Document Title	Specification of Update and Configuration Management
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	888

Document Status	published
Part of AUTOSAR Standard	Adaptive Platform
Part of Standard Release	R23-11

Document Change History

Date	Release	Changed by	Description
2023-11-23	R23-11	AUTOSAR Release Management	• Split UCM Master into SWS Vehicle Update Configuration Management
2022-11-24	R22-11	AUTOSAR Release Management	• Failing rollback clarifications • Campaign history type consolidated • Introduced production errors
2021-11-25	R21-11	AUTOSAR Release Management	• Renamed to SWS_UpdateAnd-ConfigurationManagement • UCM errors ordering • Vehicle State Manager API detailing
2020-11-30	R20-11	AUTOSAR Release Management	• Classic Plaftorm update specification for UCM Master • Refactored UCM Master API • Simplified UCM Master State Machine • Detailed campaign history information

▽

△

2019-11-28	R19-11	AUTOSAR Release Management	• Introduced UCM Master concept • Software Package state machine updated for processing while streaming • Reviewed UCM State Machine • Added new security analysis appendix • Changed Document Status from Final to published
2019-03-29	19-03	AUTOSAR Release Management	• Updating Package Management state machine • New requirements for robustness against reset • Improving specification item atomicity • Fixing errors in chapter Service Interfaces
2018-10-31	18-10	AUTOSAR Release Management	• Updated interaction other functional clusters like PER and EMO/SM • Introduction of vehicle package distribution
2018-03-29	18-03	AUTOSAR Release Management	• Extended and updated service interface • Introduction of Software Package • Introduction to securing update process
2017-10-27	17-10	AUTOSAR Release Management	• Initial release

Disclaimer

Contents

1 Introduction and functional overview

This software specification contains the functional description and interfaces of the functional cluster Update and Configuration Management which belongs to the AUTOSAR Adaptive Platform Services. Update and Configuration Management has the responsibility of installing, updating and removing software on an AUTOSAR Adaptive Platform in a safe and secure way while not sacrificing the dynamic nature of the AUTOSAR Adaptive Platform.

The Update and Configuration Management functional cluster is responsible for:

- Version reporting of the software present in the AUTOSAR Adaptive Platform
- Receiving and buffering software updates
- Checking that enough resources are available to ensure a software update
- Performing software updates and providing log messages and progress information
- Validating the outcome of a software update
- Providing rollback functionality to restore a known functional state in case of failure

In addition to updating and changing software on the AUTOSAR Adaptive Platform, the Update and Configuration Management is also responsible for updates and changes to the AUTOSAR Adaptive Platform itself, including all functional clusters, the underlying POSIX OS and its kernel with the responsibilities defined above.

In order to allow flexibility in how Update and Configuration Management is used, it will expose its functionality via ara::com service interfaces, not direct APIs. This ensures that the user of the functional cluster Update and Configuration Management does not have to be located on the same ECU.

2 Acronyms and abbreviations

The glossary below includes acronyms and abbreviations relevant to the UCM module that are not included in the [1, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
DM	AUTOSAR Adaptive Diagnostic Management
UCM	Update and Configuration Management
V-UCM	V-UCM is distributing packages and coordinating an update campaign in a vehicle
Backend	Backend is a server hosting Software Packages
OTA Client	OTA Client is an Adaptive Application in communication with Backend Over The Air
Vehicle Driver Application	Vehicle Driver Application is an Adaptive Application in communication with Vehicle Driver Human to Machine Interface
Application Error	Errors returned by UCM
Boot options	Boot Manager Configuration
VCI	Vehicle Communication Interface
MVCI	Modular Vehicle Communication Interface
D-PDU API	Diagnostic Protocol Data Unit Application Programming Interface
RDF	Root Description File
MDF	Module Description File
integrity check	verification method proving there has not been any alteration of the artefact content
dependency check	verification method proving that all configured dependencies will be fulfilled after finishing the activation.

Some technical terms used in this document are already defined in the corresponding document mentioned in the table below. This is to avoid duplicate definition of the technical term. And to refer to the correct document.

Term	Description
Adaptive Application	see [1] AUTOSAR Glossary
Application	see [1] AUTOSAR Glossary
AUTOSAR Adaptive Platform	see [1] AUTOSAR Glossary
AUTOSAR Classic Platform	see [1] AUTOSAR Glossary
Electronic Control Unit	see [1] AUTOSAR Glossary
Adaptive Platform Foundation	see [1] AUTOSAR Glossary
Adaptive Platform Services	see [1] AUTOSAR Glossary
Manifest	see [1] AUTOSAR Glossary
Executable	see [1] AUTOSAR Glossary
Functional Cluster	see [1] AUTOSAR Glossary
Machine	see [1] AUTOSAR Glossary
Service	see [1] AUTOSAR Glossary
Service Interface	see [1] AUTOSAR Glossary
Service Discovery	see [1] AUTOSAR Glossary
Execution Management	see [2] AUTOSAR Execution Management
MachineFG	see [2] AUTOSAR Execution Management
State Management	see [3] AUTOSAR State Management
Function Group	see [3] AUTOSAR State Management
Communication Management	see [4] AUTOSAR Communication Management

Platform Health Management	see [5] AUTOSAR Platform Health Management
Software Cluster	see [1] AUTOSAR Glossary
Software Package	see [1] AUTOSAR Glossary
Vehicle Package	see [1] AUTOSAR Glossary
Vehicle State Manager	see [1] AUTOSAR Glossary

Table 2.1: Reference to Technical Terms

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] Specification of Execution Management
AUTOSAR_AP_SWS_ExecutionManagement

[3] Specification of State Management
AUTOSAR_AP_SWS_StateManagement

[4] Specification of Communication Management
AUTOSAR_AP_SWS_CommunicationManagement

[5] Specification of Platform Health Management
AUTOSAR_AP_SWS_PlatformHealthManagement

[6] General Requirements specific to Adaptive Platform
AUTOSAR_AP_RS_General

[7] Explanation of Adaptive Platform Software Architecture
AUTOSAR_AP_EXP_SWArchitecture

[8] Requirements on Update and Configuration Management
AUTOSAR_AP_RS_UpdateAndConfigurationManagement

[9] Explanation of Adaptive Platform Design
AUTOSAR_AP_EXP_PlatformDesign

[10] Specification of Vehicle Update and Configuration Management
AUTOSAR_AP_SWS_VehicleUpdateAndConfigurationManagement

[11] Specification of Manifest
AUTOSAR_AP_TPS_ManifestSpecification

[12] Specification of Persistency
AUTOSAR_AP_SWS_Persistency

3.2 Related specification

See chapter 3.1.

3.3 Further applicable specification

AUTOSAR provides a general specification [6] which is also applicable for UCM. The specification RS General shall be considered as additional and required specification for implementation of UCM.

4 Constraints and assumptions

4.1 Known Limitations

UCM is not responsible to initiate the update process. UCM realizes a service interface to achieve this operation. The user of this service interface is responsible to verify that the vehicle is in a updatable state before executing a software update procedure on demand. It is also in the responsibility of the user to communicate with other AUTOSAR Adaptive Platforms or AUTOSAR Classic Platforms within the vehicle.

The UCM receives a locally available software package for processing. The software package is usually downloaded from the OEM backend. The download of the software packages has to be done by another application, i.e. UCM does not manage the connection to the OEM backend. Prior to triggering their processing, the software packages have to be transferred to UCM by using the provided ara::com interface.

The UCM update process is designed to cover updates on use case with single AUTOSAR Adaptive Platform. UCM can update Adaptive Applications, the AUTOSAR Adaptive Platform itself, including all functional clusters and the underlying OS.

The UCM is not responsible for enforcing authentication and access control to the provided interfaces. The document currently does not provide any mechanism for the confidentiality protection as well as measures against denial of service attacks. The assumption is that the platform preserves the integrity of parameters exchanged between UCM and its user.

The possibility to restart a specific application instead of a Machine reboot depends of the kind of update and application, is therefore implementation specific and is defined in the Software Package manifest.

UCM does only support updates of ARA::COM and UDS (ISO-14229) compliant ECUs. UCM is not controlling any action done by diagnostic tool directly updating a Classic platform. For instance UCM cannot protect against downgrading of a Software Cluster in a Classic platform by a diagnostic tool.

4.2 Applicability to car domains

No restrictions to applicability.

5 Dependencies to other functional clusters

This chapter provides an overview of the dependencies to other Functional Clusters in the AUTOSAR Adaptive Platform. Section 5.1 “Provided Interfaces” lists the interfaces provided by `Update and Configuration Management` to other Functional Clusters. Section 5.2 “Required Interfaces” lists the interfaces required by `Update and Configuration Management`.

A detailed technical architecture documentation of the AUTOSAR Adaptive Platform is provided in [7].

5.1 Provided Interfaces

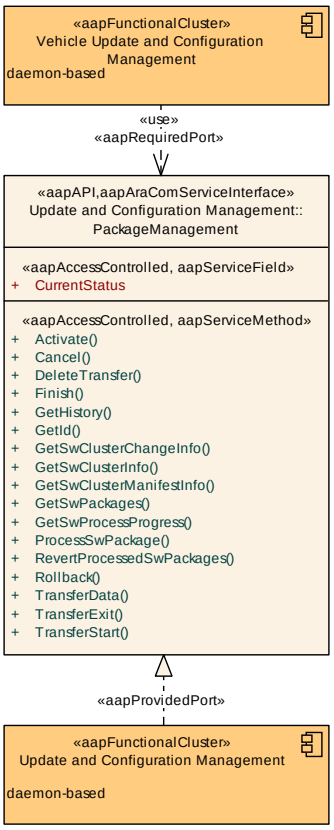

Figure 5.1: Interfaces provided by Update and Configuration Management to other Functional Clusters

Figure 5.1 shows interfaces provided by `UpdateAndConfigurationManagement` to other Functional Clusters within the AUTOSAR Adaptive Platform. Table 5.1 provides a complete list of interfaces provided to other Functional Clusters within the AUTOSAR Adaptive Platform.

Interface	*Functional Cluster*	*Purpose*
`PackageManagement`	Vehicle Update and Configuration Management	This interface is used to control different `Update and Configuration Management` instances and e.g., applications implementing the same interface located within the vehicle that act as an adapter to install software packages on third-party systems. `Vehicle Update and Configuration Management` is able to differentiate between the service instances by matching the result of `GetId` with an ID provided in a `Software Package`.

Table 5.1: Interfaces provided to other Functional Clusters

5.2 Required Interfaces

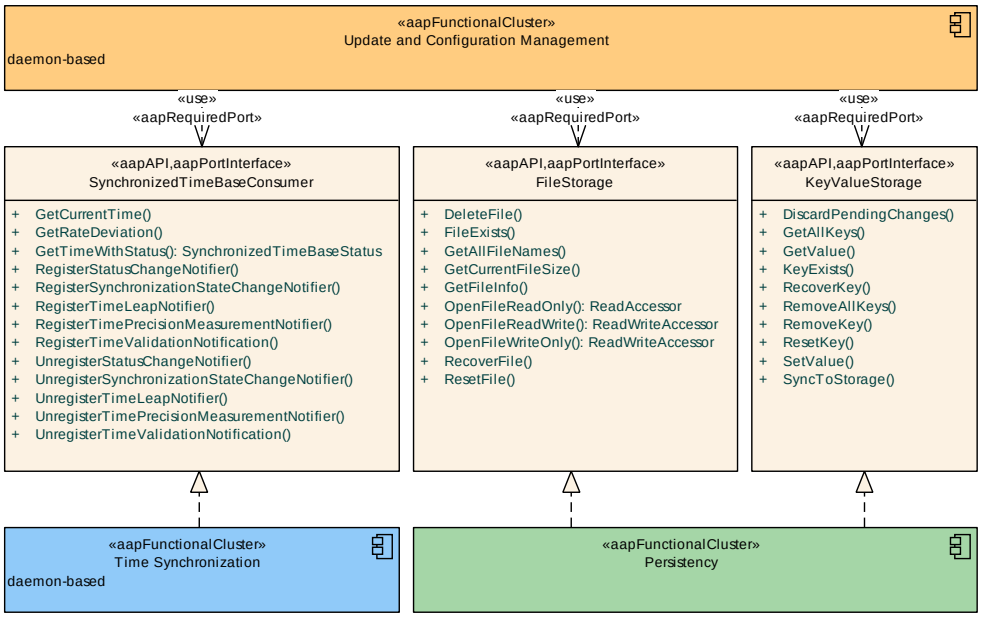

Figure 5.2: Interfaces required by Update and Configuration Management from other Functional Clusters

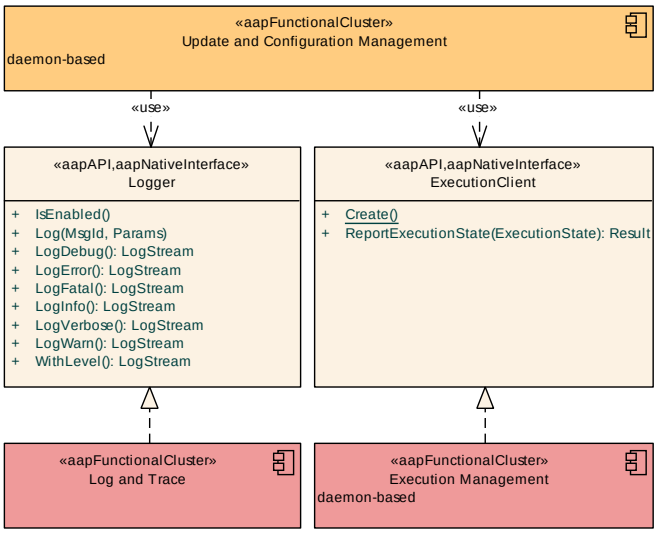

Figure 5.3: Interfaces required by Update and Configuration Management from other Functional Clusters

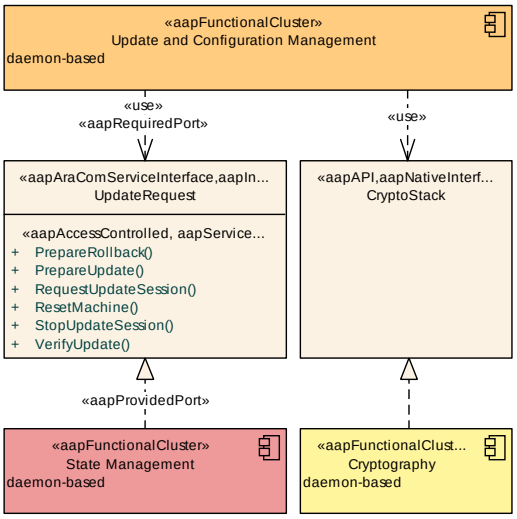

Figure 5.4: Interfaces required by Update and Configuration Management from other Functional Clusters

Figures 5.2, 5.3, and 5.4 show interfaces required by `UpdateAndConfiguration-Management` from other Functional Clusters within the AUTOSAR Adaptive Platform. Table 5.2 provides a complete list of required interfaces from other Functional Clusters within the AUTOSAR Adaptive Platform.

Functional Cluster	*Interface*	*Purpose*
Cryptography	CryptoStack	This interface may be used e.g., to verify the integrity and authenticity of `Software Package`s.
Execution Management	ExecutionClient	This interface shall be used by the daemon process(es) inside `Update and Configuration Management` to report their execution state to `Execution Management`.

△

Functional Cluster	*Interface*	*Purpose*
Log and Trace	Logger	`Update and Configuration Management` shall use this interface to log standardized messages.
Persistency	KeyValueStorageOperations	`Update and Configuration Management` should use this interface to store its internal state.
Persistency	FileStorageOperations	`Update and Configuration Management` should use this interface to store files, for example received software packages.
Persistency	FileStorage	`Update and Configuration Management` should use this interface to store files, for example received software packages.
Persistency	KeyValueStorage	`Update and Configuration Management` should use this interface to store its internal state.
Persistency	ReadAccessor	`Update and Configuration Management` should use this interface to store files, for example received software packages.
Persistency	ReadWriteAccessor	This interface should be used to store files, for example downloaded packages.
Platform Health Management	SupervisedEntity	This interface should be used to supervise the daemon process(es) of `Update and Configuration Management`.
State Management	UpdateRequest	This interface is used to interact with `State Management` of the Adaptive Platform during an update.
Time Synchronization	SynchronizedTimeBaseConsumer	`Update and Configuration Management` shall use this interface to get latest timestamp.

Table 5.2: Interfaces required from other Functional Clusters

5.3 Interfaces to Adaptive State Management

`UCM` relies on `State Management` and its provided `UpdateRequest` Service Interface to perform the necessary `Function Group` state changes needed to activate the newly installed, updated or removed software.

Certain applications can conflict with the update process or the newly updated package, and they need to be stopped during the update process. This could be achieved by putting the machine to a safe `Machine` state, by activating a combination of suitable `Function Groups` and its states. It is the responsibility of the platform integrator to define this state or `Function Groups`. The Adaptive Application accessing the `UCM`, should make sure that the platform is switched to this state (using interfaces from `State Management`), before starting the update.

6 Requirements Tracing

The following tables reference the requirements specified in [8] 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_EM_00014]	`Execution Management` shall support a Trusted Platform.	[SWS_UCM_00202]
[RS_SM_00001]	`State Management` shall coordinate and control multiple sets of `Applications`.	[SWS_UCM_00242]
[RS_UCM_00001]	`UCM` shall support installing new software on `AUTOSAR Adaptive Platform`	[SWS_UCM_00001] [SWS_UCM_00017] [SWS_UCM_00073] [SWS_UCM_00099] [SWS_UCM_00131] [SWS_UCM_00137] [SWS_UCM_00165] [SWS_UCM_00240] [SWS_UCM_00266] [SWS_UCM_00305]
[RS_UCM_00002]	`UCM` shall support reporting version information for an `AUTOSAR Adaptive Platform`	[SWS_UCM_00004] [SWS_UCM_00039] [SWS_UCM_00040] [SWS_UCM_00069] [SWS_UCM_00071] [SWS_UCM_00077] [SWS_UCM_00078] [SWS_UCM_00079] [SWS_UCM_00112] [SWS_UCM_00130] [SWS_UCM_00131] [SWS_UCM_00175] [SWS_UCM_00176] [SWS_UCM_00185] [SWS_UCM_00311] [SWS_UCM_00312] [SWS_UCM_00319] [SWS_UCM_CONSTR_00001] [SWS_UCM_CONSTR_00002] [SWS_UCM_CONSTR_00014]
[RS_UCM_00003]	`UCM` shall support updating installed software on Adaptive Platform	[SWS_UCM_00017] [SWS_UCM_00073] [SWS_UCM_00165] [SWS_UCM_00190] [SWS_UCM_00257] [SWS_UCM_00306]
[RS_UCM_00004]	`UCM` shall support uninstalling software on `AUTOSAR Adaptive Platform`	[SWS_UCM_00001] [SWS_UCM_00073] [SWS_UCM_00137] [SWS_UCM_00165] [SWS_UCM_00184] [SWS_UCM_00266] [SWS_UCM_00273]
[RS_UCM_00005]	`UCM` shall make sure that persistent data owned by uninstalled software is deleted	[SWS_UCM_00184] [SWS_UCM_00273]
[RS_UCM_00006]	`UCM` shall verify `Software Package` authenticity and integrity using strong cryptographic techniques	[SWS_UCM_00038] [SWS_UCM_00039] [SWS_UCM_00040] [SWS_UCM_00077] [SWS_UCM_00079] [SWS_UCM_00092] [SWS_UCM_00098] [SWS_UCM_00136] [SWS_UCM_00200]
[RS_UCM_00007]	`UCM` shall check that software dependencies are fulfilled	[SWS_UCM_00026] [SWS_UCM_00027] [SWS_UCM_00136] [SWS_UCM_00161] [SWS_UCM_00231] [SWS_UCM_00260] [SWS_UCM_00313] [SWS_UCM_00314] [SWS_UCM_00315] [SWS_UCM_00316] [SWS_UCM_00317] [SWS_UCM_00318]
[RS_UCM_00008]	`UCM` shall support a recovery mechanism in case of failed activation	[SWS_UCM_00005] [SWS_UCM_00024] [SWS_UCM_00107] [SWS_UCM_00110] [SWS_UCM_00111] [SWS_UCM_00126] [SWS_UCM_00127] [SWS_UCM_00131] [SWS_UCM_00146] [SWS_UCM_00155] [SWS_UCM_00162] [SWS_UCM_00163] [SWS_UCM_00164] [SWS_UCM_00264] [SWS_UCM_00282] [SWS_UCM_00299] [SWS_UCM_00302]

▽

△

Requirement	Description	Satisfied by
[RS_UCM_00010]	UCM shall support reporting of Software Packages downloaded for AUTOSAR Adaptive Platform	[SWS_UCM_00038] [SWS_UCM_00039] [SWS_UCM_00040] [SWS_UCM_00069] [SWS_UCM_00077] [SWS_UCM_00079] [SWS_UCM_00131] [SWS_UCM_CONSTR_00001] [SWS_UCM_CONSTR_00002]
[RS_UCM_00011]	UCM shall support reporting software versions which have been installed and will be activated when new versions are activated	[SWS_UCM_00030] [SWS_UCM_00038] [SWS_UCM_00039] [SWS_UCM_00040] [SWS_UCM_00077] [SWS_UCM_00078] [SWS_UCM_00079] [SWS_UCM_00131] [SWS_UCM_00185] [SWS_UCM_00191] [SWS_UCM_00192] [SWS_UCM_00193] [SWS_UCM_00194] [SWS_UCM_00195] [SWS_UCM_00196] [SWS_UCM_00197] [SWS_UCM_00198] [SWS_UCM_00199] [SWS_UCM_00286] [SWS_UCM_00287] [SWS_UCM_00311] [SWS_UCM_00312] [SWS_UCM_CONSTR_00001] [SWS_UCM_CONSTR_00002] [SWS_UCM_CONSTR_00014]
[RS_UCM_00012]	UCM shall check the consistency of transferred Software Package	[SWS_UCM_00029] [SWS_UCM_00038] [SWS_UCM_00039] [SWS_UCM_00040] [SWS_UCM_00077] [SWS_UCM_00079] [SWS_UCM_00092] [SWS_UCM_00104] [SWS_UCM_00136] [SWS_UCM_00207] [SWS_UCM_00213] [SWS_UCM_00267] [SWS_UCM_CONSTR_00012]
[RS_UCM_00013]	UCM shall check that it has enough resources to receive, process and store the Software Package and associated data	[SWS_UCM_00007] [SWS_UCM_00008] [SWS_UCM_00010] [SWS_UCM_00087] [SWS_UCM_00088] [SWS_UCM_00136] [SWS_UCM_00140] [SWS_UCM_00145] [SWS_UCM_00206] [SWS_UCM_00217] [SWS_UCM_00243] [SWS_UCM_00275] [SWS_UCM_00276] [SWS_UCM_00283] [SWS_UCM_00289] [SWS_UCM_00305] [SWS_UCM_00329]
[RS_UCM_00014]	UCM shall check that correct amount of data has been transferred for the Software Package	[SWS_UCM_00136] [SWS_UCM_00204] [SWS_UCM_00205] [SWS_UCM_00243]
[RS_UCM_00015]	UCM shall remove all unneeded data after Software Package processing has finished	[SWS_UCM_00020] [SWS_UCM_00131] [SWS_UCM_00265] [SWS_UCM_00285]
[RS_UCM_00018]	UCM shall announce when an application has been installed, updated or uninstalled	[SWS_UCM_00021] [SWS_UCM_00131] [SWS_UCM_00259]
[RS_UCM_00019]	UCM shall support simultaneous transfers multiple Software Packages	[SWS_UCM_00007] [SWS_UCM_00008] [SWS_UCM_00010] [SWS_UCM_00031] [SWS_UCM_00075] [SWS_UCM_00087] [SWS_UCM_00088] [SWS_UCM_00098] [SWS_UCM_00140] [SWS_UCM_00145] [SWS_UCM_00148] [SWS_UCM_00203] [SWS_UCM_00204] [SWS_UCM_00205] [SWS_UCM_00206] [SWS_UCM_00208] [SWS_UCM_00212] [SWS_UCM_00214] [SWS_UCM_00215] [SWS_UCM_00216] [SWS_UCM_00275] [SWS_UCM_00276] [SWS_UCM_00283]

▽

△

Requirement	Description	Satisfied by
[RS_UCM_00020]	UCM shall support cancellation of an update or install operation	[SWS_UCM_00003] [SWS_UCM_00167] [SWS_UCM_00234] [SWS_UCM_00235] [SWS_UCM_00236] [SWS_UCM_00237] [SWS_UCM_00239] [SWS_UCM_00278] [SWS_UCM_00279]
[RS_UCM_00021]	UCM shall support atomic activation of installed or updated Software Clusters	[SWS_UCM_00022] [SWS_UCM_00025] [SWS_UCM_00094] [SWS_UCM_00131] [SWS_UCM_00241] [SWS_UCM_00259] [SWS_UCM_00260] [SWS_UCM_00280]
[RS_UCM_00023]	UCM shall provide an interface to read progress of the update	[SWS_UCM_00018] [SWS_UCM_00131] [SWS_UCM_00220]
[RS_UCM_00024]	UCM shall provide an interface to read the state of UCM	[SWS_UCM_00019] [SWS_UCM_00044] [SWS_UCM_00080] [SWS_UCM_00081] [SWS_UCM_00083] [SWS_UCM_00084] [SWS_UCM_00085] [SWS_UCM_00131] [SWS_UCM_00147] [SWS_UCM_00149] [SWS_UCM_00150] [SWS_UCM_00151] [SWS_UCM_00152] [SWS_UCM_00153] [SWS_UCM_00154] [SWS_UCM_00166] [SWS_UCM_00168] [SWS_UCM_00169] [SWS_UCM_00258] [SWS_UCM_00293] [SWS_UCM_00300] [SWS_UCM_00301]
[RS_UCM_00025]	UCM shall support receiving of Software Package data	[SWS_UCM_00007] [SWS_UCM_00008] [SWS_UCM_00010] [SWS_UCM_00031] [SWS_UCM_00032] [SWS_UCM_00087] [SWS_UCM_00088] [SWS_UCM_00098] [SWS_UCM_00131] [SWS_UCM_00140] [SWS_UCM_00145] [SWS_UCM_00165] [SWS_UCM_00166] [SWS_UCM_00167] [SWS_UCM_00168] [SWS_UCM_00169] [SWS_UCM_00206] [SWS_UCM_00217] [SWS_UCM_00219] [SWS_UCM_00243] [SWS_UCM_00272] [SWS_UCM_00275] [SWS_UCM_00276] [SWS_UCM_00283] [SWS_UCM_00294]
[RS_UCM_00026]	UCM shall process installation of new Software Packages, updates and removal of existing Software Packages sequentially	[SWS_UCM_00017] [SWS_UCM_00044] [SWS_UCM_00122] [SWS_UCM_00184] [SWS_UCM_00218] [SWS_UCM_00219] [SWS_UCM_00240] [SWS_UCM_00257] [SWS_UCM_00258] [SWS_UCM_00261] [SWS_UCM_00262] [SWS_UCM_00263] [SWS_UCM_00265] [SWS_UCM_00273] [SWS_UCM_00277] [SWS_UCM_00281]
[RS_UCM_00027]	UCM shall be able to safely recover from unexpected interruption.	[SWS_UCM_00157] [SWS_UCM_00158] [SWS_UCM_00270] [SWS_UCM_00302]
[RS_UCM_00028]	UCM shall support updating Functional Clusters	[SWS_UCM_00100] [SWS_UCM_00245] [SWS_UCM_00306]
[RS_UCM_00029]	UCM shall support updating the underlying Operating System	[SWS_UCM_00101] [SWS_UCM_00245]
[RS_UCM_00030]	UCM shall be able to verify the updated software during activation	[SWS_UCM_00107] [SWS_UCM_00111] [SWS_UCM_00126] [SWS_UCM_00127] [SWS_UCM_00146] [SWS_UCM_00155] [SWS_UCM_00162] [SWS_UCM_00163] [SWS_UCM_00164] [SWS_UCM_00260] [SWS_UCM_00264]
[RS_UCM_00031]	UCM shall prevent installation of arbitrary previous version of an Adaptive Application or the Adaptive Platform	[SWS_UCM_00103] [SWS_UCM_00190]

▽

△

Requirement	Description	Satisfied by
[RS_UCM_00032]	UCM shall provide an interface to return UCM's action history	[SWS_UCM_00115] [SWS_UCM_00131] [SWS_UCM_00132] [SWS_UCM_00133] [SWS_UCM_00134] [SWS_UCM_00135] [SWS_UCM_00160] [SWS_UCM_00271] [SWS_UCM_00292]
[RS_UCM_00044]	UCM Initialization	[SWS_UCM_00274]
[RS_UCM_00045]	UCM shall report production errors	[SWS_UCM_00302] [SWS_UCM_00303] [SWS_UCM_00320] [SWS_UCM_00321] [SWS_UCM_00322] [SWS_UCM_00323] [SWS_UCM_00324] [SWS_UCM_00325] [SWS_UCM_00326] [SWS_UCM_00327]
[RS_VUCM_00035]	V-UCM shall coordinate software update in a vehicle across multiple Electronic Control Units	[SWS_UCM_00309]
[RS_VUCM_00036]	V-UCM shall use platform communication services for interacting with UCMs	[SWS_UCM_00173]
[RS_VUCM_00037]	V-UCM shall ensure it is safe to perform any modification to the vehicle	[SWS_UCM_00313] [SWS_UCM_00314] [SWS_UCM_00315] [SWS_UCM_00316] [SWS_UCM_00317] [SWS_UCM_00318]
[RS_VUCM_00039]	V-UCM shall prevent processing of compromised Vehicle Packages	[SWS_UCM_00200]

Table 6.1: RequirementsTracing

7 Functional specification

7.1 Software Cluster lifecycle

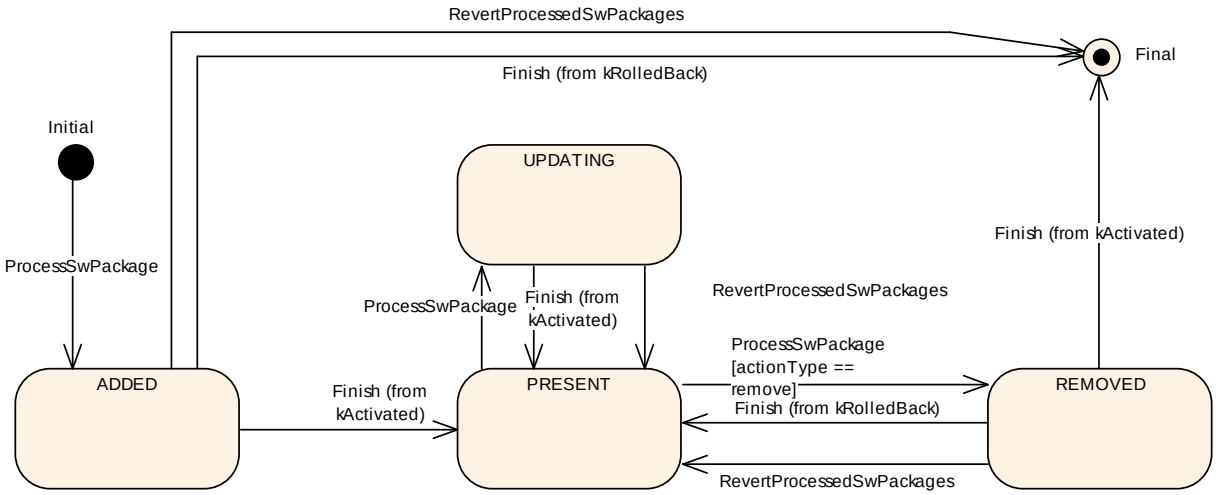

Figure 7.1: State Machine for a `Software Cluster`

The state machine in Fig. 7.1 describes the life-cycle states of a `Software Cluster`. These states are reported with `GetSwClusterChangeInfo` method.

[SWS_UCM_00191] **`Software Cluster` life-cycle state `kAdded`** ⌈A `Software Cluster` state shall be `kAdded` after the `Software Cluster` is successfully processed with `ProcessSwPackage` method call on the `AUTOSAR Adaptive Platform` and if it was not previously present in the `AUTOSAR Adaptive Platform` and before activation is finished.⌋*(RS_UCM_00011)*

[SWS_UCM_00192] **`Software Cluster` life-cycle state transition from `kAdded` to `kPresent`** ⌈A `Software Cluster` state shall change from `kAdded` to `kPresent` after a successful activation of a newly added `Software Cluster` with `Finish` method call.⌋*(RS_UCM_00011)*

[SWS_UCM_00195] **`Software Cluster` life-cycle state `kUpdating`** ⌈A `Software Cluster` state shall be `kUpdating` after a successful processing of the updated `Software Cluster` with `ProcessSwPackage` method call and before activation is finished.⌋*(RS_UCM_00011)*

[SWS_UCM_00193] **`Software Cluster` life-cycle state transition from `kUpdating` to `kPresent`** ⌈A `Software Cluster` state shall change from `kUpdating` to `kPresent` after a successful activation of the updated `Software Cluster` with `Finish` method call, or after reverting the `Software Cluster` update with a `RevertProcessedSwPackages` method call.⌋*(RS_UCM_00011)*

[SWS_UCM_00196] **`Software Cluster` life-cycle state `kRemoved`** ⌈A `Software Cluster` state shall be `kRemoved` after successful completion of method `ProcessSwPackage` which involves the removal of the existed `Software Cluster` and before activation is finished.⌋*(RS_UCM_00011)*

[SWS_UCM_00194] **Software Cluster life-cycle state transition from kRemoved to kPresent in case of RevertProcessedSwPackages call** ⌈A Software Cluster state shall change from kRemoved to kPresent after a successful call to RevertProcessedSwPackages method in case the Software Cluster was previously requested to be removed by ProcessSwPackage method call.⌋*(RS_UCM_00011)*

[SWS_UCM_00286] **Software Cluster life-cycle state transition from kRemoved to kPresent in case of Finish call** ⌈A Software Cluster state shall change from kRemoved to kPresent after a successful call to Finish method in case a Software Cluster being removed has to be rolled back after a failing activation.⌋*(RS_UCM_00011)*

[SWS_UCM_00197] End of Software Cluster life-cycle state from state kAdded in case of RevertProcessedSwPackages call ⌈A Software Cluster shall reach the end of its life-cycle from kAdded after a successful removal of a newly added Software Cluster with RevertProcessedSwPackages method call in case the Software Cluster was previously requested to be added by ProcessSwPackage method call.⌋*(RS_UCM_00011)*

[SWS_UCM_00287] End of Software Cluster life-cycle state from state kAdded in case of Finish call ⌈A Software Cluster shall reach the end of its life-cycle from kAdded after a successful removal of a newly added Software Cluster with Finish method call in case the newly added Software Cluster has to be rolled back after a failing activation.⌋*(RS_UCM_00011)*

[SWS_UCM_00198] End of Software Cluster life-cycle state from state kRemoved ⌈A Software Cluster shall reach the end of its life-cycle if it is successfully removed with a Finish method call and the Software Cluster is in state kRemoved.⌋*(RS_UCM_00011)*

[SWS_UCM_00199] Reporting of Software Cluster reaching end of life-cycle ⌈Any Software Cluster reaching the end of its life-cycle shall not be reported by UCM any more.⌋*(RS_UCM_00011)*

7.2 Technical Overview

One of the declared goals of AUTOSAR Adaptive Platform is the ability to flexibly update the software and its configuration through over-the-air updates. During the life-cycle of an AUTOSAR Adaptive Platform, UCM is responsible to perform software modifications on the machine and to retain consistency of the whole system.

The UCM Functional Cluster provides a service interface that exposes its functionality to retrieve AUTOSAR Adaptive Platform software information and consistently execute software updates. Since ara::com is used, the client using the UCM service interface can be located on the same AUTOSAR Adaptive Platform, but also remote clients are possible.

The service interface has been primarily designed with the goal to make it possible to use standard diagnostic services for downloading and installing software updates for the AUTOSAR Adaptive Platform. However, the methods and fields in the service interface are designed in such a way that they can be used in principle by any Adaptive Application. UCM does not impose any specific protocol on how data is transferred to the AUTOSAR Adaptive Platform and how package processing is controlled. In particular UCM does not expose diagnostic services.

As shown in Figure 7.2, whether the use case is an over-the-air update or garage update done through diagnostics, it is not visible to the UCM. The UCM Client abstracts the use case from the UCM and forwards the data stream and sequence control commands to the UCM. Later in this document, the term UCM Client is used to describe an Adaptive Application that consumes UCM PackageManagement services through UCM ara::com API. Diagnostic Application and V-UCM are two examples of such UCM Clients.

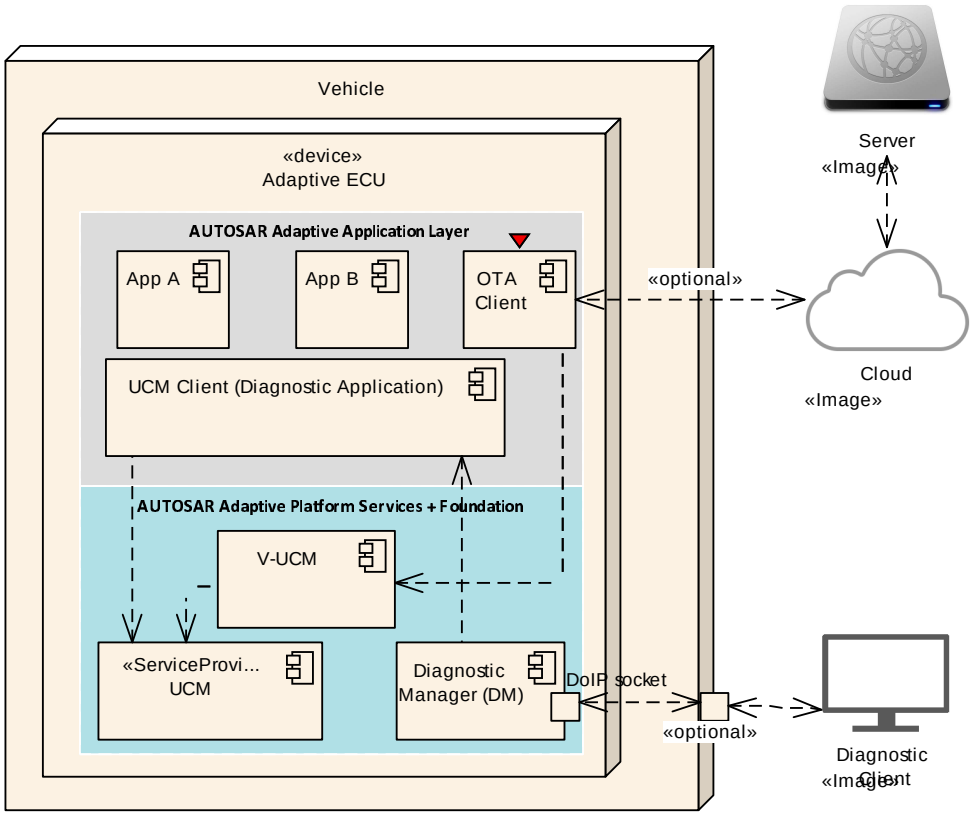

Figure 7.2: Architecture overview for diagnostic use case

7.2.1 UCM Diagnostic Application

	Diagnostic Application as UCM Client	Diagnostic Application as V-UCM Client
Purpose	Update standalone ECU/Machine without involvement of V-UCM	Update ECU/Machine as part of vehicle update through V-UCM

	Diagnostic Application as UCM Client	Diagnostic Application as V-UCM Client
Flow	Diagnostic Tool -> Diagnostic Manager -> Diagnostic App -> UCM	Diagnostic Tool -> Diagnostic Manager ->Diagnostic App -> V-UCM
Instance	One instance per standalone Adaptive ECU/Machine	One instance per vehicle
Artifacts handled	Receives Software Packages	Receives Vehicle Packages and Software Packages
UCM API (Service)	Package Management	Vehicle Package Management
ECUs/Machines being updated	Adaptive only (incl. Classic Machines on Adaptive ECU, if needed)	Any ECU (Adaptive, Classic, Proprietary)
Implemented by	ECU Vendor and/or OEM	OEM
References	• Figure 7.2, Figure 10.1, Figure 10.2, Figure 10.3 in this document • Figure "Vehicle Update Architecture" in AUTOSAR_EXP_PlatformDesign [9] • Figure "Interfaces of UCM" in AUTOSAR_EXP_SWArchitecture [7]	• Figure "Example of V-UCM architecture overview within a vehicle" and Figure "Classic platform update with V-UCM and diagnostic tool" in [10] document, Figure 10.2 in this document

Table 7.1: The usage of UCM Diagnostic Application

7.2.2 Software Package Management

The UCM update sequence consists three different phases:

- Software Package transfer: A phase in which, one or several Software Packages are transferred from the UCM's Client Application to the internal buffer of the UCM. For further information see chapter 7.3.
- Software Package processing: A phase in which the UCM performs the operation (kInstall, kUpdate, kRemove) on the relevant SoftwareCluster. For further information see chapter 7.5.
- Activation: A phase in which the UCM checks the dependencies of the SoftwareClusters that have been involved in the operation, then activates them and finally check that all the SoftwareClusters can be executed properly (via State Management) prior to finishing the update. For further information see chapter 7.6

7.2.2.1 Software Package

[SWS_UCM_00122] Software Package utilization ⌈The unit for deployment that the UCM shall take as input is called Software Package, see [1]. Each Software Package shall address a single SoftwareCluster.⌋*(RS_UCM_00026)*

A SoftwareCluster can act in two roles:

- 'Sub'-SoftwareCluster : It is a SoftwareCluster without diagnostic target address, containing processes, executables and further elements
- 'Root'-SoftwareCluster : It is a SoftwareCluster with a diagnostic target address that may reference several other 'Sub'-SoftwareClusters, which thus form a logical group.

A SoftwareCluster can be of the following categories expressed by the attribute SoftwareCluster.category :

- APPLICATION_LAYER: the SoftwareCluster can be removed by UCM
- PLATFORM_CORE: the SoftwareCluster cannot be removed as it would break the system.
- PLATFORM: the SoftwareCluster is part of the platform software and can be removed

[SWS_UCM_00245] Software Cluster category ⌈UCM shall not remove a SoftwareCluster that has installationBehavior set to value cannotBeRemoved. In case of such an attempt, UCM shall raise ApplicationError kSwclRemovalDenied.⌋*(RS_UCM_00028, RS_UCM_00029)*

A Software Package has to be modelled as a so-called SoftwareCluster which describes the content of a Software Package that is downloaded or uploaded to the AUTOSAR Adaptive Platform, see [11].

The term Software Package is used for the "physical", uploadable Software Package that is processed by UCM whereas the term SoftwareCluster is used for the modeling element. In the model, the content of a SoftwareCluster is define by references to all required model elements. The SoftwareCluster and the related model elements define the content of the manifest that is part of the Software Package. The Software Package format and the update scope are described in chapter "Content of a Software Package" as well as in [9].

[SWS_UCM_CONSTR_00012]{DRAFT} ⌈The SoftwareCluster aggregation of ArtifactChecksum shall not include the uri of this same SoftwareCluster manifest.⌋*(RS_UCM_00012)*

The uri attribute in ArtifactChecksum is referring to the artifact contained in the SoftwareCluster.

7.2.2.2 Content of a Software Package

Each Software Package addresses a single SoftwareCluster and contains manifests, executables and further data (depending on the role of the SoftwareCluster) as the example sketched in Figure 7.3.

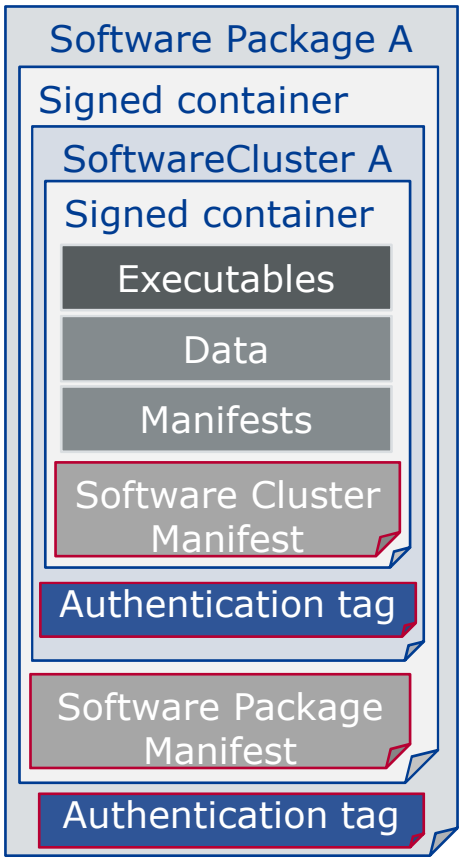

Figure 7.3: Software Package content description

A single Software Package is designed in a way that it could contain one or several executables of Adaptive Applications, kernel or firmware updates, or updated configuration and calibration data to be deployed on the AUTOSAR Adaptive Platform.

The Software Package manifest is recommended to be sent at the beginning in order for UCM to have early information of for instance memory usage or streaming.

An exemplary implementation of the adaptive workflow with Software Packages can be seen in chapter Methodology and Manifest in [9]. For more details on the Software Package class, you can refer to SoftwarePackage

[SWS_UCM_00112] Software Cluster and version ⌈SoftwareCluster's manifest shall include a name and a version following description of StrongRevisionLabelString.⌋ *(RS_UCM_00002)*

[SWS_UCM_00319] Semantic versionning ⌈UCM shall compute SoftwareCluster dependency check comparing only MajorVersion and MinorVersion.⌋*(RS_UCM_00002)*

[SWS_UCM_CONSTR_00001] ⌈If any content (for instance an executable or persistent data) of an already installed SoftwareCluster is modified by an incoming Software Package, then the version number of the incoming SoftwareCluster indicated in the Software Package shall be higher than the version number of the already installed SoftwareCluster.⌋*(RS_UCM_00002, RS_UCM_00010, RS_UCM_00011)*

If the constraint is violated, an error will be raised according to [SWS_UCM_00103].

A higher version number is achieved by an increment of any of the MajorVersion, the MinorVersion, or the PatchVersion.

For SoftwareCluster dependency check [SWS_UCM_00319] or Software Package compatibility against UCM [SWS_UCM_00161], PatchVersion and additional labels of StrongRevisionLabelString are not considered.

If there is a need to downgrade a failing SoftwareCluster (for instance, malfunction in the field that was not detected at activation), it will therefore be needed to repackage the same old SoftwareCluster that was properly working with an higher version number.

[SWS_UCM_00130] Software Cluster and version error ⌈If SoftwareCluster's manifest does not contain any SoftwareCluster.version following description of StrongRevisionLabelString, UCM shall raise the ApplicationErrorkInvalidPackageManifest.⌋*(RS_UCM_00002)*

[SWS_UCM_CONSTR_00014]{DRAFT} **Software Package and Software Cluster shortNames** ⌈SoftwarePackage and the referenced SoftwareCluster shall share the same shortName in order to be able to compare their versions.⌋*(RS_UCM_00002, RS_UCM_00011)*

The shortName of SoftwareCluster is by definition unique in its context as described into TPS Manifest document [11]. The applicable context for a SoftwareCluster is the complete vehicle, otherwise there would be conflicts of SoftwareCluster shortNames within a dependency model for instance. It is responsibility of integrator and tooling to make sure about shortName uniqueness within vehicle and it is typically applied by adding information to SoftwareCluster shortName like uri or architectural tree position (example: VirtualMachinename-SWCLshortName or filesystemUri-SWCLshortName)

7.2.2.3 Applications Persisted Data

Updating and rolling back of persisted data is handled completely by the application using persistency without involvement of UCM. A detailed explanation can be found in

the Persistency Specification [12]. An exception here is the removal of persistent data after a SoftwareCluster is removed.

[SWS_UCM_00184] Persistent data clean-up after Software Cluster removal ⌈UCM shall remove persistent data of a removed SoftwareCluster by using the information given in the application manifest, namely deploymentUri.uri and persistencyCentralStorageURI, in order to leave the AUTOSAR Adaptive Platform and the file system clean.⌋*(RS_UCM_00026, RS_UCM_00005, RS_UCM_00004)*

[SWS_UCM_00273]{DRAFT} **Persistent data clean-up after Software Cluster update that removes a process** ⌈UCM shall remove persistent data of a removed process by using the information given in the execution manifest, namely deploymentUri.uri and persistencyCentralStorageURI in order to leave the AUTOSAR Adaptive Platform and the file system clean.⌋*(RS_UCM_00026, RS_UCM_00005, RS_UCM_00004)*

Persistent data can include administrative and backup data.

[SWS_UCM_00305]{DRAFT} **Persistent data uri at Software Cluster installation** ⌈In the case of installation of a Software Cluster, UCM shall create the new uris following content of execution manifest, namely deploymentUri.uri and persistencyCentralStorageURI.⌋*(RS_UCM_00001, RS_UCM_00013)*

[SWS_UCM_00306]{DRAFT} **Persistent data uri change at update** ⌈In the case of uri change of persistent storages, UCM shall move the existing folders containing the persistent storages from the old uri to the new uri following content of execution manifest, namely deploymentUri.uri and persistencyCentralStorageURI.⌋*(RS_UCM_00003, RS_UCM_00028)*

[SWS_UCM_00329]{DRAFT} **Activate kPersistencyAllocationFailed** ⌈After Activate method is called and if the sum of all Software Clusters maximumAllowedSize exceeds available size defined by maxAvailablePersistencyStorageSpace which is reserved for persistency, then UCM shall raise ApplicationError kPersistencyAllocationFailed.⌋*(RS_UCM_00013)*

7.2.3 Runtime dependencies

Processes within a SoftwareCluster can have functional dependencies toward other SoftwareClusters.

Dependencies are described in the SoftwareCluster metamodel, see [11]. This dependency model allows to confirm for instance if there are missing or conflicting services within Machine or within the whole vehicle, or if required libraries located in another Software Cluster would be missing or conflicting with the being updated Software Cluster.

The rationale is, if UCM has to process several Software Packages, then execution dependencies may not be fulfilled at all times during the Software Packages process but must be fulfilled before changes can be activated.

At activation, UCM is starting a session with State Management (SM) which is requesting Execution Management (EM) to change states of FunctionGroups (Sequence diagram 10.4). Execution Management uses Execution Manifest which contains modelled execution dependencies. If those dependencies are not met at Verify state, Execution Management will report an error to State Management forwarding this error to UCM which will rollback.

7.2.4 Update scope and State Management

Software Package processed by UCM can contain Adaptive Applications, updates to AUTOSAR Adaptive Platform itself or to the underlying OS. Update type depends on the content of the Software Package.

[SWS_UCM_00099] Update of Adaptive Application ⌈UCM shall be able to update Adaptive Applications⌋*(RS_UCM_00001)*

[SWS_UCM_00100]{DRAFT} **Update of Functional Clusters** ⌈UCM shall be able to update all Functional Clusters, including UCM itself.⌋*(RS_UCM_00028)*

[SWS_UCM_00101]{DRAFT} **Update of Host** ⌈UCM shall be able to update the underlying OS hosting the AUTOSAR Adaptive Platform.⌋*(RS_UCM_00029)*

Definition of an updatable state with respect to the system setup is the OEM responsibility. Based on the system setup and the application, the system might need to be switched into a predefined state, to free resource to speed up the update, to block normal usage of software which might cause interruptions to update process and to block using functionality which might be interrupted by the update sequence.

[SWS_UCM_00257] Update session ⌈To confirm the system is in an updatable state, UCM shall start an update session by calling State Management UpdateRequest Service Interface RequestUpdateSession method after its dependency check triggered by Activate method call successfully completes.⌋*(RS_UCM_00026, RS_UCM_00003)*

[SWS_UCM_00258] Update session rejected ⌈If State Management UpdateRequest Service Interface RequestUpdateSession method call raises error kRejected, UCM shall transition from kActivating to kReady states and Activate method call shall return ApplicationError kUpdateSessionRejected.⌋*(RS_UCM_00026, RS_UCM_00024)*

If update session could be recurrently rejected, it is up to implementer to cache the dependency check result in order to avoid unnecessary computation and compute it only once.

During the update session, the minimum applications required for the Update process should be executed. This way system is more robust, more resources are free and user is blocked from using applications, of which failure could cause safety risk to the user.

Update of some components require a Machine reset to be performed. These components should be configured to be part of Function Group MachineFG, as the update sequence of Function Group MachineFG includes a Machine reset. Execution Management, State Management, Communication Management and UCM itself are good examples which probably require a Machine reset to activate the update. Other such components could be applications involved in the update sequence or applications involved in safety monitoring. Further details on Function Group MachineFG can be found in State Management.

7.3 Transferring Software Packages

To speed up the overall data transmission time, the package transfer is decoupled from the processing and activation process. This section describes requirements for initiation of a data transfer, the data transmission and ending of the data transmission.

Each Software Package gets its own state as soon as it is being transferred to UCM. The state machines in Fig. 7.4 specify the lifecycle of a Software Package that is transferred to and processed by UCM. During this lifecycle, a Software Package is uniquely identified with an id that UCM provides to the client.

The UCM has the possibility to keep the Software Package in kTransferred states in case it failed and retry later: transferring Software Package can be costly, if it is authenticated, there could be no reason to delete it if the update has not been successfully finished.

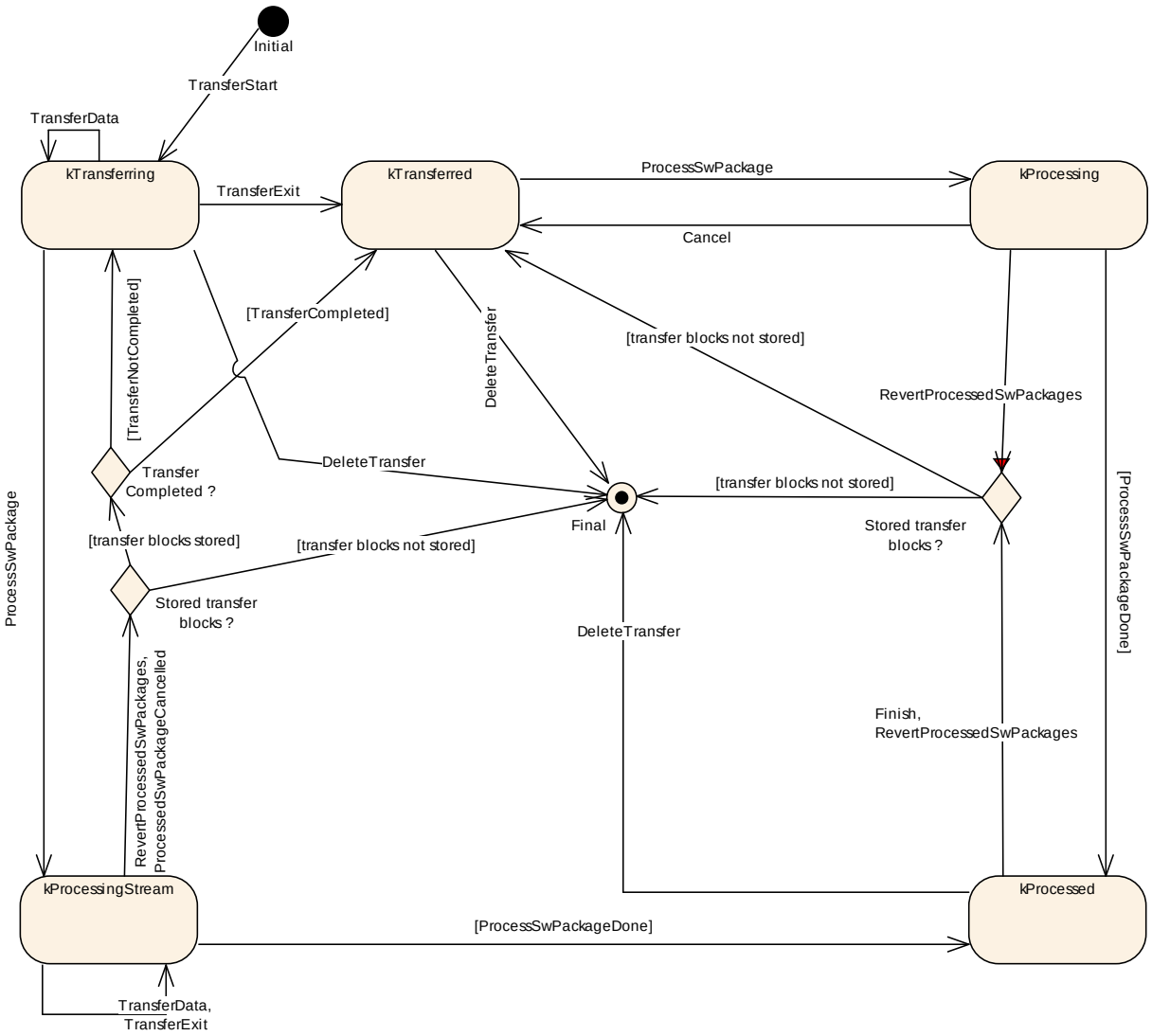

Figure 7.4: State Machine representing `Software Packages` lifecycle

[SWS_UCM_00007] Data transfer at any time ⌈`UCM` shall provide support to transfer `Software Packages` at any time when `UCM` is running. Transferring is decoupled from the `UCM` Package Management states.⌋ *(RS_UCM_00013, RS_UCM_00019, RS_UCM_00025)*

[SWS_UCM_00272]{DRAFT} **Transfer block size** ⌈`TransferStart` shall return `blockSize` parameter to indicate the maximum block size (unit: bytes, as defined by `maxBlockSize`) to be allowed to transfer in one `TransferData` method call.⌋ *(RS_UCM_00025)*

The block size should be aligned to flashing capability in case of Classic Platform capability for instance.

[SWS_UCM_00088] Preparation of data transfer ⌈Data transfer shall be prepared with the method `TransferStart`. In the preparation step the number of bytes to be transferred is provided by the client and `UCM` assigns an `id` for the `Software Package` to be transferred.⌋ *(RS_UCM_00013, RS_UCM_00019, RS_UCM_00025)*

While a Software Package is being transferred, if UCM receives a subsequent TransferStart call targeting another Software Package, UCM should make sure that the sum of the size of both Software Packages (the one being transferred and the one requested to be transferred) does not exceed the size of the UCM buffer. Otherwise, the TransferStart should raise the ApplicationError kInsufficientMemory and the newly requested transmission should be rejected as described above.

[SWS_UCM_00008] Executing the data transfer ⌈After successful call of TransferStart method, the transmission of the Software Package block-wise shall be supported by the method TransferData.⌋ *(RS_UCM_00013, RS_UCM_00019, RS_UCM_00025)*

[SWS_UCM_00145] Sequential order of data transfer ⌈The method TransferData shall support the parameter blockCounter that shall start with 0x01 and be incremented by one for each subsequent block.⌋ *(RS_UCM_00013, RS_UCM_00019, RS_UCM_00025)*

[SWS_UCM_00010] End of data transfer ⌈After transmission of a Software Package is completed, the transmission can be finished with method TransferExit.⌋ *(RS_UCM_00013, RS_UCM_00019, RS_UCM_00025)*

Software Package contains authentication and integrity tags, which are used during the transfer sequence to authenticate the content of the Software Package.

[SWS_UCM_00075] Multiple data transfers in parallel ⌈Handling of multiple data transfers in parallel shall be supported by UCM.⌋ *(RS_UCM_00019)*

If UCM provide enough buffering resources for Software Packages, several packages could be transferred (in parallel) before they are processed one after the other. The processing (i.e. unpacking and actually applying changes to the AUTOSAR Adaptive Platform) of Software Packages described by the state kProcessing is further detailed in Sect. 7.5.

[SWS_UCM_00021] Deleting transferred Software Packages ⌈UCM shall provide a method DeleteTransfer that shall delete the targeted Software Package and free the resources reserved to store that Software Package.⌋ *(RS_UCM_00018)*

[SWS_UCM_00069]{OBSOLETE} **Report information on Software Packages** ⌈UCM shall provide a method GetSwPackages of the interface service PackageManagement to provide the Software Packages' identifiers, names, versions, states, consecutive bytes received and consecutive blocks received.⌋ *(RS_UCM_00010, RS_UCM_00002)*

At the invocation of method GetSwPackages of the service interface PackageManagement, UCM returns the Software Packages' identifiers, names, versions, states, consecutive bytes received and consecutive blocks received.

If Software Package is in kTransferring state, it is not possible to get versions or names as manifest could not be complete or accessible, therefore method GetSwPackages should return empty values except for TransferID, ConsecutiveBytesReceived and ConsecutiveBlocksReceived at this particular state.

[SWS_UCM_00216] Validity of TransferId ⌈The TransferId of a Software Package shall be invalidated for further use when it reaches final lifecycle state.⌋*(RS_UCM_00019)*

7.3.1 Error handling in TransferStart

TransferStart allocates resources for the client transfer.

[SWS_UCM_00140] UCM insufficient memory ⌈TransferStart method shall raise the ApplicationError kInsufficientMemory if the UCM buffer has not enough resources to store the corresponding Software Package.⌋*(RS_UCM_00013, RS_UCM_00019, RS_UCM_00025)*

7.3.2 Error handling in TransferData

TransferData executes the following checks. It is recommended to follow the specified order.

[SWS_UCM_00275]{DRAFT} **TransferData error handling order** ⌈TransferData method shall check the following error conditions and return the respective error code.

1. [SWS_UCM_00208]
2. [SWS_UCM_00203]
3. [SWS_UCM_00204]
4. [SWS_UCM_00243]
5. [SWS_UCM_00205]
6. [SWS_UCM_00206]
7. [SWS_UCM_00289]
8. [SWS_UCM_00207]
9. [SWS_UCM_00294]
10. [SWS_UCM_00098]
11. [SWS_UCM_00092]
12. [SWS_UCM_00245]
13. [SWS_UCM_00103]

⌋*(RS_UCM_00013, RS_UCM_00019, RS_UCM_00025)*

[SWS_UCM_00208] **TransferData OperationNotPermitted** ⌈Calling TransferData after calling TransferExit for a specific TransferId shall raise the error ApplicationError kOperationNotPermitted⌋*(RS_UCM_00019)*

[SWS_UCM_00203] **TransferData InvalidTransferId** ⌈TransferData shall raise the error ApplicationError kInvalidTransferId in case an invalid TransferId (An ID that was not initiated by TransferStart or marked invalid by DeleteTransfer) is sent by the client.⌋*(RS_UCM_00019)*

[SWS_UCM_00204] **TransferData IncorrectBlock** ⌈TransferData shall raise ApplicationError kIncorrectBlock upon receipt of a block counter value that is successfully transmitted to UCM before or upon receipt of an unexpected block counter value.⌋*(RS_UCM_00014, RS_UCM_00019)*

[SWS_UCM_00243] **Too big block size received by UCM** ⌈In the case the received block size with TransferData exceeds the block size returned by TransferStart for the same TransferId, UCM shall raise the ApplicationError kIncorrectBlockSize.⌋*(RS_UCM_00013, RS_UCM_00014, RS_UCM_00025)*

[SWS_UCM_00205] **TransferData IncorrectSize** ⌈In case the transferred Software package size exceeds the provided size in TransferStart, TransferData shall raise ApplicationError kIncorrectSize⌋*(RS_UCM_00014, RS_UCM_00019)*

[SWS_UCM_00206] **TransferData InsufficientMemory** ⌈TransferData shall raise the error ApplicationError kInsufficientMemory if resources to store the Software Package ceased to exist during the transfer operation.⌋*(RS_UCM_00013, RS_UCM_00019, RS_UCM_00025)*

[SWS_UCM_00289]{DRAFT} **TransferData TransferFailed** ⌈TransferData shall raise the error ApplicationError kTransferFailed if UCM cannot persist transferred block.⌋*(RS_UCM_00013)*

[SWS_UCM_00207]{DRAFT} **TransferData BlockInconsistent** ⌈If UCM checks consistency of Block for each TransferData, UCM shall raise the error ApplicationError kBlockInconsistent in case Consistency check for transferred block fails.⌋*(RS_UCM_00012)*

The kBlockInconsistent error is intended to be used by the Flashing Adapter. The Flashing Adapter can calculate additional consistency information for each block internally, e.g. a CRC32 checksum. It can then use UDS protocol to send block data and checksum to the target ECU. In case checksum verification fails, the Flashing Adapter can report the kBlockInconsistent error to the V-UCM or diagnostic client application.

As described in section 7.2.2.2 and [11], each Software Package has an authentication tag CryptoServiceCertificate which protects integrity and authenticity. Therefore additional consistency check information is not needed. If authentication check fails, kAuthenticationFailed error is intended to be used instead.

[SWS_UCM_00294]{DRAFT} **Unsupported package format for UCM** ⌈In the case the Software Package archiving format is not supported, UCM TransferData

method shall return ApplicationError kUnsupportedPackageFormat.⌋(*RS_UCM_00025*)

[SWS_UCM_00098]{DRAFT} **Software Package Authentication failure** ⌈UCM shall raise the ApplicationError kAuthenticationFailed, if the Software Package authentication check fails.⌋(*RS_UCM_00006, RS_UCM_00019, RS_UCM_00025*)

This error can happen when TransferData, TransferExit and ProcessSwPackage methods are called. When kAuthenticationFailed error is raised, it is up to client to decide if a DeleteTransfer will be called or not. The behaviour may vary depending on the life cycle, meaning R&D phase or on the field phase.

TransferData checks the package version format in accordance to [SWS_UCM_00161] (kIncompatiblePackageVersion).

TransferData checks if the Software Cluster to be removed has attribute installationBehavior set to cannotBeRemoved. If this is the case, UCM shall not remove it in accordance to [SWS_UCM_00245].

TransferData checks if the Software Cluster version being updated is older than currently present in Machine in accordance to [SWS_UCM_00103] (kOldVersion).

7.3.3 Error handling in TransferExit

[SWS_UCM_00276]{DRAFT} **TransferExit error handling order** ⌈TransferExit method shall check the following error conditions and return the respective error code.

1. [SWS_UCM_00148]
2. [SWS_UCM_00212]
3. [SWS_UCM_00087]
4. [SWS_UCM_00294]
5. [SWS_UCM_00098]
6. [SWS_UCM_00092]
7. [SWS_UCM_00161]
8. [SWS_UCM_00213]
9. [SWS_UCM_00245]
10. [SWS_UCM_00103]

⌋(*RS_UCM_00013, RS_UCM_00019, RS_UCM_00025*)

[SWS_UCM_00148] Transfer sequence order ⌈Calling TransferExit without calling TransferData at least once or after TransferExit is called for a specific TransferID, shall raise the ApplicationError kOperationNotPermitted.⌋*(RS_UCM_00019)*

[SWS_UCM_00212] TransferExit InvalidTransferId ⌈TransferExit shall raise the error ApplicationError kInvalidTransferId in case an invalid TransferId is sent by the client.⌋*(RS_UCM_00019)*

[SWS_UCM_00087] Insufficient amount of data transferred ⌈When TransferExit method is called, UCM shall check if all blocks of the Software Package have been transferred according to the size parameter of TransferStart. If not UCM shall return ApplicationError kInsufficientData.⌋*(RS_UCM_00013, RS_UCM_00019, RS_UCM_00025)*

TransferExit checks if the Software Package archiving format is supported in accordance to [SWS_UCM_00294] (kUnsupportedPackageFormat).

TransferExit checks authentication in accordance to [SWS_UCM_00098] (kAuthenticationFailed).

[SWS_UCM_00092] Software Package integrity ⌈When TransferData or TransferExit method is called, UCM shall raise the ApplicationError kPackageInconsistent if the Software Package integrity check fails. This Software Package integrity check may be realized by the UCM via a Software Package Checksum check or via other mechanisms.⌋*(RS_UCM_00012, RS_UCM_00006)*

TransferExit checks the package version format in accordance to [SWS_UCM_00161] (kIncompatiblePackageVersion).

[SWS_UCM_00213] TransferExit kInvalidPackageManifest ⌈TransferExit shall raise the error ApplicationError kInvalidPackageManifest upon receival of an invalid manifest.⌋*(RS_UCM_00012)*

TransferExit checks if the Software Cluster to be removed has attribute installationBehavior set to cannotBeRemoved. If this is the case, UCM shall not remove it in accordance to [SWS_UCM_00245].

TransferExit checks if the Software Cluster version being updated is older than currently present in Machine in accordance to [SWS_UCM_00103] (kOldVersion).

7.3.4 Error handling in DeleteTransfer

[SWS_UCM_00283]{DRAFT} **DeleteTransfer error handling order** ⌈DeleteTransfer method shall check the following error conditions and return the respective error code.

1. [SWS_UCM_00214]

2. [SWS_UCM_00215]

⌋(*RS_UCM_00013*, *RS_UCM_00019*, *RS_UCM_00025*)

`DeleteTransfer` checks if the supplied parameter TransferId is valid.

[SWS_UCM_00214] **`DeleteTransfer` InvalidTransferId** ⌈`DeleteTransfer` shall raise the error `ApplicationError` `kInvalidTransferId` in case an invalid TransferId is sent by the client.⌋(*RS_UCM_00019*)

[SWS_UCM_00215] **`DeleteTransfer` OperationNotPermitted** ⌈Calling `DeleteTransfer` during processing or during the processing stream shall raise the error `ApplicationError` `kOperationNotPermitted`.⌋(*RS_UCM_00019*)

7.4 Processing of Software Packages from a stream

It is also possible to process a `Software Package` while the transfer is still ongoing. The following requirements apply for this use case.

[SWS_UCM_00165] **Processing from stream** ⌈The UCM may support calling `ProcessSwPackage` directly from stream without waiting to receive the `Software Package` completely.⌋(*RS_UCM_00001*, *RS_UCM_00003*, *RS_UCM_00004*, *RS_UCM_00025*)

[SWS_UCM_00166] **Processing from stream state** ⌈If UCM supports processing from stream and is in state `kIdle` or `kReady`, the method `ProcessSwPackage` for a `Software Package` in state `kTransferring` shall set this `Software Package` to state `kProcessingStream`.⌋(*RS_UCM_00024*, *RS_UCM_00025*)

[SWS_UCM_00167]{DRAFT} **Cancelling streamed packages** ⌈When `Cancel` is called, `UCM` shall remove all temporary and processed data of a `Software Package` in state `kProcessingStream`.⌋(*RS_UCM_00020*, *RS_UCM_00025*)

[SWS_UCM_00168] **Transferring while processing from stream** ⌈`Software Package` state shall remain in `kProcessingStream` when `TransferData` is called.⌋(*RS_UCM_00024*, *RS_UCM_00025*)

[SWS_UCM_00169] **Finishing transfer while processing from stream** ⌈`Software Package` state shall be set to `kProcessed` when `TransferExit` is called and the `Software Package` is completely processed.⌋(*RS_UCM_00024*, *RS_UCM_00025*)

[SWS_UCM_00200] **Failing authentication** ⌈`UCM` shall delete the `Software Package` and its related data processed by `ProcessSwPackage` call if authentication is failing at `TransferExit` or `ProcessSwPackage` call.⌋(*RS_VUCM_00039*, *RS_UCM_00006*)

7.5 Processing Software Packages

In contrast to package transmission, only one Software Package can be processed at the same time to ensure consistency of the system. In the following, a software or package processing can involve any combination of an installation, update or removal of applications, configuration data, calibration data or manifests. It is up to the vendor-specific metadata inside a Software Package to describe the tasks UCM has to perform for its processing. For a removal, this might involve metadata describing which data needs to be deleted. Nevertheless, the communication sequence between the triggering application of the software modification and UCM is the same in any case. For an update of an existing application, the Software Package can contain only partial data, e.g. just an updated version of the execution manifest. Any UCM Client need to confirm that UCM is in kIdle CurrentStatus state before starting any update (process/activate).

[SWS_UCM_00001]{OBSOLETE} **Starting the package processing** ⌈UCM shall provide a method ProcessSwPackage to process transferred Software Package. id corresponding to Software Package shall be provided for this method.⌋*(RS_UCM_00001, RS_UCM_00004)*

At the invocation of method ProcessSwPackage, UCM processes transferred Software Package with id argument corresponding to this Software Package.

[SWS_UCM_00137] Processing several update Software Packages ⌈UCM shall support processing of several Software Packages, not in parallel, by calling method ProcessSwPackage several times in sequence.⌋*(RS_UCM_00001, RS_UCM_00004)*

During package processing, the progress is provided.

[SWS_UCM_00018]{OBSOLETE} **Providing Progress Information** ⌈UCM shall provide a method GetSwProcessProgress to query the progress of executing the ProcessSwPackage method call for provided TransferId. Parameter progress shall be set to a value representing the progress between 0% and 100% (0x00 ... 0x64).⌋*(RS_UCM_00023)*

At the invocation of method GetSwProcessProgress, UCM returns the progress of executing the ProcessSwPackage method call for provided TransferId. Parameter progress will be set to a value representing the progress between 0% and 100% (0x00 ... 0x64).

[SWS_UCM_00003] Cancelling the package processing ⌈On call of Cancel method, UCM shall abort the running package processing task, undo the changes to the Software Cluster for which processing started and free the reserved resources used for it.⌋*(RS_UCM_00020)*

[SWS_UCM_00024] Revert all processed Software Packages ⌈UCM shall provide a method RevertProcessedSwPackages to revert all changes done with ProcessSwPackage.⌋*(RS_UCM_00008)*

The main difference between a RevertProcessedSwPackages and a Rollback is that the former can only be performed before the successful activation of the targeted Software Package(s) while the latter can only be performed after such activation.

Depending on the capabilities of UCM and of the updated target, RevertProcessedSwPackages is used to revert all the changes that have been applied by ProcessSwPackage. Cancel is also used to revert the changes of the Software Package for which processing started by ProcessSwPackage method call and identified by TransferId. For example, if an application with large resource files is updated "in place" (i.e. in the same partition) then it might not be feasible to revert the update. In this case, to perform a rollback the triggering application could download a Software Package to restore a stable version of the application.

7.5.1 Error handling during Processing Software Packages

[SWS_UCM_00277]{DRAFT} **ProcessSwPackage error handling order** ⌈ProcessSwPackage method shall check the following error conditions and return the respective error code.

1. [SWS_UCM_00219]
2. [SWS_UCM_00017]
3. [SWS_UCM_00218]
4. [SWS_UCM_00098]
5. [SWS_UCM_00161]
6. [SWS_UCM_00029]
7. [SWS_UCM_00285]
8. [SWS_UCM_00231]
9. [SWS_UCM_00217]
10. [SWS_UCM_00267]
11. [SWS_UCM_00104]
12. [SWS_UCM_00245]
13. [SWS_UCM_00103]
14. [SWS_UCM_00150]

⌋*(RS_UCM_00026)*

[SWS_UCM_00219] **ProcessSwPackage OperationNotPermitted** ⌈ProcessSwPackage shall raise the error ApplicationError kOperationNotPermitted in case the processing of the specified Software Package is already done.⌋*(RS_UCM_00025, RS_UCM_00026)*

[SWS_UCM_00017] Sequential Software Package Processing ⌈Once method ProcessSwPackage has been called by a client, further calls to the same method shall be rejected with ApplicationError kServiceBusy as long as CurrentStatus is different than kProcessing.⌋*(RS_UCM_00001, RS_UCM_00003, RS_UCM_00026)*

[SWS_UCM_00218] ProcessSwPackage InvalidTransferId ⌈ProcessSwPackage shall raise the error ApplicationError kInvalidTransferId in case an invalid TransferId is sent by the client.⌋*(RS_UCM_00026)*

ProcessSwPackage checks authentication in accordance to [SWS_UCM_00098] (kAuthenticationFailed)

[SWS_UCM_00161] Check Software Package version compatibility against UCM version ⌈At ProcessSwPackage, TransferData or TransferExit calls, UCM shall raise ApplicationError kIncompatiblePackageVersion if the MajorVersion and MinorVersion of minimumSupportedUcmVersion attribute of the Software Package is less than the current MajorVersion and MinorVersion of UCM as available in version attribute.⌋*(RS_UCM_00007)*

The Software Package is generated by a tooling including a packager which version could not match with the UCM version, leading to manifest interpretation issues for instance.

[SWS_UCM_00029] Consistency Check of Manifest ⌈UCM shall validate the content of the manifest against the schema defined for the meta-data(eg: for missing parameter or for value out of range of the parameter) and shall raise the ApplicationError kInvalidPackageManifest if it finds discrepancies there.⌋*(RS_UCM_00012)*

[SWS_UCM_00285]{DRAFT} **Removing or updating a Software Cluster not existing in the Machine** ⌈If a Software Package's action is to remove or update a Software Cluster that is not at one of the states kPresent, kRemoved, kUpdating and kAdded, UCM shall raise ApplicationError kSoftwareClusterMissing when ProcessSwPackage is called.⌋*(RS_UCM_00015)*

[SWS_UCM_00231]{DRAFT} **ProcessSwPackage IncompatibleDelta** ⌈ProcessSwPackage shall raise the error ApplicationError kIncompatibleDelta if deltaPackageApplicableVersion is different from the currently installed version of the referenced SoftwareCluster.⌋*(RS_UCM_00007)*

[SWS_UCM_00217]{DRAFT} **ProcessSwPackage InsufficientMemory** ⌈ProcessSwPackage method shall raise the ApplicationError kInsufficientMemory if the UCM buffer has not enough resources to process the corresponding Software Package.⌋*(RS_UCM_00013, RS_UCM_00025)*

[SWS_UCM_00267]{DRAFT} **Error when checksum is not recognised at processing time** ⌈If checksum attribute of ArtifactChecksum or CryptoProvider are not recognised, UCM shall raise the ApplicationError kInvalidChecksumDescription.⌋*(RS_UCM_00012)*

[SWS_UCM_00104] Integrity Check of processed Package ⌈UCM shall raise the ApplicationError kProcessedSoftwarePackageInconsistent if integrity check of the processed Software Packages fails.⌋ *(RS_UCM_00012)*

This operation is realized by the UCM to verify that it did not corrupt any files during the processing. This integrity check is vendor specific and may be realized by the UCM by checking the payload Checksum or by any other mechanisms.

ProcessSwPackage checks if the Software Cluster to be removed has attribute installationBehavior set to cannotBeRemoved. If this is the case, UCM shall not remove it in accordance to [SWS_UCM_00245].

ProcessSwPackage checks if the Software Cluster version being updated is older than currently present in Machine in accordance to [SWS_UCM_00103] (kOldVersion).

[SWS_UCM_00150] Cancellation of a Software Package processing ⌈ProcessSwPackage method shall raise the ApplicationError kProcessSwPackageCancelled if the Cancel method has been called during the processing of a Software Package.⌋ *(RS_UCM_00024)*

7.5.2 Error handling for Cancel

[SWS_UCM_00278] Cancel error handling order ⌈Cancel method shall check the following error conditions and return the respective error code.

1. [SWS_UCM_00234]
2. [SWS_UCM_00235]

⌋ *(RS_UCM_00020)*

[SWS_UCM_00234] Cancel OperationNotPermitted ⌈Cancel shall raise the error ApplicationError kOperationNotPermitted in case the targeted Software Package processing has not yet started or has been already finished.⌋ *(RS_UCM_00020)*

[SWS_UCM_00235] Cancel InvalidTransferId ⌈Cancel shall raise the error ApplicationError kInvalidTransferId in case an invalid TransferId is sent by the client.⌋ *(RS_UCM_00020)*

7.5.3 Error handling for RevertProcessedSwPackages

[SWS_UCM_00279] RevertProcessedSwPackages error handling order ⌈RevertProcessedSwPackages method shall check the following error conditions and return the respective error code.

1. [SWS_UCM_00237]

2. [SWS_UCM_00236]

⌋(*RS_UCM_00020*)

[SWS_UCM_00237] `RevertProcessedSwPackages` **OperationNotPermitted** ⌈`RevertProcessedSwPackages` method call shall raise the error `ApplicationError kOperationNotPermitted` in case the processed `Software Package`s are successfully activated or it is called at other states than `kReady` (`Software Package`(s) are finished being processed) or `kProcessing` states.⌋(*RS_UCM_00020*)

[SWS_UCM_00236] `RevertProcessedSwPackages` **NotAbleToRevertPackages** ⌈`RevertProcessedSwPackages` shall raise the error `ApplicationError kNotAbleToRevertPackages` in case reverting of processed `Software Package`s have failed.⌋(*RS_UCM_00020*)

7.5.4 Error handling for GetSwProcessProgress

[SWS_UCM_00220] `GetSwProcessProgress` **InvalidTransferId** ⌈`GetSwProcessProgress` shall raise the error `ApplicationError kInvalidTransferId` in case an invalid TransferId is sent by the client.⌋(*RS_UCM_00023*)

7.6 Activation and Rollback

`UCM` should notify the activation or rollback of `Software Packages` to other `Functional Clusters` of the `AUTOSAR Adaptive Platform`. Vendor specific solution dictates to which modules this information is available, in which form and if this is done directly when change is done or when change is executed.

7.6.1 Activation

The `SoftwareCluster` state `kPresent` does not express whether a `SoftwareCluster` is currently executed or not. You can refer to chapter 7.1 Software Cluster Lifecycle for more details about `kPresent` state and sequence diagram 10.4 for more details about activation.

An activation of `SoftwareCluster`s is triggered by an `Activate` method call. At beginning of activation, `UCM` is asking `State Management` for an update session. Once granted, `UCM` is requesting `State Management` to stop running processes from the outdated `SoftwareCluster`s. When processes stopped, `UCM` makes available to the `AUTOSAR Adaptive Platform` the updated or installed `SoftwareCluster`s, the core action step of the activation. A verification of the activated `SoftwareCluster`s is then performed by requesting `State Management` changing the `SoftwareCluster`s `Function Groups` modes to `kVerify`. For an example of activation sequence, you can refer to chapter 10.4

[SWS_UCM_00293]{DRAFT} **`VerifyUpdate` method** ⌈At `kVerifying` state and before triggering to `kActivated` state, `UCM` shall call the `State Management` `UpdateRequest` Service Interface `VerifyUpdate` method passing the list of `Function Groups` defined in `SoftwareCluster` `claimedFunctionGroup` attribute of the class.⌋*(RS_UCM_00024)*

[SWS_UCM_00107] Activated state ⌈`UCM` state `kActivated` shall be set after the new versions of updated `SoftwareCluster`s have been verified.⌋*(RS_UCM_00008, RS_UCM_00030)*

The state management [3] on the level of execution is handled by the `UCM`'s client controlling the update process.

`UCM` has to be able to update several `SoftwareCluster`s for an update campaign. However, these `SoftwareCluster`s could have dependencies not satisfied if updates are processed and activated one by one. Therefore, `UCM` splits the activation action from the general package processing.

[SWS_UCM_00027]{OBSOLETE} **Delta Package version applicability** ⌈Applicable version of `SoftwareCluster` on which to apply delta shall be included into related `SoftwarePackage`'s `deltaPackageApplicableVersion` attribute.⌋*(RS_UCM_00007)*

Applicable version of a `SoftwareCluster` on which to apply delta is included into related `SoftwarePackage`'s `deltaPackageApplicableVersion` attribute

[SWS_UCM_00025]{OBSOLETE} **Activation of `SoftwareCluster`s** ⌈At the invocation of method `Activate`, `UCM` shall enable execution of any pending changes from the previously processed `Software Packages`.⌋*(RS_UCM_00021)*

Every call to `ProcessSwPackage` makes necessary preparations of possible actions on the `Software Cluster` (`ActionType` [SWS_UCM_00132]) : `kInstall`, `kRemove`, `kUpdate`. The `Activate` call finalises the started actions during processing and then `UCM` applies changes at activation that were still pending from processing, like for instance updating the list of processes managed by `Execution Management`.

After `Activate`, the new set of `SoftwareCluster`s can be started. Activation covers all the processed `Software Packages` for all the clients.

[SWS_UCM_00022] Activation of `Software Clusters` ⌈`UCM` shall activate all the `Software Clusters` extracted from the `Software Packages` when `Activate` is called.⌋*(RS_UCM_00021)*

The activation method could lead to a full system reset. When `Software Package` updates underlying OS, `AUTOSAR Adaptive Platform` or any `Adaptive Application` which is configured to be part of `Function Group` `MachineFG`, the execution of updated software occurs through system reset by calling `State Management` `UpdateRequest` Service Interface `ResetMachine` method. Meta-data of `Software Package` defines the activation method.

In principle, it is possible to activate multiple versions of the same SoftwareCluster in one activation step. This could be useful for example with delta package updates but does not apply to firmware updates. The specification does not prohibit to create this kind of chained updates. The decision to use chained updates should be based on safety aspects and the applicability of the underlying update technology, if the update is for a classic or an adaptive platform, if a file system is involved or if the used platform even support it.

7.6.1.1 Error handling for Activate

[SWS_UCM_00281] Activate error handling order ⌈Activate method shall check the following error conditions and return the respective error code.

1. [SWS_UCM_00241]
2. [SWS_UCM_00329]
3. [SWS_UCM_00026]
4. [SWS_UCM_00258]
5. [SWS_UCM_00242]
6. [SWS_UCM_00280]

⌋*(RS_UCM_00026)*

[SWS_UCM_00241] Activate OperationNotPermitted ⌈Activate shall raise the error ApplicationError kOperationNotPermitted in case the UCM state is not kReady.⌋*(RS_UCM_00021)*

[SWS_UCM_00026] Dependency Check ⌈During the UCM state kActivating, UCM shall perform a dependency check to ensure that all the Software Clusters having dependencies are not missing any necessary Software Cluster as defined by dependsOn and do not conflict towards each other as defined by conflictsTo, otherwise return ApplicationError kMissingDependencies.⌋*(RS_UCM_00007)*

If Activate method cannot establish an Update Session with State Management, it returns kUpdateSessionRejected, see [SWS_UCM_00258].

[SWS_UCM_00242] Activate PrepareUpdateFailed ⌈Activate shall raise the error ApplicationError kPrepareUpdateFailed in case of activation state transition failure from State Management side.⌋*(RS_SM_00001)*

[SWS_UCM_00280] Activate VerificationFailed ⌈Activate shall raise the error ApplicationError kVerificationFailed in case of verification failure returned by State Management.⌋*(RS_UCM_00021)*

7.6.2 Rollback

[SWS_UCM_00005]{OBSOLETE} **Rollback to the software prior to Finish the update process** ⌈UCM shall provide a method Rollback to recover from an activation that went wrong.⌋*(RS_UCM_00008)*

Rollback can be called in the case of A/B partitions or UCM uses some other solution to maintain backups of updated or removed Software Packages.

[SWS_UCM_00110] Rolling-back the software update ⌈At kRollingBack state, UCM shall disable the changes done by the software update by calling State Management UpdateRequest Service Interface PrepareRollback method for each Function Group of the processed Software Cluster in the update session. Then UCM shall call State Management UpdateRequest Service Interface ResetMachine method if any Software Cluster requires a machine reboot to be rolled back.⌋*(RS_UCM_00008)*

If a reset of the Machine is not necessary, an implementation specific way to inform Execution Management that a Software Cluster was updated can be performed.

[SWS_UCM_00299]{DRAFT} **Verify rolled back Software Clusters** ⌈After a UCM successful Rollback using call State Management UpdateRequest Service Interface PrepareRollback method and optional Machine reset or manifest reparse, UCM shall call State Management UpdateRequest Service Interface VerifyUpdate method to confirm that all Software Clusters impacted by update are still safe to be launched.⌋*(RS_UCM_00008)*

[SWS_UCM_00302]{DRAFT} **Rollback failing is triggering production error** ⌈When a Rollback is failing, UCM shall report UCM_FAILED_ROLLBACK production error.⌋*(RS_UCM_00045, RS_UCM_00008, RS_UCM_00027)*

7.6.2.1 Error handling for Rollback

[SWS_UCM_00282] Rollback error handling order ⌈Rollback method shall check the following error conditions and return the respective error code.

1. [SWS_UCM_00239]

⌋*(RS_UCM_00008)*

[SWS_UCM_00239] Rollback OperationNotPermitted ⌈Rollback shall raise the error ApplicationError kOperationNotPermitted in case UCM current state is not kActivated nor kVerifying.⌋*(RS_UCM_00020)*

7.6.3 Boot options

During update process the executed software is switched from original software to updated software and in case of rollback, from updated software to original version. Which version of software is executed is dependent on the UCM state and this is managed by the UCM. In case of platform and OS update the switch between software versions occurs through system reset and depending on the system design the Execution Management [2] might be started before UCM. In this case there can't be direct interface between UCM and Execution Management [2] to define which versions of software would be executed. Instead this would be controlled through persistent controls which are referred as Boot options in this document.

[SWS_UCM_00094] Management of executable software ⌈UCM shall manage which version of software is available for the Execution Management [2] to launch.⌋*(RS_UCM_00021)*

During the kActivating state, UCM modifies the Boot options so that in the next restart for the updated software the new versions will be executed. In the kRollingBack state, UCM modifies the Boot options so that in the next restart of the updated software the original versions will be executed.

7.6.4 Finishing activation

[SWS_UCM_00020] Finishing the packages activation ⌈UCM shall provide a method Finish to commit all the changes and clean up all temporary data of the processed Software Packages.⌋*(RS_UCM_00015)*

UCM should also remove Software Packages, logs or any older versions of changed software to save storage space. It is up to implementer to remove or not the Software Packages.

[SWS_UCM_00259] Ending the update session ⌈UCM shall call State Management UpdateRequest Service Interface StopUpdateSession method when UCM is exiting the kCleaningUp state.⌋*(RS_UCM_00021, RS_UCM_00018)*

[SWS_UCM_00240] Finish OperationNotPermitted ⌈Finish shall raise the error ApplicationError kOperationNotPermitted in case there are no activated nor rolled-back Software Packages pending finalization (i.e UCM state is not kActivated nor kRolledBack.⌋*(RS_UCM_00001, RS_UCM_00026)*

For UCM to be able to free all unneeded resources while processing the Finish request, it is up to the vendor and platform specific implementation to make sure that obsolete versions of changed SoftwareClusters aren't executed anymore.

7.7 Status Reporting

Once Software Packages are transferred to UCM, they are ready to be processed to finally apply changes to the AUTOSAR Adaptive Platform. In contrast to the transmission, the processing and activation tasks have to happen in a strict sequential order.

To give an overview of the update sequence, the global state of UCM is described in this section. The details of the processing and activation phases and the methods are specified in the 7.5 and 7.6.

The global state of UCM can be queried using the field CurrentStatus. The state machine for CurrentStatus is shown in Fig. 7.5. This diagram does not include behaviour after a reset. Examples can be found of how UCM and its CurrentStatus field behave including reset management in chapter 10 Sequence Diagram.

[SWS_UCM_00019] Status Field of Package Management ⌈The global state of UCM shall be provided using the field CurrentStatus⌋ *(RS_UCM_00024)*

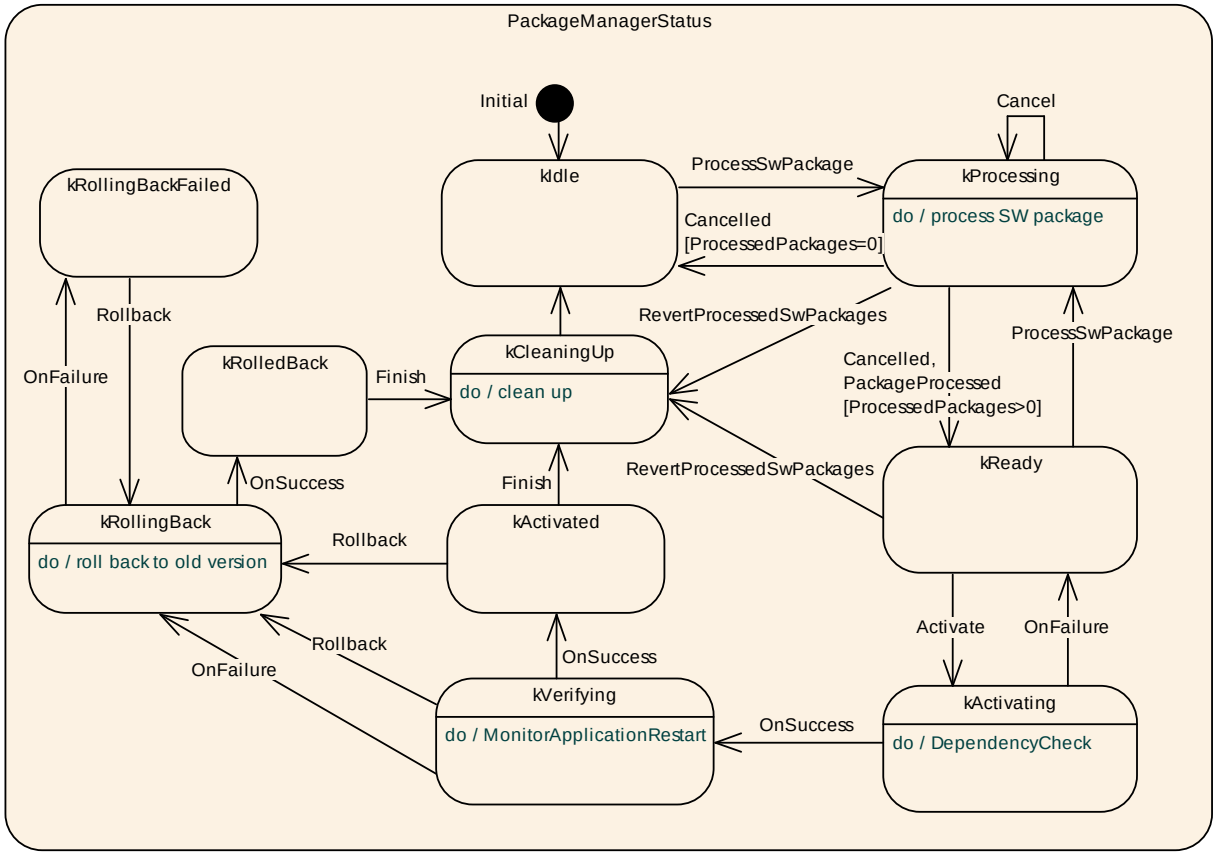

Figure 7.5: State Machine for the package processing using service interface: PackageManagement

UCM supported method calls for each value of field CurrentStatus are shown in Fig. 7.5.

[SWS_UCM_00080] Idle state of Package Management ⌈kIdle shall be the default state.⌋*(RS_UCM_00024)*

[SWS_UCM_00149] Return to the Idle state from Processing state ⌈kIdle state shall be set when ProcessSwPackage returns with error code kProcessSwPackageCancelled and if no other Software Packages were previously processed during this processing operation.⌋*(RS_UCM_00024)*

[SWS_UCM_00151] Entering the Ready state of Package Management after a Cancel call ⌈If ProcessSwPackage has been cancelled, UCM shall return error code kProcessSwPackageCancelled and set state to kReady only if at least one other Software Package was previously processed during this processing operation.⌋*(RS_UCM_00024)*

[SWS_UCM_00081] Processing state of Package Management ⌈kProcessing state shall be set only if ProcessSwPackage has been called. This shall only be possible, if CurrentStatus is reported as kIdle or kReady.⌋*(RS_UCM_00024)*

[SWS_UCM_00266] OperationNotPermitted error and UCM state ⌈UCM shall return ApplicationError kOperationNotPermitted if ProcessSwPackage is called by a client with UCM at CurrentStatus state different than kIdle, kProcessing or kReady.⌋*(RS_UCM_00001, RS_UCM_00004)*

[SWS_UCM_00083] Entering the Ready state of Package Management after a successful processing operation ⌈kReady state shall be set after a Software Package processing has been completed successfully.⌋*(RS_UCM_00024)*

[SWS_UCM_00265] state transition due to ProcessSwPackage error ⌈If ProcessSwPackage raises an ApplicationError other than kProcessSwPackageCancelled, it shall transition from kProcessing to kIdle if no other Software Packages were previously processed during this processing operation, or kReady if at least one other Software Package was previously processed before the failed processing operation, and shall perform clean-up actions.⌋*(RS_UCM_00015, RS_UCM_00026)*

Clean-up actions could be similar to the cancel call by for instance deleting files, folders or artefacts of the processed Software Cluster.

[SWS_UCM_00152] Entering the Ready state of Package Management after a missing dependency ⌈kReady state shall be set when Activate fails due to an ApplicationError kMissingDependencies.⌋*(RS_UCM_00024)*

[SWS_UCM_00084] Entering the kActivating state of Package Management ⌈kActivating shall be set when Activate is called. This triggers the dependency check and returns ApplicationError kMissingDependencies if this check fails.⌋*(RS_UCM_00024)*

[SWS_UCM_00153] Action in kActivating state of Package Management ⌈When kActivating is set and after the State Management UpdateRequest Service Interface RequestUpdateSession method call by UCM, the UCM shall call the State Management UpdateRequest Service Interface PrepareUpdate method for the

concerned Software Cluster including a list of all Function Groups belonging to that Software Cluster.⌋(*RS_UCM_00024*)

[SWS_UCM_00260]{DRAFT} **PrepareUpdate, VerifyUpdate and PrepareRollback orders** ⌈UCM shall compute the order of the State Management UpdateRequest Service Interface PrepareUpdate, VerifyUpdate and PrepareRollback method calls from the dependency model included in the Software Cluster manifests.⌋ (*RS_UCM_00007, RS_UCM_00021, RS_UCM_00030*)

[SWS_UCM_00261] PrepareUpdate, VerifyUpdate and PrepareRollback synchronous calls ⌈Calls to State Management UpdateRequest Service Interface PrepareUpdate, VerifyUpdate and PrepareRollback methods shall not be concurrent.⌋(*RS_UCM_00026*)

[SWS_UCM_00262]{DRAFT} **Update preparation rejected** ⌈If any call of the State Management UpdateRequest Service Interface PrepareUpdate method returns error kRejected too many times (maximumNumberOfRetries) or for too long (retryIntervalTime with role prepareUpdate), UCM shall transition from kActivating to kReady states.⌋(*RS_UCM_00026*)

[SWS_UCM_00263] Update preparation failure ⌈If any one of the State Management UpdateRequest Service Interface PrepareUpdate method returns error kFailed, UCM shall transition from kActivating to kReady states.⌋(*RS_UCM_00026*)

[SWS_UCM_00154] Entering the Verifying state of Package Management ⌈kVerifying shall be set when the dependency check have been performed successfully (all dependencies are satisfied) and that the preparation of the Software Clusters by the State Management has been successfully performed.⌋(*RS_UCM_00024*)

The machine could most likely be restarted in case a A/B partition is used. In case the A/B partition is not used, all affected Function Groups or the platform could be restarted. Immediately after the processed Software Package has been restarted, a system check has to be performed in order to make sure the machine is able to start up as expected. With this check it is verified that other safety relevant software like Functional Cluster Platform Health Management [5] is running and user can be protected from any issues caused by the update after the update has finished.

An update could most likely require to reparse the manifests after performing the atomic activation of the Software Clusters (switching A/B partition, changing symlinks, etc.) if a machine reset is not needed.

[SWS_UCM_00085] Entering the kActivated state of Package Management ⌈kActivated state shall be set when the VerifyUpdate method of State Management service interface UpdateRequest is returned successfully.⌋(*RS_UCM_00024*)

By a successful return of VerifyUpdate, UCM assumes all impacted Function Groups (the ones related to the processed Software Package) have been successfully restarted and verified.

kVerifying state gives the client controlling the update process a chance to perform verification test by calling State Management UpdateRequest Service Interface [SWS_SM_91017] VerifyUpdate method, though functionality in verify state can be limited. Client can also coordinate the results over several AUTOSAR Adaptive Platforms and still perform a Rollback if verification indicates the need for it.

If the system check is successful, the client can decide either to Rollback the current active processing so that the previous processed working software gets started, or to perform Finish so that the changes of processed software become permanent. By calling Finish a clean-up is initiated and in case of A/B partition, a swap between the partitions happens and the newly inactive partition becomes a copy of the newly active partition. In case Finish succeeds (including the clean-up), the current CurrentStatus changes to kIdle.

For Rollback the update software needs to be deactivated and possibly reactivated from original version, e.g. self-update of UCM. For this reason Rollback is also performed through two states, similarly as activation. Calling Rollback sets UCM into kRollingBack state where original software version is made executable and where original software is activated by the State Management. This is started by calling State Management UpdateRequest Service Interface [SWS_SM_91017] PrepareRollback method for each Software Cluster. On success, UCM goes to kRollingBack state. In this state all the changes introduced during update process have been deactivated and can be cleaned by calling Finish.

[SWS_UCM_00126] Entering the kRollingBack state after a Rollback call ⌈The state kRollingBack shall be set when Rollback is called.⌋*(RS_UCM_00008, RS_UCM_00030)*

[SWS_UCM_00155] Entering the kRolling-Back state after a failure in the kVerifying state ⌈The state kRollingBack shall be set if any of the State Management UpdateRequest Service Interface VerifyUpdate method calls returns the result kFailed.⌋*(RS_UCM_00008, RS_UCM_00030)*

[SWS_UCM_00264]{DRAFT} **Update verification rejected** ⌈If any call of the State Management UpdateRequest Service Interface VerifyUpdate returns error kRejected too many times (maximumNumberOfRetries) or for too long (retryIntervalTime with role prepareUpdate), UCM shall transition to kRollingBack state.⌋*(RS_UCM_00030, RS_UCM_00008)*

[SWS_UCM_00111] Entering the kRollingBack state ⌈The state kRollingBack shall be set when UCM calls State Management UpdateRequest Service Interface PrepareRollback method.⌋*(RS_UCM_00008, RS_UCM_00030)*

[SWS_UCM_00300]{DRAFT} **Software Cluster failing to rollback** ⌈If Rollback is failing, UCM CurrentStatus shall transition from kRollingBack to kRollingBackFailed.⌋*(RS_UCM_00024)*

[SWS_UCM_00301]{DRAFT} **Retry ro Rollback again when UCM is in kRollingBackFailed state** ⌈If `Rollback` method is called while being at `kRollingBackFailed`, `UCM CurrentStatus` shall transition from `kRollingBackFailed` to `kRollingBack`.⌋*(RS_UCM_00024)*

[SWS_UCM_00146] Entering the Cleaning-up state after a Finish call ⌈The state `kCleaningUp` shall be set when `Finish` is called and the `UCM` starts to perform cleanup actions.⌋*(RS_UCM_00008, RS_UCM_00030)*

[SWS_UCM_00162] Entering the Cleaning-up state after a `RevertProcessedSwPackages` call ⌈The state `kCleaningUp` shall be set when `RevertProcessedSwPackages` is called in `kProcessing` or `kReady` states and the `UCM` starts to perform cleanup actions.⌋*(RS_UCM_00008, RS_UCM_00030)*

[SWS_UCM_00163] Action in Cleaning-up state ⌈When `kCleaningUp` state is set, the `UCM` shall clean up all data of the processed packages that are not needed anymore.⌋*(RS_UCM_00008, RS_UCM_00030)*

[SWS_UCM_00164] Cleaning up of Software Packages ⌈In `kCleaningUp` state, the `UCM` may remove (from the UCM buffer for instance) the "physical" Software Package (e.g. zip file) that was used to transport the the SoftwareCluster to the `UCM`.⌋*(RS_UCM_00008, RS_UCM_00030)*

[SWS_UCM_00127] Finishing update sequence ⌈`kIdle` shall be set when `Finish` is called and the clean-up has been successfully performed. This finishes the update sequence and next sequence can be started.⌋*(RS_UCM_00008, RS_UCM_00030)*

[SWS_UCM_00147] Return to the Idle state from Cleaning-up state ⌈`kIdle` state shall be set when the Clean-up operation has been completed successfully.⌋*(RS_UCM_00024)*

7.8 Robustness against reset

Failure during over-the-air updates could lead into corrupted or inconsistent software configuration and further updates might be blocked. For this reason `UCM` needs to be robust against interruptions like power downs.

[SWS_UCM_00157] Detection of reset ⌈At start up `UCM` shall identify if uncontrolled reset occurred.⌋*(RS_UCM_00027)*

The way for `UCM` to detect uncontrolled reset is project specific. `UCM` could use hardware platform specific registers to detect Soft/Hard reset. Or it could access PHM Functional Cluster to detect uncontrolled reset. `UCM` could also check that the `CurrentStatus` persistent field is not `kIdle` or `kVerifying`.

[SWS_UCM_00158] Cleanup of interrupted actions ⌈After an uncontrolled reset, `UCM` shall check non volatile memory integrity, recover processed artifacts in case it is corrupted and resume interrupted actions in order to return the system into a state from where `UCM` can continue serving its Clients.⌋*(RS_UCM_00027)*

After an uncontrolled reset, it can be possible as an example for UCM to confirm consistency of any processed artifacts based on ArtifactChecksum class associated to SoftwareCluster. If checksum value of an artifact does not match, it can be deleted and processed again.

[SWS_UCM_00270] UCM internal state persistency ⌈UCM shall persist CurrentStatus state field to be able to resume on-going update after an intended or unintended reboot.⌋ *(RS_UCM_00027)*

7.8.1 Boot monitoring

Activation failure during OS and Platform-self updates can lead to a state in which the system is not able to reach a point where UCM and the client are able to function as expected and thus not able to execute the rollback. For these cases the system should include component which is responsible to monitor that the OS and platform will start up correctly. In case of failure, the Boot monitoring component should trigger a reset or modify the boot options to trigger a rollback.

7.9 History

[SWS_UCM_00115] History ⌈GetHistory method shall retrieve all actions that have been performed by UCM within a specific time window input parameter.⌋ *(RS_UCM_00032)*

In the case the UCM Client requests a rollback after a successful activation, CurrentStatus field transitioning to kActivated, GetHistory method will later return HistoryType, with subelement resolution of type ResultType equal to kActivatedAndRolledBack.

[SWS_UCM_00292] History elements ordering ⌈UCM shall return from GetHistory method a vector of HistoryType sorted in an increasing chronological order.⌋ *(RS_UCM_00032)*

[SWS_UCM_00160] Processing results records ⌈When UCM is entering kVerifying, UCM shall save activation time based on timeBaseResource and activation result of processed Software Packages in the history.⌋ *(RS_UCM_00032)*

[SWS_UCM_00271]{DRAFT} **Keeping history of failure error code** ⌈UCM shall keep in HistoryType subelement failureError the last failure error code as described in [SWS_UCM_00136]. If no error occurred, the stored value shall be 0.⌋ *(RS_UCM_00032)*

[SWS_UCM_00303]{DRAFT} **failing to record history** ⌈If UCM is failing to record a new entry in history, UCM shall report a production error: UCM_HISTORY_RECORD_FAILED. Any successful history update shall report a pass to this production error.⌋ *(RS_UCM_00045)*

7.10 Version Reporting

[SWS_UCM_00004]{OBSOLETE} **Report software information** ⌈UCM shall provide a method GetSwClusterInfo of the interface service PackageManagement to provide the identifiers and versions of the SoftwareClusters that are in state kPresent.⌋*(RS_UCM_00002)*

[SWS_UCM_00030] Report changes ⌈UCM shall provide a method GetSwClusterChangeInfo of the interface service PackageManagement to provide the identifiers and versions of the SoftwareCluster that are in state kAdded, kUpdating or kRemoved.⌋*(RS_UCM_00011)*

[SWS_UCM_00185]{OBSOLETE} **Provide SoftwareCluster general information** ⌈At the invocation of method GetSwClusterDescription, UCM shall return the version, type approval, license and release notes of the SoftwareCluster that are in state kPresent.⌋*(RS_UCM_00002, RS_UCM_00011)*

[SWS_UCM_00311]{DRAFT} **Provide SoftwareCluster general information** ⌈In case the SoftwareCluster referred by its name as input parameter to GetSwClusterManifestInfo is not in state kPresent, UCM shall raise ApplicationError kSoftwareClusterMissing.⌋*(RS_UCM_00002, RS_UCM_00011)*

7.11 Securing Software Updates

UCM provides service interface using ara::com. There is no authentication of the client in UCM's update sequence.

For authentication of the Software Package, you can refer to 7.3

[SWS_UCM_00103]{DRAFT} **Update to Software Cluster version which is not newer than currently present and than previously removed** ⌈If the version of a SoftwarePackage (returned by GetSwPackages) is less than or equal to the version of currently present SoftwareCluster (returned by GetSwClusterInfo), the UCM method TransferExit or TransferData shall raise the ApplicationError kOldVersion, report a production error UCM_OLD_VERSION_PACKAGE and delete the rejected Software Package.⌋*(RS_UCM_00031)*

[SWS_UCM_00190] Reinstallation of older Software Cluster version than previously removed ⌈New Software Clusters getting installed shall be compared with the history of all installed Software Clusters to prevent installation of a Software Cluster with a lower or equal version than previously installed.⌋*(RS_UCM_00003, RS_UCM_00031)*

[SWS_UCM_CONSTR_00002]{DRAFT} **UCM confidential information handling** ⌈The PackageManagement interface shall only be mapped via ara::com to a secure endpoint using secure communication channel providing confidentiality protection.⌋*(RS_UCM_00002, RS_UCM_00010, RS_UCM_00011)*

The GetSwClusterInfo, GetSwClusterChangeInfo, GetHistory and GetSwPackages methods are using data that could identify vehicle user and therefore should be protected for confidentiality.

[SWS_UCM_00202]{DRAFT} **Trusted Platform compliance** ⌈UCM shall ensure that after processing updates, all the necessary changes to comply with the Trusted Platform are applied.⌋*(RS_EM_00014)*

The authentication tag of the Trusted Platform corresponding to the updated/removed/added executable files should also be updated/removed/added. See also Chapter 7.10 of the Execution Management [2] for details on the Trusted Platform.

7.12 Functional cluster lifecycle

[SWS_UCM_00274]{DRAFT} **UCM initialization** ⌈UCM shall offer its services only after its internal initialization has been completed, and then report kRunning state to Execution Management.⌋*(RS_UCM_00044)*

This requirement prevents calling UCM subordinate API while internal initialization is on-going. The concrete initialization tasks are implementation specific.

7.12.1 Shutdown behaviour

There are no requirements of shutdown behaviour from UCM functional cluster.

8 API specification

There are no APIs defined in this release.

9 Service Interfaces

9.1 Type definitions

This chapter lists all types provided by the UCM.

The following figure is informative and only meant to support reader having global view of UCM types and service interface.

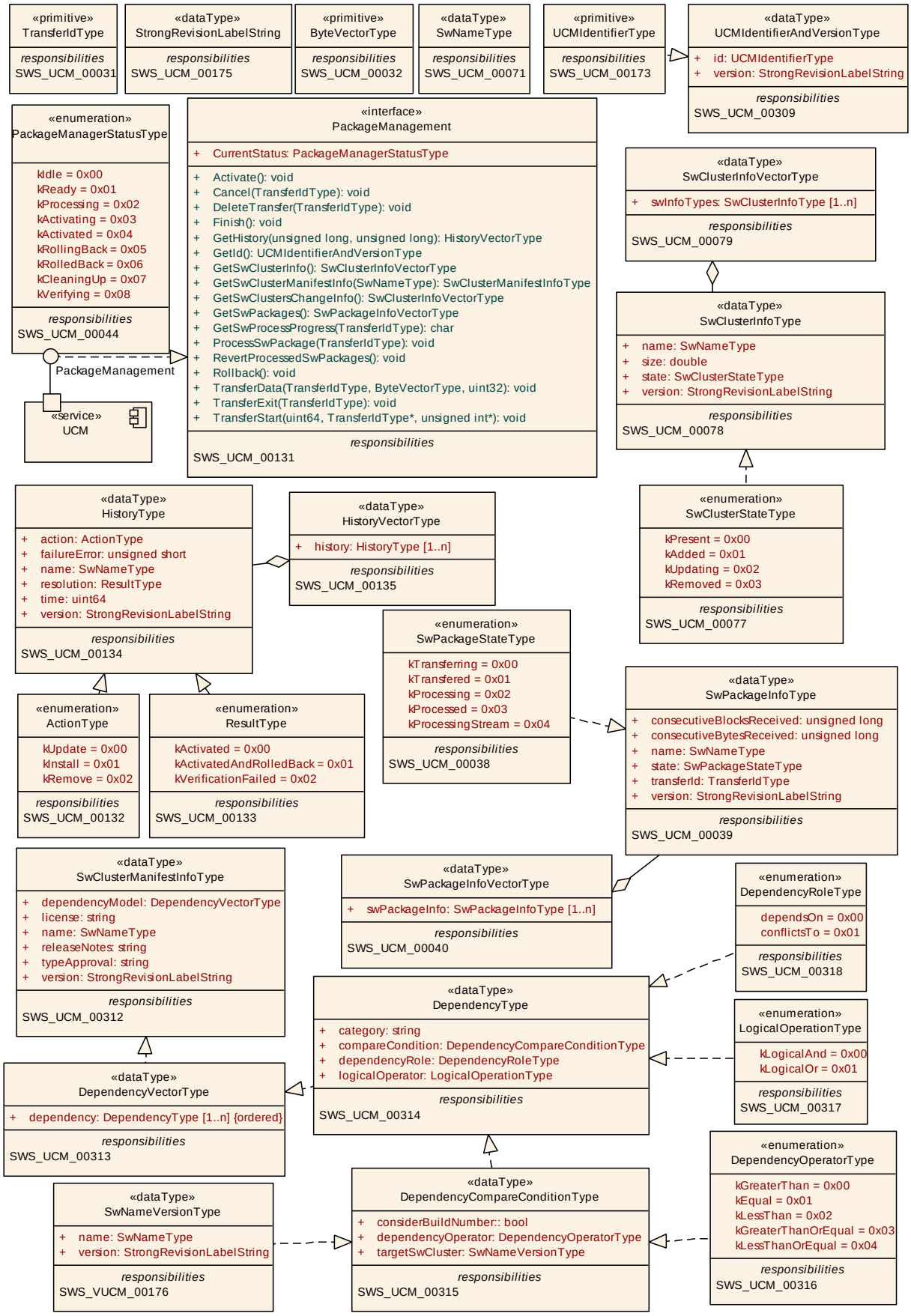

Figure 9.1: UCM composite structure

9.1.1 UCMIdentifierType

[SWS_UCM_00173]{DRAFT} **Definition of ImplementationDataType UCMIdentifier Type** ⌈

Name	UCMIdentifierType
Namespace	ara::ucm
Kind	STRING
Derived from	-
Description	UCM Module Instantiation Identifier.

⌋(*RS_VUCM_00036*)

9.1.2 UCMIdentifierAndVersionType

[SWS_UCM_00309]{DRAFT} **Definition of ImplementationDataType UCMIdentifier AndVersionType** ⌈

Name	UCMIdentifierAndVersionType
Namespace	ara::ucm
Kind	STRUCTURE
Sub-elements	id `UCMIdentifierType` version `StrongRevisionLabelString`
Derived from	-
Description	Represents UCM Module Instantion number and version of UCM.

⌋(*RS_VUCM_00035*)

9.1.3 TransferIdType

[SWS_UCM_00031]{DRAFT} **Definition of ImplementationDataType TransferId Type** ⌈

Name	TransferIdType
Namespace	ara::ucm
Kind	ARRAY `<uint8_t>`
Array size	16
Derived from	-
Description	Represents a handle identifier used to reference a particular transfer request.

⌋(*RS_UCM_00019*, *RS_UCM_00025*)

9.1.4 SwNameType

[SWS_UCM_00071]{DRAFT} **Definition of ImplementationDataType SwNameType** ⌈

Name	SwNameType
Namespace	ara::ucm
Kind	STRING
Derived from	-
Description	SoftwareCluster or SoftwarePackage shortName attribute inherited from referrable meta Class.

⌋(*RS_UCM_00002*)

9.1.5 StrongRevisionLabelString

[SWS_UCM_00175] Definition of ImplementationDataType StrongRevisionLabel String ⌈

Name	StrongRevisionLabelString
Namespace	ara::ucm
Kind	STRING
Derived from	-
Description	Primitive type representing SoftwareCluster (SoftwarePackage) version.

⌋(*RS_UCM_00002*)

9.1.6 SwNameVersionType

[SWS_UCM_00176]{DRAFT} **Definition of ImplementationDataType SwNameVersionType** ⌈

Name	SwNameVersionType
Namespace	ara::ucm
Kind	STRUCTURE
Sub-elements	name `SwNameType` version `StrongRevisionLabelString`
Derived from	-
Description	Represents the information of a Software Package (Software Cluster) name and version.

⌋(*RS_UCM_00002*)

9.1.7 ByteVectorType

[SWS_UCM_00032]{DRAFT} **Definition of ImplementationDataType ByteVector Type** ⌈

Name	ByteVectorType
Namespace	ara::ucm
Kind	VECTOR <uint8_t>
Derived from	-
Description	Byte vector representing raw data.

⌋(*RS_UCM_00025*)

9.1.8 SwPackageStateType

[SWS_UCM_00038] Definition of ImplementationDataType SwPackageStateType ⌈

Name	SwPackageStateType	
Namespace	ara::ucm	
Kind	TYPE_REFERENCE	
Derived from	uint8_t	
Description	Represents the state of a Software Package on the Platform.	
Range / Symbol	***Limit***	***Description***
kTransferring	0x00	Software package is being transferred, i.e. not completely received.
kTransferred	0x01	Software package is completely transferred and ready to be processed.
kProcessing	0x02	Software package is currently being processed.
kProcessed	0x03	Software package processing finished.
kProcessingStream	0x04	Software package is being processed from a stream.

⌋(*RS_UCM_00006*, *RS_UCM_00010*, *RS_UCM_00011*, *RS_UCM_00012*)

9.1.9 SwPackageInfoType

[SWS_UCM_00039] Definition of ImplementationDataType SwPackageInfoType ⌈

Name	SwPackageInfoType
Namespace	ara::ucm
Kind	STRUCTURE

▽

△

Sub-elements	name `SwNameType` version `StrongRevisionLabelString` transferId `TransferIdType` consecutiveBytesReceived `uint64_t` consecutiveBlocksReceived `uint64_t` state `SwPackageStateType`
Derived from	-
Description	Represents the information of a Software Package.

⌋(*RS_UCM_00002*, *RS_UCM_00006*, *RS_UCM_00010*, *RS_UCM_00011*, *RS_UCM_00012*)

9.1.10 SwPackageInfoVectorType

[SWS_UCM_00040]{DRAFT} **Definition of ImplementationDataType SwPackageInfoVectorType** ⌈

Name	SwPackageInfoVectorType
Namespace	ara::ucm
Kind	VECTOR <`SwPackageInfoType`>
Derived from	-
Description	Represents a dynamic size array of Software Packages

⌋(*RS_UCM_00002*, *RS_UCM_00006*, *RS_UCM_00010*, *RS_UCM_00011*, *RS_UCM_00012*)

9.1.11 SwClusterStateType

[SWS_UCM_00077] Definition of ImplementationDataType SwClusterStateType ⌈

Name	SwClusterStateType	
Namespace	ara::ucm	
Kind	TYPE_REFERENCE	
Derived from	`uint8_t`	
Description	Represents the state of a SoftwareCluster on the adaptive platform.	
Range / Symbol	***Limit***	***Description***
kPresent	0x00	State of a SoftwareCluster that is installed on the adaptive platform and installation has finished.
kAdded	0x01	State of a SoftwareCluster that has been newly installed.
kUpdating	0x02	State of a SoftwareCluster that has been updated.
kRemoved	0x03	State of a SoftwareCluster that is being updated.

⌋(*RS_UCM_00002*, *RS_UCM_00006*, *RS_UCM_00010*, *RS_UCM_00011*, *RS_UCM_00012*)

9.1.12 SwClusterInfoType

[SWS_UCM_00078]{DRAFT} **Definition of ImplementationDataType SwClusterInfoType** ⌈

Name	SwClusterInfoType
Namespace	ara::ucm
Kind	STRUCTURE
Sub-elements	name `SwNameType` version `StrongRevisionLabelString` state `SwClusterStateType` size `uint64_t`
Derived from	-
Description	Represents the information of a SoftwareCluster.

⌋(*RS_UCM_00002*, *RS_UCM_00011*)

9.1.13 SwClusterInfoVectorType

[SWS_UCM_00079]{DRAFT} **Definition of ImplementationDataType SwClusterInfoVectorType** ⌈

Name	SwClusterInfoVectorType
Namespace	ara::ucm
Kind	VECTOR <`SwClusterInfoType`>
Derived from	-
Description	Represents a dynamic size array of SoftwareClusters

⌋(*RS_UCM_00002*, *RS_UCM_00006*, *RS_UCM_00010*, *RS_UCM_00011*, *RS_UCM_00012*)

9.1.14 PackageManagementStatusType

[SWS_UCM_00044]{DRAFT} **Definition of ImplementationDataType PackageManagementStatusType** ⌈

Name	PackageManagementStatusType	
Namespace	ara::ucm	
Kind	TYPE_REFERENCE	
Derived from	`uint8_t`	
Description	Represents the state of UCM.	
Range / Symbol	***Limit***	***Description***
kIdle	0x00	UCM is ready to start processing if software packages are present.

▽

△

kReady	0x01	UCM has processed one or several packages and waits for additional packages, activation or reversion of processed packages.
kProcessing	0x02	UCM is currently in the middle of processing a Software Package, i.e. a client has called ProcessSwPackage.
kActivating	0x03	UCM is performing the dependency check and preparing the activation of the processed Software packages.
kActivated	0x04	Software changes introduced with processed Software Packages has been activated and executed.
kRollingBack	0x05	UCM is reverting changes introduced with processed packages.
kRolledBack	0x06	Software changes introduced with processed Software Packages has been deactivated and original software is executed.
kCleaningUp	0x07	Making sure that the system is in a clean state.
kVerifying	0x08	UCM (via State Management) is checking that the processed packages have been properly restarted.
kRollingBackFailed	0x09	UCM failed to revert changes introduced with processed packages.

⌋(*RS_UCM_00024*, *RS_UCM_00026*)

9.1.15 ActionType

[SWS_UCM_00132] Definition of ImplementationDataType ActionType ⌈

Name	ActionType	
Namespace	ara::ucm	
Kind	TYPE_REFERENCE	
Derived from	`uint8_t`	
Description	Represents the UCM action.	
Range / Symbol	***Limit***	***Description***
kUpdate	0x00	Update of a SoftwareCluster.
kInstall	0x01	Installation of a new SoftwareCluster.
kRemove	0x02	Removal of a SoftwareCluster.

⌋(*RS_UCM_00032*)

9.1.16 ResultType

[SWS_UCM_00133] Definition of ImplementationDataType ResultType ⌈

Name	ResultType	
Namespace	ara::ucm	
Kind	TYPE_REFERENCE	
Derived from	`uint8_t`	
Description	Represents the result of UCM action.	
Range / Symbol	***Limit***	***Description***
kActivated	0x00	Activation was successful.

▽

△

kActivatedAndRolledBack	0x01	UCM was activated but rolled back by its Client.
kVerificationFailed	0x02	UCM's action failed.

⌋(*RS_UCM_00032*)

9.1.17 HistoryType

[SWS_UCM_00134]{DRAFT} **Definition of ImplementationDataType HistoryType** ⌈

Name	HistoryType
Namespace	ara::ucm
Kind	STRUCTURE
Sub-elements	time `uint64_t` name `SwNameType` version `StrongRevisionLabelString` action `ActionType` resolution `ResultType` failureError `uint64_t`
Derived from	-
Description	Time refers to the verification time of the software cluster (when UCM Subordinate enters kVerifying). UCM shall get time from Time Sync Functional Cluster via UcmToTimeBase ResourceMapping.timeBaseResource

⌋(*RS_UCM_00032*)

9.1.18 HistoryVectorType

[SWS_UCM_00135]{DRAFT} **Definition of ImplementationDataType HistoryVector Type** ⌈

Name	HistoryVectorType
Namespace	ara::ucm
Kind	VECTOR <`HistoryType`>
Derived from	-
Description	Represents a list of UCM actions

⌋(*RS_UCM_00032*)

9.1.19 SwClusterManifestInfoType

[SWS_UCM_00312]{DRAFT} **Definition of ImplementationDataType SwClusterManifestInfoType** ⌈

Name	SwClusterManifestInfoType
Namespace	ara::ucm
Kind	STRUCTURE
Sub-elements	name `SwNameType` version `StrongRevisionLabelString` typeApproval `StringType` license `StringType` releaseNotes `StringType` dependencyModel `DependencyVectorType`
Derived from	-
Description	Represents the manifest information of a Software Cluster.

⌋(*RS_UCM_00002*, *RS_UCM_00011*)

9.1.20 DependencyType

[SWS_UCM_00314]{DRAFT} **Definition of ImplementationDataType Dependency Type** ⌈

Name	DependencyType
Namespace	ara::ucm
Kind	STRUCTURE
Sub-elements	dependencyRole `DependencyRoleType` category `StringType` logicalOperator `LogicalOperationType` compareCondition `DependencyCompareConditionType`
Derived from	-
Description	Represents dependencies between Software Clusters

⌋(*RS_UCM_00007*, *RS_VUCM_00037*)

9.1.21 DependencyVectorType

[SWS_UCM_00313]{DRAFT} **Definition of ImplementationDataType Dependency VectorType** ⌈

Name	DependencyVectorType
Namespace	ara::ucm
Kind	VECTOR <`DependencyType`>

▽

△

Derived from	-
Description	Represents a vector of dependencies between Software Clusters

⌋(*RS_UCM_00007*, *RS_VUCM_00037*)

9.1.22 DependencyRoleType

[SWS_UCM_00318]{DRAFT} **Definition of ImplementationDataType Dependency RoleType** ⌈

Name	DependencyRoleType	
Namespace	ara::ucm	
Kind	TYPE_REFERENCE	
Derived from	`uint8_t`	
Description	Represents the logical operation to be applied between dependencies of Software Clusters	
Range / Symbol	***Limit***	***Description***
kDependOn	0x00	depends of
kConflictsTo	0x01	conflicts to

⌋(*RS_UCM_00007*, *RS_VUCM_00037*)

9.1.23 LogicalOperationType

[SWS_UCM_00317]{DRAFT} **Definition of ImplementationDataType LogicalOperationType** ⌈

Name	LogicalOperationType	
Namespace	ara::ucm	
Kind	TYPE_REFERENCE	
Derived from	`uint8_t`	
Description	Represents the logical operation to be applied between dependencies of Software Clusters	
Range / Symbol	***Limit***	***Description***
kLogicalAnd	0x00	Logical AND
kLogicalOr	0x01	Logical OR

⌋(*RS_UCM_00007*, *RS_VUCM_00037*)

9.1.24 DependencyCompareConditionType

[SWS_UCM_00315]{DRAFT} **Definition of ImplementationDataType DependencyCompareConditionType** ⌈

Name	DependencyCompareConditionType
Namespace	ara::ucm
Kind	STRUCTURE
Sub-elements	targetSwCluster `SwNameVersionType` dependencyOperator `DependencyOperatorType` considerBuildNumber `bool`
Derived from	-
Description	operator to be applied to target Software Cluster's version

⌋(*RS_UCM_00007*, *RS_VUCM_00037*)

9.1.25 DependencyOperatorType

[SWS_UCM_00316]{DRAFT} **Definition of ImplementationDataType DependencyOperatorType** ⌈

Name	DependencyOperatorType	
Namespace	ara::ucm	
Kind	TYPE_REFERENCE	
Derived from	`uint8_t`	
Description	Represents the dependency operator to be applied between versions of Software Cluster	
Range / Symbol	***Limit***	***Description***
kGreaterThan	0x00	Greater than
kEqual	0x01	Equal
kLessThan	0x02	Less than
kGreaterThanOrEqual	0x03	Greater than or equal
kLessThanOrEqual	0x04	Less than or equal

⌋(*RS_UCM_00007*, *RS_VUCM_00037*)

9.2 Provided Service Interfaces

9.2.1 Package Management

This chapter lists all provided service interfaces of the `UCM`.

Port

[SWS_UCM_00073]{DRAFT} **Definition of Port PackageManagement provided by functional cluster UCM** ⌈

Name	PackageManagement		
Kind	ProvidedPort	***Interface***	PackageManagement
Description	Provide services like receiving, processing and activating Software Packages. Providing history, update status and Software Package and Software Cluster information.		
Variation			

⌋*(RS_UCM_00001, RS_UCM_00003, RS_UCM_00004)*

Service Interface

[SWS_UCM_00131]{DRAFT} **Definition of ServiceInterface PackageManagement** ⌈

Name	PackageManagement
Namespace	ara::ucm

Field	CurrentStatus
Description	The current status of UCM.
Type	PackageManagementStatusType
HasGetter	true
HasNotifier	true
HasSetter	false

Method	Activate	
Description	This method activates the Software Clusters extracted from the processed Software Packages.	
FireAndForget	false	
Application Errors	kOperationNotPermitted	The operation is not supported in the current context.
Application Errors	kMissingDependencies	Activation is not allowed because dependencies are missing.
Application Errors	kPersistencyAllocationFailed	UCM failed to allocate persistent data.
Application Errors	kUpdateSessionRejected	Start of an update session was rejected by State Management
Application Errors	kPrepareUpdateFailed	Error during update preparation step.
Application Errors	kVerificationFailed	State Management returned verification failure

Method	Cancel	
Description	This method aborts an ongoing processing of a Software Package.	
FireAndForget	false	
Parameter	id	
	Description	The Transfer ID.
	Type	TransferIdType
	Variation	
	Direction	IN

▽

△

Application Errors	kOperationNotPermitted	The operation is not supported in the current context.
Application Errors	kInvalidTransferId	The Transfer ID is invalid.

Method	DeleteTransfer	
Description	Delete a transferred Software Package.	
FireAndForget	false	
Parameter	id	
	Description	Transfer ID of the currently running request.
	Type	TransferIdType
	Variation	
	Direction	IN
Application Errors	kInvalidTransferId	The Transfer ID is invalid.
Application Errors	kOperationNotPermitted	The operation is not supported in the current context.

Method	Finish	
Description	This method finishes the processing for the current set of processed Software Packages. It does a cleanup of all data of the processing including the sources of the Software Packages.	
FireAndForget	false	
Application Errors	kOperationNotPermitted	The operation is not supported in the current context.

Method	GetHistory	
Description	Getter method to retrieve all actions that have been performed by UCM.	
FireAndForget	false	
Parameter	timestampGE	
	Description	Earliest timestamp (inclusive)
	Type	uint64_t
	Variation	
	Direction	IN
Parameter	timestampLT	
	Description	Latest timestamp (exclusive)
	Type	uint64_t
	Variation	
	Direction	IN
Parameter	history	
	Description	The history of all actions that have been performed by UCM.
	Type	HistoryVectorType
	Variation	
	Direction	OUT

Method	GetId	
Description	Get the UCM Instance Identifier.	
FireAndForget	false	
Parameter	id	
	Description	UCM Module Instantiation Identifier.
	Type	UCMIdentifierType
	Variation	
	Direction	OUT

Method	GetSwClusterChangeInfo	
Description	This method returns a list pending changes to the set of SoftwareClusters on the adaptive platform. The returned list includes all SoftwareClusters that are to be added, updated or removed. The list of changes is extended in the course of processing Software Packages.	
FireAndForget	false	
Parameter	swInfo	
	Description	List of SoftwareClusters that are in state kAdded,kUpdating or kRemoved.
	Type	SwClusterInfoVectorType
	Variation	
	Direction	OUT

Method	GetSwClusterInfo	
Description	This method returns a list with information of all SoftwareClusters that are in state kPresent.	
FireAndForget	false	
Parameter	swInfo	
	Description	List of installed SoftwareClusters that are in state kPresent.
	Type	SwClusterInfoVectorType
	Variation	
	Direction	OUT

Method	GetSwClusterManifestInfo	
Description	This method returns the general information of a Software Cluster that are in state kPresent.	
FireAndForget	false	
Parameter	swName	
	Description	Name of the Software Cluster that is in state kPresent.
	Type	SwNameType
	Variation	
	Direction	IN
Parameter	swInfo	
	Description	Manifest information of Software Cluster at state kPresent.
	Type	SwClusterManifestInfoType
	Variation	
	Direction	OUT

Method	GetSwPackages
Description	This method returns the Software Packages that available in UCM.
FireAndForget	false
Parameter	packages

▽

△

	Description	List of Software Packages.
	Type	`SwPackageInfoVectorType`
	Variation	
	Direction	OUT

Method	GetSwProcessProgress	
Description	Get the progress (0 - 100%) of the currently processed Software Package.	
FireAndForget	false	
Parameter	id	
	Description	The Transfer ID of the Software Package.
	Type	`TransferIdType`
	Variation	
	Direction	IN
Parameter	progress	
	Description	The progress of the current package processing (0% - 100%). 0x00 ... 0x64, 0xFF for "No information available"
	Type	`uint8_t`
	Variation	
	Direction	OUT
Application Errors	`kInvalid-TransferId`	The Transfer ID is invalid.

Method	ProcessSwPackage	
Description	Process a previously transferred Software Package or a partly transferred Software Package from a stream.	
FireAndForget	false	
Parameter	id	
	Description	The Transfer ID of the Software Package which should be processed.
	Type	`TransferIdType`
	Variation	
	Direction	IN
Application Errors	`kOperationNotPermitted`	The operation is not supported in the current context.
Application Errors	`kServiceBusy`	Another processing is already ongoing and therefore the current processing request has to be rejected.
Application Errors	`kInvalid-TransferId`	The Transfer ID is invalid.
Application Errors	`kAuthenticationFailed`	Package authentication failed.
Application Errors	`kIncompatiblePackageVersion`	The version of the Software or Vehicle Package to be processed is not compatible with the current version of UCM or UCM Master.
Application Errors	`kInvalid-PackageManifest`	Package manifest could not be read.
Application Errors	`kSoftwareCluster-Missing`	The Software Cluster is not present in the Machine.
Application Errors	`kIncompatibleDelta`	Delta package dependency check failed.

▽

△

Application Errors	kInsufficientMemory	Insufficient memory to perform operation.
Application Errors	kInvalidChecksumDescription	Checksum attribute not recognised.
Application Errors	kProcessedSoftwarePackageInconsistent	The processed Software Package integrity check has failed.
Application Errors	kSwclRemovalDenied	Attempt to remove PLATFORM_CORE Software Cluster.
Application Errors	kOldVersion	Software Package version is too old.
Application Errors	kProcessSwPackageCancelled	The processing operation has been interrupted by a Cancel() call.

Method	RevertProcessedSwPackages	
Description	Revert the changes done by processing (ProcessSwPackage) of one or several software packages.	
FireAndForget	false	
Application Errors	kOperationNotPermitted	The operation is not supported in the current context.
Application Errors	kNotAbleToRevertPackages	RevertProcessedSwPackages failed.

Method	Rollback	
Description	Rollback the system to the state before the packages were processed.	
FireAndForget	false	
Application Errors	kOperationNotPermitted	The operation is not supported in the current context.

Method	TransferData	
Description	Block-wise transfer of a Software Package to UCM.	
FireAndForget	false	
Parameter	id	
	Description	Transfer ID.
	Type	TransferIdType
	Variation	
	Direction	IN
Parameter	data	
	Description	Data block of the Software Package.
	Type	ByteVectorType
	Variation	
	Direction	IN
Parameter	blockCounter	
	Description	Block counter value of the current block.
	Type	uint64_t
	Variation	

▽

△

	Direction	IN
Application Errors	kOperationNotPermitted	The operation is not supported in the current context.
Application Errors	kInvalidTransferId	The Transfer ID is invalid.
Application Errors	kIncorrectBlock	The same block number is received twice.
Application Errors	kIncorrectBlockSize	The size of the block exceeds the provided block size from TransferStart or TransferVehiclePackage.
Application Errors	kIncorrectSize	The size of the Software or Vehicle Package exceeds the provided size in TransferStart.
Application Errors	kInsufficientMemory	Insufficient memory to perform operation.
Application Errors	kTransferFailed	UCM cannot persist transferred block.
Application Errors	kBlockInconsistent	Consistency check for transferred block failed.
Application Errors	kUnsupportedPackageFormat	The Vehicle Package or Software Package archiving format is not supported.
Application Errors	kAuthenticationFailed	Package authentication failed.
Application Errors	kInvalidPackageManifest	Package manifest could not be read.
Application Errors	kIncompatiblePackageVersion	The version of the Software or Vehicle Package to be processed is not compatible with the current version of UCM or UCM Master.
Application Errors	kPackageInconsistent	Package integrity check failed.
Application Errors	kSwclRemovalDenied	Attempt to remove PLATFORM_CORE Software Cluster.
Application Errors	kOldVersion	Software Package version is too old.

Method	TransferExit	
Description	Finish the transfer of a Software Package to UCM.	
FireAndForget	false	
Parameter	id	
	Description	Transfer ID of the currently running request.
	Type	TransferIdType
	Variation	
	Direction	IN
Application Errors	kOperationNotPermitted	The operation is not supported in the current context.
Application Errors	kInvalidTransferId	The Transfer ID is invalid.
Application Errors	kInsufficientData	TransferExit has been called but total transferred data size does not match expected data size provided with TransferStart call.
Application Errors	kAuthenticationFailed	Package authentication failed.

▽

△

Application Errors	kUnsupportedPackageFormat	The Vehicle Package or Software Package archiving format is not supported.
Application Errors	kPackageInconsistent	Package integrity check failed.
Application Errors	kIncompatiblePackageVersion	The version of the Software or Vehicle Package to be processed is not compatible with the current version of UCM or UCM Master.
Application Errors	kInvalidPackageManifest	Package manifest could not be read.
Application Errors	kSwclRemovalDenied	Attempt to remove PLATFORM_CORE Software Cluster.
Application Errors	kOldVersion	Software Package version is too old.

Method	TransferStart	
Description	Start the transfer of a Software Package after having received a Vehicle Package. The size of the Software Package to be transferred to UCM must be provided. UCM will generate a Transfer ID for subsequent calls to TransferData, TransferExit, ProcessSwPackage, DeleteTransfer.	
FireAndForget	false	
Parameter	size	
	Description	Size (in bytes) of the Software Package to be transferred.
	Type	uint64_t
	Variation	
	Direction	IN
Parameter	id	
	Description	Return TransferId.
	Type	TransferIdType
	Variation	
	Direction	OUT
Parameter	blockSize	
	Description	Size of the blocks to be received with TransferData method.
	Type	uint32_t
	Variation	
	Direction	OUT
Application Errors	kInsufficientMemory	Insufficient memory to perform operation.

⌋(*RS_UCM_00001*, *RS_UCM_00002*, *RS_UCM_00008*, *RS_UCM_00010*, *RS_UCM_00011*, *RS_UCM_00015*, *RS_UCM_00018*, *RS_UCM_00021*, *RS_UCM_00023*, *RS_UCM_00024*, *RS_UCM_00025*, *RS_UCM_00032*)

9.3 Required Interface

9.3.1 State Management Update Request

UCM requires the UpdateRequest Service Interface [SWS_SM_91017] provided by State Management

Port

[SWS_UCM_00288]{DRAFT} **Definition of Port UpdateRequest required by functional cluster UCM** ⌈

Name	UpdateRequest		
Kind	RequiredPort	***Interface***	`UpdateRequest`
Description	The UpdateRequest interface is intended to be used by UCM to interact with StateManagement to perform updates, installation and removal of SoftwareClusters.		
Variation			

⌋()

9.4 Application Errors

9.4.1 Application Error Domain

9.4.1.1 UCMErrorDomain

This section lists all application errors of the UCM.

[SWS_UCM_00136]{DRAFT} **Definition of Application Error Domain of functional cluster UCM** ⌈

Name	***Code***	***Description***
kAuthenticationFailed	8	Package authentication failed.
kBlockInconsistent	25	Consistency check for transferred block failed.
kIncompatibleDelta	29	Delta package dependency check failed.
kIncompatiblePackageVersion	24	The version of the Software or Vehicle Package to be processed is not compatible with the current version of UCM or UCM Master.
kIncorrectBlock	2	The same block number is received twice.
kIncorrectBlockSize	30	The size of the block exceeds the provided block size from Transfer Start or TransferVehiclePackage.
kIncorrectSize	3	The size of the Software or Vehicle Package exceeds the provided size in TransferStart.
kInsufficientData	6	TransferExit has been called but total transferred data size does not match expected data size provided with TransferStart call.
kInsufficientMemory	1	Insufficient memory to perform operation.
kInvalidChecksumDescription	35	Checksum attribute not recognised.
kInvalidPackageManifest	13	Package manifest could not be read.
kInvalidTransferId	4	The Transfer ID is invalid.
kMissingDependencies	21	Activation is not allowed because dependencies are missing.
kNotAbleToRevertPackages	15	RevertProcessedSwPackages failed.
kOldVersion	9	Software Package version is too old.
kOperationNotPermitted	5	The operation is not supported in the current context.
kPackageInconsistent	7	Package integrity check failed.
kPersistencyAllocationFailed	41	UCM failed to allocate persistent data.

▽

△

kPrepareUpdateFailed	19	Error during update preparation step.
kProcessSwPackageCancelled	22	The processing operation has been interrupted by a Cancel() call.
kProcessedSoftwarePackageInconsistent	23	The processed Software Package integrity check has failed.
kServiceBusy	12	Another processing is already ongoing and therefore the current processing request has to be rejected.
kSoftwareClusterMissing	37	The Software Cluster is not present in the Machine.
kSwclRemovalDenied	39	Attempt to remove PLATFORM_CORE Software Cluster.
kTransferFailed	38	UCM cannot persist transferred block.
kUnexpectedPackage	32	The Software Package name does not correspond to the RequestedPackage field value.
kUnsupportedPackageFormat	40	The Vehicle Package or Software Package archiving format is not supported.
kUpdateSessionRejected	33	Start of an update session was rejected by State Management
kVerificationFailed	36	State Management returned verification failure

⌋(*RS_UCM_00006*, *RS_UCM_00007*, *RS_UCM_00012*, *RS_UCM_00013*, *RS_UCM_00014*)

10 Sequence diagrams

The following sequence charts are simplified examples and have no normative meaning. The relevant definitions are in chapter 7 only.

10.1 Update process

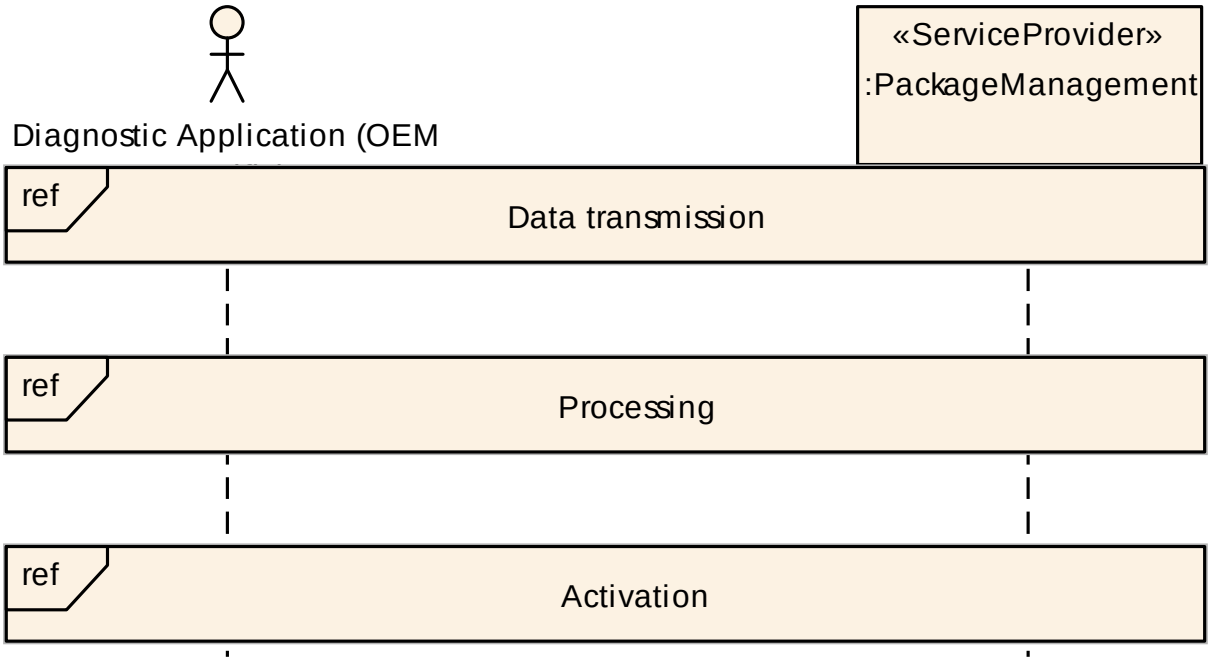

Figure 10.1: Sequence diagram showing the update process

10.2 Data transmission

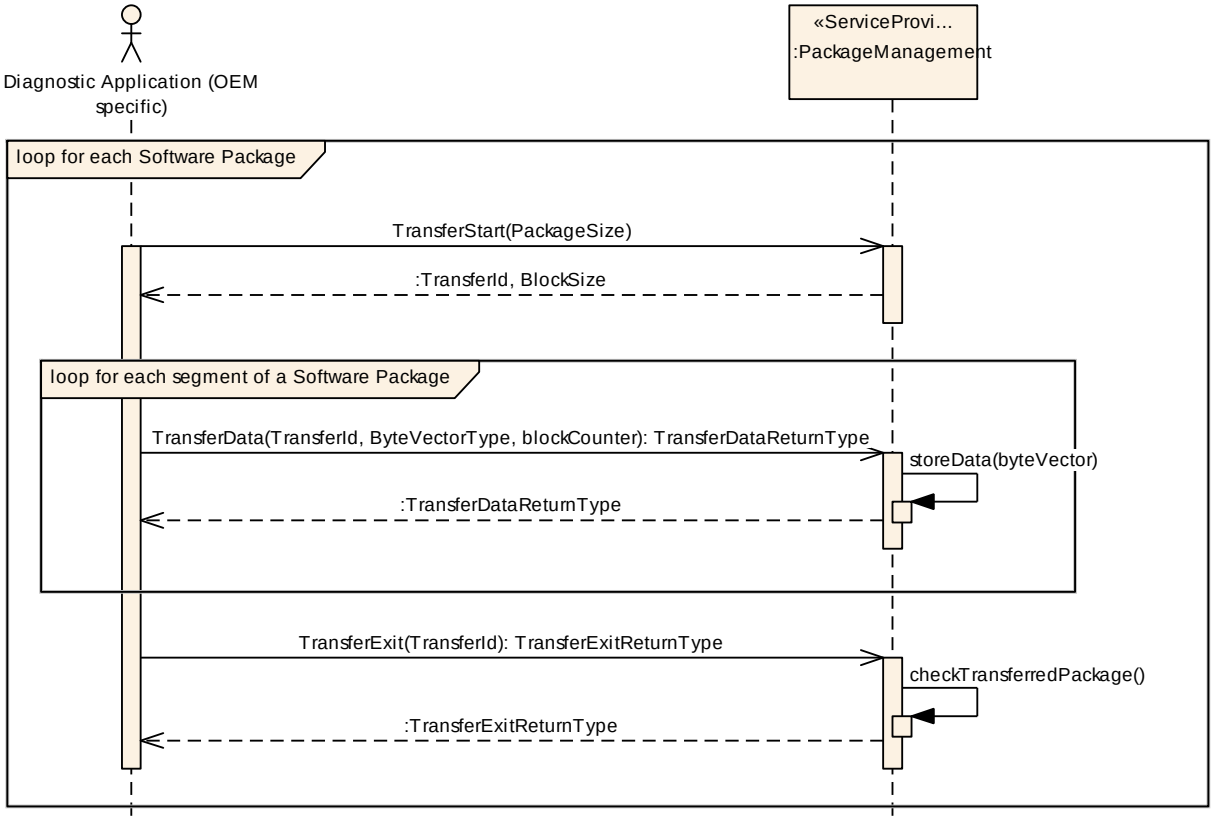

Figure 10.2: Sequence diagram showing the data transmission

10.3 Package processing

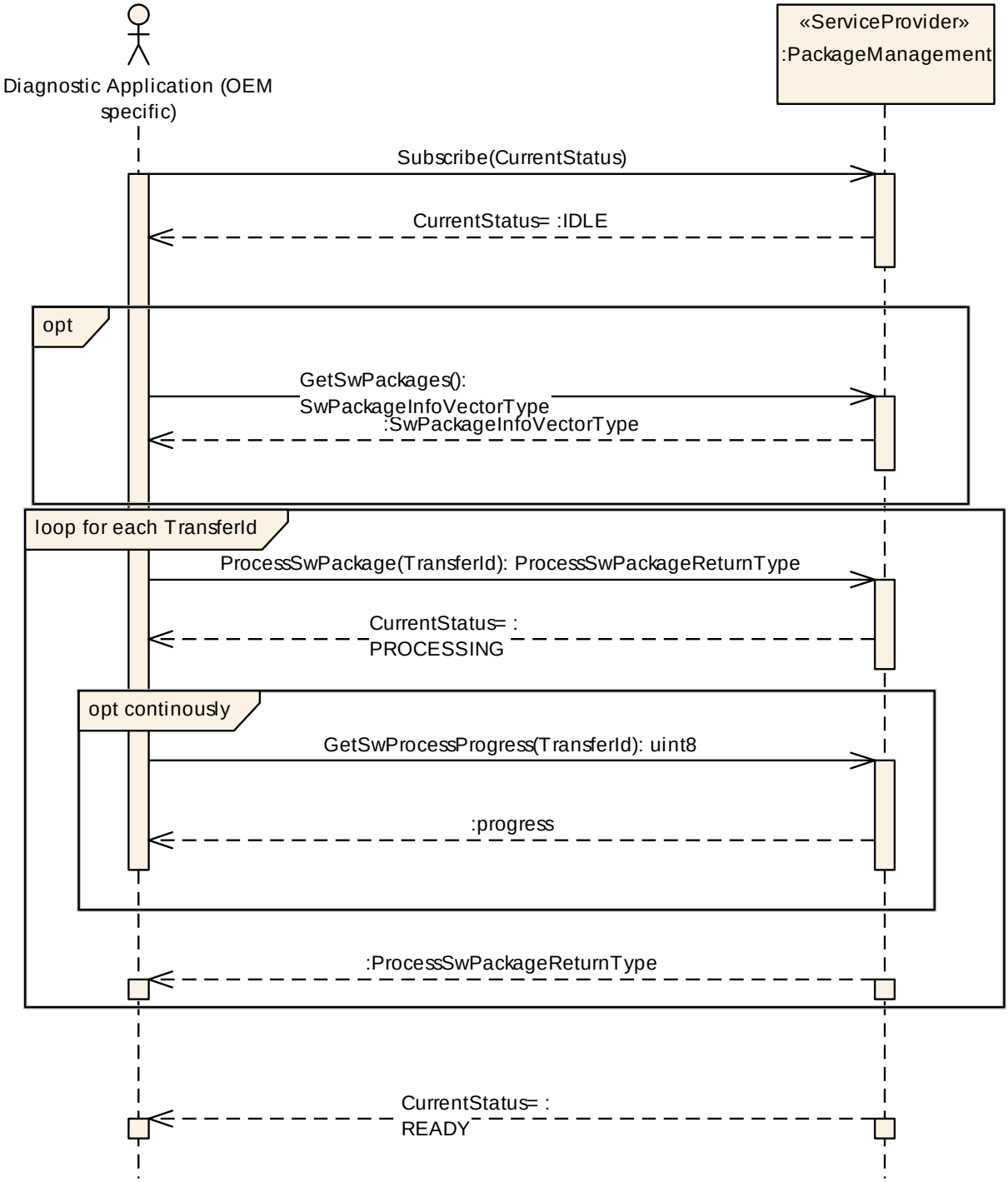

Figure 10.3: Sequence diagram showing the package processing

10.4 Activation

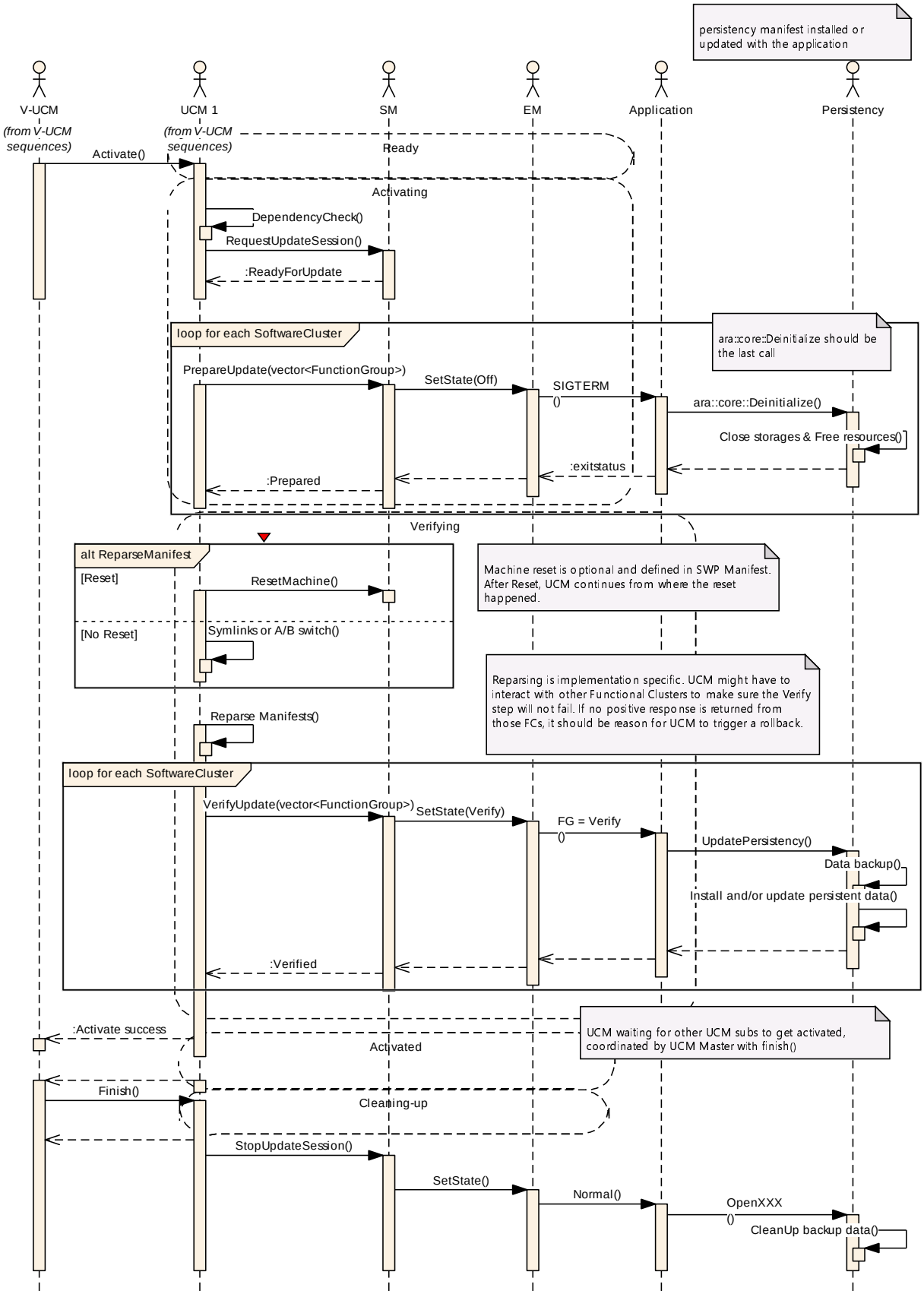

Figure 10.4: Sequence diagram showing the activation process

10.5 Failing activation

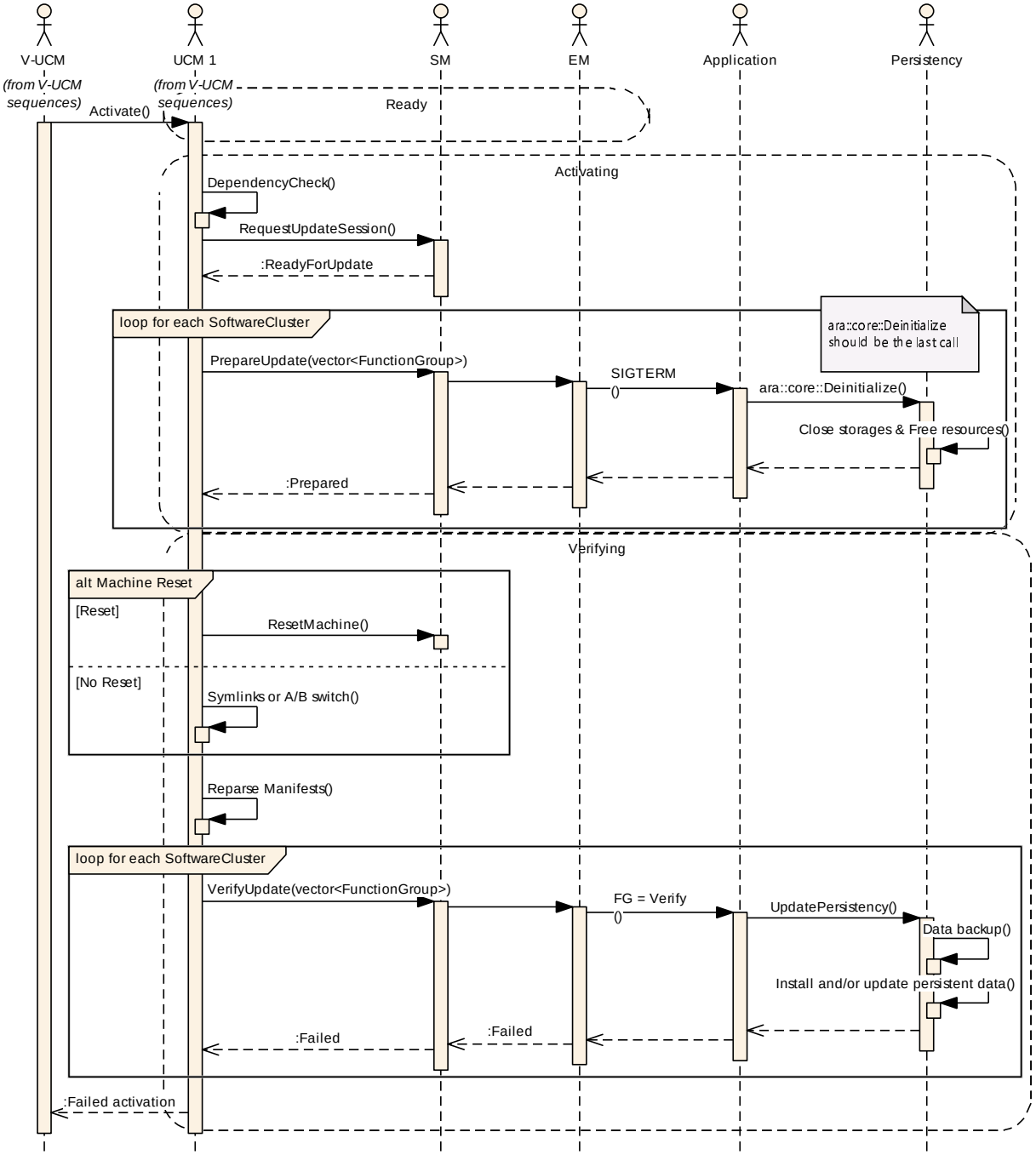

Figure 10.5: Sequence diagram showing an activation failing

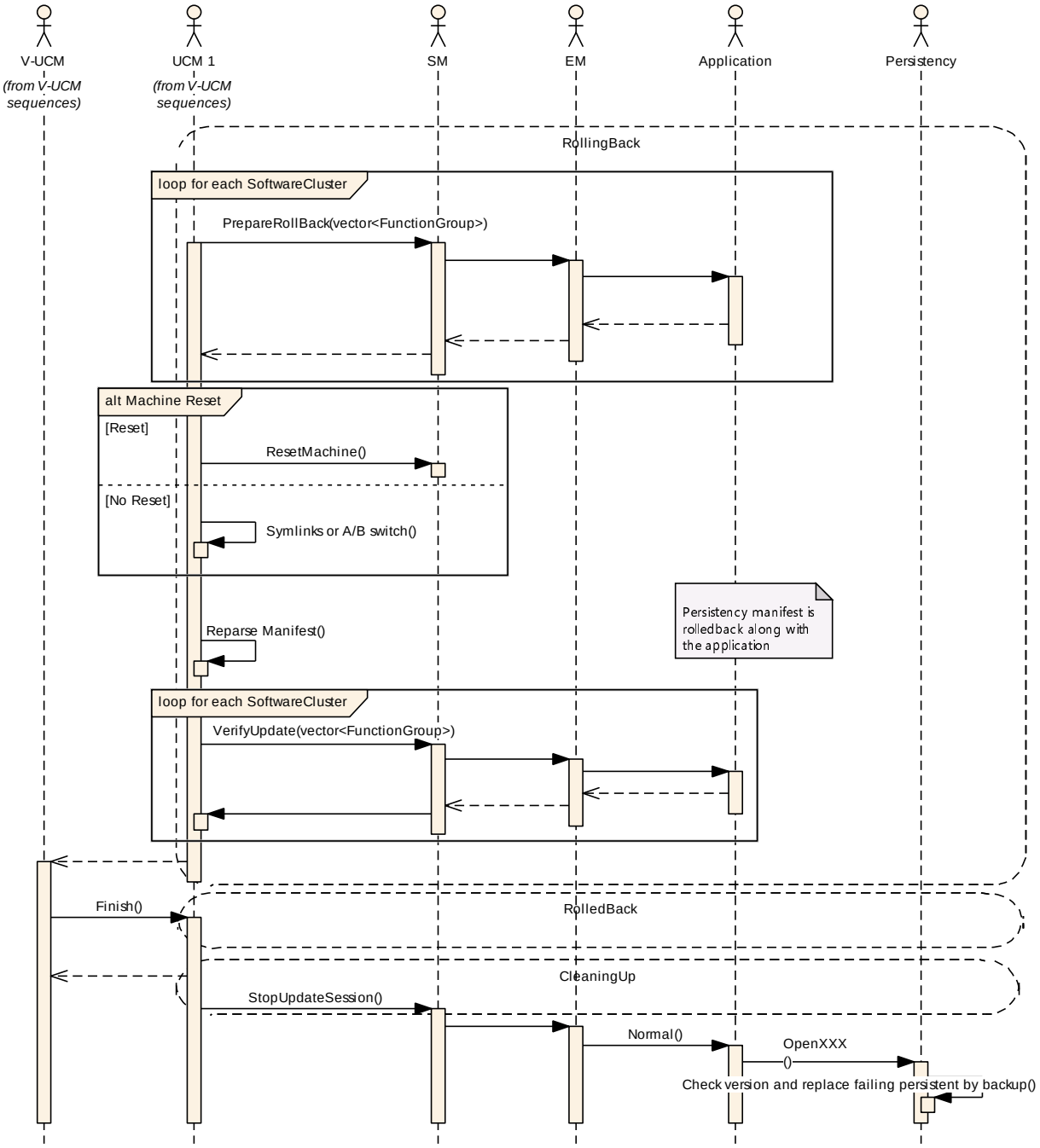

Figure 10.6: Sequence diagram showing an activation failing

10.6 Failing rollback

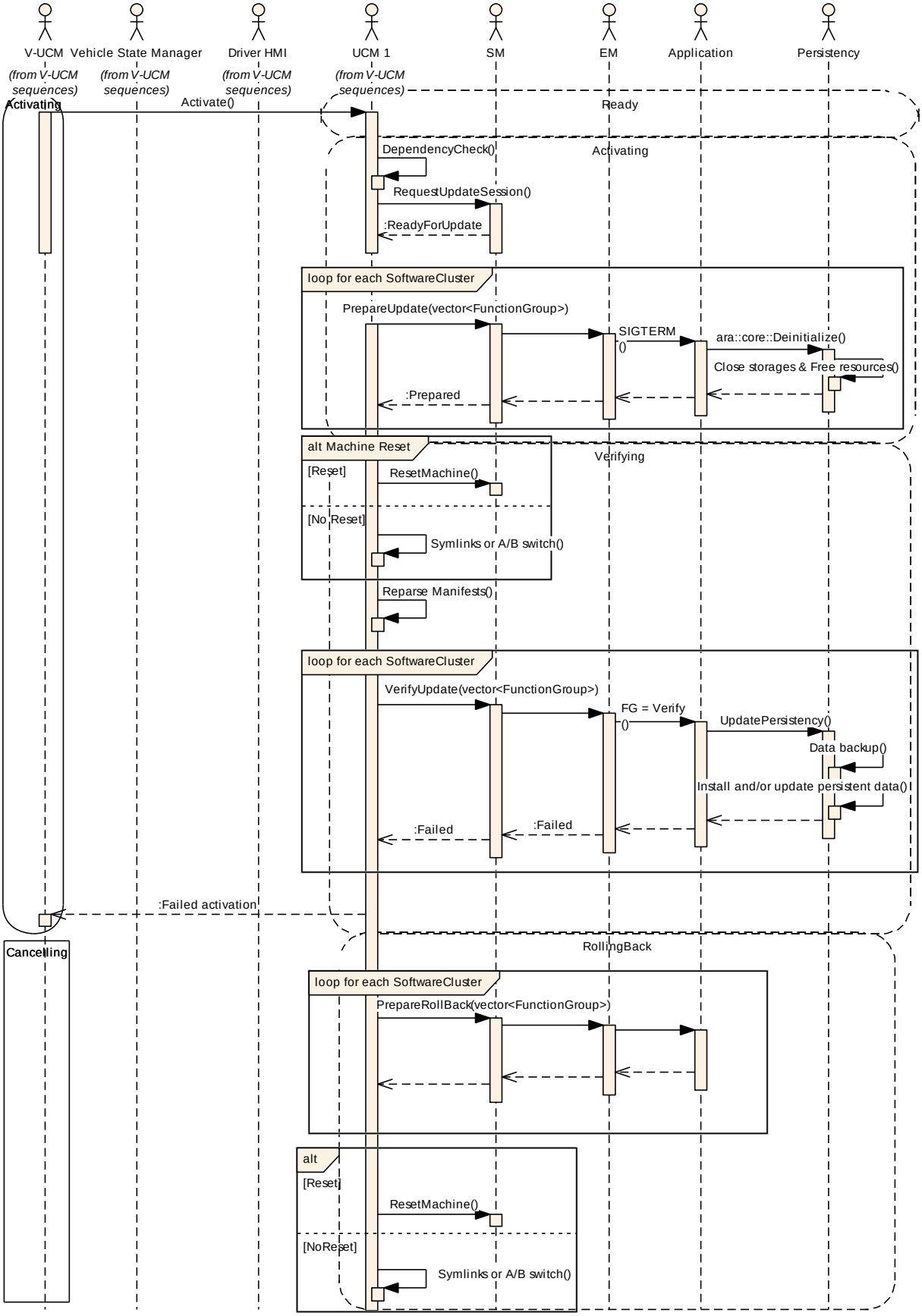

Figure 10.7: Sequence diagram showing a rollback failing

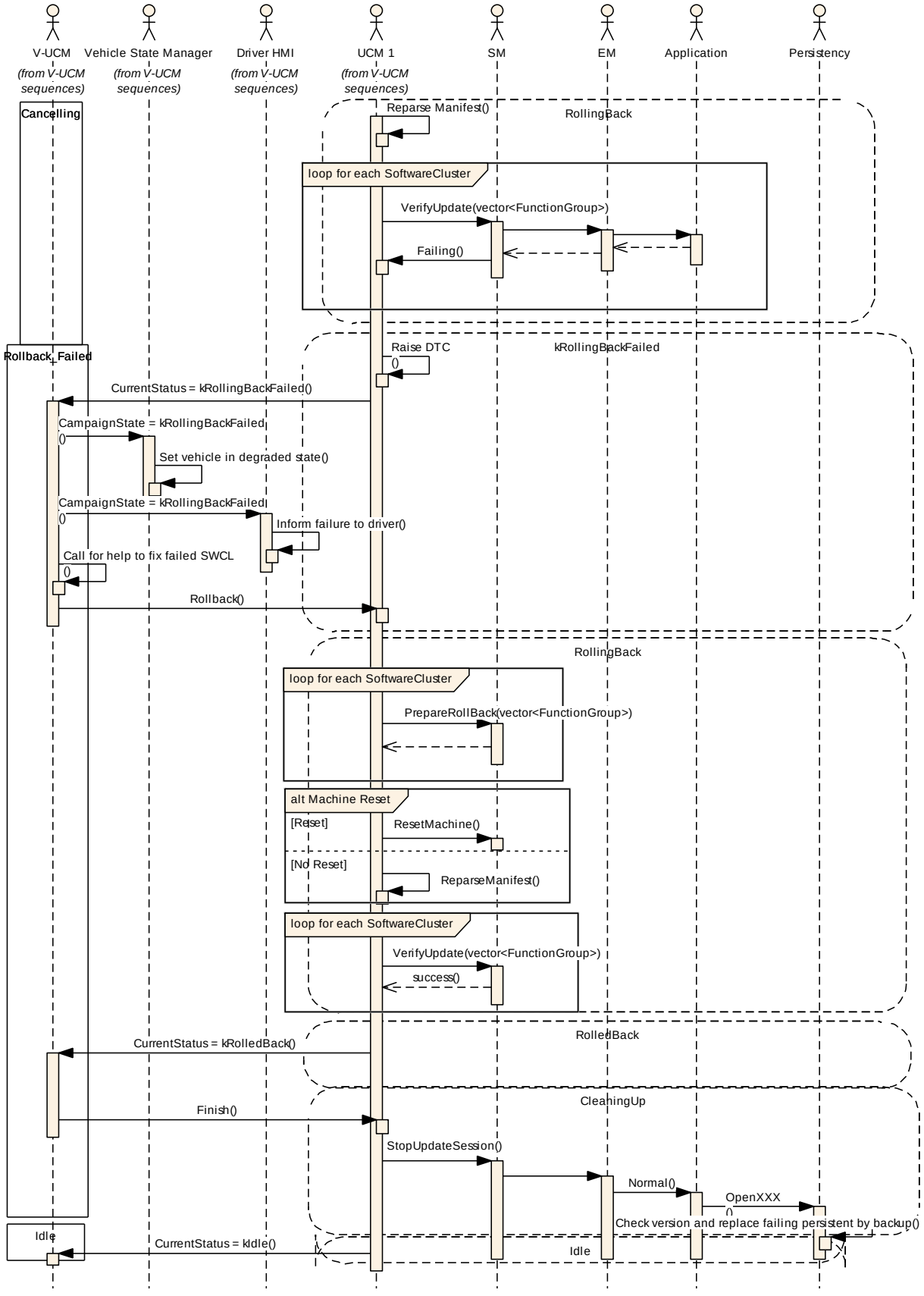

Figure 10.8: Sequence diagram showing a rollback failing

10.7 V-UCM simplified vehicle update

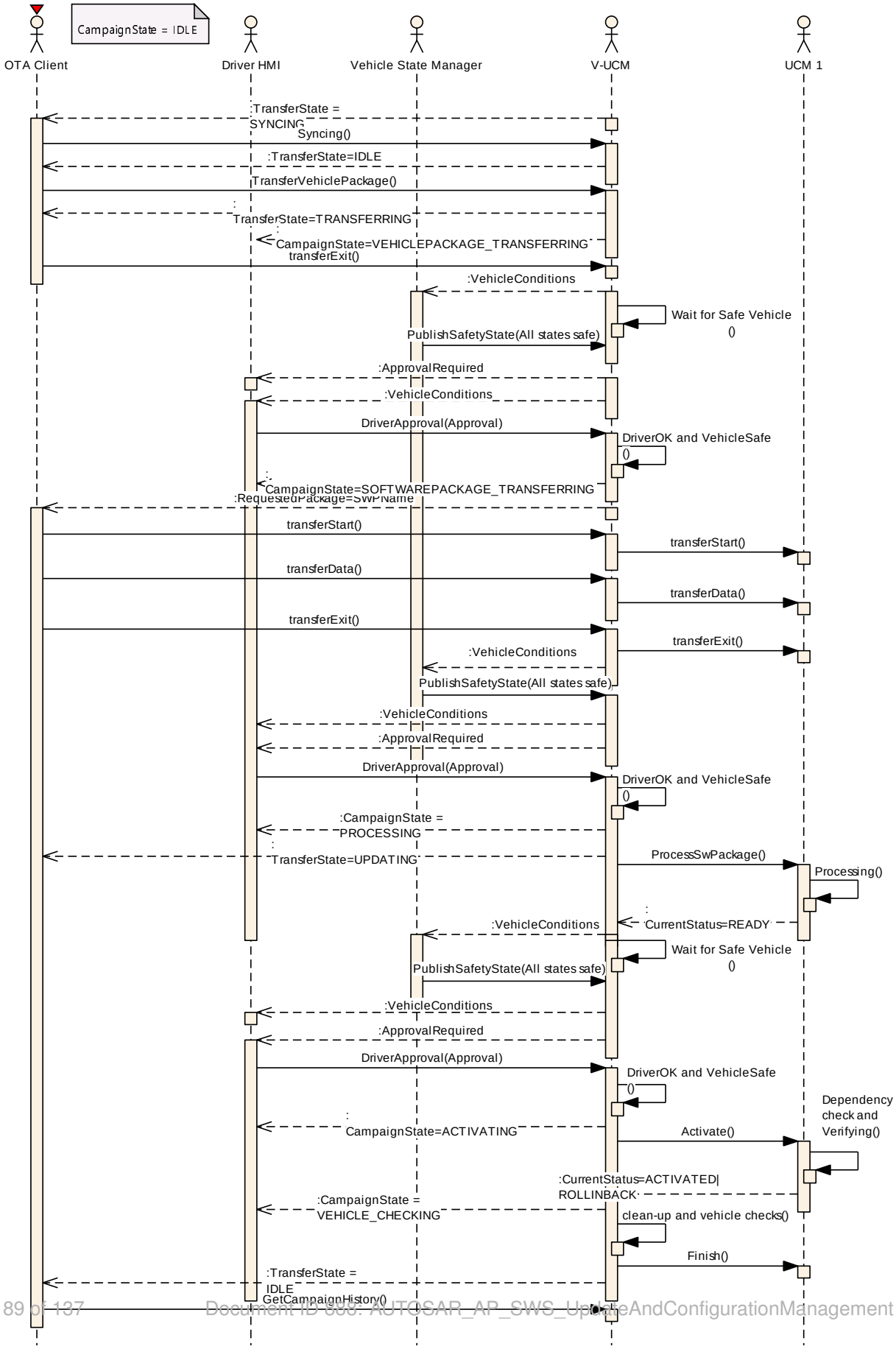

Figure 10.9: Sequence diagram showing vehicle update

A Mentioned Manifest Elements

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

Chapter is generated.

Class	**ArtifactChecksum**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class provides the ability to associate a checksum with a given artifact identified by its URI.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	SoftwareCluster.artifactChecksum			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
checksumValue	String	0..1	attr	This attributes carries the serialized checksum of the corresponding artifact.
uri	UriString	0..1	attr	This attribute represents the URI of the artifact on which the checksum shall be computed. **Stereotypes:** atpIdentityContributor

Table A.1: ArtifactChecksum

Class	**CryptoServiceCertificate**			
Package	M2::AUTOSARTemplates::SystemTemplate::SecureCommunication			
Note	This meta-class represents the ability to model a cryptographic certificate. **Tags:** atp.recommendedPackage=CryptoServiceCertificates			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDesignElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
algorithmFamily	CryptoCertificate AlgorithmFamilyEnum	0..1	attr	This attribute represents a description of the family of crypto algorithm used to generate public key and signature of the cryptographic certificate.
format	CryptoCertificateFormat Enum	0..1	attr	This attribute can be used to provide information about the format used to create the certificate
maximum Length	PositiveInteger	0..1	attr	This attribute represents the ability to define the maximum length of the certificate in bytes.
nextHigher Certificate	CryptoService Certificate	0..1	ref	The reference identifies the next higher certificate in the certificate chain.
serverName Identification	String	0..1	attr	Server Name Indication (SNI) is needed if the IP address hosts multiple servers (on the same port), each of them using a different certificate. If the client sends the SNI to the Server in the client hello, the server looks the SNI up in its certificate list and uses the certificate identified by the SNI.

Table A.2: CryptoServiceCertificate

Class	***Identifiable*** (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables.			
Base	*ARObject*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ARPackage, *AbstractDoIpLogicAddressProps*, *AbstractEvent*, *AbstractImplementationDataTypeElement*, *AbstractSecurityEventFilter*, *AbstractSecurityIdsmInstanceFilter*, *AbstractServiceInstance*, *AbstractSignalBasedToISignalTriggeringMapping*, AdaptiveSwcInternalBehavior, ApApplicationEndpoint, ApplicationEndpoint, ApplicationError, AppliedStandard, ArtifactChecksum, ArtifactLocator, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, AutosarOperationArgumentInstance, AutosarVariableInstance, *BuildActionEntity*, BuildActionEnvironment, Chapter, CheckpointTransition, ClassContentConditional, ClientIdDefinition, ClientServerOperation, Code, *CollectableElement*, ComManagementMapping, *CommConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEventGroup, CouplingPort, *CouplingPortStructuralElement*, CryptoCertificate, CryptoKeySlot, CryptoProvider, *CryptoServiceMapping*, DataPrototypeGroup, DataTransformation, DdsCpDomain, DdsCpPartition, DdsCpQosProfile, DdsCpTopic, DdsDomainRange, DependencyOnArtifact, *DiagEventDebounceAlgorithm*, DiagnosticAuthTransmitCertificateEvaluation, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticDebounceAlgorithmProps, DiagnosticFunctionInhibitSource, DiagnosticParameterElement, *DiagnosticRoutineSubfunction*, DiagnosticSovdMethodPrimitive, DltApplication, DltArgument, DltMessage, DoIpInterface, DoIpLogicAddress, DoIpRoutingActivation, E2EProfileConfiguration, End2EndEventProtectionProps, End2EndMethodProtectionProps, EndToEndProtection, EthernetWakeupSleepOnDatalineConfig, EventHandler, EventMapping, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeatureMapElement, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FieldMapping, FireAndForgetMethodMapping, FlexrayArTpNode, FlexrayTpPduPool, *FrameTriggering*, GeneralParameter, GlobalSupervision, GlobalTimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HealthChannel*, *HeapUsage*, HwAttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, *IEEE1722TpAcfBus*, *IEEE1722TpAcfBusPart*, IPSecRule, IPv6ExtHeaderFilterList, ISignalToIPduMapping, ISignalTriggering, *IdentCaption*, ImpositionTime, InternalTriggeringPoint, Keyword, LifeCycleState, Linker, MacMulticastGroup, MacSecKayParticipant, McDataInstance, MemorySection, MemoryUsage, MethodMapping, ModeDeclaration, ModeDeclarationMapping, ModeSwitchPoint, NetworkEndpoint, *NmCluster*, *NmNode*, *PackageableElement*, ParameterAccess, PduActivationRoutingGroup, PduToFrameMapping, PduTriggering, PerInstanceMemory, *PersistencyDeploymentElement*, *PersistencyInterfaceElement*, *PhmSupervision*, *PhysicalChannel*, PortGroup, *PortInterfaceMapping*, PossibleErrorReaction, ProcessToMachineMapping, Processor, ProcessorCore, PskIdentityToKeySlotMapping, ResourceConsumption, ResourceGroup, RootSwClusterDesignComponentPrototype, RootSwComponentPrototype, RootSwCompositionPrototype, RptComponent, RptContainer, RptExecutableEntity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptServicePoint, RunnableEntityGroup, *SdgAttribute*, SdgClass, SecOcJobMapping, SecOcJobRequirement, SecureCommunicationAuthenticationProps, *SecureCommunicationDeployment*, SecureCommunicationFreshnessProps, SecurityEventContextProps, *ServiceEventDeployment*, *ServiceFieldDeployment*, ServiceInterfaceElementSecureComConfig, *ServiceMethodDeployment*, *ServiceNeeds*, SignalServiceTranslationEventProps, SignalServiceTranslationProps, SocketAddress, SoftwarePackageStep, SomeipEventGroup, SomeipProvidedEventGroup, SomeipTpChannel, *SpecElementReference*, *StackUsage*, *StateManagementActionItem*, StateManagementActionList, StateManagementStateNotification, *StateManagementStateRequest*, StaticSocketConnection, StructuredReq, SupervisionCheckpoint, SupervisionMode, SupervisionModeCondition, SwGenericAxisParamType, SwServiceArg, SwcServiceDependency, SystemMapping, *TimeBaseResource*, *TimingClock*, TimingClockSyncAccuracy, TimingCondition, *TimingConstraint*, *TimingDescription*, TimingExtensionResource, TimingModeInstance, TlsCryptoCipherSuite, TlsCryptoCipherSuiteProps, TlsJobMapping, Topic1, TpAddress, TraceableTable, TraceableText, *TracedFailure*, *TransformationProps*, TransformationTechnology, Trigger, UcmDescription, UcmRetryStrategy, UcmStep, VariableAccess, VariationPointProxy, VehicleRolloutStep, ViewMap, VlanConfig, WaitPoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=adminData xml.sequenceOffset=-40

▽

△

Class	***Identifiable*** (abstract)			
annotation	Annotation	*	aggr	Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes. **Tags:** xml.sequenceOffset=-25
category	CategoryString	0..1	attr	The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints. **Tags:** xml.sequenceOffset=-50
desc	MultiLanguageOverview Paragraph	0..1	aggr	This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question. More elaborate documentation, (in particular how the object is built or used) should go to "introduction". **Tags:** xml.sequenceOffset=-60
introduction	DocumentationBlock	0..1	aggr	This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock. **Tags:** xml.sequenceOffset=-30
uuid	String	0..1	attr	The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp. **Tags:** xml.attribute=true

Table A.3: Identifiable

Class	***PersistencyDeployment*** (abstract)
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency
Note	This abstract meta-class serves as a base class for concrete classes representing different aspects of persistency.
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadableExclusivePackageElement*, *UploadablePackageElement*
Subclasses	PersistencyFileStorage, PersistencyKeyValueStorage
Aggregated by	ARPackage.element

▽

△

Class	***PersistencyDeployment*** (abstract)			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
deploymentUri (ordered)	PersistencyDeployment Uri	*	aggr	This aggregation represents the collection of URIs relevant for the enclosing PersistencyDeployment.
maximum AllowedSize	PositiveUnlimitedInteger	0..1	attr	The value of this attribute represents the maximum size (unit: bytes) allowed at deployment time for the enclosing PersistencyDeployment.
minimum SustainedSize	PositiveInteger	0..1	attr	The value of this attribute represents the minimum size (unit: bytes) guaranteed at deployment time for the enclosing PersistencyDeployment.
redundancy Handling	PersistencyRedundancy Handling	*	aggr	This aggregation represents the chosen approaches to handle redundancy.
updateStrategy	PersistencyCollection LevelUpdateStrategy Enum	0..1	attr	This attribute shall be used to specify the update strategy of the respective PersistencyDeployment as a whole.
version	StrongRevisionLabel String	0..1	attr	The attribute represents the version of the `PersistencyFileStorage` or `PersistencyKeyValueStorage`.

Table A.4: PersistencyDeployment

Class	**PersistencyDeploymentUri**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Persistency			
Note	This meta-class represents the ability to contain URIs relevant for the persistency deployment.			
Base	*ARObject*			
Aggregated by	*PersistencyDeployment*.deploymentUri			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
uri	UriString	0..1	attr	This attribute holds the storage location for the concrete subclass of PersistencyDeployment, e.g. file on the file system.

Table A.5: PersistencyDeploymentUri

Class	**ProcessToMachineMapping**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::MachineManifest			
Note	This meta-class has the ability to associate a Process with a Machine. This relation involves the definition of further properties, e.g. timeouts.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	ProcessToMachineMappingSet.processToMachineMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
design	ProcessDesignTo MachineDesignMapping	0..1	ref	This reference represents the identification of the design-time representation for the ProcessToMachine Mapping that owns the reference.
machine	Machine	0..1	ref	This reference identifies the Machine in the context of the ProcessToMachineMapping.
nonOsModule Instantiation	NonOsModule Instantiation	0..1	ref	This supports the optional case that the process represents a platform module.
persistency CentralStorage URI	UriString	0..1	attr	This attribute identifies a central place for the mapped Process to store the list of available storages and version information.
process	Process	0..1	ref	This reference identifies the Process in the context of the ProcessToMachineMapping.

▽

△

Class	ProcessToMachineMapping			
shallNotRunOn	ProcessorCore	*	ref	This reference indicates a collection of cores onto which the mapped process shall not be executing.
shallRunOn	ProcessorCore	*	ref	This reference indicates a collection of cores onto which the mapped process shall be executing.

Table A.6: ProcessToMachineMapping

Class	*Referrable* (abstract)			
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable			
Note	Instances of this class can be referred to by their identifier (while adhering to namespace borders).			
Base	*ARObject*			
Subclasses	*AtpDefinition*, BswDistinguishedPartition, *BswModuleCallPoint*, BswModuleClientServerEntry, BswVariableAccess, CouplingPortTrafficClassAssignment, *CppImplementationDataTypeContextTarget*, *DiagnosticEnvModeElement*, EthernetPriorityRegeneration, ExclusiveAreaNestingOrder, *HwDescriptionEntity*, *ImplementationProps*, ModeTransition, *MultilanguageReferrable*, NmNetworkHandle, PncMappingIdent, *SingleLanguageReferrable*, SoConIPduIdentifier, SocketConnectionBundle, SomeipRequiredEventGroup, TimeSyncServerConfiguration, TpConnectionIdent			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
shortName	Identifier	1	attr	This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference. **Stereotypes:** atpIdentityContributor **Tags:** xml.enforceMinMultiplicity=true xml.sequenceOffset=-100
shortNameFragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:** xml.sequenceOffset=-90

Table A.7: Referrable

Class	SoftwareCluster			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to define an uploadable software-package, i.e. the SoftwareCluster shall contain all software and configuration for a given purpose. **Tags:** atp.recommendedPackage=SoftwareClusters			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
artifactChecksum	ArtifactChecksum	*	aggr	This aggregation carries the checksums for artifacts contained in the enclosing SoftwareCluster. Please note that the value of these checksums is only applicable at the time of configuration. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=artifactChecksum.shortName, artifactChecksum.uri
artifactLocator	ArtifactLocator	*	aggr	This aggregation represents the artifact locations that are relevant in the context of the enclosing SoftwareCluster

▽

△

Class	**SoftwareCluster**			
claimed FunctionGroup	ModeDeclarationGroup Prototype	*	ref	Each SoftwareCluster can reserve the usage of a given functionGroup such that no other SoftwareCluster is allowed to use it
conflictsTo	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation handles conflicts. If it yields true then the SoftwareCluster shall not be installed. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=conflictsTo
contained ARElement	ARElement	*	ref	This reference represents the collection of model elements that cannot derive from UploadablePackage Element and that contribute to the completeness of the definition of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=containedARElement
containedFibex Element	FibexElement	*	ref	This allows for referencing FibexElements that need to be considered in the context of a SoftwareCluster.
contained Package Element	UploadablePackage Element	*	ref	This reference identifies model elements that are required to complete the manifest content. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=containedPackageElement
contained Process	Process	*	ref	This reference represent the processes contained in the enclosing SoftwareCluster.
dependsOn	SoftwareCluster DependencyFormula	0..1	aggr	This aggregation can be taken to identify a dependency for the enclosing SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=dependsOn
design	SoftwareClusterDesign	*	ref	This reference represents the identification of all Software ClusterDesigns applicable for the enclosing Software Cluster. **Stereotypes:** atpUriDef
diagnostic Deployment Props	SoftwareCluster DiagnosticDeployment Props	0..1	ref	This reference identifies the applicable SoftwareCluster DiagnosticDeploymentProps that are applicable for the referencing SoftwareCluster.
installation Behavior	SoftwareCluster InstallationBehavior Enum	0..1	attr	This attribute controls the behavior of the SoftwareCluster in terms of installation.
license	Documentation	*	ref	This attribute allows for the inclusion of the full text of a license of the enclosing SoftwareCluster. In many cases open source licenses require the inclusion of the full license text to any software that is released under the respective license.
module Instantiation	AdaptiveModule Instantiation	*	ref	This reference identifies AdaptiveModuleInstantiations that need to be included with the SoftwareCluster in order to establish infrastructure required for the installation of the SoftwareCluster. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=moduleInstantiation
releaseNotes	Documentation	0..1	ref	This attribute allows for the explanations of changes since the previous version. The list of changes might require the creation of multiple paragraphs of test.
typeApproval	String	0..1	attr	This attribute carries the homologation information that may be specific for a given country.
vendorId	PositiveInteger	0..1	attr	Vendor ID of this Implementation according to the AUTOSAR vendor list.

▽

△

Class	**SoftwareCluster**			
vendor Signature	CryptoService Certificate	0..1	ref	This reference identifies the certificate that represents the vendor's signature.
version	StrongRevisionLabel String	0..1	attr	This attribute can be used to describe a version information for the enclosing SoftwareCluster.

Table A.8: SoftwareCluster

Enumeration	**SoftwareClusterInstallationBehaviorEnum**
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution
Note	This enumeration defines possible approaches for the installation behavior of a SoftwareCluster.
Aggregated by	SoftwareCluster.installationBehavior
Literal	***Description***
canBeRemoved	The enclosing SoftwareCluster can be removed from the target Machine or updated with a newer version. **Tags:** atp.EnumerationLiteralIndex=0
cannotBeRemoved	The enclosing SoftwareCluster cannot be removed from the target Machine. It can only be updated with a newer version. **Tags:** atp.EnumerationLiteralIndex=1

Table A.9: SoftwareClusterInstallationBehaviorEnum

Class	**SoftwarePackage**			
Package	M2::AUTOSARTemplates::AdaptivePlatform::SoftwareDistribution			
Note	This meta-class represents the ability to formalize the content of a software package. **Tags:** atp.recommendedPackage=SoftwarePackages			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*, *UploadableDeploymentElement*, *UploadablePackageElement*			
Aggregated by	ARPackage.element			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
actionType	SoftwarePackageAction TypeEnum	0..1	attr	This attribute defines the action to be taken in the step of processing the enclosing SoftwarePackage.
activationAction	SoftwarePackage ActivationActionEnum	0..1	attr	This attribute governs the action to be taken after the installation of the SoftwareCluster completed.
compressed Software PackageSize	PositiveInteger	0..1	attr	This size represents the size of the compressed Software Package.
deltaPackage Applicable Version	StrongRevisionLabel String	0..1	attr	This attribute identifies the version of the included SoftwareCluster for which the enclosing SoftwarePackage can be used as a delta update
estimated DurationOf Operation	TimeValue	0..1	attr	This attribute provides an estimation about how long the operation of the SoftwarePackage is going to take for its transfer, processing and activation when updated standalone (not within an update campaign)
minimum SupportedUcm Version	RevisionLabelString	0..1	attr	This attribute identifies the minimum supported version of the UCM for this SoftwarePackage.
packagerId	PositiveInteger	0..1	attr	This attribute identifies Id of the organization that provides the packager generating the SoftwarePackage.
packager Signature	CryptoService Certificate	0..1	ref	This reference identifies the certificate that represents the packager's signature.

▽

△

Class	**SoftwarePackage**			
purposeOf Update	Documentation	0..1	ref	The referenced Documentation is supposed to provide a description of the purpose of the update.
softwareCluster	SoftwareCluster	0..1	ref	This reference identifies the SoftwareCluster that belongs to the SoftwarePackage. The nature of this relation is actually more like an aggregation than a reference. But the relation is still modelled as a reference because two ARElements cannot aggregate each other.
uncompressed SoftwareCluster Size	PositiveInteger	0..1	attr	This attribute gives an indication about the storage that has to be available on the target.

Table A.10: SoftwarePackage

Primitive	**StrongRevisionLabelString**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This primitive represents a revision label which identifies an object under version control. It represents a pattern which requires three integer numbers separated by a dot, representing from left to right Major Version, MinorVersion, PatchVersion and additional labels for pre-release version and build metadata. Legal patterns are for example: 1.0.0-alpha+001 1.0.0+20130313144700 1.0.0-beta+exp.sha.5114f85 **Tags:** xml.xsd.customType=STRONG-REVISION-LABEL-STRING xml.xsd.pattern=(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(-((0|[1-9]\d*|\d*[a-zA-Z-][0-9a-z A-Z-]*)(\.(0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(\+([0-9a-zA-Z-]+(\.[0-9a-zA-Z-]+)*))? xml.xsd.type=string

Table A.11: StrongRevisionLabelString

Class	***UcmModuleInstantiation*** (abstract)			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Ucm			
Note	This meta-class represents the ability to define the deployment of a UCM instantiation.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*			
Subclasses	UcmMasterModuleInstantiation, UcmSubordinateModuleInstantiation			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
identifier	String	0..1	attr	This represents the identification of a UCM.
maxBlockSize	PositiveInteger	0..1	attr	This attribute denotes the maximum block size (unit: bytes) used in the UCM implementation.
version	StrongRevisionLabel String	0..1	attr	This attribute defines the software version of the UCM on this platform. Note that the definition of the version is required if the ability of the SoftwarePackage to require a minimum version of the UCM is utilized.

Table A.12: UcmModuleInstantiation

Class	UcmRetryStrategy			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Ucm			
Note	This meta-class describes the configuration of the retry strategy for a sub-class of UcmModule Implementation.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Aggregated by	UcmMasterModuleInstantiation.blockInconsistent, UcmMasterModuleInstantiation.serviceBusy, UcmMasterModuleInstantiation.updateSessionRejected, UcmSubordinateModuleInstantiation.prepareRollback, UcmSubordinateModuleInstantiation.prepareUpdate, UcmSubordinateModuleInstantiation.verifyUpdate			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
maximumNumberOfRetries	PositiveInteger	0..1	attr	This attribute defines the maximum number of time the UCM module instantiation shall attempt a retry.
retryIntervalTime	TimeValue	0..1	attr	This attribute defines the time (in seconds) between two retry attempts.

Table A.13: UcmRetryStrategy

Class	UcmSubordinateModuleInstantiation			
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Ucm			
Note	This meta-class represents the ability to define the deployment of a UCM Subordinate instantiation.			
Base	*ARObject*, *AdaptiveModuleInstantiation*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *NonOsModuleInstantiation*, *Referrable*, *UcmModuleInstantiation*			
Aggregated by	*AtpClassifier*.atpFeature, Machine.moduleInstantiation			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
maxAvailablePersistencyStorageSpace	PositiveInteger	0..1	attr	This attribute names the maximum amount of space available for persistent data handled by the Persistency of installed packages. The UCM needs to figure out from traversing the minimum storage requirement from existing PersistencyDeployments whether specific packages can be installed from the perspective of available storage space. Note that the minimum storage requirement of PersistencyDeployment needs to include space for the handling of the storage, which shall be calculated by the tooling that creates the manifest information inside the package.
prepareRollback	UcmRetryStrategy	0..1	aggr	This attribute identifies the configuration of prepare rollback retries initiated by the Ucm Subordinate.
prepareUpdate	UcmRetryStrategy	0..1	aggr	This attribute identifies the configuration of prepare update retries initiated by the Ucm Subordinate.
verifyUpdate	UcmRetryStrategy	0..1	aggr	This attribute identifies the configuration of verify update retries initiated by the Ucm Subordinate.

Table A.14: UcmSubordinateModuleInstantiation

Class	UcmToTimeBaseResourceMapping
Package	M2::AUTOSARTemplates::AdaptivePlatform::PlatformModuleDeployment::Ucm
Note	This meta-class maps the UCM Module Instantiation to the TimeSync Module Instantiation. **Tags:** atp.recommendedPackage=FCInteractions
Base	*ARElement*, *ARObject*, *CollectableElement*, *FunctionalClusterInteractsWithFunctionalClusterMapping*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *UploadableDeployment Element*, *UploadablePackageElement*

▽

△

Class	**UcmToTimeBaseResourceMapping**			
Aggregated by	ARPackage.element			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
timeBase Resource	TimeBaseResource	0..1	ref	This reference identifies the relevant TimeBaseResource.
ucm	UcmModuleInstantiation	0..1	ref	This reference identifies the relevant UcmModule Instantiation.

Table A.15: UcmToTimeBaseResourceMapping

B Production Errors

This chapter lists all production errors of the UCM.

B.1 ROLLBACK FAILED

[SWS_UCM_00323] Diagnostic Event: RollBack failed ⌈

Diagnostic Event (Error Name)	UCM_ROLLBACK_FAILED
Description	UCM failed to rollback one or several Software Clusters.
Monitoring condition	This DTC is set during an Updating context
Failed condition	UCM fails to rollback one or several Software Clusters
Passed condition	UCM succeeds in retrying Rollback

⌋(*RS_UCM_00045*)

B.2 HISTORY RECORD FAILED

[SWS_UCM_00320] Diagnostic Event: History recording failed ⌈

Diagnostic Event (Error Name)	UCM_HISTORY_RECORD_FAILED
Description	UCM failed to record history entry
Monitoring condition	This DTC is set during an Updating context
Failed condition	UCM fails to record history

⌋(*RS_UCM_00045*)

B.3 CANCEL FAILED

[SWS_UCM_00325] Diagnostic Event: Campaign cancelling failed ⌈

Diagnostic Event (Error Name)	UCM_CANCEL_FAILED
Description	UCM failed to stop the ongoing Processing after a call of Cancel method was received
Monitoring condition	This DTC is set during an Updating context
Failed condition	UCM fails to stop the processing of a software package on call of Cancel
Passed condition	Update sequence is completed with Finish()

⌋(*RS_UCM_00045*)

B.4 MISSING DEPENDENCIES

[SWS_UCM_00326] Diagnostic Event: Activation not possible because of missing dependencies ⌈

Diagnostic Event (Error Name)	UCM_MISSING_DEPENDENCIES
Description	Software Cluster dependencies are not fulfilled with the set of currently processed Software Clusters
Monitoring condition	This DTC is set during an Updating context
Failed condition	UCM detects an unmet dependency among the new Software Cluster set
Passed condition	UCM detects all dependencies are fulfilled

⌋(*RS_UCM_00045*)

B.5 OLD VERSION PACKAGE

[SWS_UCM_00327] Diagnostic Event: Installing old software is not allowed ⌈

Diagnostic Event (Error Name)	UCM_OLD_VERSION_PACKAGE
Description	Attempt to update to older Software Cluster version than currently present and than previously removed
Failed condition	If there has been an attempt to update a Software Cluster to older version than currently present and than previously removed, UCM shall report this failure as diagnostic error
Passed condition	If UCM succeeded to update, UCM shall report the Prepassed status

⌋(*RS_UCM_00045*)

B.6 PREPAREUPDATE FAILED

[SWS_UCM_00322] Diagnostic Event: PrepareUpdate call to SM failed ⌈

Diagnostic Event (Error Name)	UCM_PREPAREUPDATE_FAILED
Description	SM returned a negative result on call of PrepareUpdate()
Monitoring condition	This DTC is set during an Updating context
Failed condition	UCM fails to call PrepareUpdate
Passed condition	Subsequent call of PrepareUpdate succeeds

⌋(*RS_UCM_00045*)

B.7 UPDATE SESSION REJECTED

[SWS_UCM_00321] Diagnostic Event: Update session with SM rejected ⌈

Diagnostic Event (Error Name)	UCM_UPDATE_SESSION_REJECTED
Description	UCM failed to start an update session
Monitoring condition	This DTC is set during an Updating context
Failed condition	SM returns kRejected on RequestUpdateSession
Passed condition	Subsequent call of RequestUpdateSession succeeds

⌋(*RS_UCM_00045*)

B.8 VERIFICATION FAILED

[SWS_UCM_00324] Diagnostic Event: Verification with SM at activation failed ⌈

Diagnostic Event (Error Name)	UCM_VERIFICATION_FAILED
Description	SM returned a negative result on the VerifyUpdate call
Monitoring condition	This DTC is set during an Updating context
Failed condition	UCM receives a negative result on calling VerifyUpdate
Passed condition	Update sequence is completed with Finish()

⌋(*RS_UCM_00045*)

C Interfaces to other Functional Clusters (informative)

C.1 Overview

AUTOSAR decided not to standardize interfaces which are exclusively used between Functional Clusters (on platform-level only), to allow efficient implementations, which might depend e.g. on the used Operating System.

This chapter provides informative guidelines how the interaction between Functional Clusters looks like, by clustering the relevant requirements of this document. In addition, the standardized public interfaces which are accessible by user space applications (see chapter 8) can also be used for interaction between Functional Clusters.

The goal is to provide a clear understanding of Functional Cluster boundaries and interaction, without specifying syntactical details. This ensures compatibility between documents specifying different Functional Clusters and supports parallel implementation of different Functional Clusters. Details of the interfaces are up to the platform provider.

C.2 Interfaces Tables

C.2.1 UCM update notification

UCM shall provide the notification to other Functional Clusters that changes have been done to the software. This enables other functional clusters to check if updated manifests have changes relevant for the concerned Functional Cluster. This can be done through the field CurrentStatus provided by the UCM service.

D Security Analysis of Installation and Update

This chapter presents a summary for the security analysis of the UCM. Some of the threats could not be addressed by specifying AUTOSAR requirements. The main reason for not specifying the countermeasures is to allow vendors to flexibly decide on the solution that fits their setup. Here we aim to raise awareness and provide advice on the selected topics:

D.1 Securing Software Package

UCM is responsible for applying changes of the platform and applications contained in the Software Packages it receives. Therefore, integrity and authenticity of Software Packages are critical to protect system integrity. It shall be ensured that the Software Packages are neither illegitimately altered nor issued by unauthorized parties. This can be achieved by applying cryptographic techniques such as digital signatures. The period that Software Package resides in UCM before being activated shall not be neglected. It provides a window of opportunity for an attacker to tamper with the Software Package after the authentication is done at TransferExit.

Information disclosure is another security threat category that might be applicable to Software Packages. Packages that contain sensitive information, such as intellectual properties or cryptographic keys, require confidentiality protection in addition to integrity and authenticity when being persisted or transmitted over a communication channel.

Another aspect of protecting Software Update Packages is their freshness. An attacker may try to manipulate the system by downgrading the software via replaying an authentic but older Software Update Package. In this regard, the platform shall ensure that only newer packages (i.e. packages that contain newer version of installed SWCL) can be installed.

D.2 Securing Calls to UCM

UCM provides a very critical functionality in the platform that allows modifying applications and platform components. In that sense, it is critical to prevent unauthorized access to UCM, meaning only legitimate callers should be allowed to reach the UCM service interface. This is primarily enforced in the communication layer supported by the Identity and Access Management. Additionally, the calls to the UCM interface shall be protected against altering, e.g. changing API arguments. When the service and client reside on the same machine, the security relies on the integrity of the operating system and the platform. In case, the service and the client are running on different machines, a secure communication, assuring authenticity and integrity of communication, is additionally required.

Moreover, some API methods of the UCM interface returns sensitive information about the platform. This subset (GetSwClusterInfo, GetSwClusterChangeInfo, GetHistory, GetSwPackages) shall be protected against information disclosure and should only be reachable over a channel that provides confidentiality.

D.3 Suppressing Call to UCM

Multiple scenarios can be envisioned where an attacker targets suppressing the calls to UCM. The attack could block the calls to or the response from UCM. In both cases the caller of the service may assume that UCM is not responding and retries its request. This would lead to undesired overhead on the system. For such scenarios, it is recommended that both UCM and the UCM Client consider reporting security events when same calls repeatedly received at UCM or calls repeatedly fail at the caller side. This information could potentially be picked up by Intrusion Detection Systems or Anomaly Detection Systems.

D.4 Resource Starvation

According to the current specification, the available resources for transferring a Software Package is only checked when TransferStart is called but not reserved. This means, while the transfer is ongoing, the system storage can be exhausted by other processes using the same storage media. A similar case is possible for processing of Software Package, as the resources are only checked at the beginning but not reserved. In this regard, a solution could be to reserve the necessary resources for the Software Package transfer or processing from the beginning to prevent attacks aiming at such scenarios.

At the same time, reserving the resources might provide opportunity to the attacker in other scenarios. The specification allows transferring multiple Software Packages in parallel. Consequently, a misbehaving or compromised client can open unlimited number of transfer sessions causing UCM to run out of resources. To cope with this scenario, a threshold for the number of parallel transfer sessions can be defined.

D.5 Zombie Sessions

The AUTOSAR specification does not enforce any expiry time for the established transfer sessions. As a result, the resources that are hold by an ongoing session will not be released no matter how long time it takes. At the same time, in certain cases it may take a long time for larger software packages to be transferred to UCM, especially when they are received from external sources with weak connectivity on-the-fly. However, a timeout may be considered for such a transfer to prevent attackers from mounting denial of service attacks by long term allocation of resources.

E 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.

E.1 Constraint and Specification Item History of this document according to AUTOSAR Release R19-11.

E.1.1 Added Specification Items in R19-11

Number	Heading
[SWS_UCM_00009]	UCM exposing its identifier
[SWS_UCM_00105]	UCM confidential information handling
[SWS_UCM_00161]	Check Software Package version compatibility against UCM version
[SWS_UCM_00162]	Entering the Cleaning-up state after a RevertProcessedSwPackages call
[SWS_UCM_00163]	Action in Cleaning-up state
[SWS_UCM_00164]	Cleaning up of Software Packages
[SWS_UCM_00165]	Processing from stream
[SWS_UCM_00166]	Processing from stream state
[SWS_UCM_00167]	Cancelling streamed packages
[SWS_UCM_00168]	Transferring while processing from stream
[SWS_UCM_00169]	Finishing transfer while processing from stream
[SWS_UCM_00170]	Log message retrieving
[SWS_UCM_00171]	Log level changing
[SWS_UCM_00172]	Log messages removing
[SWS_UCM_00173]	UCMIdentifierType table
[SWS_UCM_00174]	SwNameVectorType table
[SWS_UCM_00175]	StrongRevisionLabelString table
[SWS_UCM_00176]	SwNameVersionType table
[SWS_UCM_00177]	SwNameVersionVectorType table
[SWS_UCM_00178]	ProvidedPort VehiclePackageManagement
[SWS_UCM_00179]	RequiredPort VehicleStateManager
[SWS_UCM_00180]	RequiredPort VehicleDriverApplication
[SWS_UCM_00181]	ProvidedInterface VehiclePackageManagement
[SWS_UCM_00182]	RequiredInterface VehicleDriverApplication
[SWS_UCM_00183]	RequiredInterface VehicleStateManager
[SWS_UCM_00210]	Transferring of software packages on `kProcessApproving` or `kProcessing` state

▽

△

Number	Heading
[SWS_UCM_01001]	UCM Master processes Vehicle Package
[SWS_UCM_01002]	UCM Master shall provide UCM services
[SWS_UCM_01003]	UCM Master checks states of UCM subordinates
[SWS_UCM_01004]	Only one UCM Master shall be active per network domain
[SWS_UCM_01005]	UCM Master is discovering UCMs in vehicle
[SWS_UCM_01006]	Vehicle Package transfer to UCM Master
[SWS_UCM_01007]	Start transfer of a Vehicle Package or Software Packageto UCM Master
[SWS_UCM_01008]	Transfer data of a Vehicle Package to UCM Master
[SWS_UCM_01009]	Exit the transfer of a Vehicle Package to UCM Master
[SWS_UCM_01010]	Delete a Vehicle Package transferred to UCM Master
[SWS_UCM_01101]	Provide information of installed Software Clusters in vehicle
[SWS_UCM_01102]	Get information of available Software Clusters in Backend
[SWS_UCM_01103]	Inform Backend of needed Software Clusters for an update
[SWS_UCM_01105]	Interaction of UCM Master with Vehicle Driver
[SWS_UCM_01106]	Exclusive use of Vehicle Driver Interface
[SWS_UCM_01107]	UCM Master provides progress information to Vehicle Driver
[SWS_UCM_01108]	Unsupported safety policy by Vehicle driver interface
[SWS_UCM_01109]	Vehicle State Manager shall provide to UCM Master a safety state
[SWS_UCM_01110]	UCM Master shall be able to set the safety policy to be computed by Vehicle State Manager
[SWS_UCM_01111]	Exclusive use of Vehicle State Manager
[SWS_UCM_01112]	Unsupported safety policy by Vehicle State Manager
[SWS_UCM_01113]	Switching vehicle into update mode
[SWS_UCM_01114]	SafetyPolicyType table
[SWS_UCM_01115]	VehicleStateManagerErrorDomain
[SWS_UCM_01116]	VehicleDriverApplicationErrorDomain
[SWS_UCM_01177]	CampaignStateType table
[SWS_UCM_01201]	Sequential orchestration of campaigns
[SWS_UCM_01203]	CampaignState field
[SWS_UCM_01204]	Initial state
[SWS_UCM_01205]	UCM Master internal state persistency
[SWS_UCM_01206]	Trigger on kTransferApproving state
[SWS_UCM_01207]	Trigger on kTransferring state
[SWS_UCM_01208]	Trigger on kProcessApproving state
[SWS_UCM_01209]	Trigger on kProcessing state
[SWS_UCM_01211]	Trigger on kActivateApproving state
[SWS_UCM_01212]	Trigger on kActivating state
[SWS_UCM_01213]	Trigger on kVehicleChecking state

▽

△

Number	Heading
[SWS_UCM_01214]	Final action on `kVehicleChecking` state
[SWS_UCM_01215]	Trigger on `kRollingBack` state
[SWS_UCM_01216]	Final action on `kRollingBack` state
[SWS_UCM_01217]	Monitoring of UCM subordinates
[SWS_UCM_01218]	Transition from `kIdle` state to `kSyncing` state
[SWS_UCM_01219]	Transition from `kSyncing` state to `kIdle` state
[SWS_UCM_01220]	Transition from `kIdle` state to `kVehiclePackageTransferring` state
[SWS_UCM_01221]	Transition from `kVehiclePackageTransferring` state to `kIdle` state
[SWS_UCM_01222]	Transition from `kVehiclePackageTransferring` state to `kTransferring` state
[SWS_UCM_01223]	Transition from `kVehiclePackageTransferring` state to `kTransferApproving` state
[SWS_UCM_01224]	Transition from `kTransferApproving` state to `kTransferring` state
[SWS_UCM_01225]	Transition from `kTransferApproving` state to `kIdle` state
[SWS_UCM_01226]	Transition from `kTransferring` state to `kTransferApproving` state
[SWS_UCM_01227]	Transition from `kTransferring` state to `kIdle` state
[SWS_UCM_01228]	Transition from `kTransferring` state to `kProcessing` state
[SWS_UCM_01229]	SafetyPolicy while processing stream
[SWS_UCM_01230]	Transition from `kTransferring` state to `kProcessApproving` state
[SWS_UCM_01231]	Transition from `kProcessApproving` state to `kProcessing` state
[SWS_UCM_01232]	Transition from `kProcessApproving` state to `kIdle` state
[SWS_UCM_01233]	Transition from `kProcessing` state to `kProcessApproving` state
[SWS_UCM_01234]	Transition from `kProcessing` state to `kActivating` state
[SWS_UCM_01235]	Transition from `kProcessing` state to `kActivateApproving` state
[SWS_UCM_01236]	Transition from `kProcessing` state to `kIdle` state
[SWS_UCM_01237]	Transition from `kActivateApproving` state to `kActivating` state
[SWS_UCM_01238]	Transition from `kActivateApproving` state to `kIdle` state
[SWS_UCM_01239]	Transition from `kActivating` state to `kRollingBack` state
[SWS_UCM_01240]	Transition from `kActivating` state to `kVehicleChecking` state
[SWS_UCM_01241]	Transition from `kVehicleChecking` state to `kRollingBack` state
[SWS_UCM_01242]	Transition from `kVehicleChecking` state to `kIdle` state
[SWS_UCM_01243]	Transition from `kRollingBack` state to `kIdle` state
[SWS_UCM_01244]	Cancellation of an update campaign shall be possible
[SWS_UCM_01245]	Cancellation during activation shall be possible
[SWS_UCM_01246]	Unreachable UCM during update campaign
[SWS_UCM_01247]	Method to read History Report
[SWS_UCM_01248]	Content of History Report
[SWS_UCM_01301]	`Vehicle Package` authentication
[SWS_UCM_01302]	`Vehicle Package` authentication failure

▽

△

Number	Heading
[SWS_UCM_01303]	Dependencies between `Software Packages`
[SWS_UCM_01304]	Confidential information protection
[SWS_UCM_CONSTR_00001]	

Table E.1: Added Specification Items in R19-11

E.1.2 Changed Specification Items in R19-11

Number	Heading
[SWS_UCM_00003]	Cancelling the package processing
[SWS_UCM_00017]	Sequential `Software Package` Processing
[SWS_UCM_00018]	Providing Progress Information
[SWS_UCM_00027]	Delta Package activation
[SWS_UCM_00071]	SwNameType table
[SWS_UCM_00081]	Processing state of Package Management
[SWS_UCM_00082]	Exit from Processing state of Package Management
[SWS_UCM_00102]	Update state
[SWS_UCM_00103]	Update to older `Software Cluster` version than currently present
[SWS_UCM_00104]	Consistency Check of processed Package
[SWS_UCM_00111]	Entering the Rolled-back state
[SWS_UCM_00112]	`Software Cluster` and version
[SWS_UCM_00126]	Entering the RollingBack state after a Rollback call
[SWS_UCM_00130]	`Software Cluster` and version error
[SWS_UCM_00146]	Entering the Cleaning-up state after a Finish call
[SWS_UCM_00149]	Return to the Idle state from Processing state
[SWS_UCM_00151]	Entering the Ready state of Package Management after a Cancel call
[SWS_UCM_00155]	Entering the RollingBack state after a failure in the Verifying state

Table E.2: Changed Specification Items in R19-11

E.1.3 Deleted Specification Items in R19-11

Number	Heading
[SWS_UCM_00012]	Log message retrieving
[SWS_UCM_00114]	ActivateOptionType table
[SWS_UCM_00144]	Log error

Table E.3: Deleted Specification Items in R19-11

E.1.4 Added Constraints in R19-11

none

E.1.5 Changed Constraints in R19-11

none

E.1.6 Deleted Constraints in R19-11

none

E.2 Constraint and Specification Item History of this document according to AUTOSAR Release R20-11.

E.2.1 Added Specification Items in R20-11

Number	Heading
[SWS_UCM_00184]	Persistent data clean-up after Software Cluster removal
[SWS_UCM_00185]	Provide `Software Cluster` general information
[SWS_UCM_00186]	
[SWS_UCM_00187]	
[SWS_UCM_00190]	Reinstallation of older `Software Cluster` version than previously removed
[SWS_UCM_00191]	`Software Cluster` life-cycle state `kAdded`
[SWS_UCM_00192]	`Software Cluster` life-cycle state transition from `kAdded` to `kPresent`
[SWS_UCM_00193]	`Software Cluster` life-cycle state transition from `kUpdated` to `kPresent`
[SWS_UCM_00194]	`Software Cluster` life-cycle state transition from `kRemoved` to `kPresent`
[SWS_UCM_00195]	`Software Cluster` life-cycle state `kUpdated`
[SWS_UCM_00196]	`Software Cluster` life-cycle state `kRemoved`
[SWS_UCM_00197]	End of `Software Cluster` life-cycle state from state `kAdded`
[SWS_UCM_00198]	End of `Software Cluster` life-cycle state from state `kRemoved`
[SWS_UCM_00199]	Reporting of `Software Cluster` reaching end of life-cycle
[SWS_UCM_00200]	Failing authentication
[SWS_UCM_00201]	Delta Package dependency error
[SWS_UCM_00202]	Trusted Platform compliance
[SWS_UCM_00203]	`TransferData` InvalidTransferId
[SWS_UCM_00204]	`TransferData` IncorrectBlock
[SWS_UCM_00205]	`TransferData` IncorrectSize

▽

△

Number	Heading
[SWS_UCM_00206]	`TransferData` InsufficientMemory
[SWS_UCM_00207]	`TransferData` BlockInconsistent
[SWS_UCM_00208]	`TransferData` OperationNotPermitted
[SWS_UCM_00209]	`TransferData` PackageInconsistent
[SWS_UCM_00211]	`TransferData` TransferInterrupted
[SWS_UCM_00212]	`TransferExit` InvalidTransferId
[SWS_UCM_00213]	`TransferExit` InvalidPackageManifest
[SWS_UCM_00214]	`DeleteTransfer` InvalidTransferId
[SWS_UCM_00215]	`DeleteTransfer` OperationNotPermitted
[SWS_UCM_00216]	Validity of TransferId
[SWS_UCM_00217]	`ProcessSwPackage` InsufficientMemory
[SWS_UCM_00218]	`ProcessSwPackage` InvalidTransferId
[SWS_UCM_00219]	`ProcessSwPackage` OperationNotPermitted
[SWS_UCM_00220]	`GetSwProcessProgress` InvalidTransferId
[SWS_UCM_00230]	`ProcessSwPackage` AuthenticationFailed
[SWS_UCM_00231]	`ProcessSwPackage` IncompatibleDelta
[SWS_UCM_00232]	`ProcessSwPackage`
[SWS_UCM_00233]	`Cancel` Operation CancelFailed
[SWS_UCM_00234]	`Cancel` OperationNotPermitted
[SWS_UCM_00235]	`Cancel` InvalidTransferId
[SWS_UCM_00236]	`RevertProcessedSwPackages` NotAbleToRevertPackages
[SWS_UCM_00237]	`RevertProcessedSwPackages` OperationNotPermitted
[SWS_UCM_00238]	`Rollback` NotAbleToRollback
[SWS_UCM_00239]	`Rollback` OperationNotPermitted
[SWS_UCM_00240]	`Finish` OperationNotPermitted
[SWS_UCM_00241]	`Activate` OperationNotPermitted
[SWS_UCM_00242]	`Activate` PreActivationFailed
[SWS_UCM_00243]	Too big block size received by UCM
[SWS_UCM_00245]	Software Cluster category
[SWS_UCM_00250]	`TransferData` AuthenticationFailed
[SWS_UCM_00251]	
[SWS_UCM_00252]	
[SWS_UCM_00253]	
[SWS_UCM_00254]	
[SWS_UCM_00255]	
[SWS_UCM_00256]	
[SWS_UCM_00257]	Update session
[SWS_UCM_00258]	Update session rejected
[SWS_UCM_00259]	Ending the update session

▽

△

Number	Heading
[SWS_UCM_00260]	PrepareUpdate, VerifyUpdate and PrepareRollback orders
[SWS_UCM_00261]	PrepareUpdate, VerifyUpdate and PrepareRollback synchronous calls
[SWS_UCM_00262]	Update preparation rejected
[SWS_UCM_00263]	Update preparation failure
[SWS_UCM_00264]	Update verification rejected
[SWS_UCM_01011]	`TransferVehiclePackage` InsufficientMemory
[SWS_UCM_01012]	`TransferVehiclePackage` InsufficientComputationPower
[SWS_UCM_01013]	Too big block size received by `UCM Master`
[SWS_UCM_01014]	Packages transferring sequence
[SWS_UCM_01015]	Invalid Vehicle Package manifest
[SWS_UCM_01016]	Invalid Package Manifest
[SWS_UCM_01017]	RequestedPackage field
[SWS_UCM_01117]	UCM Master `SafetyState` field
[SWS_UCM_01118]	UCM Master waiting for vehicle driver approval
[SWS_UCM_01119]	Report information of `Software Packages`
[SWS_UCM_01120]	Provide `Software Packages` general information
[SWS_UCM_01121]	Adaptive Platform interface provided for Flashing Adapter
[SWS_UCM_01122]	Supported physical layers by `D-PDU API` implementation
[SWS_UCM_01123]	Supported application layers by `D-PDU API` implementation
[SWS_UCM_01124]	Supported protocols by `D-PDU API` implementation
[SWS_UCM_01125]	Separation of `D-PDU API`-Software with the `MVCI` protocol module firmware
[SWS_UCM_01126]	Root description file (`RDF`)
[SWS_UCM_01127]	Module Description File (`MDF`)
[SWS_UCM_01128]	Symbolic names and IDs
[SWS_UCM_01129]	SAE J2534-1 and RP 1210a compatibility
[SWS_UCM_01130]	ComPrimitives in RawMode
[SWS_UCM_01131]	PDUIoCtl(PDU_IOCTL_RESET)
[SWS_UCM_01132]	PDUIoCtl(PDU_IOCTL_START_MSG_FILTER), PDUIoCtl(PDU_IOCTL_CLEAR_MSG_FILTER), PDUIoCtl(PDU_IOCTL_STOP_MSG_FILTER)
[SWS_UCM_01133]	PDUIoCtl(PDU_IOCTL_SEND_BREAK)
[SWS_UCM_01134]	Not used `D-PDU API` function return codes
[SWS_UCM_01178]	
[SWS_UCM_01265]	TransferState field
[SWS_UCM_01266]	Subordinate Not Available On The Network
[SWS_UCM_01267]	Vehicle State Manager Communication Error
[SWS_UCM_01268]	Vehicle Driver Interface Communication Error
[SWS_UCM_01269]	Campaign cancellation history
[SWS_UCM_01270]	New campaign disabling
[SWS_UCM_01271]	New campaign enabling

▽

△

Number	Heading
[SWS_UCM_01305]	`Vehicle Package` format
[SWS_UCM_01306]	`TransferExit` Invalid package manifest
[SWS_UCM_CONSTR_00002]	UCM confidential information handling
[SWS_UCM_CONSTR_00003]	Exclusive use of Vehicle Driver Interface
[SWS_UCM_CONSTR_00004]	Unsupported safety policy by Vehicle driver interface
[SWS_UCM_CONSTR_00005]	Safety state change
[SWS_UCM_CONSTR_00006]	Exclusive use of Vehicle State Manager
[SWS_UCM_CONSTR_00007]	Unsupported safety policy by Vehicle State Manager
[SWS_UCM_CONSTR_00008]	Switching vehicle into update mode
[SWS_UCM_CONSTR_00009]	Safety policy change
[SWS_UCM_CONSTR_00010]	UCM Client update sequence
[SWS_UCM_CONSTR_00011]	Flashing Adapter provided interface

Table E.4: Added Specification Items in R20-11

E.2.2 Changed Specification Items in R20-11

Number	Heading
[SWS_UCM_00018]	Providing Progress Information
[SWS_UCM_00020]	Finishing the packages activation
[SWS_UCM_00025]	Activation of `SoftwareCluster`s
[SWS_UCM_00026]	Dependency Check
[SWS_UCM_00027]	Delta Package activation
[SWS_UCM_00028]	`Software Package` Authentication
[SWS_UCM_00029]	Consistency Check of Manifest
[SWS_UCM_00031]	
[SWS_UCM_00032]	
[SWS_UCM_00038]	
[SWS_UCM_00039]	

▽

△

Number	Heading
[SWS_UCM_00040]	
[SWS_UCM_00044]	
[SWS_UCM_00069]	Report information on `Software Packages`
[SWS_UCM_00071]	
[SWS_UCM_00073]	
[SWS_UCM_00077]	
[SWS_UCM_00078]	
[SWS_UCM_00079]	
[SWS_UCM_00084]	Entering the kActivating state of Package Management
[SWS_UCM_00085]	Entering the kActivated state of Package Management
[SWS_UCM_00088]	Preparation of data transfer
[SWS_UCM_00092]	Software Package integrity
[SWS_UCM_00098]	`Software Package` Authentication failure
[SWS_UCM_00107]	Activated state
[SWS_UCM_00110]	Rolling-back the software update
[SWS_UCM_00111]	Entering the kRolled-Back state
[SWS_UCM_00112]	`Software Cluster` and version
[SWS_UCM_00115]	History
[SWS_UCM_00126]	Entering the kRolling-Back state after a Rollback call
[SWS_UCM_00130]	`Software Cluster` and version error
[SWS_UCM_00131]	
[SWS_UCM_00132]	
[SWS_UCM_00133]	
[SWS_UCM_00134]	
[SWS_UCM_00135]	
[SWS_UCM_00136]	
[SWS_UCM_00137]	Processing several update `Software Packages`
[SWS_UCM_00145]	Sequential order of data transfer
[SWS_UCM_00147]	Return to the Idle state from Cleaning-up state
[SWS_UCM_00148]	Transfer sequence order
[SWS_UCM_00149]	Return to the Idle state from Processing state
[SWS_UCM_00151]	Entering the Ready state of Package Management after a Cancel call
[SWS_UCM_00153]	Action in kActivating state of Package Management
[SWS_UCM_00154]	Entering the Verifying state of Package Management
[SWS_UCM_00155]	Entering the kRolling-Back state after a failure in the kVerifying state
[SWS_UCM_00158]	Cleanup of interrupted actions
[SWS_UCM_00162]	Entering the Cleaning-up state after a `RevertProcessedSwPackages` call
[SWS_UCM_00165]	Processing from stream
[SWS_UCM_00166]	Processing from stream state

▽

△

Number	Heading
[SWS_UCM_00167]	Cancelling streamed packages
[SWS_UCM_00168]	Transferring while processing from stream
[SWS_UCM_00169]	Finishing transfer while processing from stream
[SWS_UCM_00173]	
[SWS_UCM_00174]	
[SWS_UCM_00175]	
[SWS_UCM_00176]	
[SWS_UCM_00177]	
[SWS_UCM_00178]	
[SWS_UCM_00179]	
[SWS_UCM_00180]	
[SWS_UCM_00181]	
[SWS_UCM_00182]	
[SWS_UCM_00183]	
[SWS_UCM_00210]	Transferring of software packages on `kProcessing` state
[SWS_UCM_01003]	`UCM Master` checks states of `UCM` subordinates
[SWS_UCM_01006]	Start transfer of a `Vehicle Package` to `UCM Master`
[SWS_UCM_01007]	Start transfer of a `Software Package` to `UCM Master`
[SWS_UCM_01008]	Transfer data of a `Vehicle Package` or `Software Package` to `UCM Master`
[SWS_UCM_01009]	Exit the transfer of a `Vehicle Package` or `Software Package` to `UCM Master`
[SWS_UCM_01010]	Delete a `Vehicle Package` transferred to `UCM Master`
[SWS_UCM_01101]	Provide information of installed `Software Clusters` in vehicle
[SWS_UCM_01102]	Get information of available `Software Clusters` in `Backend`
[SWS_UCM_01103]	Inform `Backend` of needed `Software Clusters` for an update
[SWS_UCM_01105]	Interaction of `UCM Master` with Vehicle Driver
[SWS_UCM_01107]	`UCM Master` provides progress information to Vehicle Driver
[SWS_UCM_01109]	`UCM Master` provides a safety policy interface
[SWS_UCM_01110]	`UCM Master` SafetyState method
[SWS_UCM_01114]	
[SWS_UCM_01177]	
[SWS_UCM_01203]	CampaignState field
[SWS_UCM_01207]	Trigger on `kSoftwarePackage_Transferring` state
[SWS_UCM_01221]	Transition from `kVehiclePackageTransferring` state to `kIdle` state
[SWS_UCM_01222]	Transition from `kVehiclePackageTransferring` state to `kSoftwarePackage_Transferring` state
[SWS_UCM_01227]	Transition from `kSoftwarePackage_Transferring` state to `kIdle` state
[SWS_UCM_01228]	Transition from `kSoftwarePackage_Transferring` state to `kProcessing` state

▽

△

Number	Heading
[SWS_UCM_01229]	SafetyPolicy while processing stream
[SWS_UCM_01234]	Transition from `kProcessing` state to `kActivating` state
[SWS_UCM_01236]	Transition from `kProcessing` state to `kIdle` state
[SWS_UCM_01239]	Transition from `kActivating` state to `kCancelling` state
[SWS_UCM_01240]	Transition from `kActivating` state to `kVehicleChecking` state
[SWS_UCM_01244]	Cancellation of an update campaign shall be possible
[SWS_UCM_01245]	Cancellation during activation shall be possible
[SWS_UCM_01246]	Unreachable UCM during update campaign
[SWS_UCM_01247]	Method to read History Report
[SWS_UCM_01302]	`Vehicle Package` authentication failure
[SWS_UCM_01304]	Confidential information protection
[SWS_UCM_CONSTR_00001]	

Table E.5: Changed Specification Items in R20-11

E.2.3 Deleted Specification Items in R20-11

Number	Heading
[SWS_UCM_00011]	Updating persisted data
[SWS_UCM_00041]	LogLevelType table
[SWS_UCM_00042]	LogEntryType table
[SWS_UCM_00043]	LogVectorType table
[SWS_UCM_00082]	Exit from Processing state of Package Management
[SWS_UCM_00091]	Successful data transfer
[SWS_UCM_00096]	Entering the Rolled-back state
[SWS_UCM_00102]	Update state
[SWS_UCM_00105]	UCM confidential information handling
[SWS_UCM_00108]	Execution of the update software
[SWS_UCM_00113]	Rollback of persisted data
[SWS_UCM_00124]	Verify State
[SWS_UCM_00128]	
[SWS_UCM_00141]	`UCM` insufficient memory for parallel data transfer
[SWS_UCM_00142]	Prevent software from blocking the Rollback operation
[SWS_UCM_00143]	Log level setting
[SWS_UCM_00156]	Procurement of Checksum
[SWS_UCM_00170]	Log message retrieving

▽

△

Number	Heading
[SWS_UCM_00171]	Log level changing
[SWS_UCM_00172]	Log messages removing
[SWS_UCM_01002]	UCM Master shall provide UCM services
[SWS_UCM_01106]	Exclusive use of Vehicle Driver Interface
[SWS_UCM_01108]	Unsupported safety policy by Vehicle driver interface
[SWS_UCM_01111]	Exclusive use of Vehicle State Manager
[SWS_UCM_01112]	Unsupported safety policy by Vehicle State Manager
[SWS_UCM_01113]	Switching vehicle into update mode
[SWS_UCM_01115]	VehicleStateManagerErrorDomain
[SWS_UCM_01116]	VehicleDriverApplicationErrorDomain
[SWS_UCM_01206]	Trigger on kTransferApproving state
[SWS_UCM_01208]	Trigger on kProcessApproving state
[SWS_UCM_01211]	Trigger on kActivateApproving state
[SWS_UCM_01223]	Transition from kVehiclePackageTransferring state to kTransferApproving state
[SWS_UCM_01224]	Transition from kTransferApproving state to kTransferring state
[SWS_UCM_01225]	Transition from kTransferApproving state to kIdle state
[SWS_UCM_01226]	Transition from kTransferring state to kTransferApproving state
[SWS_UCM_01230]	Transition from kTransferring state to kProcessApproving state
[SWS_UCM_01231]	Transition from kProcessApproving state to kProcessing state
[SWS_UCM_01232]	Transition from kProcessApproving state to kIdle state
[SWS_UCM_01233]	Transition from kProcessing state to kProcessApproving state
[SWS_UCM_01235]	Transition from kProcessing state to kActivateApproving state
[SWS_UCM_01237]	Transition from kActivateApproving state to kActivating state
[SWS_UCM_01238]	Transition from kActivateApproving state to kIdle state

Table E.6: Deleted Specification Items in R20-11

E.2.4 Added Constraints in R20-11

none

E.2.5 Changed Constraints in R20-11

none

E.2.6 Deleted Constraints in R20-11

none

E.3 Constraint and Specification Item History of this document according to AUTOSAR Release R21-11.

E.3.1 Added Specification Items in R21-11

Number	Heading
[SWS_UCM_00265]	state transition due to `ProcessSwPackage` error
[SWS_UCM_00266]	OperationNotPermitted error and UCM state
[SWS_UCM_00267]	Error when checksum is not recognised at processing time
[SWS_UCM_00268]	
[SWS_UCM_00269]	
[SWS_UCM_00270]	UCM internal state persistency
[SWS_UCM_00271]	Keeping history of failure error code
[SWS_UCM_00272]	Transfer block size
[SWS_UCM_00273]	Persistent data clean-up after Software Cluster update that removes a process
[SWS_UCM_00274]	`UCM` initialization
[SWS_UCM_00275]	`TransferData` error handling order
[SWS_UCM_00276]	`TransferExit` error handling order
[SWS_UCM_00277]	`ProcessSwPackage` error handling order
[SWS_UCM_00278]	`Cancel` error handling order
[SWS_UCM_00279]	`RevertProcessedSwPackages` error handling order
[SWS_UCM_00280]	`Activate` VerificationFailed
[SWS_UCM_00281]	`Activate` error handling order
[SWS_UCM_00282]	`Rollback` error handling order
[SWS_UCM_00283]	`DeleteTransfer` error handling order
[SWS_UCM_00285]	Removing or updating a `Software Cluster` not existing in the `Machine`
[SWS_UCM_00286]	`Software Cluster` life-cycle state transition from `kRemoved` to `kPresent` in case of `Finish` call
[SWS_UCM_00287]	End of `Software Cluster` life-cycle state from state `kAdded` in case of `Finish` call
[SWS_UCM_00288]	
[SWS_UCM_00289]	`TransferData` TransferFailed
[SWS_UCM_01018]	`TransferVehiclePackage` `BusyWithCampaign`
[SWS_UCM_01019]	`UCM Master` initialization
[SWS_UCM_01135]	Get Software Clusters descriptions from a vehicle

▽

△

Number	Heading
[SWS_UCM_01136]	
[SWS_UCM_01137]	
[SWS_UCM_01138]	
[SWS_UCM_01272]	`VehicleCheck` call not permitted
[SWS_UCM_01273]	`CancelCampaign` `CancelFailed` error
[SWS_UCM_01274]	`CancelCampaign` `OperationNotPermitted` error
[SWS_UCM_-CONSTR_00012]	
[SWS_UCM_-CONSTR_00013]	Confidential information protection
[SWS_UCM_-CONSTR_00014]	Software Package and Software Cluster shortNames
[SWS_UCM_-CONSTR_00015]	Trigger on `kVehicleChecking` state

Table E.7: Added Specification Items in R21-11

E.3.2 Changed Specification Items in R21-11

Number	Heading
[SWS_UCM_00004]	Report software information
[SWS_UCM_00009]	`UCM` exposing its identifier
[SWS_UCM_00017]	Sequential `Software Package` Processing
[SWS_UCM_00020]	Finishing the packages activation
[SWS_UCM_00030]	Report changes
[SWS_UCM_00039]	
[SWS_UCM_00044]	
[SWS_UCM_00078]	
[SWS_UCM_00080]	Idle state of Package Management
[SWS_UCM_00081]	Processing state of Package Management
[SWS_UCM_00083]	Entering the Ready state of Package Management after a successful processing operation
[SWS_UCM_00084]	Entering the kActivating state of Package Management
[SWS_UCM_00085]	Entering the kActivated state of Package Management
[SWS_UCM_00092]	Software Package integrity
[SWS_UCM_00103]	Update to older `Software Cluster` version than currently present
[SWS_UCM_00104]	Integrity Check of processed Package
[SWS_UCM_00107]	Activated state

▽

△

Number	Heading
[SWS_UCM_00110]	Rolling-back the software update
[SWS_UCM_00111]	Entering the `kRollingBack` state
[SWS_UCM_00115]	History
[SWS_UCM_00126]	Entering the `kRollingBack` state after a Rollback call
[SWS_UCM_00127]	Finishing update sequence
[SWS_UCM_00130]	`Software Cluster` and version error
[SWS_UCM_00131]	
[SWS_UCM_00133]	
[SWS_UCM_00134]	
[SWS_UCM_00136]	
[SWS_UCM_00146]	Entering the Cleaning-up state after a Finish call
[SWS_UCM_00147]	Return to the Idle state from Cleaning-up state
[SWS_UCM_00149]	Return to the Idle state from Processing state
[SWS_UCM_00151]	Entering the Ready state of Package Management after a Cancel call
[SWS_UCM_00152]	Entering the Ready state of Package Management after a missing dependency
[SWS_UCM_00153]	Action in kActivating state of Package Management
[SWS_UCM_00154]	Entering the Verifying state of Package Management
[SWS_UCM_00155]	Entering the kRolling-Back state after a failure in the kVerifying state
[SWS_UCM_00162]	Entering the Cleaning-up state after a `RevertProcessedSwPackages` call
[SWS_UCM_00163]	Action in Cleaning-up state
[SWS_UCM_00164]	Cleaning up of Software Packages
[SWS_UCM_00166]	Processing from stream state
[SWS_UCM_00167]	Cancelling streamed packages
[SWS_UCM_00168]	Transferring while processing from stream
[SWS_UCM_00169]	Finishing transfer while processing from stream
[SWS_UCM_00176]	
[SWS_UCM_00181]	
[SWS_UCM_00182]	
[SWS_UCM_00183]	
[SWS_UCM_00185]	Provide `SoftwareCluster` general information
[SWS_UCM_00186]	
[SWS_UCM_00190]	Reinstallation of older `Software Cluster` version than previously removed
[SWS_UCM_00191]	`Software Cluster` life-cycle state `kAdded`
[SWS_UCM_00192]	`Software Cluster` life-cycle state transition from `kAdded` to `kPresent`
[SWS_UCM_00193]	`Software Cluster` life-cycle state transition from `kUpdated` to `kPresent`
[SWS_UCM_00194]	`Software Cluster` life-cycle state transition from `kRemoved` to `kPresent` in case of `RevertProcessedSwPackages` call

▽

△

Number	Heading
[SWS_UCM_00195]	`Software Cluster` life-cycle state `kUpdated`
[SWS_UCM_00196]	`Software Cluster` life-cycle state `kRemoved`
[SWS_UCM_00197]	End of `Software Cluster` life-cycle state from state `kAdded` in case of `RevertProcessedSwPackages` call
[SWS_UCM_00198]	End of `Software Cluster` life-cycle state from state `kRemoved`
[SWS_UCM_00200]	Failing authentication
[SWS_UCM_00209]	`TransferData` PackageInconsistent
[SWS_UCM_00210]	Transferring of software packages on `kProcessing` state
[SWS_UCM_00213]	`TransferExit` `InvalidPackageManifest`
[SWS_UCM_00214]	`DeleteTransfer` InvalidTransferId
[SWS_UCM_00215]	`DeleteTransfer` OperationNotPermitted
[SWS_UCM_00220]	`GetSwProcessProgress` InvalidTransferId
[SWS_UCM_00237]	`RevertProcessedSwPackages` OperationNotPermitted
[SWS_UCM_00239]	`Rollback` OperationNotPermitted
[SWS_UCM_00240]	`Finish` OperationNotPermitted
[SWS_UCM_00241]	`Activate` OperationNotPermitted
[SWS_UCM_00242]	`Activate` PreActivationFailed
[SWS_UCM_00243]	Too big block size received by UCM
[SWS_UCM_00251]	
[SWS_UCM_00252]	
[SWS_UCM_00253]	
[SWS_UCM_00254]	
[SWS_UCM_00255]	
[SWS_UCM_00257]	Update session
[SWS_UCM_00258]	Update session rejected
[SWS_UCM_00259]	Ending the update session
[SWS_UCM_00260]	PrepareUpdate, VerifyUpdate and PrepareRollback orders
[SWS_UCM_00261]	PrepareUpdate, VerifyUpdate and PrepareRollback synchronous calls
[SWS_UCM_00262]	Update preparation rejected
[SWS_UCM_00263]	Update preparation failure
[SWS_UCM_00264]	Update verification rejected
[SWS_UCM_01003]	`UCM Master` checks states of `UCM` subordinates
[SWS_UCM_01011]	`TransferVehiclePackage` InsufficientMemory
[SWS_UCM_01015]	Invalid Vehicle Package manifest
[SWS_UCM_01016]	Invalid Package Manifest
[SWS_UCM_01103]	Inform `Backend` of needed `Software Packages` for an update
[SWS_UCM_01109]	`UCM Master` provides a safety interface
[SWS_UCM_01114]	
[SWS_UCM_01117]	UCM Master `SafetyState` field

▽

△

Number	Heading
[SWS_UCM_01118]	UCM Master waiting for vehicle driver approval
[SWS_UCM_01203]	CampaignState field
[SWS_UCM_01204]	Initial state
[SWS_UCM_01207]	Trigger on `kSoftwarePackage_Transferring` state
[SWS_UCM_01209]	Trigger on `kProcessing` state
[SWS_UCM_01212]	Trigger on `kActivating` state
[SWS_UCM_01214]	Final action on `kVehicleChecking` state
[SWS_UCM_01215]	Trigger on `kCancelling` state
[SWS_UCM_01216]	Final action on `kCancelling` state
[SWS_UCM_01217]	Monitoring of UCM subordinates
[SWS_UCM_01218]	Transition from `kIdle` state to `kSyncing` state
[SWS_UCM_01219]	Transition from `kSyncing` state to `kIdle` state
[SWS_UCM_01220]	Transition from `kIdle` state to `kVehiclePackageTransferring` state
[SWS_UCM_01221]	Transition from `kVehiclePackageTransferring` state to `kIdle` state
[SWS_UCM_01222]	Transition from `kVehiclePackageTransferring` state to `kSoftwarePackage_Transferring` state
[SWS_UCM_01227]	Transition from `kSoftwarePackage_Transferring` state to `kIdle` state
[SWS_UCM_01228]	Transition from `kSoftwarePackage_Transferring` state to `kProcessing` state
[SWS_UCM_01229]	SafetyConditions while processing stream
[SWS_UCM_01234]	Transition from `kProcessing` state to `kActivating` state
[SWS_UCM_01236]	Transition from `kProcessing` state to `kCancelling` state
[SWS_UCM_01239]	Transition from `kActivating` state to `kCancelling` state
[SWS_UCM_01240]	Transition from `kActivating` state to `kVehicleChecking` state
[SWS_UCM_01241]	Transition from `kVehicleChecking` state to `kCancelling` state
[SWS_UCM_01242]	Transition from `kVehicleChecking` state to `kIdle` state
[SWS_UCM_01243]	Transition from `kCancelling` state to `kIdle` state
[SWS_UCM_01244]	Cancellation of an update campaign shall be possible
[SWS_UCM_01246]	Unreachable UCM during update campaign
[SWS_UCM_01247]	Method to read History Report
[SWS_UCM_01265]	TransferState field
[SWS_UCM_01270]	New campaign disabling
[SWS_UCM_-CONSTR_00002]	UCM confidential information handling
[SWS_UCM_-CONSTR_00004]	Unsupported safety by Vehicle driver interface
[SWS_UCM_-CONSTR_00005]	Safety state change
[SWS_UCM_-CONSTR_00006]	Exclusive use of Vehicle State Manager

▽

△

Number	Heading
[SWS_UCM_-CONSTR_00007]	Unsupported safety conditions by Vehicle State Manager
[SWS_UCM_-CONSTR_00008]	Switching vehicle into update mode
[SWS_UCM_-CONSTR_00009]	Safety condition change

Table E.8: Changed Specification Items in R21-11

E.3.3 Deleted Specification Items in R21-11

Number	Heading
[SWS_UCM_00093]	Transfer sequence
[SWS_UCM_00201]	Delta Package dependency error
[SWS_UCM_00211]	`TransferData` TransferInterrupted
[SWS_UCM_00230]	`ProcessSwPackage` AuthenticationFailed
[SWS_UCM_00232]	`ProcessSwPackage`
[SWS_UCM_00233]	`Cancel` Operation CancelFailed
[SWS_UCM_00250]	`TransferData` AuthenticationFailed
[SWS_UCM_01001]	`UCM Master` processes `Vehicle Package`
[SWS_UCM_01004]	Only one `UCM Master` shall be active per network domain
[SWS_UCM_01006]	Start transfer of a `Vehicle Package` to `UCM Master`
[SWS_UCM_01007]	Start transfer of a `Software Package` to `UCM Master`
[SWS_UCM_01008]	Transfer data of a `Vehicle Package` or `Software Package` to `UCM Master`
[SWS_UCM_01009]	Exit the transfer of a `Vehicle Package` or `Software Package` to `UCM Master`
[SWS_UCM_01010]	Delete a `Vehicle Package` transferred to `UCM Master`
[SWS_UCM_01012]	`TransferVehiclePackage` InsufficientComputationPower
[SWS_UCM_01102]	Get information of available `Software Clusters` in `Backend`
[SWS_UCM_01213]	Trigger on `kVehicleChecking` state
[SWS_UCM_01245]	Cancellation during activation shall be possible
[SWS_UCM_01304]	Confidential information protection
[SWS_UCM_-CONSTR_00010]	UCM Client update sequence

Table E.9: Deleted Specification Items in R21-11

E.3.4 Added Constraints in R21-11

none

E.3.5 Changed Constraints in R21-11

none

E.3.6 Deleted Constraints in R21-11

none

E.4 Constraint and Specification Item History of this document according to AUTOSAR Release R22-11.

E.4.1 Added Specification Items in R22-11

Number	Heading
[SWS_UCM_00290]	
[SWS_UCM_00291]	
[SWS_UCM_00292]	History elements ordering
[SWS_UCM_00293]	`VerifyUpdate` method
[SWS_UCM_00294]	Unsupported package format for UCM
[SWS_UCM_00296]	
[SWS_UCM_00297]	Retry Strategy for ServiceBusy
[SWS_UCM_00298]	Retry Strategy for UpdateSessionRejected
[SWS_UCM_00299]	Verify rolled back Software Clusters
[SWS_UCM_00300]	Software Cluster failing to rollback
[SWS_UCM_00301]	Retry ro Rollback again when UCM is in kRollingBackFailed state
[SWS_UCM_00302]	Rollback failing is triggering production error
[SWS_UCM_00303]	failing to record history
[SWS_UCM_01020]	Retry Strategy for BlockInconsistent
[SWS_UCM_01275]	Safety conditions during activation
[SWS_UCM_01307]	`Vehicle Package` format not supported
[SWS_UCM_01308]	Check Vehicle Package version compatibility against UCM Master version
[SWS_UCM_-CONSTR_00016]	OTA Client use of RequestedPackage field
[SWS_UCM_-CONSTR_00017]	Interaction of `UCM Master` with Vehicle Driver

Table E.10: Added Specification Items in R22-11

E.4.2 Changed Specification Items in R22-11

Number	Heading
[SWS_UCM_00001]	Starting the package processing
[SWS_UCM_00003]	Cancelling the package processing
[SWS_UCM_00004]	Report software information
[SWS_UCM_00005]	Rollback to the software prior to Finish the update process
[SWS_UCM_00008]	Executing the data transfer
[SWS_UCM_00018]	Providing Progress Information
[SWS_UCM_00022]	Activation of `Software Clusters`
[SWS_UCM_00025]	Activation of `SoftwareCluster`s
[SWS_UCM_00026]	Dependency Check
[SWS_UCM_00027]	Delta Package version applicability
[SWS_UCM_00030]	Report changes
[SWS_UCM_00031]	
[SWS_UCM_00032]	
[SWS_UCM_00038]	
[SWS_UCM_00039]	
[SWS_UCM_00040]	
[SWS_UCM_00044]	
[SWS_UCM_00069]	Report information on `Software Packages`
[SWS_UCM_00071]	
[SWS_UCM_00073]	
[SWS_UCM_00077]	
[SWS_UCM_00078]	
[SWS_UCM_00079]	
[SWS_UCM_00085]	Entering the kActivated state of Package Management
[SWS_UCM_00087]	Insufficient amount of data transferred
[SWS_UCM_00092]	Software Package integrity
[SWS_UCM_00098]	`Software Package` Authentication failure
[SWS_UCM_00099]	Update of `Adaptive Application`
[SWS_UCM_00103]	Update to older `Software Cluster` version than currently present and than previously removed
[SWS_UCM_00104]	Integrity Check of processed Package
[SWS_UCM_00107]	Activated state
[SWS_UCM_00111]	Entering the `kRollingBack` state
[SWS_UCM_00120]	Runtime dependencies check
[SWS_UCM_00131]	
[SWS_UCM_00132]	
[SWS_UCM_00133]	
[SWS_UCM_00134]	

△

Number	Heading
[SWS_UCM_00135]	
[SWS_UCM_00136]	
[SWS_UCM_00137]	Processing several update `Software Packages`
[SWS_UCM_00151]	Entering the Ready state of Package Management after a Cancel call
[SWS_UCM_00154]	Entering the Verifying state of Package Management
[SWS_UCM_00155]	Entering the kRolling-Back state after a failure in the kVerifying state
[SWS_UCM_00160]	Processing results records
[SWS_UCM_00161]	Check Software Package version compatibility against `UCM` version
[SWS_UCM_00167]	Cancelling streamed packages
[SWS_UCM_00173]	
[SWS_UCM_00175]	
[SWS_UCM_00176]	
[SWS_UCM_00177]	
[SWS_UCM_00178]	
[SWS_UCM_00179]	
[SWS_UCM_00180]	
[SWS_UCM_00181]	
[SWS_UCM_00182]	
[SWS_UCM_00183]	
[SWS_UCM_00184]	Persistent data clean-up after Software Cluster removal
[SWS_UCM_00185]	Provide `SoftwareCluster` general information
[SWS_UCM_00186]	
[SWS_UCM_00187]	
[SWS_UCM_00193]	`Software Cluster` life-cycle state transition from `kUpdating` to `kPresent`
[SWS_UCM_00195]	`Software Cluster` life-cycle state `kUpdating`
[SWS_UCM_00200]	Failing authentication
[SWS_UCM_00202]	Trusted Platform compliance
[SWS_UCM_00203]	`TransferData` InvalidTransferId
[SWS_UCM_00204]	`TransferData` IncorrectBlock
[SWS_UCM_00205]	`TransferData` IncorrectSize
[SWS_UCM_00206]	`TransferData` InsufficientMemory
[SWS_UCM_00207]	`TransferData` BlockInconsistent
[SWS_UCM_00208]	`TransferData` OperationNotPermitted
[SWS_UCM_00219]	`ProcessSwPackage` OperationNotPermitted
[SWS_UCM_00231]	`ProcessSwPackage` IncompatibleDelta
[SWS_UCM_00242]	`Activate` PrepareUpdateFailed
[SWS_UCM_00245]	Software Cluster category
[SWS_UCM_00251]	

▽

△

Number	Heading
[SWS_UCM_00252]	
[SWS_UCM_00253]	
[SWS_UCM_00254]	
[SWS_UCM_00255]	
[SWS_UCM_00256]	
[SWS_UCM_00257]	Update session
[SWS_UCM_00258]	Update session rejected
[SWS_UCM_00262]	Update preparation rejected
[SWS_UCM_00263]	Update preparation failure
[SWS_UCM_00264]	Update verification rejected
[SWS_UCM_00265]	state transition due to `ProcessSwPackage` error
[SWS_UCM_00266]	OperationNotPermitted error and UCM state
[SWS_UCM_00268]	
[SWS_UCM_00269]	
[SWS_UCM_00270]	UCM internal state persistency
[SWS_UCM_00272]	Transfer block size
[SWS_UCM_00273]	Persistent data clean-up after Software Cluster update that removes a process
[SWS_UCM_00274]	`UCM` initialization
[SWS_UCM_00275]	`TransferData` error handling order
[SWS_UCM_00276]	`TransferExit` error handling order
[SWS_UCM_00277]	`ProcessSwPackage` error handling order
[SWS_UCM_00280]	`Activate` VerificationFailed
[SWS_UCM_00281]	`Activate` error handling order
[SWS_UCM_00282]	`Rollback` error handling order
[SWS_UCM_00285]	Removing or updating a `Software Cluster` not existing in the `Machine`
[SWS_UCM_00286]	`Software Cluster` life-cycle state transition from `kRemoved` to `kPresent` in case of `Finish` call
[SWS_UCM_00287]	End of `Software Cluster` life-cycle state from state `kAdded` in case of `Finish` call
[SWS_UCM_00288]	
[SWS_UCM_01017]	RequestedPackage field
[SWS_UCM_01018]	`TransferVehiclePackage` `BusyWithCampaign`
[SWS_UCM_01101]	Provide information of installed `Software Clusters` in vehicle
[SWS_UCM_01105]	Interaction of `UCM Master` with Vehicle Driver
[SWS_UCM_01109]	`UCM Master` provides a safety interface
[SWS_UCM_01114]	
[SWS_UCM_01117]	UCM Master `SafetyState` field
[SWS_UCM_01118]	UCM Master waiting for vehicle driver approval
[SWS_UCM_01119]	Report information of `Software Packages`

▽

△

Number	Heading
[SWS_UCM_01120]	Provide `Software Packages` general information
[SWS_UCM_01135]	Get Software Clusters descriptions from a vehicle
[SWS_UCM_01136]	
[SWS_UCM_01137]	
[SWS_UCM_01138]	
[SWS_UCM_01177]	
[SWS_UCM_01178]	
[SWS_UCM_01203]	CampaignState field
[SWS_UCM_01212]	Trigger on `kActivating` state
[SWS_UCM_01214]	Final action on `kVehicleChecking` state
[SWS_UCM_01215]	Trigger on `kCancelling` state
[SWS_UCM_01216]	Final action on `kCancelling` state
[SWS_UCM_01218]	Transition from `kIdle` state to `kSyncing` state
[SWS_UCM_01219]	Transition from `kSyncing` state to `kIdle` state
[SWS_UCM_01220]	Transition from `kIdle` state to `kVehiclePackageTransferring` and `kTransferring` states
[SWS_UCM_01221]	Transition from `kVehiclePackageTransferring` state and `kTransferring` state to `kCancelling` state
[SWS_UCM_01227]	Transition from `kSoftwarePackage_Transferring` state and `kTransferring` state to `kCancelling` state
[SWS_UCM_01228]	Transition from `kSoftwarePackage_Transferring` state and `kTransferring` state to `kProcessing` state and `kUpdating` state
[SWS_UCM_01234]	Transition from `kProcessing` state to `kActivating` state
[SWS_UCM_01236]	Transition from `kProcessing` state and `kUpdating` state to `kCancelling` state
[SWS_UCM_01239]	Transition from `kActivating` state and `kUpdating` state to `kCancelling` state
[SWS_UCM_01241]	Transition from `kVehicleChecking` state and `kUpdating` state to `kCancelling` state
[SWS_UCM_01242]	Transition from `kVehicleChecking` state and `kUpdating` state to `kIdle` state
[SWS_UCM_01243]	Transition from `kCancelling` state to `kIdle` state
[SWS_UCM_01244]	Cancellation of an update campaign shall be possible
[SWS_UCM_01246]	Unreachable UCM during update campaign
[SWS_UCM_01248]	Content of History Report
[SWS_UCM_01266]	Subordinate Not Available On The Network
[SWS_UCM_01267]	Vehicle State Manager Communication Error
[SWS_UCM_01268]	Vehicle Driver Interface Communication Error
[SWS_UCM_01271]	New campaign enabling
[SWS_UCM_01272]	`VehicleCheck` call not permitted
[SWS_UCM_01301]	`Vehicle Package` authentication

▽

△

Number	Heading
[SWS_UCM_01305]	`Vehicle Package` format
[SWS_UCM_01306]	`TransferExit` Invalid package manifest
[SWS_UCM_-CONSTR_00004]	Unsupported safety by Vehicle driver interface
[SWS_UCM_-CONSTR_00005]	Safety state change
[SWS_UCM_-CONSTR_00006]	Exclusive use of Vehicle State Manager
[SWS_UCM_-CONSTR_00007]	Unsupported safety conditions by Vehicle State Manager
[SWS_UCM_-CONSTR_00009]	Safety condition change
[SWS_UCM_-CONSTR_00013]	Confidential information protection
[SWS_UCM_-CONSTR_00015]	Trigger on `kVehicleChecking` state

Table E.11: Changed Specification Items in R22-11

E.4.3 Deleted Specification Items in R22-11

Number	Heading
[SWS_UCM_00009]	`UCM` exposing its identifier
[SWS_UCM_00028]	`Software Package` Authentication
[SWS_UCM_00086]	Unsupported method calls
[SWS_UCM_00174]	
[SWS_UCM_00209]	`TransferData` PackageInconsistent
[SWS_UCM_00238]	`Rollback` NotAbleToRollback
[SWS_UCM_01107]	`UCM Master` provides progress information to Vehicle Driver
[SWS_UCM_01110]	`UCM Master` SafetyState method

Table E.12: Deleted Specification Items in R22-11

E.4.4 Added Constraints in R22-11

none

E.4.5 Changed Constraints in R22-11

none

E.4.6 Deleted Constraints in R22-11

none

E.5 Constraint and Specification Item History of this document according to AUTOSAR Release R23-11.

E.5.1 Added Specification Items in R23-11

Number	Heading
[SWS_UCM_00305]	Persistent data uri at `Software Cluster` installation
[SWS_UCM_00306]	Persistent data uri change at update
[SWS_UCM_00309]	Definition of ImplementationDataType UCMIdentifierAndVersionType
[SWS_UCM_00311]	Provide `SoftwareCluster` general information
[SWS_UCM_00312]	Definition of ImplementationDataType SwClusterManifestInfoType
[SWS_UCM_00313]	Definition of ImplementationDataType DependencyVectorType
[SWS_UCM_00314]	Definition of ImplementationDataType DependencyType
[SWS_UCM_00315]	Definition of ImplementationDataType DependencyCompareConditionType
[SWS_UCM_00316]	Definition of ImplementationDataType DependencyOperatorType
[SWS_UCM_00317]	Definition of ImplementationDataType LogicalOperationType
[SWS_UCM_00318]	Definition of ImplementationDataType DependencyRoleType
[SWS_UCM_00319]	Semantic versionning
[SWS_UCM_00320]	Diagnostic Event: History recording failed
[SWS_UCM_00321]	Diagnostic Event: Update session with SM rejected
[SWS_UCM_00322]	Diagnostic Event: PrepareUpdate call to SM failed
[SWS_UCM_00323]	Diagnostic Event: RollBack failed
[SWS_UCM_00324]	Diagnostic Event: Verification with SM at activation failed
[SWS_UCM_00325]	Diagnostic Event: Campaign cancelling failed
[SWS_UCM_00326]	Diagnostic Event: Activation not possible because of missing dependencies
[SWS_UCM_00327]	Diagnostic Event: Installing old software is not allowed
[SWS_UCM_00329]	Activate `kPersistencyAllocationFailed`

Table E.13: Added Specification Items in R23-11

E.5.2 Changed Specification Items in R23-11

Number	Heading
[SWS_UCM_00026]	Dependency Check
[SWS_UCM_00038]	Definition of ImplementationDataType SwPackageStateType
[SWS_UCM_00039]	Definition of ImplementationDataType SwPackageInfoType
[SWS_UCM_00077]	Definition of ImplementationDataType SwClusterStateType
[SWS_UCM_00078]	Definition of ImplementationDataType SwClusterInfoType
[SWS_UCM_00131]	Definition of ServiceInterface PackageManagement
[SWS_UCM_00132]	Definition of ImplementationDataType ActionType
[SWS_UCM_00133]	Definition of ImplementationDataType ResultType
[SWS_UCM_00134]	Definition of ImplementationDataType HistoryType
[SWS_UCM_00135]	Definition of ImplementationDataType HistoryVectorType
[SWS_UCM_00136]	Definition of Application Error Domain of functional cluster UCM
[SWS_UCM_00161]	Check Software Package version compatibility against `UCM` version
[SWS_UCM_00175]	Definition of ImplementationDataType StrongRevisionLabelString
[SWS_UCM_00184]	Persistent data clean-up after Software Cluster removal
[SWS_UCM_00185]	Provide `SoftwareCluster` general information
[SWS_UCM_00203]	`TransferData` InvalidTransferId
[SWS_UCM_00236]	`RevertProcessedSwPackages` NotAbleToRevertPackages
[SWS_UCM_00245]	Software Cluster category
[SWS_UCM_00262]	Update preparation rejected
[SWS_UCM_00264]	Update verification rejected
[SWS_UCM_00266]	OperationNotPermitted error and UCM state
[SWS_UCM_00271]	Keeping history of failure error code
[SWS_UCM_00273]	Persistent data clean-up after Software Cluster update that removes a process
[SWS_UCM_00274]	`UCM` initialization
[SWS_UCM_00278]	`Cancel` error handling order
[SWS_UCM_00279]	`RevertProcessedSwPackages` error handling order
[SWS_UCM_00281]	`Activate` error handling order
[SWS_UCM_00292]	History elements ordering

Table E.14: Changed Specification Items in R23-11

E.5.3 Deleted Specification Items in R23-11

Number	Heading
[SWS_UCM_00120]	Runtime dependencies check
[SWS_UCM_00177]	
[SWS_UCM_00178]	
[SWS_UCM_00179]	
[SWS_UCM_00180]	
[SWS_UCM_00181]	
[SWS_UCM_00182]	
[SWS_UCM_00183]	
[SWS_UCM_00186]	
[SWS_UCM_00187]	
[SWS_UCM_00210]	Transferring of software packages on `kProcessing` state
[SWS_UCM_00251]	
[SWS_UCM_00252]	
[SWS_UCM_00253]	
[SWS_UCM_00254]	
[SWS_UCM_00255]	
[SWS_UCM_00256]	
[SWS_UCM_00268]	
[SWS_UCM_00269]	
[SWS_UCM_00290]	
[SWS_UCM_00291]	
[SWS_UCM_00296]	
[SWS_UCM_00297]	Retry Strategy for ServiceBusy
[SWS_UCM_00298]	Retry Strategy for UpdateSessionRejected
[SWS_UCM_01003]	`UCM Master` checks states of `UCM` subordinates
[SWS_UCM_01005]	`UCM Master` is discovering `UCM`s in vehicle
[SWS_UCM_01011]	`TransferVehiclePackage` InsufficientMemory
[SWS_UCM_01013]	Too big block size received by `UCM Master`
[SWS_UCM_01014]	Packages transferring sequence
[SWS_UCM_01015]	Invalid Vehicle Package manifest
[SWS_UCM_01016]	Invalid Package Manifest
[SWS_UCM_01017]	RequestedPackage field
[SWS_UCM_01018]	`TransferVehiclePackage BusyWithCampaign`
[SWS_UCM_01019]	`UCM Master` initialization
[SWS_UCM_01020]	Retry Strategy for BlockInconsistent
[SWS_UCM_01101]	Provide information of installed `Software Clusters` in vehicle

△

Number	Heading
[SWS_UCM_01103]	Inform Backend of needed Software Packages for an update
[SWS_UCM_01105]	Interaction of UCM Master with Vehicle Driver
[SWS_UCM_01109]	UCM Master provides a safety interface
[SWS_UCM_01114]	
[SWS_UCM_01117]	UCM Master SafetyState field
[SWS_UCM_01118]	UCM Master waiting for vehicle driver approval
[SWS_UCM_01119]	Report information of Software Packages
[SWS_UCM_01120]	Provide Software Packages general information
[SWS_UCM_01121]	Adaptive Platform interface provided for Flashing Adapter
[SWS_UCM_01122]	Supported physical layers by D-PDU API implementation
[SWS_UCM_01123]	Supported application layers by D-PDU API implementation
[SWS_UCM_01124]	Supported protocols by D-PDU API implementation
[SWS_UCM_01125]	Separation of D-PDU API-Software with the MVCI protocol module firmware
[SWS_UCM_01126]	Root description file (RDF)
[SWS_UCM_01127]	Module Description File (MDF)
[SWS_UCM_01128]	Symbolic names and IDs
[SWS_UCM_01129]	SAE J2534-1 and RP 1210a compatibility
[SWS_UCM_01130]	ComPrimitives in RawMode
[SWS_UCM_01131]	PDUIoCtl(PDU_IOCTL_RESET)
[SWS_UCM_01132]	PDUIoCtl(PDU_IOCTL_START_MSG_FILTER), PDUIoCtl(PDU_IOCTL_CLEAR_MSG_FILTER), PDUIoCtl(PDU_IOCTL_STOP_MSG_FILTER)
[SWS_UCM_01133]	PDUIoCtl(PDU_IOCTL_SEND_BREAK)
[SWS_UCM_01134]	Not used D-PDU API function return codes
[SWS_UCM_01135]	Get Software Clusters descriptions from a vehicle
[SWS_UCM_01136]	
[SWS_UCM_01137]	
[SWS_UCM_01138]	
[SWS_UCM_01177]	
[SWS_UCM_01178]	
[SWS_UCM_01201]	Sequential orchestration of campaigns
[SWS_UCM_01203]	CampaignState field
[SWS_UCM_01204]	Initial state
[SWS_UCM_01205]	UCM Master internal state persistency
[SWS_UCM_01207]	Trigger on kSoftwarePackage_Transferring state
[SWS_UCM_01209]	Trigger on kProcessing state
[SWS_UCM_01212]	Trigger on kActivating state
[SWS_UCM_01214]	Final action on kVehicleChecking state
[SWS_UCM_01215]	Trigger on kCancelling state

▽

△

Number	Heading
[SWS_UCM_01216]	Final action on `kCancelling` state
[SWS_UCM_01217]	Monitoring of UCM subordinates
[SWS_UCM_01218]	Transition from `kIdle` state to `kSyncing` state
[SWS_UCM_01219]	Transition from `kSyncing` state to `kIdle` state
[SWS_UCM_01220]	Transition from `kIdle` state to `kVehiclePackageTransferring` and `kTransferring` states
[SWS_UCM_01221]	Transition from `kVehiclePackageTransferring` state and `kTransferring` state to `kCancelling` state
[SWS_UCM_01222]	Transition from `kVehiclePackageTransferring` state to `kSoftwarePackage_Transferring` state
[SWS_UCM_01227]	Transition from `kSoftwarePackage_Transferring` state and `kTransferring` state to `kCancelling` state
[SWS_UCM_01228]	Transition from `kSoftwarePackage_Transferring` state and `kTransferring` state to `kProcessing` state and `kUpdating` state
[SWS_UCM_01229]	SafetyConditions while processing stream
[SWS_UCM_01234]	Transition from `kProcessing` state to `kActivating` state
[SWS_UCM_01236]	Transition from `kProcessing` state and `kUpdating` state to `kCancelling` state
[SWS_UCM_01239]	Transition from `kActivating` state and `kUpdating` state to `kCancelling` state
[SWS_UCM_01240]	Transition from `kActivating` state to `kVehicleChecking` state
[SWS_UCM_01241]	Transition from `kVehicleChecking` state and `kUpdating` state to `kCancelling` state
[SWS_UCM_01242]	Transition from `kVehicleChecking` state and `kUpdating` state to `kIdle` state
[SWS_UCM_01243]	Transition from `kCancelling` state to `kIdle` state
[SWS_UCM_01244]	Cancellation of an update campaign shall be possible
[SWS_UCM_01246]	Unreachable UCM during update campaign
[SWS_UCM_01247]	Method to read History Report
[SWS_UCM_01248]	Content of History Report
[SWS_UCM_01265]	TransferState field
[SWS_UCM_01266]	Subordinate Not Available On The Network
[SWS_UCM_01267]	Vehicle State Manager Communication Error
[SWS_UCM_01268]	Vehicle Driver Interface Communication Error
[SWS_UCM_01269]	Campaign cancellation history
[SWS_UCM_01270]	New campaign disabling
[SWS_UCM_01271]	New campaign enabling
[SWS_UCM_01272]	`VehicleCheck` call not permitted
[SWS_UCM_01273]	`CancelCampaign` `CancelFailed` error
[SWS_UCM_01274]	`CancelCampaign` `OperationNotPermitted` error
[SWS_UCM_01275]	Safety conditions during activation

▽

△

Number	Heading
[SWS_UCM_01301]	`Vehicle Package` authentication
[SWS_UCM_01302]	`Vehicle Package` authentication failure
[SWS_UCM_01303]	Dependencies between `Software Packages`
[SWS_UCM_01305]	`Vehicle Package` format
[SWS_UCM_01306]	`TransferExit` Invalid package manifest
[SWS_UCM_01307]	`Vehicle Package` format not supported
[SWS_UCM_01308]	Check Vehicle Package version compatibility against UCM Master version

Table E.15: Deleted Specification Items in R23-11

E.5.4 Added Constraints in R23-11

none

E.5.5 Changed Constraints in R23-11

none

E.5.6 Deleted Constraints in R23-11

Number	Heading
[SWS_UCM_-CONSTR_-00003]	Exclusive use of Vehicle Driver Interface
[SWS_UCM_-CONSTR_-00004]	Unsupported safety by Vehicle driver interface
[SWS_UCM_-CONSTR_-00005]	Safety state change
[SWS_UCM_-CONSTR_-00006]	Exclusive use of Vehicle State Manager
[SWS_UCM_-CONSTR_-00007]	Unsupported safety conditions by Vehicle State Manager

▽

△

Number	Heading
[SWS_UCM_-CONSTR_-00008]	Switching vehicle into update mode
[SWS_UCM_-CONSTR_-00009]	Safety condition change
[SWS_UCM_-CONSTR_-00011]	Flashing Adapter provided interface
[SWS_UCM_-CONSTR_-00013]	Confidential information protection
[SWS_UCM_-CONSTR_-00015]	Trigger on `kVehicleChecking` state
[SWS_UCM_-CONSTR_-00016]	OTA Client use of RequestedPackage field
[SWS_UCM_-CONSTR_-00017]	Interaction of `UCM Master` with Vehicle Driver

Table E.16: Deleted Constraints in R23-11