger State Management to change its internal state.		
Variation			

Table 9.2: Port - TriggerIn_{State}

Service Interface

Name	TriggerIn_{StateGroup}
NameSpace	ara::sm

Table 9.3: Service Interfaces - TriggerIn

Fields

Name	Trigger
Description	Value to be evaluated by State Management in a projectspecific way.
Type	`project_specific`
HasGetter	false
HasNotifier	false
HasSetter	true

Table 9.4: Service Interface TriggerIn - Field: Trigger

9.2.2 State Management TriggerOut

Port

Name	TriggerOut_{State}		
Kind	ProvidedPort	***Interface***	`TriggerOut`
Description	To be used by Adaptive (Platform) Applications to be informed when State Management has changed its internal state.		
Variation			

Table 9.5: Port - TriggerOut_{State}

Service Interface

Name	TriggerOut_{StateGroup}
NameSpace	ara::sm

Table 9.6: Service Interfaces - TriggerOut

Fields

Name	Notifier
Description	To be set by State Management in a projectspecific way to inform Adaptive (Platform) Applications about changes within StateManagement
Type	`project_specific`
HasGetter	true
HasNotifier	true
HasSetter	false

Table 9.7: Service Interface TriggerOut - Field: Notifier

9.2.3 State Management TriggerInOut

Port

Name	TriggerInOut_{State}		
Kind	ProvidedPort	***Interface***	`TriggerInOut`
Description	To be used by Adaptive (Platform) Applications to tigger State Management to change its internal state and to get information when it is carried out.		
Variation			

Table 9.8: Port - TriggerInOut_{State}

Service Interface

Name	TriggerInOut_{StateGroup}
NameSpace	ara::sm

Table 9.9: Service Interfaces - TriggerInOut

Fields

Name	Trigger
Description	Value to be evaluated by State Management in a projectspecific way.
Type	`project_specific`
HasGetter	false
HasNotifier	false
HasSetter	true

Table 9.10: Service Interface TriggerInOut - Field: Trigger

Name	Notifier
Description	To be set by State Management in a projectspecific way to inform Adaptive (Platform) Applications about changes within StateManagement
Type	`project_specific`
HasGetter	true
HasNotifier	true
HasSetter	false

Table 9.11: Service Interface TriggerInOut - Field: Notifier

9.2.4 FunctionGroupState

The FunctionGroupState interface is obsolete. In current release of this document this interface is still available due to compatibility reasons for PHM, UCM and DIAG

Port

Name	State_{FunctionGroup}		
Kind	ProvidedPort	***Interface***	FunctionGroupState
Description	Provides handling of FunctionGroupStates.		
Variation			

Table 9.12: Port - State_{FunctionGroup}

Service Interface

Name	FunctionGroupState
NameSpace	ara::sm

Table 9.13: Service Interfaces - FunctionGroupState

Fields

Name	FunctionGroupState
Description	Contains the current status of the Function Group {Function Group}.
Type	FunctionGroupStates
HasGetter	true
HasNotifier	true
HasSetter	false

Table 9.14: Service Interface FunctionGroupState - Field: FunctionGroupState

Methods

Name	RequestState	
Description	Requests a new Function Group State or Machine State and gathers the ownership for this Function Group.	
Parameter	FunctionGroupState_or_MachineState	
	Description	Function Group State or Machine State to be set.
	Type	FunctionGroupStates
	Variation	
	Direction	IN
Application Error Set	FunctionGroupStateErrorSet	The potential errors values returned using State Managements Function Group State change service interface

Table 9.15: Service Interface FunctionGroupState - Method: RequestState

Name	ReleaseRequest	
Description	Releases the current ownership of a Function Group or Machine State.	
Application Error Set	`FunctionGroupStateErrorSet`	The potential errors values returned using State Managements Function Group State change service interface

Table 9.16: Service Interface FunctionGroupState - Method: ReleaseRequest

9.3 Application Errors

This chapter lists all application errors of `State Management` Port

9.3.1 Application Error Domain

Name	*Code*	*Description*
kSuccess	0	FunctionGroup State change request was executed successfully
kInvalid	1	FunctionGroup State change request was invalid e,g, unknown state.
kFailed	2	FunctionGroup State change request failed due to other reason.
kDelay	3	FunctionGroup State change request was delayed due to ownership.

Table 9.17: Application Errors of FunctionGroupStateErrors

9.3.2 Application Error Set

Error Set Name	FunctionGroupStateErrorSet
Description	The potential errors values returned using State Managements Function Group State change service interface
Reference	`kInvalid`, `kFailed`, `kDelay`

Table 9.18: Application Errors of Set FunctionGroupStateErrorSet

9.4 State Management Required Interfaces

9.4.1 Update and Config Management

9.4.1.1 PackageManagement CurrentStatus

Port

Name	PackageManagement		
Kind	RequiredPort	***Interface***	`PackageManagement`
Description			
Variation			

Table 9.19: Port - PackageManagement

9.4.2 Network Management

9.4.2.1 NetworkManagement NetworkState

Port

Name	NetworkState_{NetworkHandle}		
Kind	RequiredPort	***Interface***	NetworkState
Description	Provides information about network status per NetworkHandle. Intended to be only used by State Management!		
Variation	FOR NetworkHandle : MODEL.filterType("NetworkHandle");		

Table 9.20: Port - NetworkState_{NetworkHandle}

A Used Interfunctional Cluster Interfaces

No IFC-Interfaces are provided by State Management.

B Not applicable requirements

C History of Constraints and Specification Items

Please note that the lists in this chapter also include constraints and specification items that have been removed from the specification in a later version. These constraints and specification items do not appear as hyperlinks in the document.

C.1 Constraint and Specification Item History of this document according to AUTOSAR Release 19-03

C.1.1 Added Traceables in 19-03

Number	Heading
[SWS_SM_00020]	InternalState Propagation
[SWS_SM_00021]	InternalState Influence

▽

△

Number	Heading
[SWS_SM_00202]	Reset Execution

Table C.1: Added Traceables in 19-03

C.1.2 Changed Traceables in 19-03

Number	Heading
[SWS_SM_00002]	Function Group State Change Request
[SWS_SM_00003]	Function Group State Retrieval
[SWS_SM_00004]	Function Group State Change Request Result
[SWS_SM_00006]	Function Group Calibration Support
[SWS_SM_00200]	Prevent Shutdown during to Update Session
[SWS_SM_00201]	Supervision of Shutdown Prevention
[SWS_SM_00302]	NetworkHandle to FunctionGroupState
[SWS_SM_00401]	Execution Management Results
[SWS_SM_00402]	Function Group State Change Results
[SWS_SM_00500]	Virtualized/hierarchical State Management
[SWS_SM_00501]	Virtualized/hierarchical State Management internal State

Table C.2: Changed Traceables in 19-03

C.1.3 Deleted Traceables in 19-03

Number	Heading
[SWS_SM_00010]	Component (states)
[SWS_SM_00011]	Component (states) Handling
[SWS_SM_00012]	Component (states) Registration
[SWS_SM_00013]	Component (states) Configuration
[SWS_SM_00014]	Component (states) Enforcement
[SWS_SM_00015]	Component (states) Transitions
[SWS_SM_00102]	Component States for Reset

Table C.3: Deleted Traceables in 19-03

C.1.4 Added Constraints in 19-03

none

C.1.5 Changed Constraints in 19-03

none

C.1.6 Deleted Constraints in 19-03

none