Document Title	Specification of Software Cluster Connection module
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	974

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R20-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2020-11-30	R20-11	AUTOSAR Release Management	Initial release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

Known Limitations

- Not all VFB communication features are supported
- The Proxy Modules are not specified for all BSW Services
- No run-time separation is defined

Additional limitations are described in document [1]

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Basic Software module Software Cluster Connection.

2 Acronyms and abbreviations

The glossary below includes acronyms and abbreviations relevant to Requirements on Software Cluster Connection that are not included in the AUTOSAR Glossary [2].

Abbreviation / Acronym:	Description:
SwCluC	Software Cluster Connection

Table 2.1: Acronyms and Abbreviations

Terms:	Description:
Software Cluster	A Software Cluster groups all AUTOSAR artifacts that are relevant to deploy software on a machine. The full definition is given in document [2]
Software Cluster Connection	The Software Cluster Connection is the BSW module that provides the features to • connect the Binary Objects deployed on the same machine • substitute not locally-available BSW modules in an Applicative Software Cluster, whose interfaces are required for the integrated SW, by so called Proxy Modules. • implement the VFB communication features between Software Clusters together with RTE with the means of an RTE Implementation Plug-In

Terms:	Description:
machine	A machine consists of a set of computing resources - such as microcontroller cores, memory or peripheral (e.g. communication) devices - and has the ability to execute software applications. The representation of a machine in the AUTOSAR Classic Platform could be done with an EcuInstance, but note that this semantic is currently in clarification. Further information is given in document [2].
binary-identical	Bit for Bit identical
Binary Object	A set of files, which contains the binary executable code and data. This binary executable code and data will not be modified again, before programming it on the target ECU.
Binary Manifest	The Binary Manifest is the well-defined interface of the Software Cluster's Binary Object, providing the meta information of a resources and information - so called handles - to access such a resource.
Applicative Software Cluster	A Software Cluster that mainly contains software components, and only selected BSW modules (e.g. a Service module, transformers, e.t.c.)
Host Software Cluster	The single Software Cluster that contains the major part of the BSW, and especially the micro controller dependent lower layer BSW Modules, e.g. OS and MCAL.
Substitution Software Cluster	The single Software Cluster that can override the provided resources of other Software Clusters for bug fixing purpose.
Proxy Module	A Proxy Module substitutes a BSW module in an Applicative Software Cluster. A Proxy module itself is split into High Proxy Module and Low Proxy Module. The High Proxy Module provides dedicated interfaces for modules in higher layers or same layer, and the functionality to connect them via the Binary Manifest to the Low Proxy Module in the Host Software Cluster.
High Proxy Module	The part of the Proxy Module residing in an Applicative Software Cluster.
Low Proxy Module	The part of the Proxy Module residing in the Host Software Cluster.
Os High Proxy	A type of Proxy Module implementing Os APIs in the Applicative Software Cluster.
Os Low Proxy	A type of proxy Module implementing an Os abstraction in the Host Software Cluster.
NvM High Proxy	A type of Proxy Module substituting the NVRAM Manager in the Applicative Software Cluster.
NvM Low Proxy	A type of Proxy Module connecting the NvM High Proxy Modules to the NVRAM Manager in the Host Software Cluster.
RTE Implementation Plug-In	A RTE Implementation Plug-In is a part of the overall RTE implementation, which is not provided by the RTE Generator, but from an additional source (e.g. a Plug-In Generator or a manually implemented source code).
RTE Implementation Plug-In Service	A RTE Implementation Plug-In Service is a single entry point into the RTE Implementation Plug-In implementing a low level service for the RTE. For instance access to a specific buffer.

Terms:	Description:
Local Software Cluster Communication Plug-In	A Local Software Cluster Communication Plug-In is an RTE Implementation Plug-In, which handles the communication locally inside a Software Cluster. This includes the Transformer handling, if a DataMapping exist for the according Communication Graph
Cross Software Cluster Communication Plug-In	A Cross Software Cluster Communication Plug-In is an RTE Implementation Plug-In that handles the communication towards other Software Clusters. This includes the Transformer handling, if intra ECU transformation is configured.
Communication Graph	The sum of all AbstractAccessPoints to elements of PortInterfaces, instantiated in PortPrototypes which are connected to each other; or the sum of all accesses from BswModuleEntitys to interface elements in a BswModuleDescriptions connected to each other.
Data Communication Graph	The sum of all VariableAccesses to VariableDataPrototypes instantiated in PortPrototypes, which are connected to each other; or the sum of all VariableAccesses to VariableDataPrototypes in the InternalBehavior; or the sum of all BswVariableAccesses to VariableDataPrototypes in BswModuleDescriptions connected to each other.
Parameter Communication Graph	The sum of all ParameterAccesses to ParameterDataPrototypes instantiated in PortPrototypes, which are connected to each other; or the sum of all ParameterAccesses to ParameterDataPrototypes in the InternalBehavior.
Client Server Communication Graph	The sum of all ServerCallPoints to operations instantiated in PortPrototypes, which are connected to each other, including the associated server runnable.
Trigger Communication Graph	The sum of all ExternalTriggeringPoints for triggers instantiated in PortPrototypes, which are connected to each other, including the associated triggered runnable.
Mode Communication Graph	The sum of all ModeAccessPoints and ModeSwitchPoints to ModeDeclarationGroupPrototypes instantiated in PortPrototypes, which are connected to each other; or the sum of all managedModeGroups and accessedModeGroups to ModeDeclarationGroupPrototypes in BswModuleDescriptions connected to each other.
mode manager	Entering and leaving modes is initiated by a *mode manager*. A *mode manager* is either a software component that provides a p-port typed by a ModeSwitchInterface, or a BSW module that defines in its BswModuleDescription a ModeDeclarationGroupPrototype in the role providedModeGroup.
mode switch notification	The communication of a mode switch from the mode manager to the mode user, using either the ModeSwitchInterface or providedModeGroup and requiredModeGroup ModeDeclarationGroupPrototypes.
mode switch port	The port for receiving (or sending) a mode switch notification. For this purpose, a mode switch port is typed by a ModeSwitchInterface.

Terms:	Description:
mode user	An *AUTOSAR SW-C* or *AUTOSAR Basic Software Module* that depends on modes, is called a mode user. The dependency can occur through a SwcModeSwitchEvent/BswModeSwitchEvent, a ModeAccessPoint for a provided/required mode switch port, or a accessedModeGroup for a providedModeGroup/requiredModeGroup ModeDeclarationGroupPrototype.
on-entry ExecutableEntity	A RunnableEntity that is triggered by a SwcModeSwitchEvent with *ModeActivationKind* 'entry'; or a BswSchedulableEntity that is triggered by a BswModeSwitchEvent with *ModeActivationKind* 'entry'.
on-exit ExecutableEntity	A RunnableEntity that is triggered by a SwcModeSwitchEvent with *ModeActivationKind* 'exit'; or a BswSchedulableEntity that is triggered by a BswModeSwitchEvent with *ModeActivationKind* 'exit'.
on-transition ExecutableEntity	A RunnableEntity that is triggered by a SwcModeSwitchEvent with *ModeActivationKind* 'transition'; or a BswSchedulableEntity that is triggered by a BswModeSwitchEvent with *ModeActivationKind* 'transition'.
trigger port	A PortPrototype, which is typed by an TriggerInterface
trigger sink	A *trigger sink* relies on the activation of RunnableEntity or a BswSchedulableEntity, if a particular Trigger is raised. A *trigger sink* has a dedicated require trigger port(s) and / or requiredTrigger Trigger(s) to communicate to the trigger source(s).
trigger source	A *trigger source* administrates the particular Trigger, and informs the RTE or *Basic Software Scheduler* if the Trigger is raised. A *trigger source* has dedicated provide trigger port(s) and / or releasedTrigger Trigger(s) to communicate to the trigger sink(s).
triggered BswSchedulableEntity	A BswSchedulableEntity that is triggered at least by one BswExternalTriggerOccurredEvent or BswInternalTriggerOccurredEvent. In particular cases, the *Trigger Event Communication* or the *Inter Basic Software Schedulable Entity Triggering* is implemented by the *Basic Software Scheduler* as a direct or trusted function call of the triggered ExecutableEntity, by the triggering ExecutableEntity.
triggered ExecutableEntity	A RunnableEntity that is triggered by at least one ExternalTriggerOccurredEvent / InternalTriggerOccurredEvent; or a BswSchedulableEntity that is triggered by at least one BswExternalTriggerOccurredEvent/ BswInternalTriggerOccurredEvent. In particular cases, the Trigger Event Communication or the *Inter Runnable Triggering* is implemented by RTE or *Basic Software Scheduler* as a direct or trusted function call of the triggered ExecutableEntity, by the triggering ExecutableEntity.
triggered runnable	A RunnableEntity that is triggered at least by one ExternalTriggerOccurredEvent or InternalTriggerOccurredEvent. In particular cases, the Trigger Event Communication or the *Inter Runnable Triggering* is implemented by RTE as a direct or trusted function call of the *triggered runnable*, by the triggering runnable.

Table 2.2: Terms

3 Related documentation

3.1 Input documents & related standards and norms

[1] Explanation of CP Software Cluster Design And Integration Guideline
AUTOSAR_EXP_CPSwClusterDesignAndIntegrationGuideline

[2] Glossary
AUTOSAR_TR_Glossary

[3] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral

[4] Requirements on Software Cluster Connection module
AUTOSAR_SRS_SoftwareClusterConnection

[5] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral

[6] System Template
AUTOSAR_TPS_SystemTemplate

[7] Specification of CRC Routines
AUTOSAR_SWS_CRCLibrary

[8] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping

[9] Specification of RTE Software
AUTOSAR_SWS_RTE

[10] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration

[11] Software Component Template
AUTOSAR_TPS_SoftwareComponentTemplate

[12] Specification of Operating System
AUTOSAR_SWS_OS

[13] Specification of NVRAM Manager
AUTOSAR_SWS_NVRAMManager

[14] Specification of Platform Types
AUTOSAR_SWS_PlatformTypes

[15] Specification of Standard Types
AUTOSAR_SWS_StandardTypes

[16] ISO 17356-3: Road vehicles – Open interface for embedded automotive applications – Part 3: OSEK/VDX Operating System (OS)

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [3, SWS BSW General], which is also valid for Software Cluster Connection.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Software Cluster Connection.

4 Constraints and assumptions

4.1 Limitations

The specification currently supports a limited number of BSW modules. In addition, the available VFB communication features are restricted.

4.2 Applicability to car domains

The specification focus on larger ECUs centralizing software functionality - so called domain or zone controllers. It assumes that the software components which are mapped to different Software Clusters are rather loosely coupled w.r.t. to interfaces and time domain. The software components supposed to implement mainly control loop software - usually time driven but may also react on limited numbers of sporadic events. Further information can be found in document [1].

5 Dependencies to other modules

5.1 Binary Manifest

5.1.1 Header File structure

[SWS_SwCluC_00013]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall provide the header file SwCluC_BManif.h.

⌋()

[SWS_SwCluC_00014]{DRAFT} ⌈The header file SwCluC_BManif.h shall include all header files configured in SwCluCBManifHeaderIncludes in the containers SwCluCBManifProvideResourceEntryGroup and SwCluCBManifRequireResourceEntryGroup.⌋()

6 Requirements Tracing

The following tables reference the requirements specified in [4], SRS_SoftwareClusterConnection and [5], SRS_BSWGeneral 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
[SRS_BSW_00161]	No description	[SWS_SwCluC_00999]
[SRS_BSW_00301]	No description	[SWS_SwCluC_00999]
[SRS_BSW_00304]	No description	[SWS_SwCluC_00999]
[SRS_BSW_00359]	No description	[SWS_SwCluC_00999]
[SRS_BSW_00409]	No description	[SWS_SwCluC_00999]
[SRS_BSW_00410]	No description	[SWS_SwCluC_00999]
[SRS_BSW_00427]	No description	[SWS_SwCluC_00999]
[SRS_BSW_00456]	No description	[SWS_SwCluC_00999]
[SRS_BSW_00375]	No description	[SWS_SwCluC_00999]
[SRS_BSW_00005]	Modules of the μC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	[SWS_SwCluC_00999]
[SRS_BSW_00006]	The source code of software modules above the μC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	[SWS_SwCluC_00999]
[SRS_BSW_00007]	All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.	[SWS_SwCluC_00999]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_SwCluC_11000] [SWS_SwCluC_11001] [SWS_SwCluC_12000] [SWS_SwCluC_12100]
[SRS_BSW_00160]	Configuration files of AUTOSAR Basic SW module shall be readable for human beings	[SWS_SwCluC_00999]
[SRS_BSW_00162]	The AUTOSAR Basic Software shall provide a hardware abstraction layer	[SWS_SwCluC_00999]
[SRS_BSW_00164]	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	[SWS_SwCluC_00999]

△

Requirement	Description	Satisfied by
[SRS_BSW_00168]	SW components shall be tested by a function defined in a common API in the Basis-SW	[SWS_SwCluC_00999]
[SRS_BSW_00170]	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	[SWS_SwCluC_00999]
[SRS_BSW_00302]	All AUTOSAR Basic Software Modules shall only export information needed by other modules	[SWS_SwCluC_00999]
[SRS_BSW_00306]	AUTOSAR Basic Software Modules shall be compiler and platform independent	[SWS_SwCluC_00999]
[SRS_BSW_00307]	Global variables naming convention	[SWS_SwCluC_00999]
[SRS_BSW_00308]	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	[SWS_SwCluC_00999]
[SRS_BSW_00309]	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	[SWS_SwCluC_00999]
[SRS_BSW_00310]	API naming convention	[SWS_Rte_89020] [SWS_Rte_89021] [SWS_Rte_89022] [SWS_Rte_89023] [SWS_Rte_91000] [SWS_Rte_91113] [SWS_Rte_91114] [SWS_Rte_91115] [SWS_Rte_91117] [SWS_Rte_91119] [SWS_Rte_91121] [SWS_Rte_91122] [SWS_SwCluC_10000] [SWS_SwCluC_10001] [SWS_SwCluC_10002] [SWS_SwCluC_10003] [SWS_SwCluC_10004] [SWS_SwCluC_10005] [SWS_SwCluC_10006] [SWS_SwCluC_10007] [SWS_SwCluC_10008] [SWS_SwCluC_10011] [SWS_SwCluC_10020] [SWS_SwCluC_10021] [SWS_SwCluC_10022] [SWS_SwCluC_10023] [SWS_SwCluC_10032] [SWS_SwCluC_10033] [SWS_SwCluC_10034] [SWS_SwCluC_11000] [SWS_SwCluC_11001] [SWS_SwCluC_12000] [SWS_SwCluC_12100]
[SRS_BSW_00312]	Shared code shall be reentrant	[SWS_SwCluC_00999]
[SRS_BSW_00314]	All internal driver modules shall separate the interrupt frame definition from the service routine	[SWS_SwCluC_00999]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_SwCluC_02137] [SWS_SwCluC_02138] [SWS_SwCluC_02139]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00325]	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	[SWS_SwCluC_00999]
[SRS_BSW_00327]	Error values naming convention	[SWS_SwCluC_02140]
[SRS_BSW_00328]	All AUTOSAR Basic Software Modules shall avoid the duplication of code	[SWS_SwCluC_00999]
[SRS_BSW_00330]	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	[SWS_SwCluC_00999]
[SRS_BSW_00331]	All Basic Software Modules shall strictly separate error and status information	[SWS_SwCluC_00999]
[SRS_BSW_00333]	For each callback function it shall be specified if it is called from interrupt context or not	[SWS_NvM_00467] [SWS_NvM_00469]
[SRS_BSW_00335]	Status values naming convention	[SWS_SwCluC_00999]
[SRS_BSW_00337]	Classification of development errors	[SWS_SwCluC_02140]
[SRS_BSW_00339]	Reporting of production relevant error status	[SWS_SwCluC_00999]
[SRS_BSW_00342]	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	[SWS_SwCluC_00999]
[SRS_BSW_00343]	The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit	[SWS_SwCluC_00999]
[SRS_BSW_00344]	BSW Modules shall support link-time configuration	[SWS_SwCluC_00999]
[SRS_BSW_00346]	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	[SWS_SwCluC_00999]
[SRS_BSW_00347]	A Naming seperation of different instances of BSW drivers shall be in place	[SWS_SwCluC_00999]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00348]	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	[SWS_SwCluC_00999]
[SRS_BSW_00353]	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	[SWS_SwCluC_00999]
[SRS_BSW_00357]	For success/failure of an API call a standard return type shall be defined	[SWS_SwCluC_00999]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_SwCluC_11000] [SWS_SwCluC_11001] [SWS_SwCluC_12000] [SWS_SwCluC_12100]
[SRS_BSW_00360]	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	[SWS_NvM_00467] [SWS_NvM_00469] [SWS_SwCluC_00999]
[SRS_BSW_00361]	All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header	[SWS_SwCluC_00999]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_SwCluC_02137] [SWS_SwCluC_02138] [SWS_SwCluC_02139]
[SRS_BSW_00371]	The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules	[SWS_SwCluC_00999]
[SRS_BSW_00373]	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	[SWS_SwCluC_00999]
[SRS_BSW_00377]	A Basic Software Module can return a module specific types	[SWS_SwCluC_00999]
[SRS_BSW_00378]	AUTOSAR shall provide a boolean type	[SWS_SwCluC_00999]
[SRS_BSW_00380]	Configuration parameters being stored in memory shall be placed into separate c-files	[SWS_SwCluC_00999]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00385]	List possible error notifications	[SWS_SwCluC_02140]
[SRS_BSW_00404]	BSW Modules shall support post-build configuration	[SWS_SwCluC_00999]
[SRS_BSW_00405]	BSW Modules shall support multiple configuration sets	[SWS_SwCluC_00999]
[SRS_BSW_00406]	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	[SWS_SwCluC_00999]
[SRS_BSW_00411]	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	[SWS_SwCluC_00999]
[SRS_BSW_00413]	An index-based accessing of the instances of BSW modules shall be done	[SWS_SwCluC_00999]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_SwCluC_00999]
[SRS_BSW_00415]	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	[SWS_SwCluC_00999]
[SRS_BSW_00416]	The sequence of modules to be initialized shall be configurable	[SWS_SwCluC_00999]
[SRS_BSW_00417]	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	[SWS_SwCluC_00999]
[SRS_BSW_00419]	If a pre-compile time configuration parameter is implemented as "const" it should be placed into a separate c-file	[SWS_SwCluC_00999]
[SRS_BSW_00422]	Pre-de-bouncing of error status information is done within the DEM	[SWS_SwCluC_00999]
[SRS_BSW_00423]	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	[SWS_SwCluC_00999]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00424]	BSW module main processing functions shall not be allowed to enter a wait state	[SWS_SwCluC_00999]
[SRS_BSW_00425]	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	[SWS_SwCluC_00999]
[SRS_BSW_00426]	BSW Modules shall ensure data consistency of data which is shared between BSW modules	[SWS_SwCluC_00999]
[SRS_BSW_00432]	Modules should have separate main processing functions for read/receive and write/ transmit data path	[SWS_SwCluC_00999]
[SRS_BSW_00433]	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	[SWS_SwCluC_00999]
[SRS_BSW_00437]	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	[SWS_SwCluC_00999]
[SRS_BSW_00438]	Configuration data shall be defined in a structure	[SWS_SwCluC_00999]
[SRS_BSW_00439]	Enable BSW modules to handle interrupts	[SWS_SwCluC_00999]
[SRS_BSW_00440]	The callback function invocation by the BSW module shall follow the signature provided by RTE to invoke servers via Rte_Call API	[SWS_SwCluC_00999]
[SRS_BSW_00441]	Naming convention for type, macro and function	[SWS_SwCluC_01000] [SWS_SwCluC_01002]
[SRS_BSW_00447]	Standardizing Include file structure of BSW Modules Implementing Autosar Service	[SWS_SwCluC_00999]
[SRS_BSW_00448]	Module SWS shall not contain requirements from Other Modules	[SWS_SwCluC_00999]
[SRS_BSW_00449]	BSW Service APIs used by Autosar Application Software shall return a Std_ ReturnType	[SWS_SwCluC_00999]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00450]	A Main function of a un-initialized module shall return immediately	[SWS_SwCluC_00999]
[SRS_BSW_00451]	Hardware registers shall be protected if concurrent access to these registers occur	[SWS_SwCluC_00999]
[SRS_BSW_00452]	Classification of runtime errors	[SWS_SwCluC_00999]
[SRS_BSW_00453]	BSW Modules shall be harmonized	[SWS_SwCluC_00999]
[SRS_BSW_00454]	An alternative interface without a parameter of category DATA_ REFERENCE shall be available.	[SWS_SwCluC_00999]
[SRS_BSW_00457]	Callback functions of Application software components shall be invoked by the Basis SW	[SWS_NvM_00467] [SWS_NvM_00469] [SWS_NvM_00539] [SWS_NvM_00540] [SWS_SwCluC_00999]
[SRS_BSW_00458]	Classification of production errors	[SWS_SwCluC_00999]
[SRS_BSW_00459]	It shall be possible to concurrently execute a service offered by a BSW module in different partitions	[SWS_SwCluC_00999]
[SRS_BSW_00460]	Reentrancy Levels	[SWS_SwCluC_00999]
[SRS_BSW_00461]	Modules called by generic modules shall satisfy all interfaces requested by the generic module	[SWS_SwCluC_00999]
[SRS_BSW_00463]	Naming convention of callout prototypes	[SWS_SwCluC_00999]
[SRS_BSW_00466]	Classification of extended production errors	[SWS_SwCluC_00999]
[SRS_BSW_00467]	The init / deinit services shall only be called by BswM or EcuM	[SWS_SwCluC_00999]
[SRS_BSW_00469]	Fault detection and healing of production errors and extended production errors	[SWS_SwCluC_00999]
[SRS_BSW_00470]	Execution frequency of production error detection	[SWS_SwCluC_00999]
[SRS_BSW_00471]	Do not cause dead-locks on detection of production errors - the ability to heal from previously detected production errors	[SWS_SwCluC_00999]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00472]	Avoid detection of two production errors with the same root cause.	[SWS_SwCluC_00999]
[SRS_BSW_00473]	Classification of transient faults	[SWS_SwCluC_00999]
[SRS_BSW_00477]	The functional interfaces of AUTOSAR BSW modules shall be specified in C90	[SWS_SwCluC_00999]
[SRS_BSW_00478]	Timing limits of main functions	[SWS_SwCluC_00999]
[SRS_BSW_00479]	Interfaces for handling request from external devices	[SWS_SwCluC_00999]
[SRS_BSW_00480]	NullPointer Errors shall follow a naming rule	[SWS_SwCluC_02140]
[SRS_BSW_00481]	Invalid configuration set selection errors shall follow a naming rule	[SWS_SwCluC_00999]
[SRS_BSW_00483]	BSW Modules shall handle buffer alignments internally	[SWS_SwCluC_00999]
[SRS_BSW_00484]	Input parameters of scalar and enum types shall be passed as a value.	[SWS_SwCluC_00999]
[SRS_BSW_00485]	Input parameters of structure type shall be passed as a reference to a constant structure	[SWS_SwCluC_00999]
[SRS_BSW_00486]	Input parameters of array type shall be passed as a reference to the constant array base type	[SWS_SwCluC_00999]
[SRS_BSW_00487]	Errors for module initialization shall follow a naming rule	[SWS_SwCluC_02140]
[SRS_BSW_00488]	Classification of security events	[SWS_SwCluC_00999]
[SRS_LIBS_08533]	No description	[SWS_NvM_00454] [SWS_NvM_00540] [SWS_NvM_00764]
[SRS_Mem_00016]	The NVRAM manager shall provide functionality to read out data associated with an NVRAM block from the non-volatile memory	[SWS_NvM_00454] [SWS_NvM_00764]
[SRS_Mem_00017]	The NVRAM manager shall provide functionality to store data associated with an NVRAM block in the non-volatile memory	[SWS_NvM_00455] [SWS_NvM_00793]

▽

△

Requirement	Description	Satisfied by
[SRS_Mem_00018]	The NVRAM manager shall provide functionality to restore an NVRAM block's associated data from ROM defaults	[SWS_NvM_00456] [SWS_NvM_00813]
[SRS_Mem_00020]	The NVRAM manager shall provide functionality to read out the status of read/write operations	[SWS_NvM_00451]
[SRS_Mem_00127]	The NVRAM manager shall allow enabling/ disabling a write protection for each NVRAM block individually	[SWS_NvM_00450]
[SRS_Mem_08007]	The NVRAM manager shall provide a service for the selection of valid dataset NV blocks	[SWS_NvM_00448]
[SRS_Mem_08011]	The NVRAM manager shall provide a service to invalidate a block of data in the non-volatile memory	[SWS_NvM_00459]
[SRS_Mem_08544]	The NVRAM manager shall provide a service to erase the NV block(s) associated with an NVRAM block	[SWS_NvM_00457]
[SRS_Mem_08545]	The NVRAM Manager shall provide a service for marking the permanent RAM data block of an NVRAM block valid	[SWS_NvM_00453]
[SRS_Mem_08546]	It shall be possible to protect permanent RAM data blocks against data loss due to reset	[SWS_NvM_00548]
[SRS_Mem_08560]	Each NVRAM block shall be configurable for shared access	[SWS_NvM_00535]
[SRS_Rte_00183]	RTE Read API returning the data Element value	[SWS_Rte_91122]
[SRS_Rte_00300]	RTE Implementation Plug-Ins for explicit communication	[SWS_Rte_89020] [SWS_Rte_89021] [SWS_Rte_91000] [SWS_Rte_91119] [SWS_Rte_91122]
[SRS_Rte_00301]	RTE Implementation Plug-Ins for implicit communication	[SWS_Rte_89020] [SWS_Rte_89021] [SWS_Rte_91000] [SWS_Rte_91119] [SWS_Rte_91122]
[SRS_Rte_00306]	Standardized interfaces for RTE Implementation Plug-Ins	[SWS_Rte_89020] [SWS_Rte_89021] [SWS_Rte_89022] [SWS_Rte_89023] [SWS_Rte_91000] [SWS_Rte_91113] [SWS_Rte_91114] [SWS_Rte_91115] [SWS_Rte_91117] [SWS_Rte_91119] [SWS_Rte_91121] [SWS_Rte_91122]

▽

△

Requirement	Description	Satisfied by
[SRS_Rte_00312]	RTE Implementation Plug-Ins for transformers in client server communication	[SWS_Rte_89022] [SWS_Rte_89023]
[SRS_Rte_00317]	RTE Implementation Plug-Ins for transformers in trigger communication	[SWS_Rte_89022] [SWS_Rte_91117]
[SRS_Rte_00319]	RTE Implementation Plug-Ins for parameter communication	[SWS_Rte_91121]
[SRS_Rte_00321]	RTE Implementation Plug-Ins for mode communication	[SWS_Rte_91113] [SWS_Rte_91114] [SWS_Rte_91115]
[SRS_SwCluC_00001]	Easy target machine interpretation	[SWS_SwCluC_00001] [SWS_SwCluC_00003] [SWS_SwCluC_00004] [SWS_SwCluC_00005] [SWS_SwCluC_00006] [SWS_SwCluC_00007] [SWS_SwCluC_00008] [SWS_SwCluC_00009] [SWS_SwCluC_00010] [SWS_SwCluC_00011] [SWS_SwCluC_00012] [SWS_SwCluC_00015] [SWS_SwCluC_00016] [SWS_SwCluC_00017] [SWS_SwCluC_00018] [SWS_SwCluC_00019] [SWS_SwCluC_00020] [SWS_SwCluC_00021] [SWS_SwCluC_00022] [SWS_SwCluC_00023] [SWS_SwCluC_00024] [SWS_SwCluC_00025] [SWS_SwCluC_00026] [SWS_SwCluC_00027] [SWS_SwCluC_00028] [SWS_SwCluC_00029] [SWS_SwCluC_00030] [SWS_SwCluC_00031] [SWS_SwCluC_00032] [SWS_SwCluC_00033] [SWS_SwCluC_00034] [SWS_SwCluC_00035] [SWS_SwCluC_00036] [SWS_SwCluC_00037] [SWS_SwCluC_00038] [SWS_SwCluC_00039] [SWS_SwCluC_00040] [SWS_SwCluC_00041] [SWS_SwCluC_00042] [SWS_SwCluC_00070] [SWS_SwCluC_00071] [SWS_SwCluC_00072] [SWS_SwCluC_00079] [SWS_SwCluC_00080] [SWS_SwCluC_00081] [SWS_SwCluC_00082] [SWS_SwCluC_00083] [SWS_SwCluC_00084] [SWS_SwCluC_00085] [SWS_SwCluC_CONSTR_00073] [SWS_SwCluC_CONSTR_00074]
[SRS_SwCluC_00002]	Identification of intended connections by unique IDs	[SWS_SwCluC_00015] [SWS_SwCluC_00021] [SWS_SwCluC_00034] [SWS_SwCluC_00035] [SWS_SwCluC_00043]
[SRS_SwCluC_00003]	Bidirectional connections	[SWS_SwCluC_00024] [SWS_SwCluC_00026]
[SRS_SwCluC_00004]	Connection multiplicity	[SWS_SwCluC_00027] [SWS_SwCluC_00045] [SWS_SwCluC_00046] [SWS_SwCluC_00047] [SWS_SwCluC_00089]
[SRS_SwCluC_00005]	Substitute Resource Providers	[SWS_SwCluC_00008] [SWS_SwCluC_00054] [SWS_SwCluC_00055] [SWS_SwCluC_CONSTR_00087]
[SRS_SwCluC_00006]	C API	[SWS_SwCluC_00002] [SWS_SwCluC_00062] [SWS_SwCluC_00063] [SWS_SwCluC_00064] [SWS_SwCluC_00065] [SWS_SwCluC_00066] [SWS_SwCluC_00067] [SWS_SwCluC_00068] [SWS_SwCluC_00069] [SWS_SwCluC_01000] [SWS_SwCluC_01002] [SWS_SwCluC_10000] [SWS_SwCluC_10001] [SWS_SwCluC_10002] [SWS_SwCluC_10003] [SWS_SwCluC_10004] [SWS_SwCluC_10005] [SWS_SwCluC_10006] [SWS_SwCluC_10007] [SWS_SwCluC_10008] [SWS_SwCluC_10009] [SWS_SwCluC_10010] [SWS_SwCluC_10011] [SWS_SwCluC_10020] [SWS_SwCluC_10021] [SWS_SwCluC_10022] [SWS_SwCluC_10023] [SWS_SwCluC_10032] [SWS_SwCluC_10033] [SWS_SwCluC_10034]
[SRS_SwCluC_00009]	Support missing interface partners	[SWS_SwCluC_00015] [SWS_SwCluC_00019] [SWS_SwCluC_00030] [SWS_SwCluC_00034] [SWS_SwCluC_00035] [SWS_SwCluC_00044]

▽

△

Requirement	Description	Satisfied by
[SRS_SwCluC_00010]	Static safeguard of Software Cluster connections	[SWS_SwCluC_00015] [SWS_SwCluC_00022] [SWS_SwCluC_03005] [SWS_SwCluC_03033] [SWS_SwCluC_03034] [SWS_SwCluC_03035] [SWS_SwCluC_03036] [SWS_SwCluC_03037] [SWS_SwCluC_03038] [SWS_SwCluC_03039] [SWS_SwCluC_03040] [SWS_SwCluC_03041] [SWS_SwCluC_03042] [SWS_SwCluC_03043] [SWS_SwCluC_03044] [SWS_SwCluC_03045] [SWS_SwCluC_03046] [SWS_SwCluC_03047]
[SRS_SwCluC_00011]	Separation of immutable memory and memory modifiable at the connection phase	[SWS_SwCluC_00001] [SWS_SwCluC_00003] [SWS_SwCluC_00004] [SWS_SwCluC_00005] [SWS_SwCluC_00006] [SWS_SwCluC_00007] [SWS_SwCluC_00008] [SWS_SwCluC_00009] [SWS_SwCluC_00010] [SWS_SwCluC_00011] [SWS_SwCluC_00012] [SWS_SwCluC_00015] [SWS_SwCluC_00028] [SWS_SwCluC_00029] [SWS_SwCluC_00030] [SWS_SwCluC_00031] [SWS_SwCluC_00032] [SWS_SwCluC_00033] [SWS_SwCluC_00034] [SWS_SwCluC_00035] [SWS_SwCluC_00042] [SWS_SwCluC_00070] [SWS_SwCluC_00071] [SWS_SwCluC_00072] [SWS_SwCluC_00079] [SWS_SwCluC_00080] [SWS_SwCluC_00081] [SWS_SwCluC_00082] [SWS_SwCluC_00083] [SWS_SwCluC_00084] [SWS_SwCluC_00085]
[SRS_SwCluC_00012]	direct linkage	[SWS_SwCluC_00057] [SWS_SwCluC_00058] [SWS_SwCluC_00059] [SWS_SwCluC_00060] [SWS_SwCluC_00061]
[SRS_SwCluC_00013]	Initialization with C-compiler and linker means	[SWS_SwCluC_00001] [SWS_SwCluC_00003] [SWS_SwCluC_00004] [SWS_SwCluC_00005] [SWS_SwCluC_00006] [SWS_SwCluC_00007] [SWS_SwCluC_00008] [SWS_SwCluC_00009] [SWS_SwCluC_00010] [SWS_SwCluC_00011] [SWS_SwCluC_00012] [SWS_SwCluC_00029] [SWS_SwCluC_00030] [SWS_SwCluC_00031] [SWS_SwCluC_00032] [SWS_SwCluC_00033] [SWS_SwCluC_00034] [SWS_SwCluC_00035] [SWS_SwCluC_00042] [SWS_SwCluC_00070] [SWS_SwCluC_00071] [SWS_SwCluC_00072]
[SRS_SwCluC_00014]	Standardized persistence in memory	[SWS_SwCluC_00001] [SWS_SwCluC_00003] [SWS_SwCluC_00004] [SWS_SwCluC_00005] [SWS_SwCluC_00006] [SWS_SwCluC_00007] [SWS_SwCluC_00008] [SWS_SwCluC_00009] [SWS_SwCluC_00010] [SWS_SwCluC_00011] [SWS_SwCluC_00012] [SWS_SwCluC_00015] [SWS_SwCluC_00016] [SWS_SwCluC_00017] [SWS_SwCluC_00018] [SWS_SwCluC_00019] [SWS_SwCluC_00020] [SWS_SwCluC_00021] [SWS_SwCluC_00022] [SWS_SwCluC_00023] [SWS_SwCluC_00024] [SWS_SwCluC_00025] [SWS_SwCluC_00026] [SWS_SwCluC_00027] [SWS_SwCluC_00028] [SWS_SwCluC_00029] [SWS_SwCluC_00030] [SWS_SwCluC_00031] [SWS_SwCluC_00032] [SWS_SwCluC_00033] [SWS_SwCluC_00034] [SWS_SwCluC_00035] [SWS_SwCluC_00036] [SWS_SwCluC_00037] [SWS_SwCluC_00038] [SWS_SwCluC_00039] [SWS_SwCluC_00040] [SWS_SwCluC_00041] [SWS_SwCluC_00042] [SWS_SwCluC_00070] [SWS_SwCluC_00071] [SWS_SwCluC_00072] [SWS_SwCluC_00079] [SWS_SwCluC_00080] [SWS_SwCluC_00081] [SWS_SwCluC_00082] [SWS_SwCluC_00083] [SWS_SwCluC_00084] [SWS_SwCluC_00085] [SWS_SwCluC_CONSTR_00073] [SWS_SwCluC_CONSTR_00074]
[SRS_SwCluC_00100]	Cross Software Cluster Communication Plug-Ins	[SWS_SwCluC_03000] [SWS_SwCluC_03001] [SWS_SwCluC_03002] [SWS_SwCluC_03003] [SWS_SwCluC_03004] [SWS_SwCluC_03006] [SWS_SwCluC_03007] [SWS_SwCluC_03008] [SWS_SwCluC_03009] [SWS_SwCluC_03010] [SWS_SwCluC_03011] [SWS_SwCluC_03064] [SWS_SwCluC_03065] [SWS_SwCluC_03068] [SWS_SwCluC_03101] [SWS_SwCluC_03102] [SWS_SwCluC_03103] [SWS_SwCluC_03106] [SWS_SwCluC_03107] [SWS_SwCluC_03108] [SWS_SwCluC_03109] [SWS_SwCluC_03110] [SWS_SwCluC_03111] [SWS_SwCluC_03112] [SWS_SwCluC_03113] [SWS_SwCluC_03120] [SWS_SwCluC_03121] [SWS_SwCluC_03122] [SWS_SwCluC_03123] [SWS_SwCluC_03124] [SWS_SwCluC_03126] [SWS_SwCluC_03130] [SWS_SwCluC_03131] [SWS_SwCluC_03132] [SWS_SwCluC_03133] [SWS_SwCluC_03134] [SWS_SwCluC_03135] [SWS_SwCluC_03136] [SWS_SwCluC_03137] ▽

▽

△

Requirement	Description	Satisfied by
		△ [SWS_SwCluC_03138] [SWS_SwCluC_03139] [SWS_SwCluC_03140] [SWS_SwCluC_03142] [SWS_SwCluC_03143] [SWS_SwCluC_03144] [SWS_SwCluC_03145] [SWS_SwCluC_03146] [SWS_SwCluC_03147] [SWS_SwCluC_03150] [SWS_SwCluC_03151] [SWS_SwCluC_03152] [SWS_SwCluC_03153] [SWS_SwCluC_03154] [SWS_SwCluC_03155] [SWS_SwCluC_03156] [SWS_SwCluC_03157] [SWS_SwCluC_03158] [SWS_SwCluC_03159] [SWS_SwCluC_03160] [SWS_SwCluC_03161] [SWS_SwCluC_03162] [SWS_SwCluC_03163] [SWS_SwCluC_03164] [SWS_SwCluC_03165] [SWS_SwCluC_03166] [SWS_SwCluC_03167] [SWS_SwCluC_03168] [SWS_SwCluC_03169] [SWS_SwCluC_03170] [SWS_SwCluC_03171] [SWS_SwCluC_03173] [SWS_SwCluC_03174] [SWS_SwCluC_03175] [SWS_SwCluC_03176] [SWS_SwCluC_03177] [SWS_SwCluC_11000] [SWS_SwCluC_11001] [SWS_SwCluC_CONSTR_03066] [SWS_SwCluC_CONSTR_03067] [SWS_SwCluC_CONSTR_03069]
[SRS_SwCluC_00101]	'1:n' Sender-receiver communication	[SWS_SwCluC_03034] [SWS_SwCluC_03035] [SWS_SwCluC_03036] [SWS_SwCluC_03037] [SWS_SwCluC_03038] [SWS_SwCluC_03039] [SWS_SwCluC_03040] [SWS_SwCluC_03041] [SWS_SwCluC_03042] [SWS_SwCluC_03043] [SWS_SwCluC_03044] [SWS_SwCluC_03064] [SWS_SwCluC_03065] [SWS_SwCluC_03068] [SWS_SwCluC_03101] [SWS_SwCluC_03102] [SWS_SwCluC_03103] [SWS_SwCluC_03106] [SWS_SwCluC_03107] [SWS_SwCluC_03108] [SWS_SwCluC_03109] [SWS_SwCluC_03110] [SWS_SwCluC_03111] [SWS_SwCluC_03112] [SWS_SwCluC_03113] [SWS_SwCluC_03120] [SWS_SwCluC_03121] [SWS_SwCluC_03122] [SWS_SwCluC_03123] [SWS_SwCluC_03124] [SWS_SwCluC_03126] [SWS_SwCluC_03130] [SWS_SwCluC_03131] [SWS_SwCluC_03132] [SWS_SwCluC_03133] [SWS_SwCluC_03135] [SWS_SwCluC_03136] [SWS_SwCluC_03137] [SWS_SwCluC_03138] [SWS_SwCluC_03139] [SWS_SwCluC_03140] [SWS_SwCluC_03142] [SWS_SwCluC_03143] [SWS_SwCluC_03144] [SWS_SwCluC_03156] [SWS_SwCluC_03157] [SWS_SwCluC_03158] [SWS_SwCluC_03159]
[SRS_SwCluC_00102]	'n:1' Sender-receiver communication	[SWS_SwCluC_03034] [SWS_SwCluC_03035] [SWS_SwCluC_03036] [SWS_SwCluC_03037] [SWS_SwCluC_03038] [SWS_SwCluC_03039] [SWS_SwCluC_03040] [SWS_SwCluC_03041] [SWS_SwCluC_03042] [SWS_SwCluC_03043] [SWS_SwCluC_03044]
[SRS_SwCluC_00103]	'n:1' Client-server communication	[SWS_SwCluC_03035] [SWS_SwCluC_03036] [SWS_SwCluC_03037] [SWS_SwCluC_03038] [SWS_SwCluC_03039] [SWS_SwCluC_03040] [SWS_SwCluC_03041] [SWS_SwCluC_03042] [SWS_SwCluC_03043] [SWS_SwCluC_03044] [SWS_SwCluC_03047] [SWS_SwCluC_03134] [SWS_SwCluC_03145] [SWS_SwCluC_03146] [SWS_SwCluC_03147] [SWS_SwCluC_03150] [SWS_SwCluC_03151] [SWS_SwCluC_03152] [SWS_SwCluC_03153] [SWS_SwCluC_03154] [SWS_SwCluC_03155] [SWS_SwCluC_03160] [SWS_SwCluC_03161] [SWS_SwCluC_03162] [SWS_SwCluC_03163] [SWS_SwCluC_03164] [SWS_SwCluC_03165] [SWS_SwCluC_03166] [SWS_SwCluC_03167] [SWS_SwCluC_03168] [SWS_SwCluC_03169] [SWS_SwCluC_03170] [SWS_SwCluC_03171] [SWS_SwCluC_03173] [SWS_SwCluC_03174] [SWS_SwCluC_03175] [SWS_SwCluC_03176] [SWS_SwCluC_03177]
[SRS_SwCluC_00104]	'1:n' Mode Switch Communication	[SWS_SwCluC_03015] [SWS_SwCluC_03016] [SWS_SwCluC_03017] [SWS_SwCluC_03018] [SWS_SwCluC_03019] [SWS_SwCluC_03022] [SWS_SwCluC_03023] [SWS_SwCluC_03024] [SWS_SwCluC_03025] [SWS_SwCluC_03026] [SWS_SwCluC_03027] [SWS_SwCluC_03028] [SWS_SwCluC_03029] [SWS_SwCluC_03030] [SWS_SwCluC_03031] [SWS_SwCluC_03045] [SWS_SwCluC_03057] [SWS_SwCluC_03061] [SWS_SwCluC_03062] [SWS_SwCluC_03063] [SWS_SwCluC_CONSTR_03020] [SWS_SwCluC_CONSTR_03021] [SWS_SwCluC_CONSTR_03032]

▽

△

Requirement	Description	Satisfied by
[SRS_SwCluC_00105]	'1:n' External Trigger communication	[SWS_SwCluC_03046] [SWS_SwCluC_03048] [SWS_SwCluC_03049] [SWS_SwCluC_03050] [SWS_SwCluC_03051] [SWS_SwCluC_03052] [SWS_SwCluC_03055] [SWS_SwCluC_03056] [SWS_SwCluC_03058] [SWS_SwCluC_03060] [SWS_SwCluC_CONSTR_03053] [SWS_SwCluC_CONSTR_03054] [SWS_SwCluC_CONSTR_03059]
[SRS_SwCluC_00106]	'1:n' Parameter Communication	[SWS_SwCluC_03006] [SWS_SwCluC_03007] [SWS_SwCluC_03008] [SWS_SwCluC_03009] [SWS_SwCluC_03010] [SWS_SwCluC_03011] [SWS_SwCluC_03034] [SWS_SwCluC_03035] [SWS_SwCluC_03036] [SWS_SwCluC_03037] [SWS_SwCluC_03038] [SWS_SwCluC_03039] [SWS_SwCluC_03040] [SWS_SwCluC_03041] [SWS_SwCluC_03042] [SWS_SwCluC_03043] [SWS_SwCluC_03044]
[SRS_SwCluC_00108]	Prevent from writing directly to memory of other Software Clusters	[SWS_SwCluC_11000] [SWS_SwCluC_11001]
[SRS_SwCluC_00206]	NV blocks in Applicative Software Cluster	[SWS_SwCluC_02101] [SWS_SwCluC_02102] [SWS_SwCluC_02103] [SWS_SwCluC_02104] [SWS_SwCluC_02105] [SWS_SwCluC_02106] [SWS_SwCluC_02107] [SWS_SwCluC_02108] [SWS_SwCluC_02109] [SWS_SwCluC_02110] [SWS_SwCluC_02111] [SWS_SwCluC_02112] [SWS_SwCluC_02113] [SWS_SwCluC_02114] [SWS_SwCluC_02115] [SWS_SwCluC_02116] [SWS_SwCluC_02117] [SWS_SwCluC_02118] [SWS_SwCluC_02119] [SWS_SwCluC_02120] [SWS_SwCluC_02121] [SWS_SwCluC_02122] [SWS_SwCluC_02123] [SWS_SwCluC_02124] [SWS_SwCluC_02125] [SWS_SwCluC_02126] [SWS_SwCluC_02127] [SWS_SwCluC_02128] [SWS_SwCluC_02129] [SWS_SwCluC_02132] [SWS_SwCluC_02133] [SWS_SwCluC_02136] [SWS_SwCluC_02137] [SWS_SwCluC_02138] [SWS_SwCluC_02139] [SWS_SwCluC_02142] [SWS_SwCluC_02144] [SWS_SwCluC_CONSTR_02134] [SWS_SwCluC_CONSTR_02135] [SWS_SwCluC_CONSTR_02141]
[SRS_SwCluC_00300]	A2L Generation Support	[SWS_SwCluC_03071] [SWS_SwCluC_03072] [SWS_SwCluC_03073] [SWS_SwCluC_03074] [SWS_SwCluC_03075] [SWS_SwCluC_03076] [SWS_SwCluC_03077]

Table 6.1: RequirementsTracing

7 Functional specification

7.1 Binary Manifest

7.1.1 Overview

With the concept of Software Clusters, the overall software of an AUTOSAR Classic Platform Architecture can be split into smaller units. In such a clustered AUTOSAR Classic Platform Architecture a resource is used to describes any capability which

- is needed to operate the software in Software Clusters

  AND

- which is provided by one Software Cluster for another Software Clusters

Please note as well document [6] with its description about CpSoftwareClusterResources.

Each Software Cluster is an independent build unit, and the result of the cluster-specific build processes are the Binary Objects. The Binary Manifests provide the means to connect the Binary Objects that are deployed on the same machine. Hence, the Binary Manifest is the well-defined interface of the Software Cluster's Binary Object.

The Binary Manifest provides any information, which is required to access a resource inside a Software Cluster, and to connect provided and required resources of Software Clusters. A resource, in this context, can be anything that is required to operate the software. For example, a sender receiver interface of the Software Cluster or a NV block. An obvious property of such a resource is, whether it is provided or required by a Software Cluster.

The Binary Manifest has the following core characteristics:

- The Binary Manifest gets created during the build of the Software Cluster, since it has to store information which might be build dependent (e.g. data and function addresses, ID values to use BSW APIs, attribute values, hashes)
- The Binary Manifest provides a C interface towards Software Cluster's implementation. This supports an abstraction in the other functional blocks of the Software Cluster Connection from the Binary Manifest's table implementation. The C interface is also accessible by implementations of CDDs requiring the usage of the Binary Manifest.
- The Binary Manifest defines an ECU-C interface to Software Cluster's build tools. This supports an abstract usage of the Binary Manifest by other functional blocks of the Software Cluster Connection, as well as the usage of the Binary Manifest by CDDs.

- The `Binary Manifest` format can be easily interpreted by the target `machine`. Note: In contrast to concepts for other domains (e.g. Java, Android), in this specification it is not the goal to provide a textual manifest (XML, JSON).
- An unique identifier per resource is used for the connection process. These are explicitly assigned since hash numbers are not suitable.
- The `Binary Manifest` shall have guarding information, to ensure that only compatible interfaces are getting connected. These guarding values consist of a hash over certain interface properties (see 7.3.2.2 and 7.5).

7.1.2 Logical structure of a Binary Manifest

The Figure 7.1 provides an overview of the conceptual elements of the `Binary Manifest` necessary for a fictional example resource.

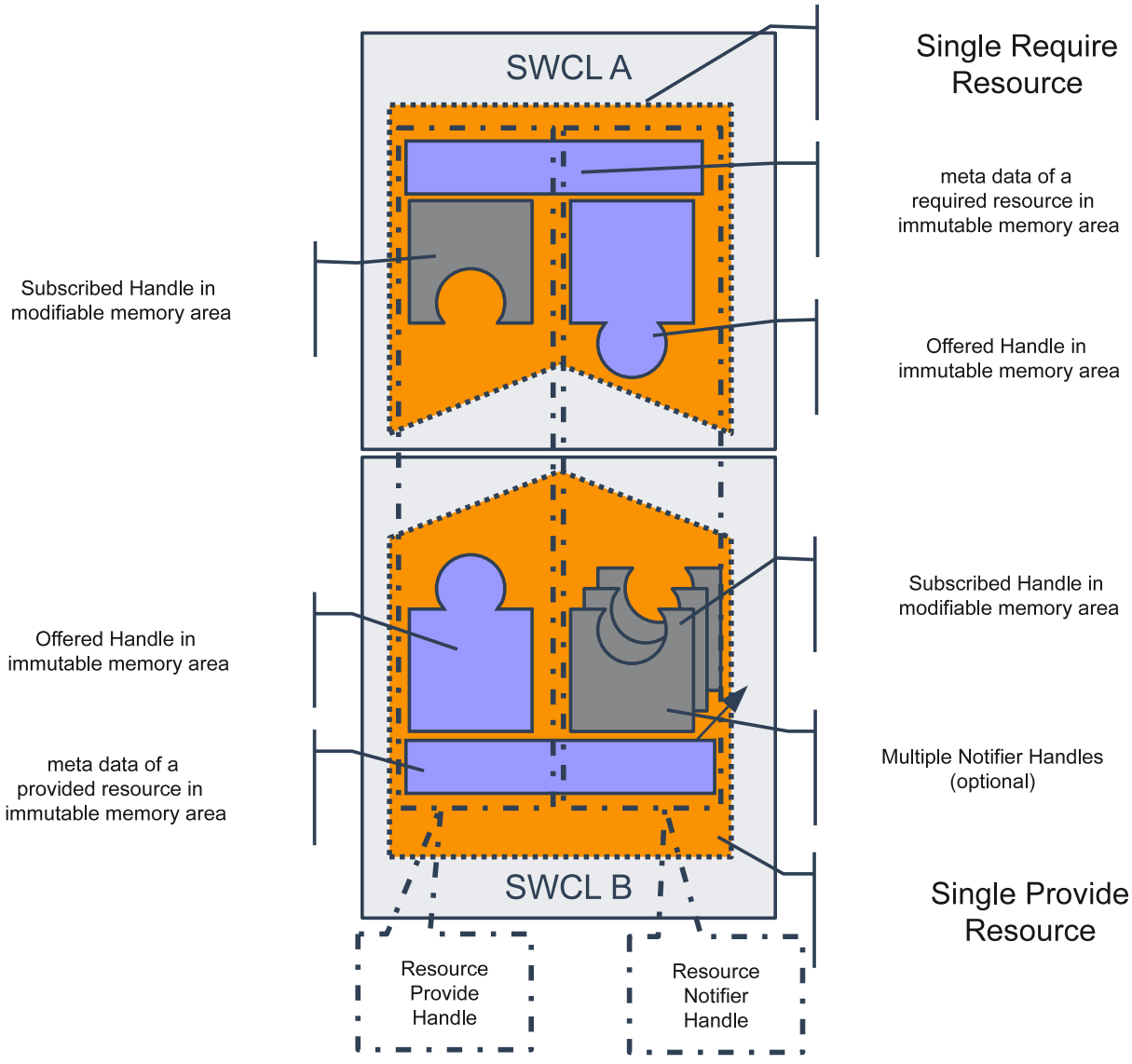

Figure 7.1: Logical structure of a Binary Manifest

The Binary Manifest provides the ability to connect required and provided resources of Software Clusters. Each resource is qualified by a set of meta data, e.g.

- the unique identifier of the resource
- the type of the resource (e.g. a sender receiver communication or connection of NV blocks)
- the nature, whether a resource is required or provided by this Software Cluster
- in case of required resources, whether it is mandatory for the operation of the Software Cluster
- guarding information (e.g. hash values) to ensure that only compatible resources are getting connected
- the number of stored handles, and where those handles can be found in the the according tables

Each type of resource is qualified by the number of handles, their purpose for the connection, and the order in which they are placed in the Binary Manifest. The handles hold the information, which the Software Cluster requesting a resource needs to access it in the providing Software Cluster.

A single handle can be

- a data pointer
- a function pointer
- a value

In some cases, a connection of a resource may require the exchange of handles in both directions. Consistently the Binary Manifest distinguishes between two types of handles on the logical level:

- Provide Handle to publish a handle from resource provider to resource requester
- Notifier Handle to publish a handle from resource requester to resource provider

An example for a Provide Handle would be a propagated API function to call an AUTOSAR Service from a Applicative Software Cluster.

An example for a Notifier Handle would be a callback notification from a connected AUTOSAR Service.

In addition, it is supported that a resource provider gets connected to multiple resource requester even if it utilizes Notifier Handles. But in this case for each potential resource requester a own set of Notifier Handles needs to be reserved for each resource requester. Hence a Notifier Handle Set holds all the Notifier Handles for one resource requester.

The Binary Manifest is composed of the following parts:

- the Binary Manifest Header. It is the central entry point, provides administrative data and references to the other parts of the manifest.
- the Interface Descriptor Table. It contains one row of meta information per Resource Entry (required or provided interface) - unique ID, properties to operate the resource at run-time, hash values about interface characteristics and semantics.
- the Offered Interface. It contains all handles that are offered to other clusters - as explained above, either because a resource is provided, or because a required resource also needs a handle in the other direction.
- the Subscribed Interface. Vice versa, it contains all handles that this Software Cluster holds that were offered by other Software Clusters
- checksums and markers (Immutable Tables Checksum, Subscribed Interface Validity Marker, Total Manifest Checksum)

Some parts of the Binary Manifest are fixed when building a Software Cluster, called immutable, while others have to be changed in the connection process, called modifiable:

- immutable
  - the Binary Manifest Header
  - the Interface Descriptor Table
  - offered handles
  - the default value for subscribed handles, to support operation in the unconnected state
- modifiable To establish a connection, this memory area has to store the handles from other SWCLs. Additionally, information might be stored to identify whether the resource is connected at all (only defaults are visible) and to which SWCL.

7.1.3 Mapping between Logical structure and Configuration structure

The logical structure shown in Figure 7.1 is also reflected in the configuration structure of the Binary Manifest.

The set of handles, relevant to connect one type of resource, is defined in the SwCluCBManifResourceType. A Resource Type is characterized by a defined set of Provide Handles and Notifier Handles, with specific types in a well defined order. This set of handles needs to be uniform for all resources of the same Resource Type, in order to be connectable cross Software Clusters.

Each required or provided resource of this Software Cluster corresponds to one Resource Entry in the Binary Manifest.

One Resource Entry corresponds to exactly one row in the Interface Descriptor Table, plus one or several handles in the Offered Interface and / or Subscribed Interface. Depending whether the resource is required or provided, Provide Handles and Notifier Handles are implemented in the Software Cluster or expected to be set by the other Software Cluster. Please note as well Table 7.1.

Therefore, the configuration provides two distinct definitions for Provide Resource Entrys as SwCluCBManifProvideResourceEntry and Require Resource Entrys as SwCluCBManifRequireResourceEntry.

From a Binary Manifest user perspective, it is an important use case to iterate over uniform Resource Entrys by an index. For this purpose, the configuration provides the ability to group Resource Entrys of the same Resource Type in Resource Entry Groups. A Provide Resource Entry Group, being a group of Provide Resource Entrys, is represented as SwCluCBManifProvideResourceEntryGroup. Correspondingly, a Require Resource Entry Group, being a group of Require Resource Entrys, is represented as SwCluCBManifRequireResourceEntryGroup.

In this and the previous section, three pairs of terms have been introduced, for three different logical layers within the Binary Manifest:

1. provide and require resources
2. Provide Handles and Notifier Handles
3. offered and subscribed handles

A single connection (between provide and require resource, 1), consists of one or more logical channels (provide and notifier handles, 2), which are implemented in the binary manifest (offered and subscribed handles, 3).

An analogy might help to clarify these terms.

Consider a satellite receiver connected to a TV via a cable, which internally contains many wires. The satellite receiver provides 'video', the TV requests it (provide and require resource). To make a single connection between provide and require side, multiple separate data flows (provide and notifier handles) are required. Some transmit the video and audio signal from the provider to the requestor, while others transmit remote control signals in the opposite direction. Both the TV and the satellite receiver have a connector with many pins (provide and notifier handles).

For the 'left audio channel' (Provide Handle), a certain pin is used. On that pin, the satellite receiver sends a signal (an offered handle implements this Provide Handle), which the TV reads from the same pin (a subscribed handle implements this Provide Handle). For the 'remote control signal' (Notifier Handle) pin, the TV sends the signal (offered handle implements the Notifier Handles), while the

satellite receiver reads the signal (subscribed handle implements the Notifier Handles).

The possible combinations of the three layers are shown in table 7.1.

7.1.4 Implementation structure of a Binary Manifest

Usually, in AUTOSAR the implementation of BSW modules is not standardized. But the various data tables of the Binary Manifest implement an interface to Software Cluster connector algorithms. Therefore, the implemented layout and semantic of the tables is standardized in this document. Furthermore, an abstract set of requirements for such connector algorithms is defined in section 7.1.5.

The Figure 7.2 provides an overview about the individual tables of the Binary Manifest, and their main relationships.

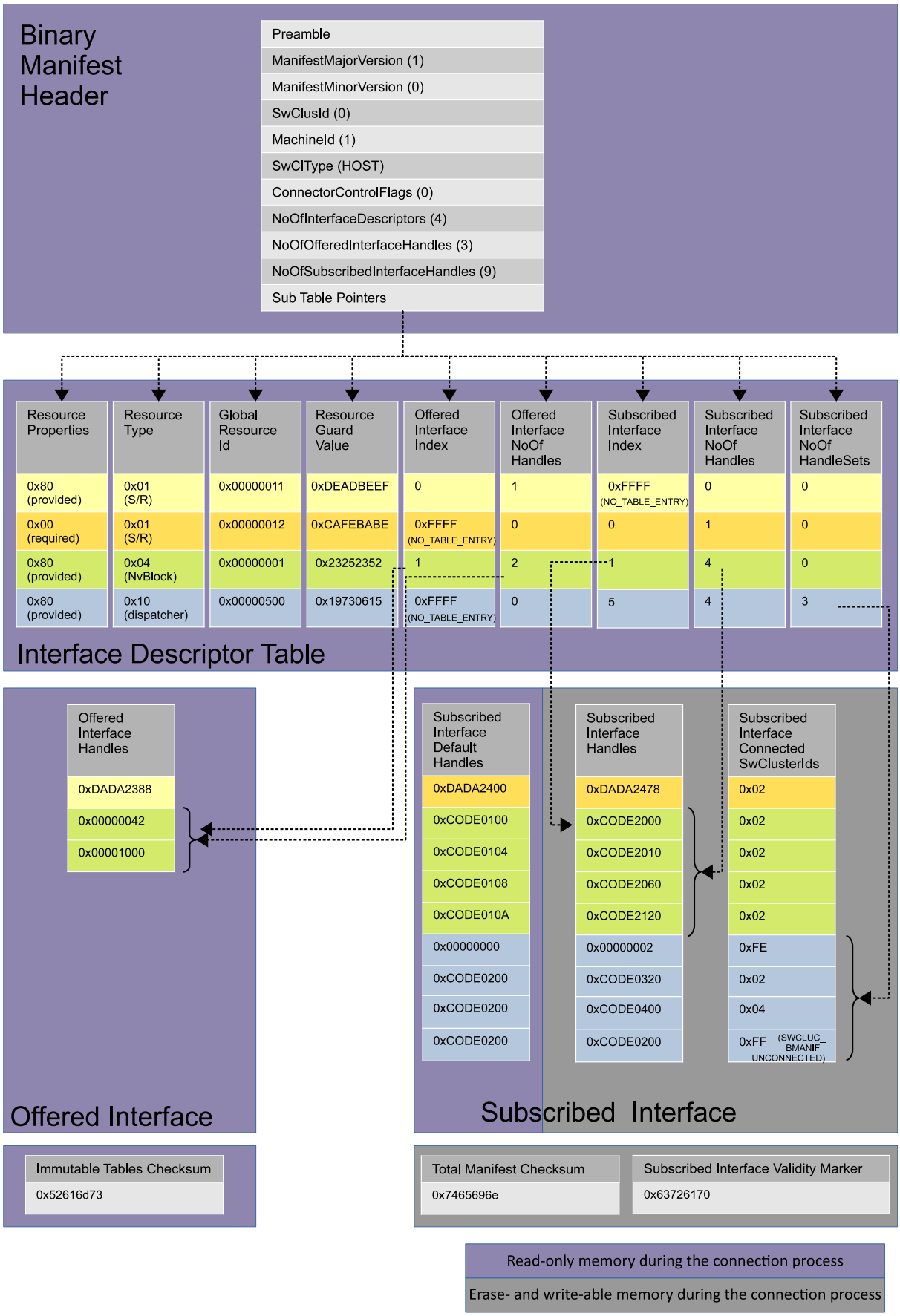

Figure 7.2: Implementation structure of a Binary Manifest

7.1.4.1 Implementation Overview

Each Software Cluster's Binary Manifest has a central entry point, called the Binary Manifest Header. It contains the AUTOSAR standardized meta data of a Software Cluster, and pointers to the sub tables (and their columns, as explained below) of the Binary Manifest. It is specified in 7.1.4.3.

Each row in the Interface Descriptor Table stores the properties of a single resource. For instance, the global resource Id. To avoid alignment gaps and padding bytes, which could occur if multiple elements are put in a structure, the table is not stored as an array of such a structure. Instead, the individual properties are organized in separate arrays. The individual arrays of the Interface Descriptor Table are required to have exactly the same number of elements and the identical order. In other words, these arrays can be seen as the columns of the Interface Descriptor Table and the identical index denotes the table row. Each row describes one resource. It is specified in 7.1.4.4.

Each of these resources can now use a dedicated number of handles in the offered and / or subscribed handle tables. The indication of the associated handles is done with the start index for the interface table(s) and the number of handles in the interface table(s). Since the number and placement of offered and subscribed handles can differ, a separate 'index' and a separate 'number of handles' for offered and subscribed handle tables exist. A special NO_TABLE_ENTRY value in the index indicates that no handle exists in that table. In this case, the number of handles is set to zero.

For example, in figure 7.2, the resource with id 0x11 (marked yellow) has 1 offered handle at index 0 and no subscribed handle (indicated by NO_TABLE_ENTRY). The resource with id 0x1 (marked green), has two handles in the offered table (starting at index 1) and four handles in the subscribed table (also starting at index 1).

The Offered Interface consists of one array - the Offered Interface Handle Column - to store the fixed handles, which are offered for the connection process (for example, in the case of a send port, the addresses of the data buffer related to the port that this software cluster provides). It is specified in 7.1.4.5

The Subscribed Interface consists out of three arrays:

- Subscribed Interface Default Handle Column offers default values for the subscribed handles, which are used in case the connection process does not have a connection partner in another Software Cluster.
- Subscribed Interface Handle Column holds the handles modified by the connection process. Those values are taken during the connection process from the Subscribed Interface Handle Column of another Software Cluster.
- The Subscribed Interface Connected SwClusterId Column holds the Software Cluster Id, from which the handle values are taken.

It is specified in 7.1.4.6.

Please note that the Subscribed Interface Handle Column and Subscribed Interface Connected SwClusterId Column need to be located in a memory area that can be erased and re-written during the connection process. In contrast, the Subscribed Interface Default Handle Column (like all the other Binary Manifest tables) needs to be located in a memory area which is read-only during the connection process. This split ensures that the connection process can be re-started at any point of time - even if the Subscribed Interface Handle Column and Subscribed Interface Connected SwClusterId Column is already erased and not yet re-written.

7.1.4.2 Multiple Notifier Sets Introduction

To support 1:n connections that require individual notifications of the requesters, the Binary Manifest supports the storage of multiple Notifier Handle Sets. In this case, the SubscribedInterfaceNoOfHandles field describes how many handles sets of other Software Clusters can be connected. Please note that the value 0 means that only a single connection is supported. 1..n means multiple connections are possible. If this is selected, the first handle entry is used to store the actual number of connected Software Clusters - similar to a dynamic length array. With this handling, it possible to preserve the MULTIPLE_NOTIFIER_SETS semantic for a Binary Manifest user, even if only at most one connection is supported.

It is also possible to connect interfaces that require more than one handle per connection. The number of handles per connection is not stored in the Binary Manifest, but if it is required, it can be calculated as (SubscribedInterfaceNoOfHandles -1) / SubscribedInterfaceNoOfHandleSets.

As an example, in figure 7.2, the resource with id 0x500 (marked blue) has 3 handle sets (the entry in the column SubscribedInterfaceNoOfHandleSets is set to 3). So at most three connections can be made to this resource. In this example, each of the three handle set consists of one handle, and since one more row is used for meta information, SubscribedInterfaceNoOfHandles is set to 3*1 + 1 = 4.

As explained above, the number of established connections is stored in the first handle entry. In this case, the entry is set to 0x2, so two connections are used and one is unused. The value 0xFE in SubscribedInterfaceConnectedClusterIds additionally indicates that this row does not contain a handle. The next two rows each show the target address of the handle and the cluster id. Since the last connection is currently unused, in the last row the handle value is set to the default value, and SubscribedInterfaceConnectedClusterIds is set to SWCLUC_BMANIF_UNCONNECTED (0xFF).

More in depth information about multiple Notifier Handle Sets can be found in 7.1.5.4. As part of that chapter, table 7.2 also shows an example with four sets, where each set consists of two handles.

7.1.4.3 Binary Manifest Header

[SWS_SwCluC_00001]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall provide exactly one instance of the Binary Manifest Header

```
const SwCluC_BManif_HeaderType SwCluC_BManif_Header =
        { <initialization> };
```

⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00002]{DRAFT} ⌈The header file SwCluC_BManif.h shall export the declaration of the Binary Manifest Header

```
extern const SwCluC_BManif_HeaderType SwCluC_BManif_Header;
```

⌋(*SRS_SwCluC_00006*)

[SWS_SwCluC_00003]{DRAFT} ⌈The element Preamble of the Binary Manifest Header shall be set to the value 0x41524350464C4558.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*, *SRS_SwCluC_00014*)

Note: The purpose of the Preamble is to serve as an obvious marker of the Binary Manifest's begin in memory. In addition, the value is chosen so that byte and word order can be detected, when the Binary Manifest is read in a byte stream or from a Binary Object file.

[SWS_SwCluC_00004]{DRAFT} ⌈The element ManifestMajorVersion of the Binary Manifest Header shall be set to the value 0x01.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00005]{DRAFT} ⌈The element ManifestMinorVersion of the Binary Manifest Header shall be set to the value 0x00.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00006]{DRAFT} ⌈The element SwClusterId of the Binary Manifest Header shall be set to the value of the configuration parameter SwCluCSoftwareClusterId of the selected SwCluCDefinition.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00007]{DRAFT} ⌈The element MachineId of the Binary Manifest Header shall be set to the value of the configuration parameter SwCluCMachineId of the selected SwCluCDefinition.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00008]{DRAFT} ⌈The element SwClusterType of the Binary Manifest Header shall be set according to the value of the configuration parameter SwCluCSoftwareClusterType of the selected SwCluCDefinition with the following encoding:

- HOST_SWCL : 0x00

- APPLICATIVE_SWCL : 0x01
- SUBSTITUTION_SWCL : 0xFF

⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*, *SRS_SwCluC_00014*, *SRS_SwCluC_00005*)

[SWS_SwCluC_00009]{DRAFT} ⌈The bit SWCLUC_BMANIF_DISABLE_ON_ECU_CONNECTION in the element ConnectorControlFlags of the Binary Manifest Header shall be set if the SwCluCBManifOnBoardConnectorControl is set to DISABLE_ON_ECU_CONNECTOR. Otherwise, the bit is not set (value 0).⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*, *SRS_SwCluC_00014*)
Note: It is not specified, if it is possible to reflash a cluster with changed addresses, how a system would detect this, and how it would behave in this case. This might change in future releases of this specification.

[SWS_SwCluC_00010]{DRAFT} ⌈The element NoOfInterfaceDescriptors of the Binary Manifest Header shall be set to the number of SwCluCBManifProvideResourceEntry + number of SwCluCBManifRequireResourceEntry containers in the configuration.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00011]{DRAFT} ⌈The element NoOfOfferedInterfaceHandles of the Binary Manifest Header shall be set to the number of offered handles.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00012]{DRAFT} ⌈The element NoOfSubscribedInterfaceHandles of the Binary Manifest Header shall be set to the number of handles in the Subscribed Interface.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00070]{DRAFT} ⌈The element ImmutableTablesChecksumPtr of the Binary Manifest Header shall reference the Immutable Tables Checksum.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00071]{DRAFT} ⌈The element TotalManifestChecksumPtr of the Binary Manifest Header shall reference the Total Manifest Checksum.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00072]{DRAFT} ⌈The element SubscribedInterfaceValidityMarkerPtr of the Binary Manifest Header shall reference the Subscribed Interface Validity Marker.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00042]{DRAFT} ⌈The elements

- ResourcePropertiesDescriptorColumnPtr
- ResourceTypeDescriptorColumnPtr

- GlobalResourceIdDescriptorColumnPtr
- ResourceGuardValueDescriptorColumnPtr
- OfferedInterfaceIndexDescriptorColumnPtr
- OfferedInterfaceNoOfHandlesDescriptorColumnPtr
- SubscribedInterfaceIndexDescriptorColumnPtr
- SubscribedInterfaceNoOfHandlesDescriptorColumnPtr
- SubscribedInterfaceNoOfHandleSetsDescriptorColumnPtr
- OfferedInterfaceHandleColumnPtr
- SubscribedInterfaceHandleDefaultColumnPtr
- SubscribedInterfaceHandleColumnPtr
- SubscribedInterfaceConnectedSwClusterIdColumnPtr

of the Binary Manifest Header shall reference the according array of the Interface Descriptor Table if the SwCluCBManifDescriptorTreatment is set to EMBED_DESCRIPTORS. Otherwise the elements are initialized to NULL_PTR.⌋*(SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00014)*

The order of the elements given by the data type SwCluC_BManif_HeaderType.

7.1.4.4 Interface Descriptor Table

[SWS_SwCluC_00015]{DRAFT} ⌈If the SwCluCBManifDescriptorTreatment is set to EMBED_DESCRIPTORS the Binary Manifest of the Software Cluster Connection shall provide exactly one instance of the Interface Descriptor Table composed out of the following column arrays:

```
1  /* descriptor table column for resource properties  */
2  const SwCluC_BManif_ResourcePropertiesType
       SwCluC_BManif_ResourcePropertiesDescriptorColumn[
       SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };
3
4  /* descriptor table column for resource type Ids  */
5  const SwCluC_BManif_ResourceTypeIdType
       SwCluC_BManif_ResourceTypeIdDescriptorColumn[
       SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };
6
7  /* descriptor table column for global resource Ids  */
8  const SwCluC_BManif_GlobalResourceIdType
       SwCluC_BManif_GlobalResourceIdDescriptorColumn[
       SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };
9
10 /* descriptor table column for guard values */
```

```
const SwCluC_BManif_ResourceGuardValueType
    SwCluC_BManif_ResourceGuardValueDescriptorColumn[
    SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };

/* descriptor table column for offered interface table index */
const SwCluC_BManif_TableIndexType
    SwCluC_BManif_OfferedInterfaceIndexDescriptorColumn[
    SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };

/* descriptor table column for number of handles in offered interface
    table*/
const SwCluC_BManif_HandleIndexType
    SwCluC_BManif_OfferedInterfaceNoOfHandlesDescriptorColumn[
    SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };

/* descriptor table column for subscribed interface table index */
const SwCluC_BManif_TableIndexType
    SwCluC_BManif_SubscribedInterfaceIndexDescriptorColumn[
    SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };

/* descriptor table column for number of handles in subscribed
    interface table*/
const SwCluC_BManif_HandleIndexType
    SwCluC_BManif_SubscribedInterfaceNoOfHandlesDescriptorColumn[
    SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };

/* descriptor table column for number of handle sets in subscribed
    interface table*/
const SwCluC_BManif_HandleIndexType
    SwCluC_BManif_SubscribedInterfaceNoOfHandleSetsDescriptorColumn[
    SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = { <initialization> };
```

⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00014*, *SRS_SwCluC_00002*, *SRS_SwCluC_00009*, *SRS_SwCluC_00010*, *SRS_SwCluC_00011*)

[SWS_SwCluC_00016]{DRAFT} ⌈For each SwCluCBManifProvideResourceEntry and SwCluCBManifRequireResourceEntry in a Software Cluster's configuration, the Binary Manifest of the Software Cluster Connection shall provide one row in the Interface Descriptor Table.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00014*)

Note: This means that each array in the Interface Descriptor Table gets one element per row. In the below requirements, the term 'element of the X column' refers to one cell of the Interface Descriptor Table at a certain row and column. In this way, the content of each cell of the Interface Descriptor Table is specified.

[SWS_SwCluC_00017]{DRAFT} ⌈The rows in the Interface Descriptor Table shall be sorted in ascending order of resource type Ids and rows with equal resource type Ids shall be sorted in turn in ascending order of global resource Ids.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00018]{DRAFT} ⌈The bit SWCLUC_BMANIF_PROVIDED_RESOURCE in the element of the 'resource properties' column shall be set, if the row be-

longs to a SwCluCBManifProvideResourceEntry.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00019]{DRAFT} ⌈The bit SWCLUC_BMANIF_MANDATORY_RESOURCE in the element of the 'resource properties' column shall be set, if the row belongs to a SwCluCBManifRequireResourceEntry where the SwCluCBManifIsMandatory is true.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00014*, *SRS_SwCluC_00009*)

[SWS_SwCluC_00020]{DRAFT} ⌈The element in the 'resource type Id' column shall be set to the SwCluCBManifResourceId of the applicable SwCluCBManifResourceType.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00014*)

Note: The applicable SwCluCBManifResourceType is referenced by the owning SwCluCBManifProvideResourceEntryGroup / SwCluCBManifRequireResourceEntryGroup

[SWS_SwCluC_00021]{DRAFT} ⌈The element in the 'global resource Id' column shall be set to the attribute value globalResourceId of the referenced CpSoftwareClusterResource as given via SwCluCBManifResourceRef. If the reference SwCluCBManifResourceRef is not set, the element in the global resource Id column shall be set to 0.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00002*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00022]{DRAFT} ⌈The element in the 'guard value' column shall be set to the SwCluCBManifResourceGuardValue of the SwCluCBManifProvideResourceEntry / SwCluCBManifRequireResourceEntry container in the configuration.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00010*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00023]{DRAFT} ⌈The element in the 'offered interface table index' column shall be set to the index of the Offered Interface where the first offered handle for this resource is allocated. In case the resource has no offered handle the value is set to NO_TABLE_ENTRY.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00024]{DRAFT} ⌈The element in the 'number of handles in offered interface table' column shall be set to number of offered handles for this resource. In case the resource has no offered handle, the value is set to 0.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00014*, *SRS_SwCluC_00003*)

[SWS_SwCluC_00025]{DRAFT} ⌈The element in the 'subscribed interface table index' column shall be set to the index of the Subscribed Interface where the first subscribed handle for this resource is allocated. In case the resource has no subscribed handle the value is set to NO_TABLE_ENTRY.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00014*)

[SWS_SwCluC_00026]{DRAFT} ⌈The element in the 'number of handles in subscribed interface table' column shall be set to number of subscribed handles for this resource. In case the resource has no subscribed handle, the value is set to 0.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00014*, *SRS_SwCluC_00003*)

[SWS_SwCluC_00027]{DRAFT} ⌈The element in the 'number of handle sets in subscribed interface table' column shall be set to the SwCluCBManifMaxNumberOfNotifierSets if the applicable SwCluCBManifResourceType has set SwCluCBManifMultipleNotifierSupport to MULTIPLE_NOTIFIER_SETS. Otherwise the value is set to 0.⌋*(SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00004)*

Depending whether a resource is provided or required by a Software Cluster, the Provide Handles and Notifier Handles need to be put either in the Offered Interface or in the Subscribed Interface. The table 7.1 defines, how many of the handles for a SwCluCBManifProvideResourceEntry / SwCluCBManifRequireResourceEntry are created in the Offered Interface or Subscribed Interface.

[SWS_SwCluC_00088]{DRAFT} **Number of offered and subscribed handles, depending on the resource direction** ⌈

	Provide Handle	Notifier Handle
Provided Resource	One handle in the Offered Interface per defined SwCluCBManifProvideHandle The handles are initialized with the SwCluCBManifProvideSymbols in the given order.	**SINGLE_NOTIFIER_SET** One handle in the Subscribed Interface per defined SwCluCBManifNotifierHandle The handles are initialized with the SwCluCBManifDefaultNotifierSymbols in the given order. The Software Cluster Ids are set to SWCLUC_BMANIF_UNCONNECTED **MULTIPLE_NOTIFIER_SETS** One handle in the Subscribed Interface per defined SwCluCBManifNotifierHandle multiplied by SwCluCBManifMaxNumberOfNotifierSets plus one The first handle is initialized to 0. The remaining handles are initialized with consecutive sets of SwCluCBManifDefaultNotifierSymbols in the given order. The Software Cluster Ids are set to SWCLUC_BMANIF_UNCONNECTED.
Required Resource	One handle in the Subscribed Interface per defined SwCluCBManifProvideHandle The handles are initialized with the SwCluCBManifDefaultProvideSymbols in the given order.	One handle in the Offered Interface per defined SwCluCBManifNotifierHandle The handles are initialized with the SwCluCBManifNotifierSymbols in the given order.

Table 7.1: Number of offered and subscribed handles, depending on the resource direction

⌋()

7.1.4.5 Offered Interface Table

[SWS_SwCluC_00028]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall provide exactly one instance of the Offered Interface Handle Column

```
const SwCluC_BManif_HandleType
    SwCluC_BManif_OfferedInterfaceHandleColumn[
    SWCLUC_BMANIF_NO_OF_OFFERED_HANDLES] = { <initialization> };
```

⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00014*, *SRS_SwCluC_00011*)

[SWS_SwCluC_00029]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall allocate and initialize the number of handles in the Offered Interface according to [SWS_SwCluC_00088].⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00014*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*)

7.1.4.6 Subscribed Interface Table

[SWS_SwCluC_00030]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall provide exactly one instance of the Subscribed Interface Default Handle Column

```
const SwCluC_BManif_HandleType
    SwCluC_BManif_SubscribedInterfaceDefaultHandleColumn[
    SWCLUC_BMANIF_NO_OF_SUBSCRIBED_HANDLES] = { <initialization> };
```

⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00014*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*, *SRS_SwCluC_00009*)

[SWS_SwCluC_00031]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall allocate and initialize the number of handles in the Subscribed Interface Handle Column according to [SWS_SwCluC_00088].⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00014*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*)

[SWS_SwCluC_00032]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall provide exactly one instance of the Subscribed Interface Handle Column

```
const  SwCluC_BManif_HandleType
    SwCluC_BManif_SubscribedInterfaceHandleColumn[
    SWCLUC_BMANIF_NO_OF_SUBSCRIBED_HANDLES] = { <initialization> };
```

⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00014*, *SRS_SwCluC_00011*, *SRS_SwCluC_00013*)

[SWS_SwCluC_00033]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall allocate and initialize the number of handles in the Subscribed Interface Handle Column according to [SWS_SwCluC_00088].⌋

(SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00011, SRS_SwCluC_00013)

[SWS_SwCluC_00034]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall provide exactly one instance of the Subscribed Interface Connected SwClusterId Column

```
const  SwCluC_BManif_SwClusterIdType
    SwCluC_BManif_SubscribedInterfaceConnectedSwClusterIdColumn[
    SWCLUC_BMANIF_NO_OF_SUBSCRIBED_HANDLES] = { <initialization> };
```

⌋*(SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00009, SRS_SwCluC_00002)*

[SWS_SwCluC_00035]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall allocate and initialize the number of handles in the Subscribed Interface Connected SwClusterId Column according to [SWS_SwCluC_00088].⌋*(SRS_SwCluC_00001, SRS_SwCluC_00014, SRS_SwCluC_00011, SRS_SwCluC_00013, SRS_SwCluC_00009, SRS_SwCluC_00002)*

7.1.4.7 Administrative Data

7.1.4.7.1 Immutable Tables Checksum

The Immutable Tables Checksum is built over all those constants of the Binary Manifest that are not changed by the Software Cluster connection step.

This includes

- the Binary Manifest Header
- all arrays of the Interface Descriptor Table
- all arrays of the Offered Interface
- the Subscribed Interface Default Handle Column

The checksum is created as part of the software build.

[SWS_SwCluC_00036]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall provide exactly one instance of the Immutable Tables Checksum

```
const uint32 SwCluC_BManif_ImmutableTablesChecksum = <initialization>;
```

⌋*(SRS_SwCluC_00001, SRS_SwCluC_00014)*

[SWS_SwCluC_00037]{DRAFT} ⌈The Immutable Tables Checksum shall be set to the value of the configuration parameter SwCluCBManifImmutableTablesChecksum.⌋*(SRS_SwCluC_00001, SRS_SwCluC_00014)*

[SWS_SwCluC_CONSTR_00073]{DRAFT} ⌈The Immutable Tables Checksum shall be calculated on the binary representation in memory of the immutable memory area (inclusive reserved memory space) in ascending order of memory address as CRC32 according to 32-bit Ethernet CRC Calculation as described in document [7].⌋*(SRS_SwCluC_00001, SRS_SwCluC_00014)*

7.1.4.7.2 Total Manifest Checksum

The Total Manifest Checksum is built over all constants of the Binary Manifest. Since the checksum calculation takes also the content of the Subscribed Interface as input, it needs to be updated after the Software Cluster connection step.

[SWS_SwCluC_00038]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall provide exactly one instance of the Total Manifest Checksum

```
const uint32 SwCluC_BManif_TotalManifestChecksum = <initialization>;
```

⌋*(SRS_SwCluC_00001, SRS_SwCluC_00014)*

[SWS_SwCluC_00039]{DRAFT} ⌈Initially (before the connection process), the Total Manifest Checksum shall be set to the value of the configuration parameter SwCluCBManifTotalManifestChecksum.⌋*(SRS_SwCluC_00001, SRS_SwCluC_00014)*

[SWS_SwCluC_CONSTR_00074]{DRAFT} ⌈The Total Manifest Checksum shall be calculated on the binary representation in memory of the immutable memory area and modifiable memory area (inclusive reserved memory space) in ascending order of memory address as CRC32, according to 32-bit Ethernet CRC Calculation as described in document [7].⌋*(SRS_SwCluC_00001, SRS_SwCluC_00014)*

7.1.4.7.3 Subscribed Interface Validity Marker

The Subscribed Interface Validity Marker indicates that all subscribed tables are written after the Software Cluster connection step. It needs to be set to the valid value after the Software Cluster connection step. The invalid /valid values are not standardized, since those values need to be chosen according to the flash technology storing the Binary Manifest.

[SWS_SwCluC_00040]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall provide exactly one instance of the Subscribed Interface Validity Marker

```
const uint32 SubscribedInterfaceValidityMarker = <initialization>;
```

⌋*(SRS_SwCluC_00001, SRS_SwCluC_00014)*

[SWS_SwCluC_00041]{DRAFT} ⌈Initially, and after a completed connection process, the Subscribed Interface Validity Marker shall be set to the value of the configuration parameter SwCluCBManifSubscribedInterfaceValidityMarker.⌋*(SRS_SwCluC_00001, SRS_SwCluC_00014)*

7.1.4.8 Memory Mapping

[SWS_SwCluC_00079]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall use the <feature> = BMANIF according to [SWS_MemMap_00040] of document [8].⌋*(SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00014)*

[SWS_SwCluC_00080]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall map the Binary Manifest Header to a constant, 32 bit aligned memory section named CONST_IMMUTABLE_HEADER.⌋*(SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00014)*

Example 7.1

```
#define SWCLUC_BMANIF_START_SEC_CONST_IMMUTABLE_HEADER_32
#include "SwCluC_MemMap.h"

const SwCluC_BManif_HeaderType SwCluC_BManif_Header = ...;

#define SWCLUC_BMANIF_STOP_SEC_CONST_IMMUTABLE_HEADER_32
#include "SwCluC_MemMap.h"
```

[SWS_SwCluC_00081]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall map the immutable columns of all tables to a constant, 32 bit aligned memory section named CONST_IMMUTABLE_COLUMNS. The immutable columns are:

- all columns of the Interface Descriptor Table
- Offered Interface Handle Column
- Subscribed Interface Default Handle Column

⌋*(SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00014)*

Example 7.2

```
#define SWCLUC_BMANIF_START_SEC_CONST_IMMUTABLE_COLUMNS_32
#include "SwCluC_MemMap.h"

const SwCluC_BManif_ResourcePropertiesType
    SwCluC_BManif_ResourcePropertiesDescriptorColumn[
    SWCLUC_BMANIF_NO_OF_DESCRIPTORS] = ...;

...

```

```
const SwCluC_BManif_HandleType
    SwCluC_BManif_SubscribedInterfaceDefaultHandleColumn[
    SWCLUC_BMANIF_NO_OF_SUBSCRIBED_HANDLES] = ...;

#define SWCLUC_BMANIF_STOP_SEC_CONST_IMMUTABLE_COLUMNS_32
#include "SwCluC_MemMap.h"
```

[SWS_SwCluC_00082]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall map the modifiable columns to a constant, 32 bit aligned memory section named CONST_MODIFIABLE_COLUMNS The modifiable columns are

- Subscribed Interface Handle Column
- Subscribed Interface Connected SwClusterId Column

.⌋*(SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00014)*

Example 7.3

```
#define SWCLUC_BMANIF_START_SEC_CONST_MODIFIABLE_COLUMNS_32
#include "SwCluC_MemMap.h"

const SwCluC_BManif_HandleType
    SwCluC_BManif_SubscribedInterfaceHandleColumn[
    SWCLUC_BMANIF_NO_OF_SUBSCRIBED_HANDLES] = ...;

const SwCluC_BManif_SwClusterIdType
    SwCluC_BManif_SubscribedInterfaceConnectedSwClusterIdColumn[
    SWCLUC_BMANIF_NO_OF_SUBSCRIBED_HANDLES] = ...;

#define SWCLUC_BMANIF_STOP_SEC_CONST_MODIFIABLE_COLUMNS_32
#include "SwCluC_MemMap.h"
```

[SWS_SwCluC_00083]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall map the Immutable Tables Checksum to a constant, 32 bit aligned memory section named CONST_IMMUTABLE_TABLES_CHECKSUM.⌋*(SRS_SwCluC_00001, SRS_SwCluC_00011, SRS_SwCluC_00014)*

Rationale: Checksums handling in memory might need a fixed location.

Example 7.4

```
#define SWCLUC_BMANIF_START_SEC_CONST_IMMUTABLE_TABLES_CHECKSUM_32
#include "SwCluC_MemMap.h"

const uint32 SwCluC_BManif_ImmutableTablesChecksum = ...;

#define SWCLUC_BMANIF_STOP_SEC_CONST_IMMUTABLE_TABLES_CHECKSUM_32
#include "SwCluC_MemMap.h"
```

[SWS_SwCluC_00084]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall map the Total Manifest Checksum to a constant,

32 bit aligned memory section named CONST_TOTAL_TABLES_CHECKSUM.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00014*)

Rationale: Checksums handling in memory might need a fixed location.

Example 7.5

```
#define SWCLUC_BMANIF_START_SEC_CONST_TOTAL_TABLES_CHECKSUM_32
#include "SwCluC_MemMap.h"

const uint32 SwCluC_BManif_TotalManifestChecksum = ...;

#define SWCLUC_BMANIF_STOP_SEC_CONST_TOTAL_TABLES_CHECKSUM_32
#include "SwCluC_MemMap.h"
```

[SWS_SwCluC_00085]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall map the Subscribed Interface Validity Marker to a constant, 32 bit aligned memory section named CONST_SUBSCRIBED_TABLES_MARKER.⌋(*SRS_SwCluC_00001*, *SRS_SwCluC_00011*, *SRS_SwCluC_00014*)

Rationale: Validity handling in memory might need a fixed location.

Example 7.6

```
#define SWCLUC_BMANIF_START_SEC_CONST_SUBSCRIBED_TABLES_MARKER_32
#include "SwCluC_MemMap.h"

const uint32 SwCluC_BManif_SubscribedInterfaceValidityMarker = ...;

#define SWCLUC_BMANIF_STOP_SEC_CONST_SUBSCRIBED_TABLES_MARKER_32
#include "SwCluC_MemMap.h"
```

7.1.5 Connecting Software Clusters

7.1.5.1 Overview

The AUTOSAR Methodology supports two different approaches to connect the Binary Objects of the different Software Clusters.

- The Software Clusters are connected after the individual and unconnected Binary Objects are programmed on the target ECU, here called On-board.
- The Software Clusters are connected before the Binary Objects are programmed on the target ECU outside the target machine, here called Off-board.

In the On-board case, the On-board Software Cluster Connector is a software, which is executed on the target ECU during the reprogramming phase. After re-

programming one or more unconnected Binary Objects, the Software Cluster Connector stores the connection result into the modifiable area of the Subscribed Interface of the Binary Manifest.

Please note: The design of the Binary Manifest assumes that a connection phase might get aborted at any point in time (e.g. power loss), and a restart or recovery of the connection phase should be supported.

In the Off-board case, the Off-board Software Cluster Connector is a tool, which is used to read the files of the Binary Objects, and to store the result of the connection in files. When reprogramming the ECU, already connected Binary Objects are used. Such an Off-board process supports additionally the variant to store meta data contained in the Interface Descriptor Table outside the Binary Objects, with the benefit that no ROM for this meta data is occupied.

Most requirements given in the section are kept universal, and can be applied for the On-board and Off-board case. If requirements are specific to one case, this is indicated in the requirement. Since AUTOSAR does not standardize the architecture of a flash boot loader, no further interfaces for the On-board Software Cluster Connector are standardized.

Some general remarks on machines:

Direct connections between Software Clusters based on the Binary Manifest are only made locally (on the same machine). Connecting Software Clusters across 'virtual' machines on the same micro controller is not possible, as it would break the separtion between the virtual machines.

7.1.5.2 Connecting Resources

[SWS_SwCluC_00043]{DRAFT} ⌈The Software Cluster Connector shall connect a required resource with a provided resource, if all of the following conditions are fulfilled:

- the machine Id of the Software Cluster providing the resource and the machine Id of the Software Cluster(s) requesting the resource are identical
- the global resource Id of provided resource and required resource(s) are equal
- the global resource Id is not 0
- the resource type Id of provided resource and required resource is equal
- the guard value of provided resource and required resource is equal
- the number of Provide Handles of provided resource and required resource is equal
- the number of Notifier Handles of provided resource and required resource is equal

- in case Notifier Handles exist, the provided resource has a Notifier Handle Sets available for the required resource

⌋(*SRS_SwCluC_00002*)

Note: The global resource Id = 0 is reserved to indicate the absence of a valid Id. The Software Cluster Connector shall only connect if the global resource Id != 0.

7.1.5.3 Handling of the Substitution Software Cluster

The purpose of the Substitution Software Cluster is to temporarily overwrite the output of a Software Cluster for development purposes. For instance, if a sender receiver signal 'Y' of Software Cluster A does not have the right functional behavior, it is possible to add a Substitution Software Cluster which provides a fixed version of sender receiver signal 'Y'. The fixed version of sender receiver signal 'Y' is provided with the identical global resource Id. In such a case, all Software Clusters requiring the sender receiver signal 'Y' are getting connected to the Substitution Software Cluster instead of the original providing Software Cluster. If the Substitution Software Cluster additionally requires the sender receiver signal 'Y', it gets connected to the original providing Software Cluster. With this principle, it is possible to take the original version of the signal as a basis for the overwriting one.

The utilization of a Substitution Software Cluster is only intended for development purposes, and not supported in productive systems. It is intended as a quick means for problem solving during development (for example, if a signal that is provided by a third party needs to be modified). At a machine, at most one Substitution Software Cluster is supported to avoid ambiguity about the effective resource provider.

[SWS_SwCluC_CONSTR_00087]{DRAFT} ⌈At a machine, at most one Substitution Software Cluster shall exist.⌋(*SRS_SwCluC_00005*)

[SWS_SwCluC_00054]{DRAFT} ⌈In case a resource is provided by the Substitution Software Cluster, the Software Cluster Connector shall connect all Software Clusters requiring the resource, except the Substitution Software Cluster itself, to the resource of the Substitution Software Cluster. This happens, regardless if another Software Cluster provides the resource as well. Nevertheless, the matching criteria defined in [SWS_SwCluC_00043] apply.⌋(*SRS_SwCluC_00005*)

[SWS_SwCluC_00055]{DRAFT} ⌈In case a resource is required by a Substitution Software Cluster, the Software Cluster Connector shall connect this required resource only to provided resources of Software Clusters not being a Substitution Software Cluster. Nevertheless, the matching criteria defined in [SWS_SwCluC_00043] apply.⌋(*SRS_SwCluC_00005*)

7.1.5.4 Multiple Notifier Sets

Note: A provided resource can optionally have the ability to handle multiple notifier sets. In this case, the number of handle sets in the Interface Descriptor Table indicates how many handle sets the Software Cluster Connector can register at most. The first handle in the Subscribed Interface Handle Column provides the number of actual registered Notifier Handle Sets. The subsequent handles in the Subscribed Interface Handle Column are store N sets of handles in a consecutive order. The order of the different handle sets written by the Software Cluster Connector are determined by the Software Cluster Id.

[SWS_SwCluC_00045]{DRAFT} ⌈In case the provided resource supports multiple Notifier Handle Sets, the Software Cluster Connector shall sort the Notifier Handle Sets in ascending order of Software Cluster Id of the according required resource(s), and place them in the Subscribed Interface Handle Column of the providing Software Cluster, starting at the second handle.⌋*(SRS_SwCluC_00004)*

[SWS_SwCluC_00046]{DRAFT} ⌈In case the provided resource supports multiple Notifier Handle Sets, the Software Cluster Connector shall write as first entry the actual number of used Notifier Handle Sets in the Subscribed Interface Handle Column.⌋*(SRS_SwCluC_00004)*

Note: the actual number of used Notifier Handle Sets is equal to the actual number of connected required resources, since each resource occupies one Notifier Handle Set.

[SWS_SwCluC_00089]{DRAFT} ⌈In case the provided resource supports multiple Notifier Handle Sets, the Software Cluster Connector shall write as first entry in the Subscribed Interface Connected SwClusterId Column the value 0xFE.⌋*(SRS_SwCluC_00004)*

Note: the value 0xFE indicates that the corresponding cell in the Subscribed Interface Handle Column holds the actual number of used notifier sets.

[SWS_SwCluC_00047]{DRAFT} ⌈In case the provide resource supports multiple Notifier Handle Sets, and less required resources are connected, the Software Cluster Connector shall fill the remaining handles in the Subscribed Interface Handle Column with the default values from the Subscribed Interface Default Handle Column of the Software Cluster providing the resource.⌋*(SRS_SwCluC_00004)*

The example in table 7.2 shows a connection of a resource provider offering at most 4 Notifier Handle sets. The resource type uses two Notifier Handles. The provided resource is connected to two different Software Clusters, and the remaining two Notifier Handle sets are filled with defaults.

Subscribed Interface Handle Column	Subscribed Interface Connected SwClusterId Column	remark
2	0xFE	actual number of used notifier sets, entry in Subscribed Interface Connected SwClusterId Column is set to 0
0xABCDABCD	2	first handle of required resource in Software Cluster 2
0xABC04711	2	second handle of required resource in Software Cluster 2
0xDADA0010	5	first handle of required resource in Software Cluster 5
0xDADA0020	5	second handle of required resource in Software Cluster 5
0xC0CAC01A	0xFF	first default handle of resource provider, entry in Subscribed Interface Connected SwClusterId Column is set to SWCLUC_BMANIF_UNCONNECTED
0xADD511FE	0xFF	second default handle of resource provider, entry in Subscribed Interface Connected SwClusterId Column is set to SWCLUC_BMANIF_UNCONNECTED
0xC0CAC01A	0xFF	first default handle of resource provider, entry in Subscribed Interface Connected SwClusterId Column is set to SWCLUC_BMANIF_UNCONNECTED
0xADD511FE	0xFF	second default handle of resource provider, entry in Subscribed Interface Connected SwClusterId Column is set to SWCLUC_BMANIF_UNCONNECTED

Table 7.2: Example of multiple notifier set connection

7.1.5.5 Unconnected Resources

[SWS_SwCluC_00044]{DRAFT} ⌈The Software Cluster Connector shall copy the Subscribed Interface Default Handle Column entries to the Subscribed Interface Handle Column, and set the related entries in the Subscribed Interface Connected SwClusterId Column to SWCLUC_BMANIF_UNCONNECTED for

- all Provide Handles of any required resources, for which no connect was applied

  AND

- all Notifier Handles of any provided resources, for which no connect was applied

⌋(*SRS_SwCluC_00009*)

7.1.5.6 Disabling of the On-board Software Cluster Connection

In general, it is possible that an ECU is equipped with an On-board Software Cluster Connector but this connector should not be used or shall only be used on specific machines.

One reason to do so is a configuration where the software on one machine requires a certification applied on the whole machine's software whereas the software on the other machine can be updated cluster-wise.

For example, such a configuration uses two machines on the same microcontroller, here called Machine A and Machine B.

Machine A needs a certification, which is only granted for the whole software on the machine. Nevertheless, the software is developed by multiple parties, with the means of Software Clusters. The Software Clusters still needs to be connected, which is done with the Off-board Software Cluster Connector before the certification is done.

For Machine B, this restriction does not apply. Here, the On-board Software Cluster Connector shall be used to do some in-field partial updates which requires a working On-board Software Cluster Connector. Since the flash-boot-loader for the whole microcontroller still has to work, the On-board Software Cluster Connector is disabled for Machine A, in order to protect it from unintended changes, caused by an On-board Software Cluster Connector run intended for Machine B.

[SWS_SwCluC_00056]{DRAFT} ⌈If the bit SWCLUC_BMANIF_DISABLE_ON_ECU_CONNECTION in the ConnectorControlFlags of the Binary Manifest Header of the Host Software Cluster is set, the Software Cluster Connection On-board for this machine is not executed.⌋*()*

7.1.5.7 Errors during software cluster connection

In some cases, the information inside the Binary Manifests might be incompatible with the cluster connector, be incomplete or inconsistent. For example, reading a Binary Manifest, which is structurally incompatible to the Software Cluster Connector's implementation, may have completely undefined results. This section lists a set of conditions that lead to an abort of the connection process.

The behavior of the system after an abort is currently not specified. The implementation of recovery mechanisms is advisable.

[SWS_SwCluC_00048]{DRAFT} ⌈The Software Cluster Connector shall detect, if the ManifestMajorVersion and ManifestMinorVersion numbers given in the Binary Manifest Header are compatible to the Software Cluster Connector's implementation. In case of any incompatible version number, the connection process for this specific machine shall be aborted.⌋*()*

If the Software Cluster Ids are not unique on the same machine, the Subscribed Interface Connected SwClusterId Column cannot be set unambiguous:

[SWS_SwCluC_00049]{DRAFT} ⌈The Software Cluster Connector shall detect, if the Software Cluster Ids of all Software Clusters belonging to same machine are not unique. In this case, the connection process for this specific machine shall be aborted⌋*()*

If a Software Cluster requires a mandatory resource that no other Software Cluster provides, the operation of the Software Cluster is not possible.

[SWS_SwCluC_00050]{DRAFT} ⌈The Software Cluster Connector shall detect, if for a Software Cluster any mandatory required resource is not provided by any other Software Cluster. In this case, the connection process for this specific Software Cluster shall be aborted.⌋*()*

In general, a resource shall only be provided once. The only exception is a second provision by the Substitution Software Cluster, which overrides the provision by another Software Cluster.

[SWS_SwCluC_00051]{DRAFT} ⌈The Software Cluster Connector shall detect, if for a machine a resource with the identical resource type id and identical global resource Id is provided twice by Software Clusters, none of which is a Substitution Software Cluster. In this case, the connection process for this specific machine shall be aborted.⌋*()*

[SWS_SwCluC_00052]{DRAFT} ⌈In case the resource does not support multiple notifier sets, but uses Notifier Handles, the Software Cluster Connector shall detect if for a machine a resource with the identical resource type id and identical global resource Id is required more than one time. In this case, the connection process for this specific machine shall be aborted.⌋*()*

[SWS_SwCluC_00053]{DRAFT} ⌈In case the resource does support multiple notifier sets, the Software Cluster Connector shall detect if for a machine a resource with the identical resource type id and identical global resource Id is required more often as notifier sets are available. In this case, the connection process for this specific machine shall be aborted.⌋*()*

7.1.6 Software Cluster Binary Manifest Descriptor

The Software Cluster Binary Manifest Descriptor describes the Binary Manifest in the Binary Object. This is mandatory for Off-board connection, but also supporting the On-board connection, since it enables to apply checks outside the target ECU.

[SWS_SwCluC_00086]{DRAFT} ⌈The Binary Manifest of the Software Cluster Connection shall provide one instance of the CpSoftwareClusterBinaryManifestDescriptor, which holds the description of the generated Binary Manifest. This includes

- the cpSoftwareCluster reference to the given CpSoftwareCluster

- the softwareClusterId of the given CpSoftwareCluster
- the metaDataFields for IMMUTABLE_TABLES_CHECKSUM, TOTAL_MANIFEST_CHECKSUM, and SUBSCRIBED_INTERFACE_VALIDITY_MARKER
- a provideResources for each Provide Resource Entry
- a requireResources for each Require Resource Entry
- a resourceDefinitions for each Resource Type

⌋()

7.1.7 Error Classification

On source code level, the implementation of the Binary Manifest is a set of constant tables accessible with a set of C-Macros. Those in turn need to be capable to be static C-initializers. Since the constant tables cannot change at runtime, any implementation of run-time detecting error code is not useful. Therefore, the following sections do not define error codes, and are marked as not applicable.

7.1.7.1 Development Errors

Development errors are not applicable for the Binary Manifest of the Software Cluster Connection.

7.1.7.2 Runtime Errors

Runtime errors are not applicable for the Binary Manifest of the Software Cluster Connection.

7.1.7.3 Transient Faults

Transient Faults are not applicable for the Binary Manifest of the Software Cluster Connection.

7.1.7.4 Production Errors

Production Errors are not applicable for the Binary Manifest of the Software Cluster Connection.

7.1.7.5 Extended Production Errors

Extended Production Errors are not applicable for the Binary Manifest of the Software Cluster Connection.

7.2 Software Cluster Base Configuration Check

The Software Cluster Base Configuration Check provides a basic mechanism, to avoid that Applicative Software Clusters with severe configuration incompatibilities are connected to the Host Software Cluster. This mechanism utilizes the guard value and mandatory feature of the Binary Manifest. The configuration information collected in the two parameters SwCluCAutoBaseConfigDescriptor and SwCluCUserBaseConfigDescriptor is used to calculate the guard value for a mandatory connection.

In case of deviating configurations in the Host Software Cluster and Applicative Software Cluster, the connection fails. In this case, the whole Applicative Software Cluster stays unconnected, and is skipped in the connection process (treated, as if it was not present). Please note as well [SWS_SwCluC_00050].

It is implementation dependent, which configuration settings are automatically collected into the SwCluCAutoBaseConfigDescriptor. But it is strongly recommended, to consider only those ECU / machine wide settings that are functional critical, in order to avoid frequent incompatibilities. For instance, compiler settings impacting the EAIB compatibility, like fill byte usage, stack usage, machine register usage.

[SWS_SwCluC_00075]{DRAFT} ⌈In the Host Software Cluster, the Software Cluster Base Configuration Check of the Software Cluster Connection shall create one Provide Resource Entry for each configured SwCluCBaseConfigurationCheck, and add it to the Binary Manifest.⌋*()*

[SWS_SwCluC_00076]{DRAFT} ⌈In the Applicative Software Cluster, the Software Cluster Base Configuration Check of the Software Cluster Connection shall create one **mandatory** Require Resource Entry for each configured SwCluCBaseConfigurationCheck, and add them to the Binary Manifest.⌋*()*

[SWS_SwCluC_00077]{DRAFT} ⌈The SwCluCBManifResourceGuardValue of the Resource Entrys of [SWS_SwCluC_00075] and [SWS_SwCluC_00076] shall be calculated out of the given multi line strings in SwCluCAutoBaseConfigDescriptor and SwCluCUserBaseConfigDescriptor.⌋*()*

7.3 Cross Cluster Communication

7.3.1 Overview

The Cross Cluster Communication implements the VFB communication between Software Clusters. For this purpose, the Cross Cluster Communication implements a Cross Software Cluster Communication Plug-In, as specified in document [9].

Due to the intended separation of Software Clusters (e.g. well separated build units, independent timing), some VFB communication features would either become very inefficient or would break the intended separation. Hence, this specification selected a subset of the most common VFB features expected for such systems, as explained in document [1].

Those are in brief:

- 1:n last is the best sender-receiver communication
- n:1 queued sender-receiver communication
- n:1 client server communication - asynchronous server calls only
- 1:n mode switch communication
- 1:n parameter communication
- limited applicability of PortInterfaceMappings, see 7.3.2.2
- DataReceivedEvents are polled
- AsynchronousServerCallReturnsEvent are polled
- OperationInvokedEvents for Cross Cluster Communication calls are polled

When designing an implementation of a Cross Cluster Communication, this intended separation has to be considered. Since each Software Clusters is built by its own, only a limited knowledge about the other Software Clusters is in place. It is also not the target of this Cross Cluster Communication to break the goal of independent development of Software Clusters by exchange of detailed Software Clusters internal design and configuration information.

Instead, the code design patterns have to deal with limited knowledge about the communication counter part.

Due to the limited knowledge about the communication partner, the execution of code might fail in partitioned systems.

[SWS_SwCluC_CONSTR_03066]{DRAFT} ⌈The implementation of a Cross Cluster Communication of an Applicative Software Clusters shall not execute code belonging to any other Software Cluster, except from special entry

points provided by the Host Software Cluster (e.g. to implement event passing)⌋ *(SRS_SwCluC_00100)*

For this reason, SynchronousServerCallPoints are not supported for Cross Cluster Communication. Please note as well [SWS_Rte_CONSTR_0xxx2] and [SWS_Rte_CONSTR_0xxx3] in document [9].

In addition, inside a partitioned system it is not guaranteed that the RAM is writable for each Software Cluster or any given partition of a Software Cluster.

[SWS_SwCluC_CONSTR_03067]{DRAFT} ⌈The implementation of a Cross Cluster Communication of an Applicative Software Cluster shall only write into the memory of other Software Clusters, if that write does not cross a Partition border. To determine this, the Applicative Software Cluster needs to determine if the CpSoftwareClusterResource is assigned to the same partition.⌋ *(SRS_SwCluC_00100)*

Please note: It might be required to restrict some communication patterns to intra-partition communication, for example to reach a certain performance. But this decreases the overall flexibility in the design and integration of Software Clusters, and therfore should only be done in rare, exceptional cases.

7.3.2 General Requirements

7.3.2.1 Cross Software Cluster Communication Plug-In

[SWS_SwCluC_03000]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall implement a Cross Software Cluster Communication Plug-In according to document [9].⌋ *(SRS_SwCluC_00100)*

[SWS_SwCluC_03000] means among other things:

[SWS_SwCluC_03001]{DRAFT} ⌈The Vendor Specific Module Definition (see document [10]) of the Software Cluster Connection shall provide a definition of the container RteRipsPluginProps as a subContainer of the container SwCluCXcc with the shortName RteRipsPluginProps.⌋ *(SRS_SwCluC_00100)*

Please note that the shortName of the EcucContainerValue related to the RteRipsPluginProps container can be freely chosen. This name will be used to determine the name part <PlugIn> in the RTE Implementation Plug-In Services according to [SWS_Rte_70034].

[SWS_SwCluC_CONSTR_03069]{DRAFT} ⌈The EcucModuleConfigurationValues of the SwCluC module shall set the RteRipsPluginCommunicationScope to RTE_RIPS_CROSS_SW_CLUSTER_COM.⌋ *(SRS_SwCluC_00100)*

Note: It is possible to use multiple Cross Software Cluster Communication Plug-Ins in one Software Cluster, but this might be rarely used in practice.

Additionally, the Cross Cluster Communication of the Software Cluster Connection has to provide all the RTE Implementation Plug-InServices according to the given input configuration.

[SWS_SwCluC_03002]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall implement the RTE Implementation Plug-In functionality for a Communication Graph, if

- a FlatInstanceDescriptor references the Communication Graph

  AND

- and the rtePluginProps.associatedCrossSwClusterComRtePlugin references the RteRipsPluginProps container of **this** Cross Cluster Communication implementation.

⌋*(SRS_SwCluC_00100)*

7.3.2.2 Cross Cluster Communication and Binary Manifest

[SWS_SwCluC_03003]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall put one Resource Entry for each associated Communication Graph into the Binary Manifest.⌋*(SRS_SwCluC_00100)*

[SWS_SwCluC_03004]{DRAFT} ⌈The SwCluCBManifResourceRef of the Resource Entry shall reference the CpSoftwareClusterCommunicationResource which is mapped by the PortElementToCommunicationResourceMapping to the Communication Graph.⌋*(SRS_SwCluC_00100)*

[SWS_SwCluC_03005]{DRAFT} ⌈The SwCluCBManifResourceGuardValue of the Resource Entry shall be calculated out of the following properties:

- data range
- resolution
- physical meaning
- encoding
- structure in memory

⌋*(SRS_SwCluC_00010)*

[SWS_SwCluC_03033]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall normalize the numerical values of attributes, before using them for the SwCluCBManifResourceGuardValue calculation, in such a way that compatible numerical values according to [TPS_GST_02501] result in the identical guard value.⌋*(SRS_SwCluC_00010)*

Note: [SWS_SwCluC_03033] shall ensure that the calculated guard value result is independent from the number format (e.g. decimal, binary, octal and hexadecimal)

of the given attributes. For instance the values, 1, 1.0, 0x000001 are semantically equivalent, and have to result in the identical guard value contribution.

Additionally, the below requirements on the guard value calculation in some cases define default values for optional existing attributes. This ensures an identical guard value on provider an requester side, even if one side expresses the default value explicitly and the other side omits it.

Only in special cases, the shortName of AUTOSAR model elements has an explicit semantic. Hence, shortNames are not relevant for the guard value, except if it is explicitly specified.

The list of attributes relevant for the guard value, specified in the following requirements, guarantee that different implementations of Software Cluster Connections treat the same set of required and provided ports as compatible, if these conditions apply:

- the elements in the PortInterfaces are compatible according to document [11] without PortInterfaceMappings
- or if a PortInterfaceMappings would be applied, it does not result in
  - Data Scaling

    NOR
  - utilization of a SubElementMapping

    NOR
  - utilization of a ClientServerOperationMapping

    NOR
  - utilization of a ClientServerApplicationErrorMapping

    NOR
  - utilization of a ModeDeclarationMappingSet

Please note:

In case the following requirements about guard value calculation list references as attributes, the calculation has to consider the attributes of the referenced model element. The fully qualified shortName (building the reference) is not part of the guard value. For example, the attribute requiredInterface is a reference to a PortInterface. In this case, the mentioned attributes of the referenced PortInterface are relevant instead of the fully qualified shortName (e.g. /Example/PortInterfaces/MySRInterface)

7.3.2.2.1 Guard value calculation for Data Communication Graphs

[SWS_SwCluC_03034]{DRAFT} ⌈The `Cross Cluster Communication` shall use the following attributes for the `SwCluCBManifResourceGuardValue` calculation for a `Data Communication Graph`:

- `PPortPrototype.providedInterface`
- `RPortPrototype.requiredInterface`
- `CpSoftwareClusterCommunicationResource.comProps.sendIndication`
  - in case of `SenderReceiverInterface`
    - `VariableDataPrototype.type`
    - `VariableDataPrototype.swDataDefProps`
    - `SenderReceiverInterface.invalidationPolicy.handleInvalid` (if existing for the `VariableDataPrototype`, else `dontInvalidate` applies)
  - in case of `NvDataInterface`
    - `VariableDataPrototype.type`
    - `VariableDataPrototype.swDataDefProps`
  - in case of `ParameterInterface`
    - `ParameterDataPrototype.type`
    - `ParameterDataPrototype.swDataDefProps`

  Additionally, attributes of `VariableDataPrototype` / `ParameterDataPrototype.type` according to [SWS_SwCluC_03035] and [SWS_SwCluC_03036] apply.

⌋*(SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00106)*

7.3.2.2.2 Guard value calculation for Client Server Communication Graphs

[SWS_SwCluC_03047]{DRAFT} ⌈The `Cross Cluster Communication` shall use the following attributes for the `SwCluCBManifResourceGuardValue` calculation for a `Client Server Communication Graph`:

- `PPortPrototype.providedInterface`
- `RPortPrototype.requiredInterface`
  - `ClientServerOperation.arguments` {ordered}

    * ArgumentDataPrototype.type
    * ArgumentDataPrototype.direction
  - ClientServerOperation.possibleError.errorCode (if existing, else one errorCode = 0 is applied)

    Additionally, attributes of ArgumentDataPrototype.type according to [SWS_SwCluC_03035] and [SWS_SwCluC_03036] apply.

⌋(*SRS_SwCluC_00010, SRS_SwCluC_00103*)

7.3.2.2.3 Guard value calculation for Mode Communication Graphs

[SWS_SwCluC_03045]{DRAFT} ⌈The Cross Cluster Communication shall use the following attributes for the SwCluCBManifResourceGuardValue calculation for a Mode Communication Graph:

- PPortPrototype.providedInterface
- RPortPrototype.requiredInterface
- ModeDeclarationGroupPrototype.type
- ModeDeclarationGroup.category
- ModeDeclarationGroup.initialMode
- ModeDeclarationGroup.modeDeclarations
- ModeDeclarationGroup.onTransitionValue (in case ModeDeclarationGroup.category == EXPLICIT_ORDER)
- ModeDeclarationGroup.modeUserErrorBehavior (if existing)
- ModeDeclarationGroup.modeManagerErrorBehavior (if existing)
- ModeDeclarationGroup.modeTransitions (if existing) for each ModeDeclaration the following attributes apply:
  - ModeDeclaration.shortName
  - ModeDeclaration.value (in case ModeDeclarationGroup.category == EXPLICIT_ORDER)

  for each ModeErrorBehavior the following attributes apply:
  - ModeErrorBehavior.errorReactionPolicy
  - ModeErrorBehavior.defaultMode (if existing)

  for each ModeTransition the following attributes apply:
  - ModeTransition.exitedMode

  - ModeTransition.enteredMode

⌋(*SRS_SwCluC_00010*, *SRS_SwCluC_00104*)

7.3.2.2.4 Guard value calculation for Trigger Communication Graphs

[SWS_SwCluC_03046]{DRAFT} ⌈The Cross Cluster Communication shall use the following attributes for the SwCluCBManifResourceGuardValue calculation for a Trigger Communication Graph:

- PPortPrototype.providedInterface
- RPortPrototype.requiredInterface
- Trigger.swImplPolicy (if existing, else standard applies)

⌋(*SRS_SwCluC_00010*, *SRS_SwCluC_00105*)

7.3.2.2.5 Guard value calculation for DataPrototypes

[SWS_SwCluC_03035]{DRAFT} ⌈The Cross Cluster Communication shall use the following attributes for the referenced ApplicationDataType for the SwCluCBManifResourceGuardValue calculation:

- in case the AutosarDataPrototype.type refers to a ApplicationPrimitiveDataType
  - ApplicationPrimitiveDataType.category
  - ApplicationPrimitiveDataType.swDataDefProps
- in case the AutosarDataPrototype.type refers to a ApplicationPrimitiveDataType
  - ApplicationArrayDataType.dynamicArraySizeProfile (if existing)
  - ApplicationArrayDataType.element
  - ApplicationArrayElement.type
  - ApplicationArrayElement.arraySizeSemantics (if existing, else fixedSize applies)
  - ApplicationArrayElement.arraySizeHandling (if existing)
  - ApplicationArrayElement.maxNumberOfElements
- in case the AutosarDataPrototype.type refers to a ApplicationPrimitiveDataType

  - ApplicationRecordDataType.elements {ordered}
  - ApplicationRecordElement.type
  - ApplicationRecordElement.isOptional (if existing, else false applies)

Additionally, the attributes of the mapped ImplementationDataType according to [SWS_SwCluC_03036] and the attributes of SwDataDefProps according to [SWS_SwCluC_03037] apply.⌋*(SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00103, SRS_SwCluC_00106)*

[SWS_SwCluC_03036]{DRAFT} ⌈The Cross Cluster Communication shall use the following attributes for the referenced or mapped ImplementationDataType for the SwCluCBManifResourceGuardValue calculation:

- in case the AutosarDataPrototype.type refers to an ImplementationDataType of category TYPE_REFERENCE or the ApplicationDataType according to [SWS_SwCluC_03036] is mapped to an ImplementationDataType of category TYPE_REFERENCE
  - ImplementationDataType.swDataDefProps

  in case the AutosarDataPrototype.type refers to an ImplementationDataType of category VALUE or the ApplicationDataType according to [SWS_SwCluC_03036] is mapped to an ImplementationDataType of category VALUE
  - ImplementationDataType.swDataDefProps
- in case the AutosarDataPrototype.type refers to an ImplementationDataType of category ARRAY or the ApplicationDataType according to [SWS_SwCluC_03036] is mapped to an ImplementationDataType of category ARRAY
  - ImplementationDataType.dynamicArraySizeProfile (if existing)
  - ImplementationDataType.swDataDefProps
  - ImplementationDataType.subElement
  - ImplementationDataTypeElement.arraySizeSemantics (if existing, else fixedSize applies)
  - ImplementationDataTypeElement.arraySizeHandling (if existing)
  - ImplementationDataTypeElement.swDataDefProps
- in case AutosarDataPrototype.type refers to an ImplementationDataType of category STRUCTURE or the ApplicationDataType according to [SWS_SwCluC_03036] is mapped to an ImplementationDataType of category STRUCTURE
  - ImplementationDataType.swDataDefProps

- ImplementationDataType.isStructWithOptionalElement (if existing, else false applies)
- ImplementationDataType.subElement {ordered}
- ImplementationDataTypeElement.arraySizeSemantics (if existing, else fixedSize applies)
- ImplementationDataTypeElement.arraySizeHandling (if existing)
- ImplementationDataTypeElement.swDataDefProps

Additionally, the attributes of SwDataDefProps according to [SWS_SwCluC_03037] apply.

⌋(*SRS_SwCluC_00010*, *SRS_SwCluC_00101*, *SRS_SwCluC_00102*, *SRS_SwCluC_00103*, *SRS_SwCluC_00106*)

[SWS_SwCluC_03037]{DRAFT} ⌈The Cross Cluster Communication shall use the following attributes for the given SwDataDefProps for the SwCluCBManifResourceGuardValue calculation:

- SwDataDefProps.additionalNativeTypeQualifier (if existing)
- SwDataDefProps.swImplPolicy (if existing, else standard applies)
- SwDataDefProps.invalidValue (if existing)
- SwDataDefProps.unit (if existing, else CompuMethod.unit applies)
- SwDataDefProps.dataConstr (if existing)
- SwDataDefProps.compuMethod (if existing)
- SwDataDefProps.swTextProps (if existing)
- SwDataDefProps.baseType (if existing)
- SwDataDefProps.swValueBlockSize (if existing)
- SwDataDefProps.swValueBlockSizeMult (if existing)
- SwDataDefProps.valueAxisDataType (if existing)
- SwDataDefProps.implementationDataType (if existing)

Here, the prioritization of SwDataDefProps attributes according to [constr_1015] applies. Only the most prior SwDataDefProps attributes are considered.

Additionally, the attributes of Unit, CompuMethod, DataConstr, and BaseType according to [SWS_SwCluC_03038], [SWS_SwCluC_03039], [SWS_SwCluC_03042], [SWS_SwCluC_03043], and [SWS_SwCluC_03044] apply.

⌋(*SRS_SwCluC_00010*, *SRS_SwCluC_00101*, *SRS_SwCluC_00102*, *SRS_SwCluC_00103*, *SRS_SwCluC_00106*)

[SWS_SwCluC_03038]{DRAFT} ⌈The `Cross Cluster Communication` shall use the following attributes for the given `Unit` for the `SwCluCBManifResourceGuardValue` calculation:

- `Unit.factorSiToUnit` (if existing, else `1` applies)
- `Unit.offsetSiToUnit` (if existing, else `0` applies)
- `Unit.physicalDimension` (if existing, else [SWS_SwCluC_03041] applies)

Additionally, the attributes of `PhysicalDimension` according to [SWS_SwCluC_03040] and [SWS_SwCluC_03041] apply.⌋*(SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00103, SRS_SwCluC_00106)*

[SWS_SwCluC_03039]{DRAFT} ⌈In case **no** `Unit` is given for the `SwCluCBManifResourceGuardValue` calculation, the `Cross Cluster Communication` shall use the following attribute values:

- `Unit.factorSiToUnit` = `1`
- `Unit.offsetSiToUnit` = `0`
- `Unit.physicalDimension` according to [SWS_SwCluC_03041]

Additionally, the attributes of `PhysicalDimension` according to [SWS_SwCluC_03041] apply.⌋*(SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00103, SRS_SwCluC_00106)*

[SWS_SwCluC_03040]{DRAFT} ⌈The `Cross Cluster Communication` shall use the following attributes for the given `PhysicalDimension` for the `SwCluCBManifResourceGuardValue` calculation:

- `PhysicalDimension.shortName`
- `PhysicalDimension.currentExp` (if existing, else `0` applies)
- `PhysicalDimension.lengthExp` (if existing, else `0` applies)
- `PhysicalDimension.luminousIntensityExp` (if existing, else `0` applies)
- `PhysicalDimension.massExp` (if existing, else `0` applies)
- `PhysicalDimension.molarAmountExp` (if existing, else `0` applies)
- `PhysicalDimension.temperatureExp` (if existing, else `0` applies)
- `PhysicalDimension.timeExp` (if existing, else `0` applies)

⌋*(SRS_SwCluC_00010, SRS_SwCluC_00101, SRS_SwCluC_00102, SRS_SwCluC_00103, SRS_SwCluC_00106)*

[SWS_SwCluC_03041]{DRAFT} ⌈In case **no** `PhysicalDimension` is given, the `Cross Cluster Communication` shall use the following attribute values for the `SwCluCBManifResourceGuardValue` calculation:

- PhysicalDimension.shortName = NoDimension
- PhysicalDimension.currentExp = 0
- PhysicalDimension.lengthExp = 0
- PhysicalDimension.luminousIntensityExp = 0
- PhysicalDimension.massExp = 0
- PhysicalDimension.molarAmountExp = 0
- PhysicalDimension.temperatureExp = 0
- PhysicalDimension.timeExp = 0

⌋(*SRS_SwCluC_00010*, *SRS_SwCluC_00101*, *SRS_SwCluC_00102*, *SRS_SwCluC_00103*, *SRS_SwCluC_00106*)

The guard value calculation has to consider the intended compatibility of CompuMethod with different category, as defined in document [11] with [constr_1176] and [constr_1192]. Please note as well [constr_1375] w.r.t the existence of attributes of CompuMethods. Nevertheless, not all attributes are considered for the guard value, in order to increase compatibility between provider and requesters.

[SWS_SwCluC_03042]{DRAFT} ⌈The Cross Cluster Communication shall use the following attributes for the given CompuMethod for the SwCluCBManifResourceGuardValue calculation:

- CompuMethod.unit
- in case the CompuMethod.category is IDENTICAL
  - it is treated like a CompuMethod where the category is LINEAR where the CompuMethod.compuPhysToInternal.compuScale yield the conversion

    $int = \frac{N_0+N_1*phys}{D_0}$ if $N_0 \sim 0$ && $N_1 \sim 1$ && $D_0 \sim 0$
- in case the CompuMethod.category is LINEAR
  - CompuMethod.compuPhysToInternal.compuScale

    OR
  - CompuMethod.compuInternalToPhys.compuScale

    where the compuInternalToPhys coefficients are converted to equivalent compuPhysToInternal coefficients.
    - CompuScale.compuNumerator.vs {ordered}
    - CompuScale.compuDenominator.v
- in case the CompuMethod.category is SCALE_LINEAR

  - CompuMethod.compuPhysToInternal.compuScales
  - CompuMethod.compuPhysToInternal.compuDefaultValue (if existing)

    OR

  - CompuMethod.compuInternalToPhys.compuScales
  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)
    * CompuScale.upperLimit
    * CompuScale.lowerLimit
    * CompuScale.compuNumerator.vs {ordered}
    * CompuScale.compuDenominator.v

- in case the CompuMethod.category is RAT_FUNC with 2 compuNumerator coefficients and 1 compuDenominator coefficient
  - this case is treated like a CompuMethod where the category is LINEAR
- in case the CompuMethod.category is RAT_FUNC not matching the LINEAR case
  - CompuMethod.compuPhysToInternal.compuScale
  - CompuMethod.compuPhysToInternal.compuDefaultValue (if existing)

    AND

  - CompuMethod.compuInternalToPhys.compuScale
  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)
    * CompuScale.upperLimit
    * CompuScale.lowerLimit
    * CompuScale.compuInverseValue.vf (if existing)
    * CompuScale.compuNumerator.vs {ordered}
    * CompuScale.compuDenominator.v {ordered}
- in case the CompuMethod.category is SCALE_RAT_FUNC
  - CompuMethod.compuPhysToInternal.compuScales
  - CompuMethod.compuPhysToInternal.compuDefaultValue (if existing)

AND

  - CompuMethod.compuInternalToPhys.compuScales
  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)
    * CompuScale.upperLimit
    * CompuScale.lowerLimit
    * CompuScale.compuInverseValue.vf (if existing)
    * CompuScale.compuNumerator.vs {ordered}
    * CompuScale.compuDenominator.v {ordered}

- in case the CompuMethod.category is TEXTTABLE
  - CompuMethod.compuInternalToPhys.compuScales
  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)
    * CompuScale.upperLimit
    * CompuScale.lowerLimit
    * CompuScale.compuInverseValue.vf (if existing)
    * CompuScale.compuConst.vt
- in case the CompuMethod.category is BITFIELD_TEXTTABLE
  - CompuMethod.compuInternalToPhys.compuScales
  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)
    * CompuScale.upperLimit
    * CompuScale.lowerLimit
    * CompuScale.mask
    * CompuScale.compuInverseValue.vf (if existing)
    * CompuScale.compuConst.vt
- in case the CompuMethod.category is SCALE_LINEAR_AND_TEXTTABLE
  - CompuMethod.compuInternalToPhys.compuScales
  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)
    * CompuScale.upperLimit

    * CompuScale.lowerLimit
    * CompuScale.mask
    * CompuScale.compuInverseValue.vf (if existing)
    * CompuScale.compuConst.vt (if existing)
    * CompuScale.compuNumerator.vs {ordered} (if existing)
    * CompuScale.compuDenominator.v (if existing)

- in case the CompuMethod.category is SCALE_RATIONAL_AND_TEXTTABLE
  - CompuMethod.compuPhysToInternal.compuScales
  - CompuMethod.compuPhysToInternal.compuDefaultValue (if existing)

    AND
  - CompuMethod.compuInternalToPhys.compuScales
  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)
    * CompuScale.upperLimit
    * CompuScale.lowerLimit
    * CompuScale.mask
    * CompuScale.compuInverseValue.vf (if existing)
    * CompuScale.compuConst.vt (if existing)
    * CompuScale.compuNumerator.vs {ordered} (if existing)
    * CompuScale.compuDenominator.v {ordered} (if existing)
- in case the CompuMethod.category is TAB_NOINTP
  - CompuMethod.compuInternalToPhys.compuScales
  - CompuMethod.compuInternalToPhys.compuDefaultValue (if existing)
    * CompuScale.upperLimit
    * CompuScale.lowerLimit
    * CompuScale.compuInverseValue.vf (if existing)
    * CompuScale.compuConst.vt (if existing)

  Additionally, the attributes of Unit according to [SWS_SwCluC_03038], [SWS_SwCluC_03039] apply.

⌋(*SRS_SwCluC_00010*, *SRS_SwCluC_00101*, *SRS_SwCluC_00102*, *SRS_SwCluC_00103*, *SRS_SwCluC_00106*)

[SWS_SwCluC_03043]{DRAFT} ⌈The `Cross Cluster Communication` shall use the following attributes for the given `DataConstr` for the `SwCluCBManifResourceGuardValue` calculation:

- `DataConstr.dataConstrRule.physConstrs.upperLimit` (if existing)
- `DataConstr.dataConstrRule.physConstrs.lowerLimit` (if existing)
- `DataConstr.dataConstrRule.physConstrs.monotony` (if existing)
- `DataConstr.dataConstrRule.physConstrs.maxGradient` (if existing)
- `DataConstr.dataConstrRule.physConstrs.maxDiff` (if existing)
- `DataConstr.dataConstrRule.internalConstrs.upperLimit` (if existing, else based on `baseTypeEncoding` and `baseTypeSize` the technical upper limit is taken)
- `DataConstr.dataConstrRule.internalConstrs.lowerLimit` (if existing, else based on `baseTypeEncoding` and `baseTypeSize` the technical lower limit is taken)
- `DataConstr.dataConstrRule.internalConstrs.monotony` (if existing, else `noMonotony`)
- `DataConstr.dataConstrRule.internalConstrs.maxGradient` (if existing)
- `DataConstr.dataConstrRule.internalConstrs.maxDiff` (if existing)

⌋(*SRS_SwCluC_00010*, *SRS_SwCluC_00101*, *SRS_SwCluC_00102*, *SRS_SwCluC_00103*, *SRS_SwCluC_00106*)

[SWS_SwCluC_03044]{DRAFT} ⌈The `Cross Cluster Communication` shall use the following attributes for the given `SwBaseType` for the `SwCluCBManifResourceGuardValue` calculation:

- `SwBaseType.baseTypeEncoding`
- `SwBaseType.baseTypeSize`
- `SwBaseType.byteOrder`
- `SwBaseType.nativeDeclaration`
- `SwBaseType.memAlignment` (if existing, else `0` applies)

⌋(*SRS_SwCluC_00010*, *SRS_SwCluC_00101*, *SRS_SwCluC_00102*, *SRS_SwCluC_00103*, *SRS_SwCluC_00106*)

7.3.2.3 Cross Cluster Communication Base Socket

In general, it is possible that the communication between Applicative Software Clusters needs some basic infrastructure from the Host Software Cluster. For instance, to pass an activity or event between Applicative Software Clusters. Usually, such event passing requires some knowledge about the overall environment, which is **NOT** available in an Applicative Software Cluster. For instance, in the Applicative Software Cluster, only a subset of the defined partitions (and related EcucPartitions plus OsApplications) are known.

[SWS_SwCluC_03012]{DRAFT} ⌈If the implementation of a Cross Cluster Communication requires some basic infrastructure from the Host Software Cluster, it shall use the means of the SwCluCXccBaseSocket configuration to link to the resource pool.⌋*()*

[SWS_SwCluC_03013]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall put one Resource Entry for each associated SwCluCXccBaseSocket into the Binary Manifest.⌋*()*

7.3.2.4 Cross Cluster Communication and McSupport Data

[SWS_SwCluC_03071]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall provide an *MC-Support* (Measurement and Calibration) description as part of its *Basic Software Module Description*.⌋*(SRS_SwCluC_00300)*

[SWS_SwCluC_03072]{DRAFT} ⌈The McSupportData element begin part of [SWS_SwCluC_03071] and its sub-structure shall be self-contained, in the sense that there is no need to deliver the whole upstream descriptions of the ECU (including the ECU Extract, Software Component descriptions, Basic Software Module descriptions, ECU Configuration Values descriptions, etc.), in order to later generate the final "A2L"-file. This means that the Cross Cluster Communication generator has to copy the required information from the upstream descriptions into the McSupportData element.⌋*(SRS_SwCluC_00300)*

[SWS_SwCluC_03073]{DRAFT} ⌈The Cross Cluster Communication generator shall export the effective SwDataDefProps (including all of the referenced and aggregated sub-elements like e.g. CompuMethod or SwRecordLayout) in the role resultingProperties, for each McDataInstance, after resolving the precedence rules defined in the SW-Component Template [11] chapter *Properties of Data Definitions*. Thereby, the ImplementationDataType properties compuMethod and dataConstraint are not taken in consideration for effective SwDataDefProps of the McDataInstance, due to their refinement nature of **C** and **AI**.⌋*(SRS_SwCluC_00300)*

[SWS_SwCluC_03074]{DRAFT} ⌈If the parameter SwCluCXccSwCluCXccDefaultDataHandling is set to DEFAULTS_AS_CALPRMS, the Cross Cluster Communication generator shall export one entry in the McSupportData, describing

the default data instance according to [SWS_SwCluC_03120], with the role mcParameterInstance, with the following attributes:

- swCalibrationAccess set to readWrite
- effective SwDataDefProps according to [SWS_SwCluC_03073]
- McDataInstance.subElements for array elements or structure elements, if applicable
- symbol set to the C-symbol name used for the default data instance

⌋(*SRS_SwCluC_00300*)

[SWS_SwCluC_03077]{DRAFT} ⌈If the parameter SwCluCXccSwCluCXccDefaultDataHandling is set to DEFAULTS_AS_CONSTANTS, the Cross Cluster Communication generator shall export one entry in the McSupportData, describing the default data instance according to [SWS_SwCluC_03120], with the role mcParameterInstance, with the following attributes:

- swCalibrationAccess set to readOnly
- effective SwDataDefProps according to [SWS_SwCluC_03073]
- McDataInstance.subElements for array elements or structure elements, if applicable
- symbol set to the C-symbol name used for the default data instance

⌋(*SRS_SwCluC_00300*)

[SWS_SwCluC_03075]{DRAFT} ⌈The Cross Cluster Communication generator shall export one entry in the McSupportData, describing the default parameter instance according to [SWS_SwCluC_03006], with the role mcParameterInstance, with the following attributes:

- swCalibrationAccess set to readWrite
- effective SwDataDefProps according to [SWS_SwCluC_03073]
- McDataInstance.subElements for array elements or structure elements, if applicable
- symbol set to the C-symbol name used for the default parameter instance

⌋(*SRS_SwCluC_00300*)

[SWS_SwCluC_03076]{DRAFT} ⌈If the swCalibrationAccess of a VariableDataPrototype instance in the delegation PPortPrototype at the CompositionSwComponentType at the CompositionSwComponentType of the rootSoftwareComposition of the Ecu Extract is set to readOnly or readWrite, the Cross Cluster Communication generator shall export one entry in the McSupportData, describing the data instance according to [SWS_SwCluC_03101], with the role mcVariableInstance, with the following attributes:

- effective SwDataDefProps (inclusive swCalibrationAccess) according to [SWS_SwCluC_03073]
- McDataInstance.subElements for array elements or structure elements, if applicable
- symbol set to the C-symbol name used for the default parameter instance
- if applicable, flatMapEntry referencing to the corresponding FlatInstanceDescriptor element of the VariableDataPrototype

⌋(*SRS_SwCluC_00300*)

7.3.3 Sender Receiver Communication

7.3.3.1 Restrictions on VFB communication features

[SWS_SwCluC_03064]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall reject the configuration, if there is a "N:1" sender-receiver communication with last-is-the-best semantic.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03065]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall reject the configuration, if there is a "1:N" sender-receiver communication with event semantic (queued communication).⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03068]{DRAFT} ⌈The implementation of a Cross Cluster Communication shall support that Data Communication Graphs between Applicative Software Clusters can be added, without modification of the Host Software Cluster.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

7.3.3.2 Transmission

[SWS_SwCluC_03101]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall provide one data instance, for each assigned Data Communication Graph

- where the referenced VariableDataPrototype is owned by a delegation PPortPrototype at the CompositionSwComponentType at the CompositionSwComponentType of the rootSoftwareComposition of the Ecu Extract AND
- where implicit communication applies to that VariableDataPrototype OR
- where explicit unqueued communication applies to that VariableDataPrototype

⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03102]{DRAFT} ⌈The data instance created according to [SWS_SwCluC_03101] shall be mapped to a VAR memory section, according to document [8].⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03103]{DRAFT} ⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall make the `data instance` created according to [SWS_SwCluC_03101], accessible to other `Software Cluster`s via a `Resource Entry` in the `Binary Manifest`.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03106]{DRAFT} ⌈When implicit or explicit unqueued communication applies, the `Rte_Rips_Write` shall update the data instance created according to [SWS_SwCluC_03101]), and return `RTE_E_OK`.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03142]{DRAFT} ⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall provide one `Resource Entry`, for each assigned `Data Communication Graph`

- where the referenced `VariableDataPrototype` is owned by a delegation `PPortPrototype` at the `CompositionSwComponentType` at the `CompositionSwComponentType` of the `rootSoftwareComposition` of the `Ecu Extract` AND
- where explicit queued communication applies to that `VariableDataPrototype`

This `Resource Entry` shall be used to link the access of the related `Rte_Rips_Write` service to the according queue instance on the receiving `Software Cluster`.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03107]{DRAFT} ⌈When explicit queued communication applies, and the `Resource Entry` created according to [SWS_SwCluC_03142] is not connected, the `Rte_Rips_Write` shall discard the write access, and return `RTE_E_OK`.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03108]{DRAFT} ⌈When explicit queued communication applies, and the `Resource Entry` created according to [SWS_SwCluC_03142] is connected, the `Rte_Rips_Write` shall check the queue status (See [SWS_SwCluC_03135]), to determine whether or not the data can be enqueued.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03109]{DRAFT} ⌈When explicit queued communication applies, and the `Resource Entry` created according to [SWS_SwCluC_03142] is connected, and the queue is not full, the `Rte_Rips_Write` shall enqueue the data and return `RTE_E_OK` (See [SWS_SwCluC_03135]).⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03110]{DRAFT} ⌈When explicit queued communication applies, and the Resource Entry created according to [SWS_SwCluC_03142] is connected, but the queue is full, the Rte_Rips_Write shall discard the write access and return RTE_E_LIMIT (See [SWS_SwCluC_03135]).⌋*(SRS_SwCluC_00100, SRS_SwCluC_00101)*

[SWS_SwCluC_03111]{DRAFT} ⌈The data instance created according to [SWS_SwCluC_03101] shall be initialized according to the NonqueuedSenderComSpec.initValue in the PPortPrototype at the CompositionSwComponentType of the rootSoftwareComposition of the Ecu Extract.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00101)*

[SWS_SwCluC_03112]{DRAFT} ⌈If acknowledgment is enabled, and the resource created according to [SWS_SwCluC_03103] is not connected, the Rte_Rips_Feedback API shall return RTE_E_UNCONNECTED.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00101)*

[SWS_SwCluC_03113]{DRAFT} ⌈If acknowledgement is enabled, and the resource created according to [SWS_SwCluC_03103] is connected, the Rte_Rips_Feedback API shall return RTE_E_TRANSMIT_ACK.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00101)*

7.3.3.3 Reception

[SWS_SwCluC_03120]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall provide one default data instance, for each assigned Data Communication Graph

- where the referenced VariableDataPrototype is owned by a delegation RPortPrototype at the CompositionSwComponentType at the CompositionSwComponentType of the rootSoftwareComposition of the Ecu Extract AND
- where implicit communication applies to that VariableDataPrototype OR
- where explicit unqueued communication applies to that VariableDataPrototype

⌋*(SRS_SwCluC_00100, SRS_SwCluC_00101)*

[SWS_SwCluC_03133]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall provide a queue data instance (where the queue size corresponds to the specified queue length), for each assigned Data Communication Graph

- where the referenced VariableDataPrototype is owned by a delegation RPortPrototype at the CompositionSwComponentType at the CompositionSwComponentType of the rootSoftwareComposition of the Ecu Extract AND

- where explicit queued communication applies to that VariableDataPrototype

⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03134]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall determine the queue size according to the queueLength attribute of the QueuedReceiverComSpec⌋(*SRS_SwCluC_00100, SRS_SwCluC_00103*)

[SWS_SwCluC_03135]{DRAFT} ⌈When [SWS_SwCluC_03133] applies, the Cross Cluster Communication of the Software Cluster Connection shall provide a data instance, representing the number of possible entries in the queue.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03122]{DRAFT} ⌈If the parameter SwCluCXccSwCluCXccDefaultDataHandling is set to DEFAULTS_AS_CONSTANTS, the default data instance created according to [SWS_SwCluC_03120] shall be mapped to a CONST memory section, according to document [8].⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03121]{DRAFT} ⌈If the parameter SwCluCXccSwCluCXccDefaultDataHandling is set to DEFAULTS_AS_CALPRMS, the default data instance created according to [SWS_SwCluC_03120] shall be mapped to a CALPRM memory section, according to document [8].⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

Please note as well that section 7.3.2.4 is relevant for calibration parameters instantiated by Software Cluster Connection.

[SWS_SwCluC_03123]{DRAFT} ⌈The data instances created according to [SWS_SwCluC_03120], [SWS_SwCluC_03133] and [SWS_SwCluC_03135] shall be mapped to a VAR memory section, according to document [8].⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03124]{DRAFT} ⌈The default data instance created according to [SWS_SwCluC_03120] shall be initialized according to the NonqueuedReceiverComSpec.initValue in the RPortPrototype at the CompositionSwComponentType of the rootSoftwareComposition of the Ecu Extract.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03126]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall use the default data instance created according to [SWS_SwCluC_03120] as default handle in the corresponding Resource Entry in the Binary Manifest.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03136]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall make the data instance created according to [SWS_SwCluC_03133] accessible from other Software Clusters, via a Resource Entry in the Binary Manifest.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03130]{DRAFT} ⌈When explicit unqueued communication via dataReceivePointByArguments applies, and the Resource Entry created according to[SWS_SwCluC_03126] is not connected, the Rte_Rips_Read shall copy the value of default data instance created according to [SWS_SwCluC_03120] to the location of the OUT parameter <data>, and shall return RTE_E_UNCONNECTED.⌋ *(SRS_SwCluC_00100, SRS_SwCluC_00101)*

[SWS_SwCluC_03131]{DRAFT} ⌈When explicit unqueued communication via dataReceivePointByValues applies, and the Resource Entry created according to [SWS_SwCluC_03126] is not connected, the Rte_Rips_DRead shall return the value of default data instance created according to [SWS_SwCluC_03120].⌋*(SRS_SwCluC_00100, SRS_SwCluC_00101)*

[SWS_SwCluC_03132]{DRAFT} ⌈When explicit unqueued communication via dataReceivePointByArguments applies, and the Resource Entry created according to [SWS_SwCluC_03126] is connected, the Rte_Rips_Read shall copy the value of the corresponding data instance of the provider to the location of the OUT parameter <data>, and shall return RTE_E_OK.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00101)*

[SWS_SwCluC_03137]{DRAFT} ⌈When explicit queued communication applies, and the resource created according to [SWS_SwCluC_03136] is not connected, the Rte_Rips_Read shall discard the read access, and return RTE_E_UNCONNECTED.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00101)*

[SWS_SwCluC_03138]{DRAFT} ⌈When explicit queued communication applies, and the Resource Entry created according to [SWS_SwCluC_03136] is connected, but no entry is available in the queue, the Rte_Rips_Read shall discard the read access, and return RTE_E_NO_DATA .⌋*(SRS_SwCluC_00100, SRS_SwCluC_00101)*

[SWS_SwCluC_03139]{DRAFT} ⌈When explicit queued communication applies, and the Resource Entry created according to [SWS_SwCluC_03136] is connected, the Rte_Rips_Read shall copy the value of the first available entry of the corresponding data instance to the location of the OUT parameter <data>, and return RTE_E_OK.⌋ *(SRS_SwCluC_00100, SRS_SwCluC_00101)*

[SWS_SwCluC_03140]{DRAFT} ⌈When [SWS_SwCluC_03139] applies, the Rte_Rips_Read shall dequeue the data from the corresponding queue.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00101)*

[SWS_SwCluC_03143]{DRAFT} ⌈If explicit unqueued communication applies, and the Resource Entry in the Binary Manifest created according to [SWS_SwCluC_03126] is not connected, the Rte_Rips_DataIsUpdated shall return FALSE.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00101)*

[SWS_SwCluC_03144]{DRAFT} ⌈If explicit unqueued communication applies, and the Resource Entry in the Binary Manifest created according to [SWS_SwCluC_03126] is connected, and the sender has updated the data since the previous execution of the corresponding Rte_Rips_Read service, the Rte_Rips_DataIsUpdated shall return TRUE

⌋(*SRS_SwCluC_00100*, *SRS_SwCluC_00101*)

7.3.4 Client Server Communication

7.3.4.1 General

[SWS_SwCluC_03145]{DRAFT} ⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall ensure that the result of a `ClientServerOperation` is dispatched to the correct client, if more than one client invokes the `ClientServerOperation`⌋(*SRS_SwCluC_00100*, *SRS_SwCluC_00103*)

[SWS_SwCluC_03146]{DRAFT} ⌈The `Cross Cluster Communication` shall support multiple `Clients` invoking the same `ClientServerOperation` on a server ('N:1' Communication where N $\geq$ 1).⌋(*SRS_SwCluC_00100*, *SRS_SwCluC_00103*)

7.3.4.2 Timeout

[SWS_SwCluC_03147]{DRAFT} ⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall ensure that timeout monitoring is performed for client-server communication.⌋(*SRS_SwCluC_00100*, *SRS_SwCluC_00103*)

7.3.4.3 Buffering

[SWS_SwCluC_03150]{DRAFT} ⌈The `Cross Cluster Communication` on the client side of the `Software Cluster Connection` shall provide a request queue (with queue length 1) for each assigned `Client Server Communication Graph`, where the referenced `ClientServerOperation` is owned by a delegation `RPortPrototype` at the `CompositionSwComponentType` of the `rootSoftwareComposition` of the `Ecu Extract`.

Note: The structure of the queue is implementation specific, but the `Cross Cluster Communication` has to store at least the IN parameters and IN/OUT parameters, the transaction handle and the status code of the communication.⌋(*SRS_SwCluC_00100*, *SRS_SwCluC_00103*)

[SWS_SwCluC_03151]{DRAFT} ⌈The `Cross Cluster Communication` on the client side of the `Software Cluster Connection` shall provide a response queue (with queue length 1), for each assigned `Client Server Communication Graph`, where the referenced `ClientServerOperation` is owned by a delegation `RPortPrototype` at the `CompositionSwComponentType` of the `rootSoftwareComposition` of the `Ecu Extract`.

Note: The structure of the queue is implementation specific, but the `Cross Cluster Communication` has to store at least the IN/OUT and OUT parameters, the

transaction handle and the status code of the communication.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00103*)

[SWS_SwCluC_03152]{DRAFT} ⌈The Cross Cluster Communication on the server side of the Software Cluster Connection shall provide a request queue with the specified queue length (See [SWS_SwCluC_03154]), for each assigned Client Server Communication Graph, where the referenced ClientServer-Operation is owned by a delegation PPortPrototype at the Composition-SwComponentType of the rootSoftwareComposition of the Ecu Extract.

Note: The structure of the queue is implementation specific, but the Cross Cluster Communication has to store at least the IN/OUT and OUT parameters, the transaction handle and the status code of the communication.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00103*)

[SWS_SwCluC_03153]{DRAFT} ⌈The Cross Cluster Communication on the server side of the Software Cluster Connection shall provide N response queues (where N corresponds to the specified queue length), for each assigned Client Server Communication Graph, where the referenced ClientServer-Operation is owned by a delegation PPortPrototype at the Composition-SwComponentType of the rootSoftwareComposition of the Ecu Extract.

Note: The structure of the queue is implementation specific, but the Cross Cluster Communication has to store at least the IN/OUT and OUT parameters, the transaction handle and the status code of the communication.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00103*)

[SWS_SwCluC_03154]{DRAFT} ⌈The Cross Cluster Communication on the server side of the Software Cluster Connection shall determine the queue length according to the following priority rules (highest priority first):

1. value of the ECU-C parameter RteServerQueueLength
2. value of the queueLength attribute of the ServerComSpec

⌋(*SRS_SwCluC_00100, SRS_SwCluC_00103*)

[SWS_SwCluC_03155]{DRAFT} ⌈The Cross Cluster Communication on the server side of the Software Cluster Connection shall handle the requests in a first-in-first-out queue.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00103*)

[SWS_SwCluC_03156]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall make the request queue instance created according to [SWS_SwCluC_03150] accessible for the Software Cluster implementing the server, via a Resource Entry in the Binary Manifest.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03157]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall make the response queue instance created

according to [SWS_SwCluC_03151] accessible to the Software Cluster, via a Resource Entry in the Binary Manifest.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03158]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall make each request queue instance created according to [SWS_SwCluC_03152] accessible to the Software Cluster, via a Resource Entry in the Binary Manifest.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

[SWS_SwCluC_03159]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall make each response queue instance created according to [SWS_SwCluC_03153] accessible to the Software Clusters, via a Resource Entry in the Binary Manifest.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00101*)

7.3.4.4 Response to Request Mapping

The Cross Cluster Communication is responsible to map a response to the corresponding request. The problem of request to response mapping is split into:

- Mapping of a response to the correct client in the Software Cluster.
- Mapping of a response to the correct request within one client in the Software Cluster.

The general approach for request response mapping is to use transaction handles.

[SWS_SwCluC_03160]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall use a transaction handle that contains three parts, of unsigned integer type:

- Software Cluster Identifier
- Client Identifier
- Client Sequence Counter

⌋(*SRS_SwCluC_00100, SRS_SwCluC_00103*)

[SWS_SwCluC_03161]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall use the transaction handle for the identification of client server transactions, communicated between the clusters.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00103*)

[SWS_SwCluC_03162]{DRAFT} ⌈The Cross Cluster Communication on the server side of the Software Cluster Connection shall return the transaction handle of the request, without modification, together with the response.⌋(*SRS_SwCluC_00100, SRS_SwCluC_00103*)

[SWS_SwCluC_03163]{DRAFT} ⌈The `Cross Cluster Communication` of the `Software Cluster Connection` shall allow only one request per client and server operation at any time.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00103)*

7.3.4.5 Client Side

[SWS_SwCluC_03164]{DRAFT} ⌈If the corresponding resource in the `Binary Manifest` representing the response queue (See [SWS_SwCluC_03157]) is not connected, the `Rte_Rips_Invoke` shall return RTE_E_UNCONNECTED immediately.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00103)*

[SWS_SwCluC_03165]{DRAFT} ⌈If the corresponding resource in the `Binary Manifest` representing the response queue (See [SWS_SwCluC_03157]) is connected, the `Rte_Rips_Invoke` shall copy the IN and IN/OUT parameters, inform the server via a status flag, and return RTE_E_OK.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00103)*

[SWS_SwCluC_03166]{DRAFT} ⌈If the corresponding resource in the `Binary Manifest` representing the response queue (See [SWS_SwCluC_03157]) is connected, the `Rte_Rips_Invoke` shall return RTE_E_LIMIT, until the server's result has been successfully passed to the client, and no timeout occurred. The IN and IN/OUT parameters shall not be modified by `Rte_Rips_Invoke` .⌋*(SRS_SwCluC_00100, SRS_SwCluC_00103)*

[SWS_SwCluC_03167]{DRAFT} ⌈If the corresponding resource in the `Binary Manifest` representing the response queue (See [SWS_SwCluC_03157]) is not connected, the `Rte_Rips_ReturnResult` shall return RTE_E_UNCONNECTED immediately.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00103)*

[SWS_SwCluC_03168]{DRAFT} ⌈If the corresponding resource in the `Binary Manifest` representing the response queue (See [SWS_SwCluC_03157]) is connected, the `Rte_Rips_ReturnResult` shall return RTE_E_NO_DATA, until the server's result has been successfully passed to the client and no timeout occurred.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00103)*

[SWS_SwCluC_03169]{DRAFT} ⌈If the corresponding resource in the `Binary Manifest` representing the response queue (See [SWS_SwCluC_03157]) is connected, and the server's result was not available within the specified timeout, the `Rte_Rips_ReturnResult` shall return RTE_E_TIMEOUT.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00103)*

[SWS_SwCluC_03170]{DRAFT} ⌈If the corresponding resource in the `Binary Manifest` representing the response queue (See [SWS_SwCluC_03157]) is connected, and an `AsynchronousServerCallReturnsEvent` exists, the `Rte_Rips_InvocationHandler` shall poll the corresponding response queue status to determine whether or not the server's result is available.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00103)*

[SWS_SwCluC_03171]{DRAFT} ⌈If the corresponding resource in the Binary Manifest representing the response queue (See [SWS_SwCluC_03157]) is connected, and an AsynchronousServerCallReturnsEvent exists, the Rte_Rips_InvocationHandler shall call the corresponding ACSR Runnable.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00103)*

7.3.4.6 Server Side

[SWS_SwCluC_03173]{DRAFT} ⌈If the corresponding resource in the Binary Manifest representing the request queue (See [SWS_SwCluC_03158]) is connected, the Rte_Rips_InvocationHandler shall poll the corresponding request queue status to determine whether or not there is a pending request.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00103)*

[SWS_SwCluC_03174]{DRAFT} ⌈If the corresponding resource in the Binary Manifest representing the request queue (See [SWS_SwCluC_03158]) is connected, the Rte_Rips_InvocationHandler shall invoke the server with IN, IN/OUT and OUT parameters of request resp. response queue.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00103)*

[SWS_SwCluC_03175]{DRAFT} ⌈If the corresponding resource in the Binary Manifest representing the response queue (See [SWS_SwCluC_03157]) is connected, the Rte_Rips_InvocationHandler shall call the corresponding server Runnable.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00103)*

[SWS_SwCluC_03176]{DRAFT} ⌈If the corresponding resource in the Binary Manifest representing the request queue (See [SWS_SwCluC_03158]) is connected, the Rte_Rips_InvocationHandler shall dequeue the server queue after successful invocation of the server queue.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00103)*

[SWS_SwCluC_03177]{DRAFT} ⌈If the corresponding resource in the Binary Manifest representing the request queue (See [SWS_SwCluC_03158]) is connected, the Rte_Rips_InvocationHandler shall set the status of the response queue (See [SWS_SwCluC_03159]) accordingly, to inform the client that the request has been processed.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00103)*

7.3.5 Modes Communication

7.3.5.1 General principles

The mode switch communication in AUTOSAR is a mixture between a data (the mode value) and an event semantic (the transition between modes). It also has some potential impact on the execution of ExecutableEntitys, via mode switch events and mode disabling dependencies. Additionally, the transitions between modes are following a strict sequence, as described in document [9].

Introducing this communication pattern to a clustered software architecture requires a technical solution that balances between the local view of software components inside a Software Cluster, and the ECU wide behavior cross several Software Clusters.

For the Software Cluster Communication, the following principles apply:

- The mode providing Software Cluster owns the leading mode machine instance, including the mode queue.
- The Host Software Cluster provides the mode switch tasks, which are used to execute the mode switch on each partition, where the mode is capable to execute runnables
- The Host Software Cluster coordinates the transfer of the mode switch notifications between the Software Clusters.
- The Host Software Cluster coordinates the conjunction of the Software Cluster individual completions of the mode switches, and notifies the mode providing Software Cluster
- Nevertheless, the mode switch itself is locally executed in each RTE in a Software Cluster. This has the consequence that the order between on-exit ExecutableEntitys, on-transition ExecutableEntitys, and on-entry ExecutableEntitys is only preserved locally inside a Software Cluster.
- The interfaces towards the RTE are only called in well-defined OS task contexts.

A sender-receiver communication can be implemented solely between Applicative Software Clusters, without impact to the Host Software Cluster. In contrast, the adding of a new mode switch communication between Applicative Software Clusters, additionally impacts the Host Software Cluster.

7.3.5.2 Software Cluster providing a mode

The Software Cluster providing the mode owns the leading mode machine instance, including its mode queue. Hence, the information about the next mode, to which the mode machine currently switches, is provided by this Software Cluster. In this specification, this mode value is called 'current on-transition value'.

The figure 7.3 illustrates the principle sequence, when a mode switch is initiated by an Applicative Software Cluster.

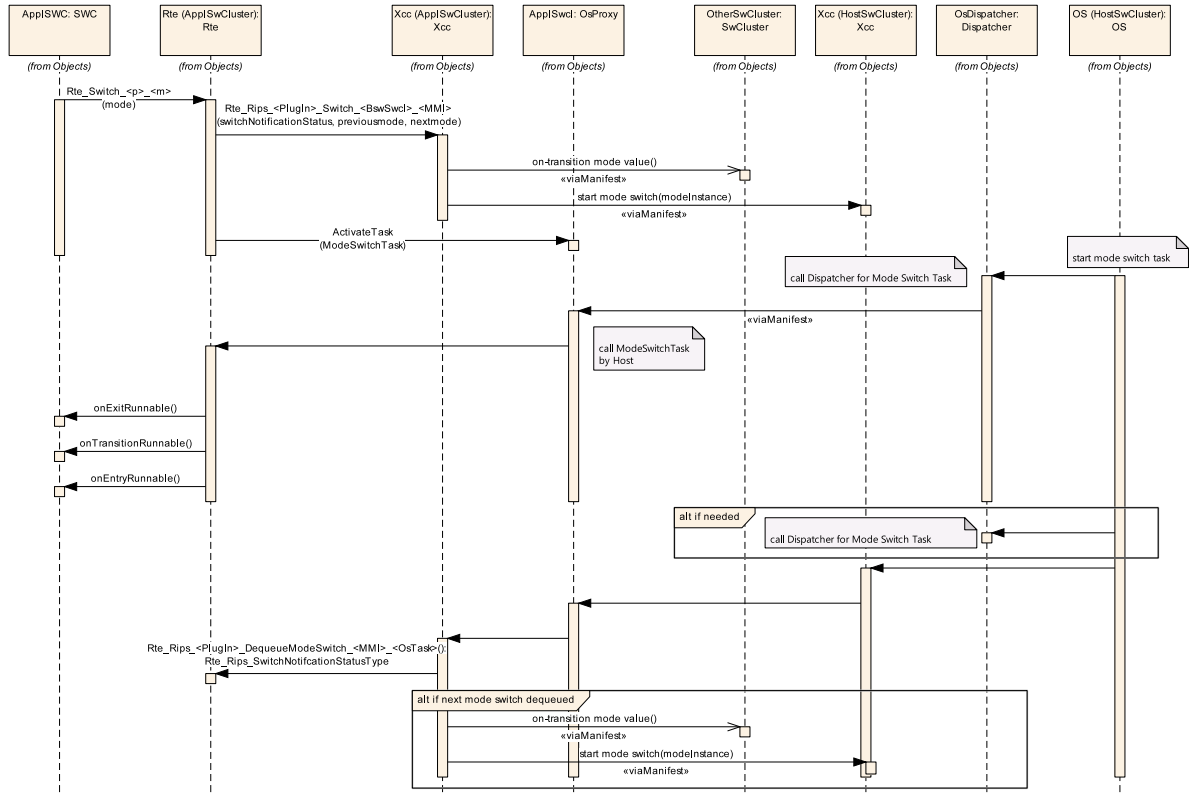

Figure 7.3: Execution of a mode switch provided by an Applicative Software Cluster

- The mode manager software component calls the Rte_Switch API to do the mode switch notification
- The Cross Cluster Communication gets notified via the call of the Rte_Rips_Switch Service.
- If the mode queue is empty, this Rte_Rips_Switch can be used to determine the first on-transition value.
- The Cross Cluster Communication in the Applicative Software Cluster informs the Cross Cluster Communication in the Host Software Cluster that a mode switch starts. This notification is a Cross Cluster Communication internal interface, which is not standardized.
- After the start of the mode switch on the host, the according mode switch tasks are scheduled (potentially on multiple partitions). This in turn, via Dispatchers, schedules cluster local proxy mode switch tasks.
- Potentially, other proxy mode switch tasks are scheduled before and after. Hence, the RTE cannot dequeue the next mode already in the context of its mode switch tasks.
- After the Cross Cluster Communication in the Host Software Cluster has determined that a mode switch task has run to end, it uses Cross

Cluster Communication to notify the RTE in the Applicative Software Cluster to dequeue the next mode.

The figure 7.4 illustrates the principle sequence, when a mode switch is initiated by a Host Software Cluster.

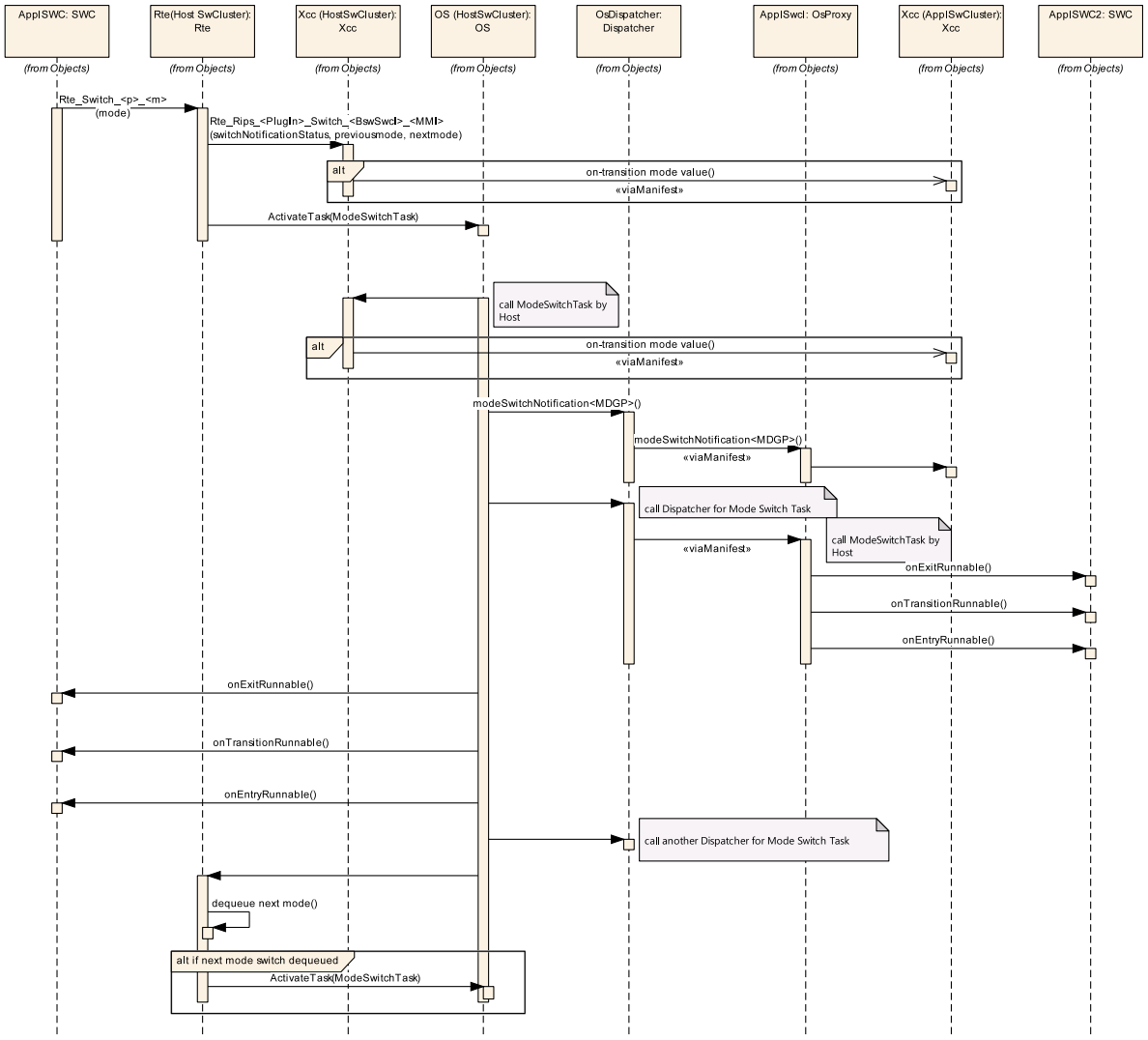

Figure 7.4: Execution of a mode switch provided by an Host Software Cluster

- The mode manager software component calls the Rte_Switch API, to do the mode switch notification
- The Cross Cluster Communication gets notified via the call of the Rte_Rips_Switch Service.
- If the mode queue is empty, this Rte_Rips_Switch can be used to determine the first on-transition value. Alternatively, this is also possible in the mode switch task(s) before first Dispatchers are scheduled.

- After the start of the mode switch on the host, the according mode switch tasks are scheduled (potentially on multiple partitions). This in turn, via Dispatchers, schedules cluster local proxy mode switch tasks.
- In contrast to the scenario in figure 7.3, the RTE can dequeue the next mode already in the context of its mode switch tasks.

These two principle sequences lead to the following requirements on the Cross Cluster Communication, in case a Software Cluster provides a mode.

[SWS_SwCluC_03022]{DRAFT} ⌈The Cross Cluster Communication shall signal the current on-transition value to other Software Clusters, before the mode switch according to [SWS_Rte_02665] starts.⌋*(SRS_SwCluC_00104)*

Note: This functionality can be implemented inside the Rte_Rips_Switch service, in case the mode queue was empty. Otherwise, this can be done shortly before the last mode switch task terminates, or when the first mode switch task starts already for the next transition. The interface to signal the current on-transition is a Cross Cluster Communication internal interface, which is not standardized.

[SWS_SwCluC_03057]{DRAFT} ⌈The Cross Cluster Communication shall add at the Software Cluster providing the mode a Provided Resource Entry, and at any Software Cluster requiring the trigger an according Require Resource Entry in the Binary Manifest.⌋*(SRS_SwCluC_00104)*

[SWS_SwCluC_03057] ensures that a missing mode manager in a clustered ECU gets detected, even if the implementation may rely additionally on the according Dispatchers.

[SWS_SwCluC_03023]{DRAFT} ⌈The Cross Cluster Communication of the Host Software Cluster shall implement a mode manager for each mode which needs to be provided by an Applicative Software Cluster.⌋*(SRS_SwCluC_00104)*

With this mode manager, it is possible to schedule the according Dispatchers for mode switch notification, and Dispatchers for Software Cluster local proxy mode switch tasks.

To ease the realization of [SWS_SwCluC_03023], all modes provided by the Applicative Software Clusters, shall be consumed by the Host Software Cluster.

[SWS_SwCluC_CONSTR_03032]{DRAFT} ⌈For each mode provided by an Applicative Software Cluster, the Ecu Extract of the Host Software Cluster shall own a required mode switch port at the CompositionSwComponentType of the rootSoftwareComposition.⌋*(SRS_SwCluC_00104)*

[SWS_SwCluC_03024]{DRAFT} ⌈The Cross Cluster Communication of the Host Software Cluster shall enqueue the current on-transition values signaled by Applicative Software Cluster, via the related mode manager. ([SWS_SwCluC_03023])⌋*(SRS_SwCluC_00104)*

Note: In case the Host Software Cluster provides the mode, the RTE implements already the leading mode machine instance.

[SWS_SwCluC_03025]{DRAFT} ⌈The Cross Cluster Communication of the Host Software Cluster shall notify the Applicative Software Cluster, when a mode switch tasks starts.⌋*(SRS_SwCluC_00104)*

Note: This notification is used by the Applicative Software Cluster to schedule the mode switch notification runnable, which enqueues the mode switch notification in the cluster local RTE.

[SWS_SwCluC_03026]{DRAFT} ⌈The Cross Cluster Communication of the Host Software Cluster shall notify the Cross Cluster Communication of the Applicative Software Cluster, when all mode switch tasks run to end.⌋*(SRS_SwCluC_00104)*

[SWS_SwCluC_03027]{DRAFT} ⌈In case the Applicative Software Cluster provides the mode, the Cross Cluster Communication of the Host Software Cluster shall notify the Cross Cluster Communication of the Applicative Software Cluster, when mode switch tasks run to end.⌋*(SRS_SwCluC_00104)*

[SWS_SwCluC_03028]{DRAFT} ⌈In case the Applicative Software Cluster provides the mode, the Cross Cluster Communication of the Applicative Software Cluster notifies the RTE via Rte_Rips_DequeueModeSwitch, when a mode switch task known by this RTE runs to end. The Cross Cluster Communication has to guarantee that the last call of Rte_Rips_DequeueModeSwitch is not done before the last on-entry ExecutableEntity in the whole clustered system terminated.⌋*(SRS_SwCluC_00104)*

Please note [SWS_Rte_70123], which guarantees a certain execution context for the RTE.

[SWS_SwCluC_03029]{DRAFT} ⌈The Cross Cluster Communication shall provide a Complex Driver Software Component on each EcucPartition, where mode switch task are configured. This component is later called mode proxy component.⌋*(SRS_SwCluC_00104)*

[SWS_SwCluC_03030]{DRAFT} ⌈The Cross Cluster Communication shall provide RunnableEntitys to detect the start and end of any mode switch task, which is related to mode communication cross Software Clusters.⌋*(SRS_SwCluC_00104)*

Please note: the RunnableEntitys [SWS_SwCluC_03030] may require additional ModeAccessPoints, as well as suitable RTEEvents. It is possible to use OsTaskExecutionEvent and a small runtime logic, to determine the ongoing mode transition, or a set of SwcModeSwitchEvents, which activate the RunnableEntity on any transition of a specific mode machine instance.

7.3.5.3 Host Software Cluster requiring a mode

As explained in section 7.3.5.2, the Host Software Cluster is already aware about existing modes communicated cross Software Clusters. Therefore, it technically receives already the modes provided by Applicative Software Clusters. In case the Host Software Cluster owns software components, which require the the mode as well, it shall connect them to its already existing mode manager.

[SWS_SwCluC_03031]{DRAFT} ⌈The Cross Cluster Communication shall provide the AssemblySwConnectors between the provided mode switch ports of the mode proxy component and the required mode switch ports at the software components requiring the mode, if the mode is required by the Host Software Cluster.⌋*(SRS_SwCluC_00104)*

7.3.5.4 Applicative Software Cluster requiring a mode

For each required mode, the Applicative Software Cluster defines OsTask(s) plus dispatcher(s), for the mode switch notification.

When the mode switch gets notified by the Host Software Cluster, the Cross Cluster Communication writes the actual on-transition mode value to the RTE of the Applicative Software Cluster with the regular Rte_Switch API related to a mode switch port.

Note: The RTE in the Applicative Software Cluster shall still a call ActivateTask(s) to trigger the mode switch task(s). This call can still be used by the Host Software Cluster to trigger the mode switch task execution in the Host Software Cluster, if needed.

For this purpose, the Cross Cluster Communication defines a RunnableEntity with an OsTaskExecutionEvent for each mode on each partition where a mode switch notification is configured. When the mode switch is executed, the Host Software Cluster schedules the mode switch task(s), which in turn schedule the 'proxy' mode switch task(s) in the Applicative Software Cluster

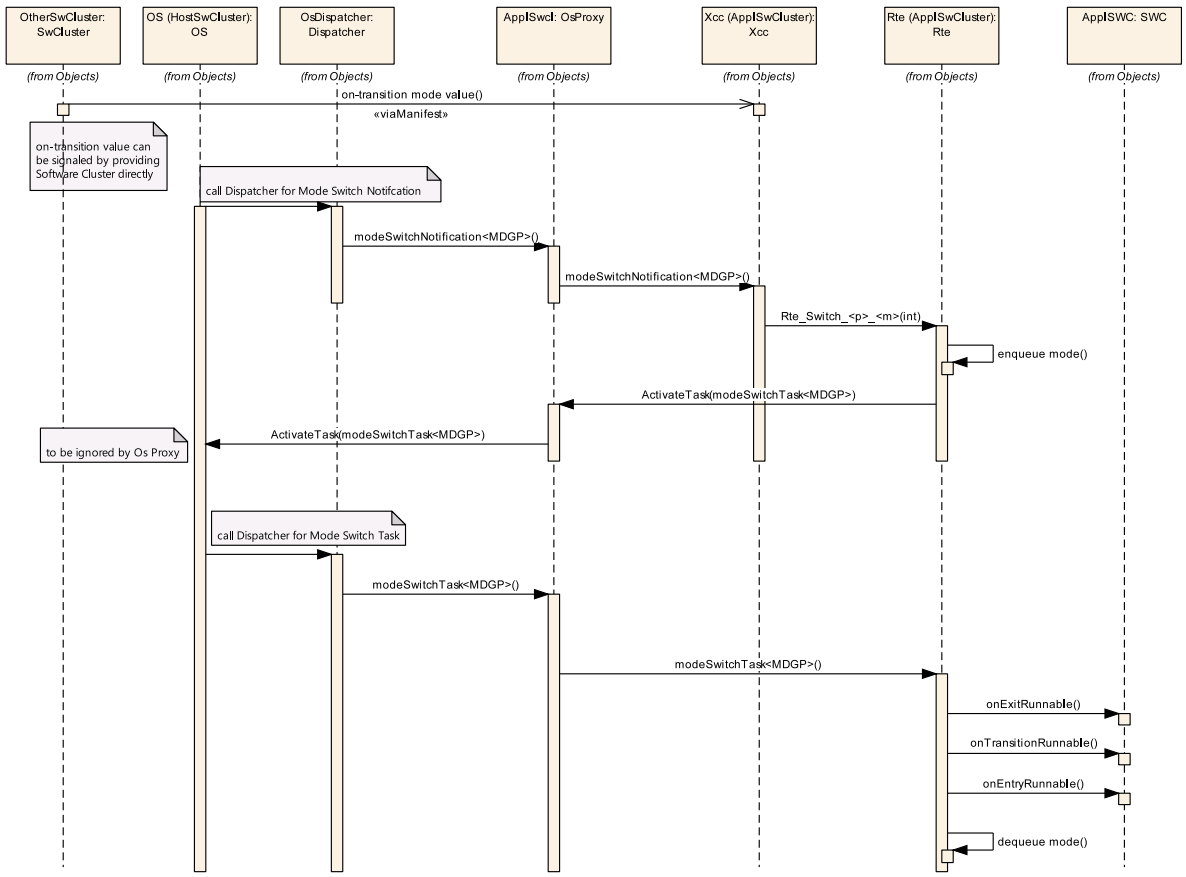

Figure 7.5: Execution of a mode switch in an Applicative Software Cluster

[SWS_SwCluC_03015]{DRAFT} ⌈The `Cross Cluster Communication` shall provide a Complex Driver Software Component on each `EcucPartition` where a `mode switch port` needs to be provided. This component is later called `mode proxy component`.⌋*(SRS_SwCluC_00104)*

[SWS_SwCluC_03016]{DRAFT} ⌈The `Cross Cluster Communication` shall provide a provided mode switch port on each `mode proxy component` where a `mode switch port` needs to be provided.⌋*(SRS_SwCluC_00104)*

[SWS_SwCluC_03017]{DRAFT} ⌈The `Cross Cluster Communication` shall provide the `AssemblySwConnector`s between the provided `mode switch ports` and the required `mode switch ports` at the software components requiring the mode.⌋*(SRS_SwCluC_00104)*

[SWS_SwCluC_03018]{DRAFT} ⌈The `Cross Cluster Communication` shall provide a `mode switch notification` runnable with an `OsTaskExecution-Event`, for each mode on each partition where a `mode switch notification` is configured. The `RunnableEntity` shall define a `ModeSwitchPoint`.⌋*(SRS_-SwCluC_00104)*

[SWS_SwCluC_03019]{DRAFT} ⌈When the mode switch notification runnable is executed, it shall write the actual on-transition value via Rte_Switch to the RTE.⌋*(SRS_SwCluC_00104)*

[SWS_SwCluC_CONSTR_03020]{DRAFT} ⌈For each EcucPartition on which the mode switch notification is configured for a ModeDeclarationGroup-Prototype required by an Applicative Software Cluster, a OsTask with a related SwCluCOsProxyOsTask and SwCluCOsProxyOsTaskDispatcher shall be configured.⌋*(SRS_SwCluC_00104)*

[SWS_SwCluC_CONSTR_03021]{DRAFT} ⌈For each required ModeDeclara-tionGroupPrototype and EcucPartition on which the mode switch noti-fication is configured, a OsTask with a related SwCluCOsProxyOsTask and SwCluCOsProxyOsTaskDispatcher shall be configured.⌋*(SRS_SwCluC_00104)*

7.3.5.5 Initialization

The approach described in section 7.3.5.2 and 7.3.5.4 leads to local mode machine instances in each Software Cluster providing or requiring the mode.

But this also implies that the initialization of the mode machine instances is executed time shifted, without explicit synchronization by the Cross Cluster Communication.

When the clustered system starts, the following procedure needs to be preserved!

- Initialization of the Cross Cluster Communications
- Execute all Rte_Init_<InitContainer> functions in Applicative Software Clusters and Host Software Cluster. This step already leads to the execution of on-entry ExecutableEntitys, triggered by initialMode and mapped to RteInitializationRunnableBatch container.
- Execute all Rte_Start in Applicative Software Clusters. This step executes the remaining on-entry ExecutableEntitys triggered by initialMode. In case the RTE implementation triggers the mode switch task to proceed the transition to the initialMode, it will happen here as well.
- Execute Rte_Start in the Host Software Cluster. This step executes the remaining on-entry ExecutableEntitys triggered by initialMode in the Host Software Cluster. In case the RTE implementation triggers the mode switch task to proceed the transition to the initialMode, the mode switch task will now be executed.

7.3.6 Trigger Communication

7.3.6.1 General principles

The trigger communication in AUTOSAR is a pure event semantic, which is used to request the execution of triggered ExecutableEntitys.

Introducing this communication pattern to a clustered software architecture, the focus lies on the use case to implement trigger sources in Applicative Software Clusters and in the Host Software Cluster. Nevertheless, the control on the execution of the triggered ExecutableEntitys is implemented in the Host Software Cluster, to guarantee an ECU wide behavior cross several Software Clusters.

For the Software Cluster Communication, the following principles apply:

- The trigger providing Applicative Software Cluster transfers the occurrence of the trigger to the Host Software Cluster.
- The Host Software Cluster provides the tasks, which are used to execute the triggered ExecutableEntitys on each partition where needed.
- If queuing of triggers is required from a dynamic perspective, this is configured and implemented in the RTE of the Host Software Cluster
- Nevertheless, the triggering itself is locally executed in each RTE inside a Software Cluster. This has the consequence that the order between triggered ExecutableEntitys is only preserved locally inside a single proxy OS Task of an Applicative Software Cluster.
- The interfaces towards the RTE are only called in well-defined OS task contexts.

Adding of a new trigger communication between Applicative Software Clusters, additionally impacts the Host Software Cluster.

7.3.6.2 Software Cluster providing a mode

The figure 7.6 illustrates the principle sequence, when a trigger is raised by an Applicative Software Cluster.

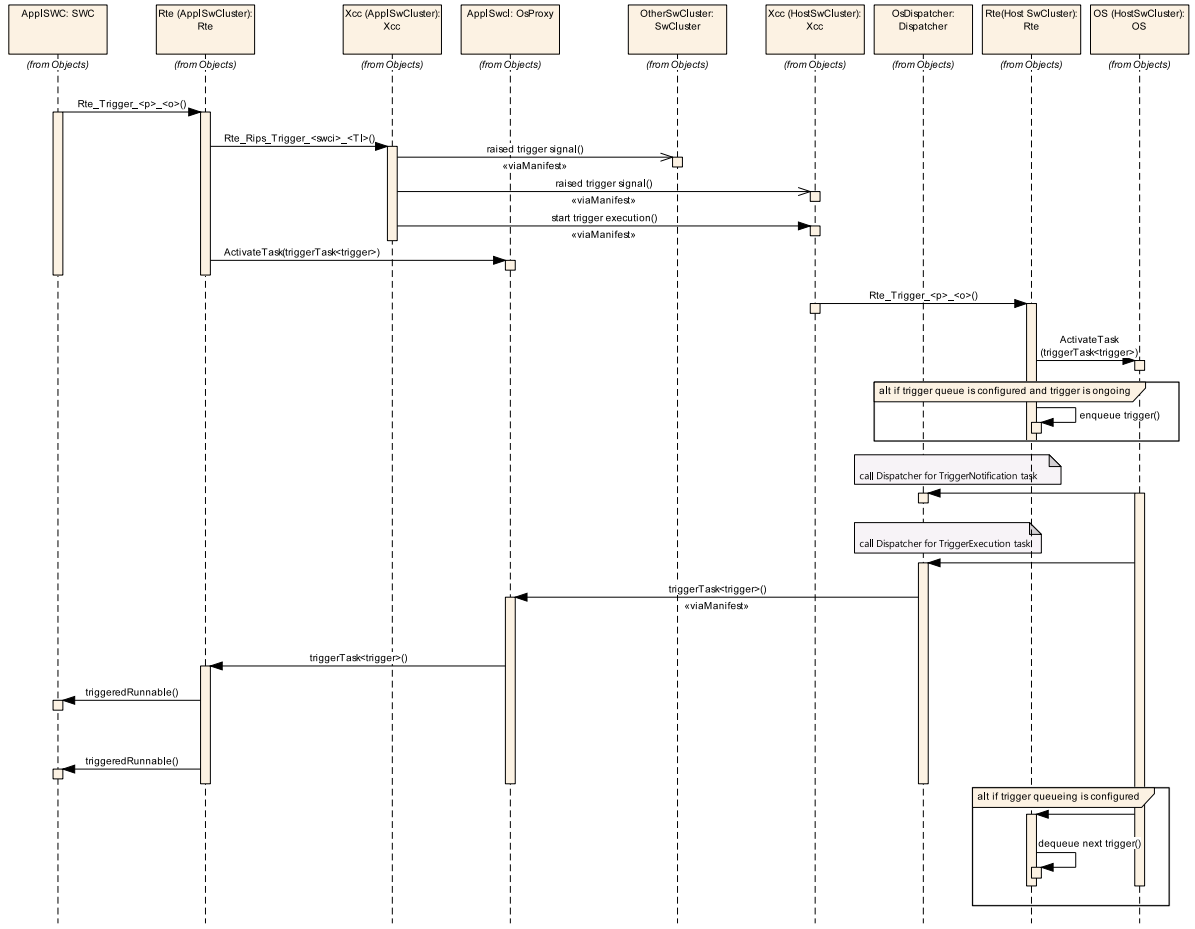

Figure 7.6: Execution of a trigger provided by an Applicative Software Cluster

- The trigger source calls the Rte_Trigger API, to raise the trigger.
- The Cross Cluster Communication gets notified, via the call of the Rte_Rips_Trigger Service.
- The Cross Cluster Communication in the Applicative Software Cluster informs the Cross Cluster Communication in the Host Software Cluster that a trigger was raised. This notification is a Cross Cluster Communication internal interface, which is not standardized.
- In the Host Software Cluster, the Cross Cluster Communication raises the trigger in the RTE, via Rte_Trigger API. In case of a configured trigger queue, this would enqueue the trigger.
- After the start of the trigger execution on the Host Software Cluster, the according tasks are scheduled (potentially on multiple partitions). This in turn schedules, via Dispatchers, the Applicative Software Cluster local proxy tasks.
- In case of a configured trigger queue, the dequeue operation is executed after the last triggered ExecutableEntitys terminated.

The figure 7.7 illustrates the principle sequence, when a trigger is raised by a Host Software Cluster.

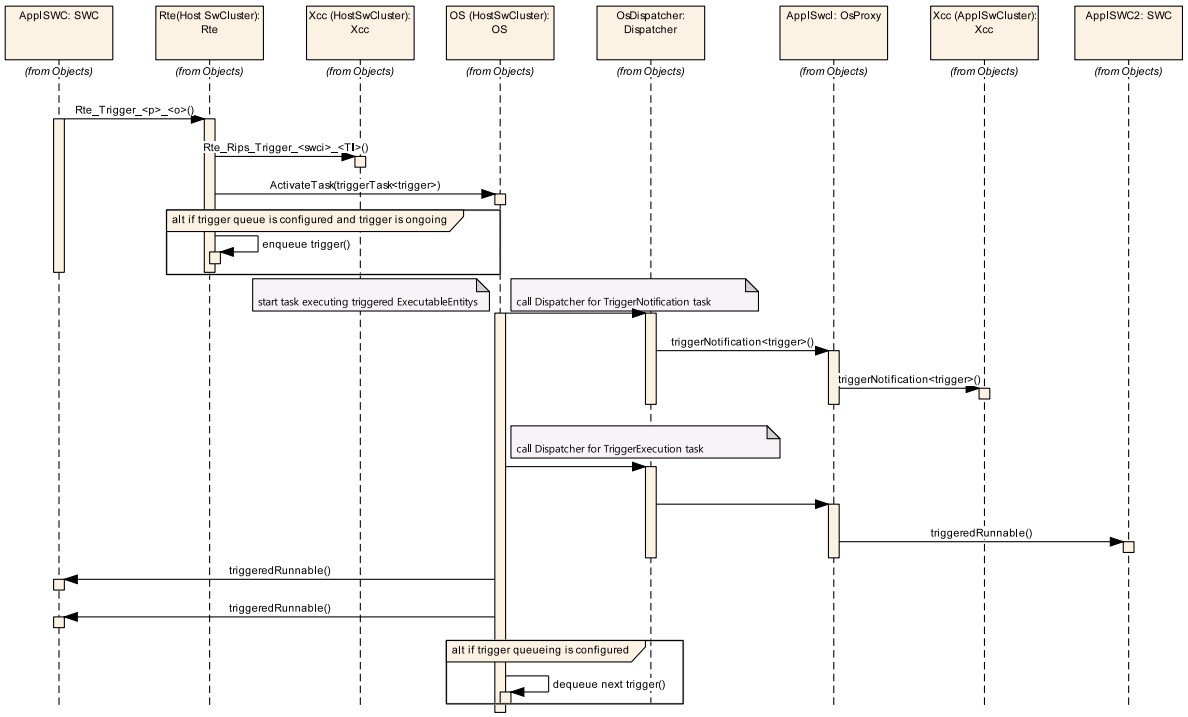

Figure 7.7: Execution of a trigger provided by an Host Software Cluster

- The trigger source calls the Rte_Trigger API, to raise the trigger.
- The Cross Cluster Communication gets notified, via the call of the Rte_Rips_Trigger Service.
- The RTE activates the tasks used for execution of triggered ExecutableEntitys. If a queue is configured, and a trigger execution is already ongoing, the RTE in the Host Software Cluster enqueues the trigger.
- After the start of the trigger execution on the Host Software Cluster, the according tasks are scheduled (potentially on multiple partitions). This in turn, schedules, via Dispatchers, the Applicative Software Cluster local proxy tasks.
- In case of a configured trigger queue, the dequeue operation is executed after the last triggered ExecutableEntitys terminated.

These two principle sequences lead to the following requirements on the Cross Cluster Communication, in case a Software Cluster provides a mode.

[SWS_SwCluC_03055]{DRAFT} ⌈The Cross Cluster Communication shall signal to other Software Clusters that a trigger was raised.⌋ *(SRS_SwCluC_00105)*

Note: This functionality can be implemented inside the Rte_Rips_Trigger service. Function wise, it is up to the implementation, if the [SWS_SwCluC_03055] is already fulfilled via the connection to the Host Software Cluster.

[SWS_SwCluC_03056]{DRAFT} ⌈The Cross Cluster Communication shall, at the Software Cluster providing the trigger, add a Provided Resource Entry and, at Software Clusters requiring the trigger, an according Require Resource Entry in the Binary Manifest.⌋*(SRS_SwCluC_00105)*

[SWS_SwCluC_03056] ensures that a missing trigger source in a clustered ECU gets detected, even if the implementation may rely on the according Dispatchers.

[SWS_SwCluC_03058]{DRAFT} ⌈The Cross Cluster Communication of the Host Software Cluster shall implement a trigger source for each trigger, which needs to be provided by an Applicative Software Cluster.⌋*(SRS_SwCluC_00105)*

With this trigger source, it is possible to schedule the according Dispatchers for trigger notification, and Dispatchers for Software Cluster local trigger execution tasks.

To ease the realization of [SWS_SwCluC_03058], all triggers provided by Applicative Software Clusters, shall be consumed by the Host Software Cluster.

[SWS_SwCluC_CONSTR_03059]{DRAFT} ⌈The Ecu Extract of the Host Software Cluster shall, for each trigger provided by an Applicative Software Cluster, own a required trigger port at the CompositionSwComponentType of the rootSoftwareComposition.⌋*(SRS_SwCluC_00105)*

[SWS_SwCluC_03060]{DRAFT} ⌈The Cross Cluster Communication of the Host Software Cluster shall raise the triggers signaled by Applicative Software Clusters, via the related trigger source. ([SWS_SwCluC_03058])⌋*(SRS_SwCluC_00105)*

Note: In case the Host Software Cluster provides the trigger, the trigger source is already part of the Software Components or BSW Modules belonging to the Host Software Cluster, which are directly interacting with the RTE or SchM.

[SWS_SwCluC_03061]{DRAFT} ⌈The Cross Cluster Communication of the Host Software Cluster shall notify the Applicative Software Cluster, when the execution of a trigger tasks starts.⌋*(SRS_SwCluC_00104)*

Note: This notification is used by the Applicative Software Cluster to schedule the trigger notification runnable, which raises the trigger at the cluster local RTE.

[SWS_SwCluC_03062]{DRAFT} ⌈The Cross Cluster Communication shall provide a Complex Driver Software Component on each EcucPartition, where a trigger source according to [SWS_SwCluC_03058] is configured. This component is later called trigger proxy component.⌋*(SRS_SwCluC_00104)*

7.3.6.3 Host Software Cluster requiring a trigger

As explained in section 7.3.6.2, the Host Software Cluster is already aware about an existing trigger communicated cross Software Clusters. Therefore, it technically already receives the triggers raised by Applicative Software Clusters. In case the Host Software Cluster owns software components that require the the trigger as well, it shall connect them to its already existing trigger source, according to [SWS_SwCluC_03058].

[SWS_SwCluC_03063]{DRAFT} ⌈The Cross Cluster Communication shall provide the AssemblySwConnectors between the provided trigger ports of the trigger proxy component, and the required trigger ports at the software components requiring the trigger, if the trigger is required by the Host Software Cluster.⌋*(SRS_SwCluC_00104)*

7.3.6.4 Applicative Software Cluster requiring a trigger

For each required trigger, the Applicative Software Cluster defines OsTask(s) plus dispatcher(s), for the trigger notification.

When the trigger gets executed by the Host Software Cluster, the Cross Cluster Communication raises trigger at the RTE of the Applicative Software Cluster, with the regular Rte_Trigger API related to a trigger port.

Note: The RTE in the Applicative Software Cluster shall still call ActivateTask(s), to trigger the task(s) used for the execution of triggered ExecutableEntitys. This call can still be used by the Host Software Cluster to trigger the task executions in the Host Software Cluster, if needed.

For this purpose, the Cross Cluster Communication defines a RunnableEntity with an OsTaskExecutionEvent, for each trigger and on each partition, a trigger notification is configured. When the trigger is executed, the Host Software Cluster schedules the task(s), which in turn schedule the 'proxy' task(s) in the Applicative Software Cluster.

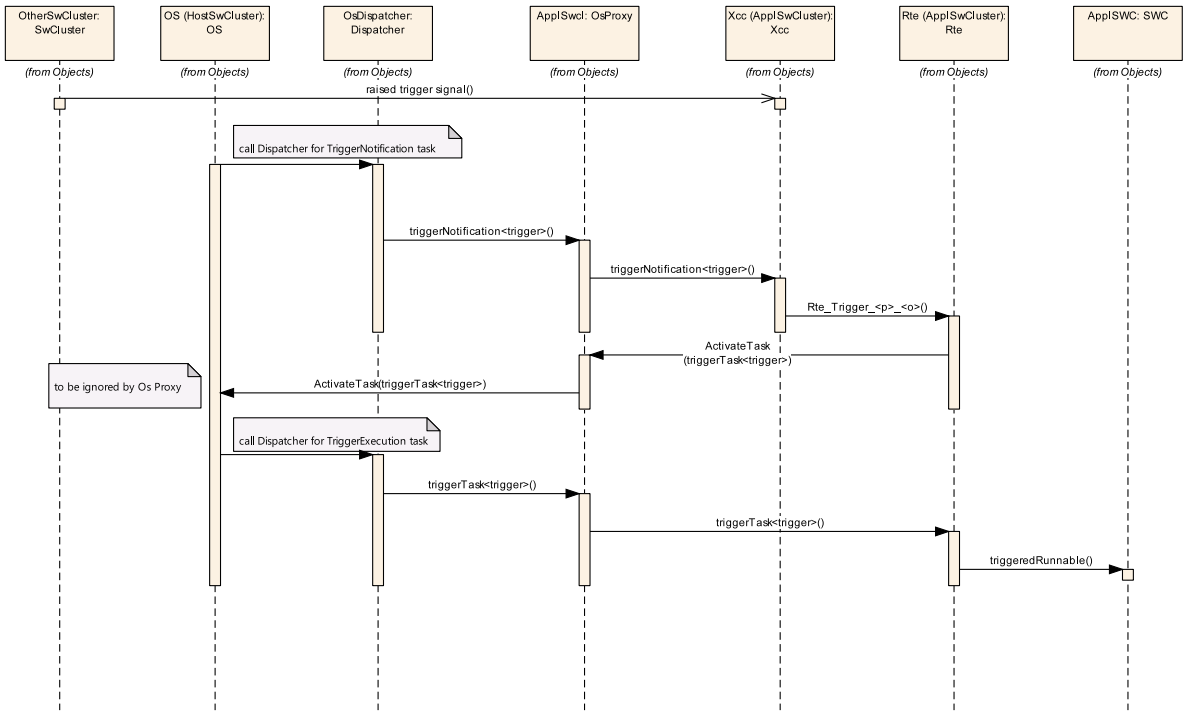

Figure 7.8: Execution of a trigger in an Applicative Software Cluster

[SWS_SwCluC_03048]{DRAFT} ⌈The `Cross Cluster Communication` shall provide a Complex Driver Software Component, on each `EcucPartition` where a `trigger port` needs to be provided. This component is later called `trigger proxy component`.⌋*(SRS_SwCluC_00105)*

[SWS_SwCluC_03049]{DRAFT} ⌈The `Cross Cluster Communication` shall provide a provided `trigger port`, on each `trigger proxy component`.⌋*(SRS_SwCluC_00105)*

[SWS_SwCluC_03050]{DRAFT} ⌈The `Cross Cluster Communication` shall provide the `AssemblySwConnector`s between the provided `trigger ports` and the required `trigger ports`, at the software components requiring the trigger.⌋ *(SRS_SwCluC_00105)*

[SWS_SwCluC_03051]{DRAFT} ⌈The `Cross Cluster Communication` shall provide a `trigger notification` runnable with an `OsTaskExecutionEvent`, for each trigger on each partition a `trigger notification` is configured. The `RunnableEntity` shall define a `ExternalTriggeringPoint`.⌋*(SRS_SwCluC_00105)*

[SWS_SwCluC_03052]{DRAFT} ⌈When the `trigger notification runnable` is executed, it shall raise the trigger via `Rte_Trigger` to the RTE.⌋*(SRS_SwCluC_00105)*

[SWS_SwCluC_CONSTR_03053]{DRAFT} ⌈For each EcucPartition on which the trigger notification is configured for a trigger required by an Applicative Software Cluster, a OsTask, with a related SwCluCOsProxyOsTask and SwCluCOsProxyOsTaskDispatcher, shall be configured.⌋*(SRS_SwCluC_00105)*

[SWS_SwCluC_CONSTR_03054]{DRAFT} ⌈For each required trigger and EcucPartition on which triggered ExecutableEntitys needs to be executed, at least one OsTask, with a related SwCluCOsProxyOsTask and SwCluCOsProxyOsTaskDispatcher, shall be configured.⌋*(SRS_SwCluC_00105)*

7.3.7 Parameter Communication

[SWS_SwCluC_03006]{DRAFT} ⌈The Cross Cluster Communication of the Software Cluster Connection shall provide one constant default parameter instance for each assigned Parameter Communication Graph, which contains NO PPortPrototype.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00106)*

[SWS_SwCluC_03007]{DRAFT} ⌈The default value instance shall be mapped to a CALPRM memory section, according to document [8].⌋*(SRS_SwCluC_00100, SRS_SwCluC_00106)*

Please note as well section 7.3.2.4, which is relevant for calibration parameters instantiated by Software Cluster Connection.

[SWS_SwCluC_03008]{DRAFT} ⌈The default value instance shall be initialized according to the ParameterRequireComSpec.initValue, in the RPortPrototype at the CompositionSwComponentType of the rootSoftwareComposition of the Ecu Extract.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00106)*

[SWS_SwCluC_03009]{DRAFT} ⌈If the corresponding resource in the Binary Manifest is connected, the Rte_Rips_Prm shall return the value of the connected parameter.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00106)*

[SWS_SwCluC_03010]{DRAFT} ⌈If the corresponding resource in the Binary Manifest is NOT connected, the Rte_Rips_Prm shall return the value of the default parameter instance of [SWS_SwCluC_03006].⌋*(SRS_SwCluC_00100, SRS_SwCluC_00106)*

Note: The behavior of Rte_Rips_Prm function, according to [SWS_SwCluC_03009] and [SWS_SwCluC_03010], can be implemented by referencing the default parameter instance as the default handle in the Binary Manifest.

[SWS_SwCluC_03011]{DRAFT} ⌈For each assigned Parameter Communication Graph, which contains the PPortPrototype, the Cross Cluster Communication of the Software Cluster Connection shall make the parameter instance, according to [SWS_Rte_80130], accessible for other Software Clusters, via a Resource Entry in the Binary Manifest.⌋*(SRS_SwCluC_00100, SRS_SwCluC_00106)*

7.3.8 Error Classification

Together with the RTE, the Cross Cluster Connection of the Software Cluster Connection implements the Run-Time Environment of the Software Cluster. The usual development errors checking API parameters are already covered by the development error detection of RTE. In addition, communication infrastructure errors are not reported as Production Errors nor as Extended Production Errors. Therefore, the following sections do not define error codes, and are marked as not applicable.

7.3.8.1 Development Errors

Development errors are not applicable for the Cross Cluster Connection of the Software Cluster Connection.

7.3.8.2 Runtime Errors

Runtime errors are not applicable for the Cross Cluster Communication of the Software Cluster Connection.

7.3.8.3 Transient Faults

Transient Faults are not applicable for the Cross Cluster Communication of the Software Cluster Connection.

7.3.8.4 Production Errors

Production Errors are not applicable for the Cross Cluster Communication of the Software Cluster Connection.

7.3.8.5 Extended Production Errors

Extended Production Errors are not applicable for the Cross Cluster Communication of the Software Cluster Connection.

7.4 Proxy Modules

7.4.1 Overview

Since an Applicative Software Cluster does not contain all BSW modules (in extreme case no BSW modules), those missing APIs, as well as the EcuC configuration elements, need to be substituted.

In the AUTOSAR Layered Software Architecture, the dependency between BSW modules can be generalized as follows:

A higher layer BSW module uses the APIs of a lower layer BSW module. In the opposite direction, the lower layer BSW module may call callback functions of the higher layer BSW module. In case of `AUTOSAR Interfaces`, additional Software Component Descriptions are required to enable the RTE generation. If the interface between those BSW modules is configurable, the ECU configuration provides the information about Symbolic Name values and ID values.

Of course, the AUTOSAR Layered Software Architecture also has horizontal dependencies. In this case, rotate the previous paragraph by 90 degrees.

It is important to state that the different BSW Module are directly using the C-interfaces of each other, without abstraction (as the RTE provides between SWCs). This, in turn, causes strong implementation dependencies, like fixed names of include files and C-functions.

Since the actual implementation and integration of the Classic Platform architecture is now split into Software Clusters, missing BSW Modules - those to which interfaces are existing - need to be substituted. The Figure 7.9 illustrates the proxy module approach, for a horizontal interface dependency.

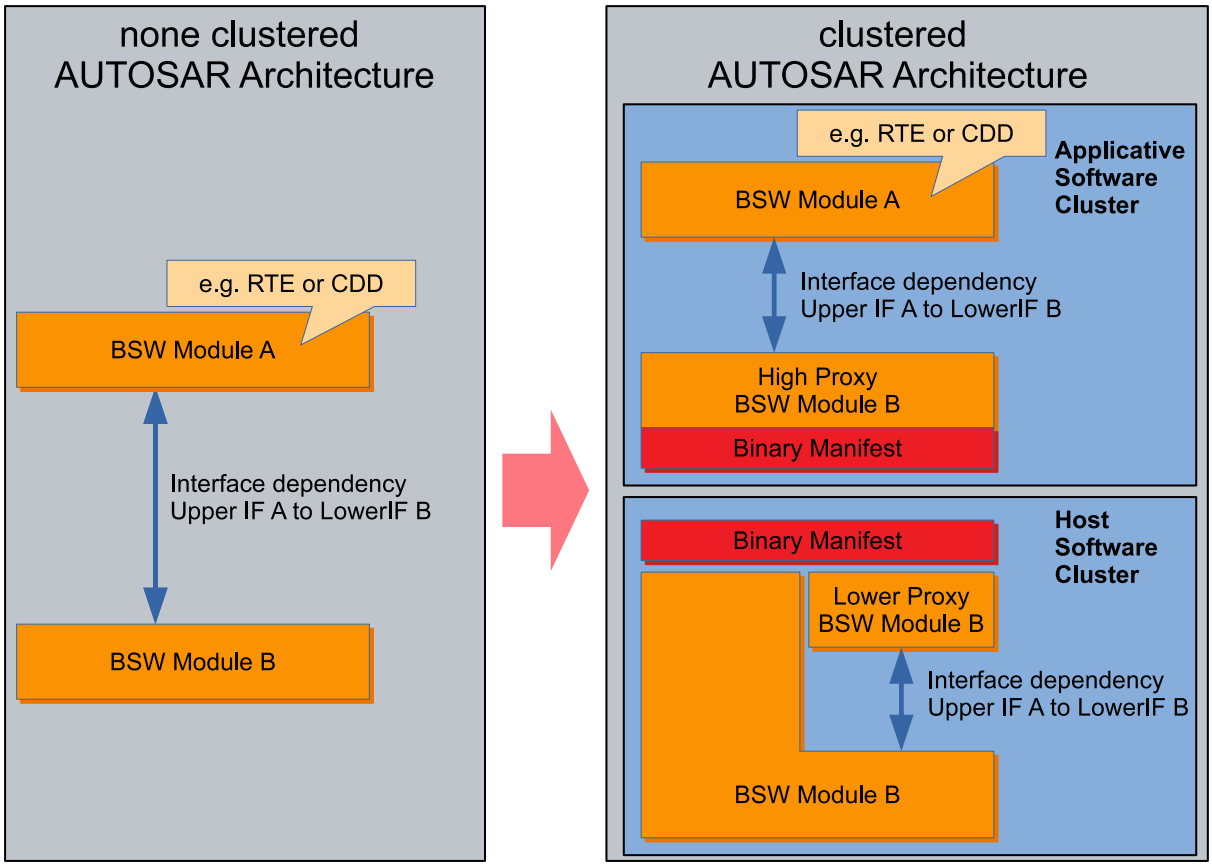

Figure 7.9: Proxy module approach

The C-interfaces of the lower layer BSW module (Module B), towards the higher layer BSW module (Module A), are provided by the High Proxy Module B, in the Software Cluster SWCL 1. The C-interfaces, ECU-C, and Component Descriptions of Module B are provided in the same name space / ARPackage structure as the original Module B. For the users of Module B (in this example Module A), the substitution of Module B by the High Proxy Module B is transparent. Usually, the offered API functions are almost without logic, and are just redirecting the calls towards the substituted module.

In the `Host Software Cluster`, the Low Proxy Module acts as a substitute for all 'user modules' in other Software Clusters. It implements the callback functions, which are normally configured in Module B.

The High Proxy Module and Low Proxy Module are connected via the means of the `Binary Manifest`. Therefore, the functions of the High and Low Proxy modules may contain functionality, which handles this `Binary Manifest` link and the situations if the link is missing. For instance, returning a reasonable error code or even providing some replacement value.

In addition, the High Proxy Module can implement functionality to localize configuration decisions. For instance, in order to support a SWCL local growing of the NV block over integration steps, it could be possible to adjust the locally used size of a NV block. Of course, this is only possible if the configured size in the real NvM is sufficiently large.

By this approach, the SWCs and BSW modules inside the Applicative Software Clusters call the common BSW inside the Host Software Cluster via proxies. When doing so, the partitioning of the ECU SW has to be considered, as described in section 7.4.2.2 Partitions.

This section specifies the general Proxy Module pattern. Additionally, requirements for specific Proxy Modules are in section 7.4.3. Additional Proxy Modules might be standardized by AUTOSAR in future releases.

7.4.2 Abstract Proxy Module Pattern

This section provides some generic design principles and common requirements, relevant for Proxy Module implementations.

Basically, a High Proxy Module provides a kind of facade, which hides the Software Cluster Connection specific mechanisms to access the real BSW Module. In the AUTOSAR Architecture, such a facade has to hide dependencies to

- Standardized Interfaces
- Standardized AUTOSAR Interfaces
  and the
- ECU Configuration

In addition to the essential need to provide the interfaces to the BSW Module user in the Applicative Software Cluster, the Proxy Module should implement abstraction functionalities. Those abstraction functionalities decouple the needed configuration in the High Proxy Module, from the configuration done in the BSW Module of the Host Software Cluster. The introduction of abstraction functionalities is a case-by-case decision. One the one hand, they might need a functional support of the according BSW Module, or might lead to functional restrictions. On the other hand, such abstractions are beneficial, since in some cases they can be used to avoid a reconfiguration of the Host Software Cluster, after change in the Applicative Software Cluster.

Please note that the following requirements only have limited applicability to an OS Proxy, since the OS APIs are already designed to be called from all the partitions.

7.4.2.1 General Proxy functionality

[SWS_SwCluC_02000]{DRAFT} ⌈The Proxy Module shall support that id values, required to access the BSW service API of the Host Software Cluster, can change, without reconfiguration or rebuild of the Applicative Software Cluster using the BSW service.⌋*()*

7.4.2.2 Partitions

Partitions are shared between the Host Software Cluster and Applicative Software Clusters. This also applies to the properties of the related OsApplications and OsTasks. Consequently, Applicative Software Clusters are not strictly separated from the Host Software Cluster. Due to performance reasons, it is possible to share the identical partiton of the Host Software Cluster with many Applicative Software Clusters. If Applicative Software Clusters share the same partition, they are also not strictly separated from each other. In contrast, if a Software Cluster has multiple partitions (e.g. on different safety levels), those partitions are separated from each other.

When a partition crossing function call is required, usually some close interaction with the Operating System is necessary - especially in case of different safety levels. However, the Operating System is not directly available inside an Applicative Software Cluster. Additionally, it needs to be considered that passing synchronous function calls cross partitions, may have an impact on the scheduling behavior of the software. Therefore, it is preferable to do this in the Host Software Cluster, or - even better - to avoid this by implementing the master satellite pattern in the related BSW modules.

As a consequence:

Inside a partition, such direct function calls to the BSW module in the Host Software Cluster are easily possible, with little overhead. If a BSW service is needed in a specific partition (including safety partition), the Host Software Cluster should offer the service interface in this partition. If this is not possible, the according transition shall be implemented in the Host Software Cluster. To support this, the Proxy Modules have to grant a partition write access to their interfaces.

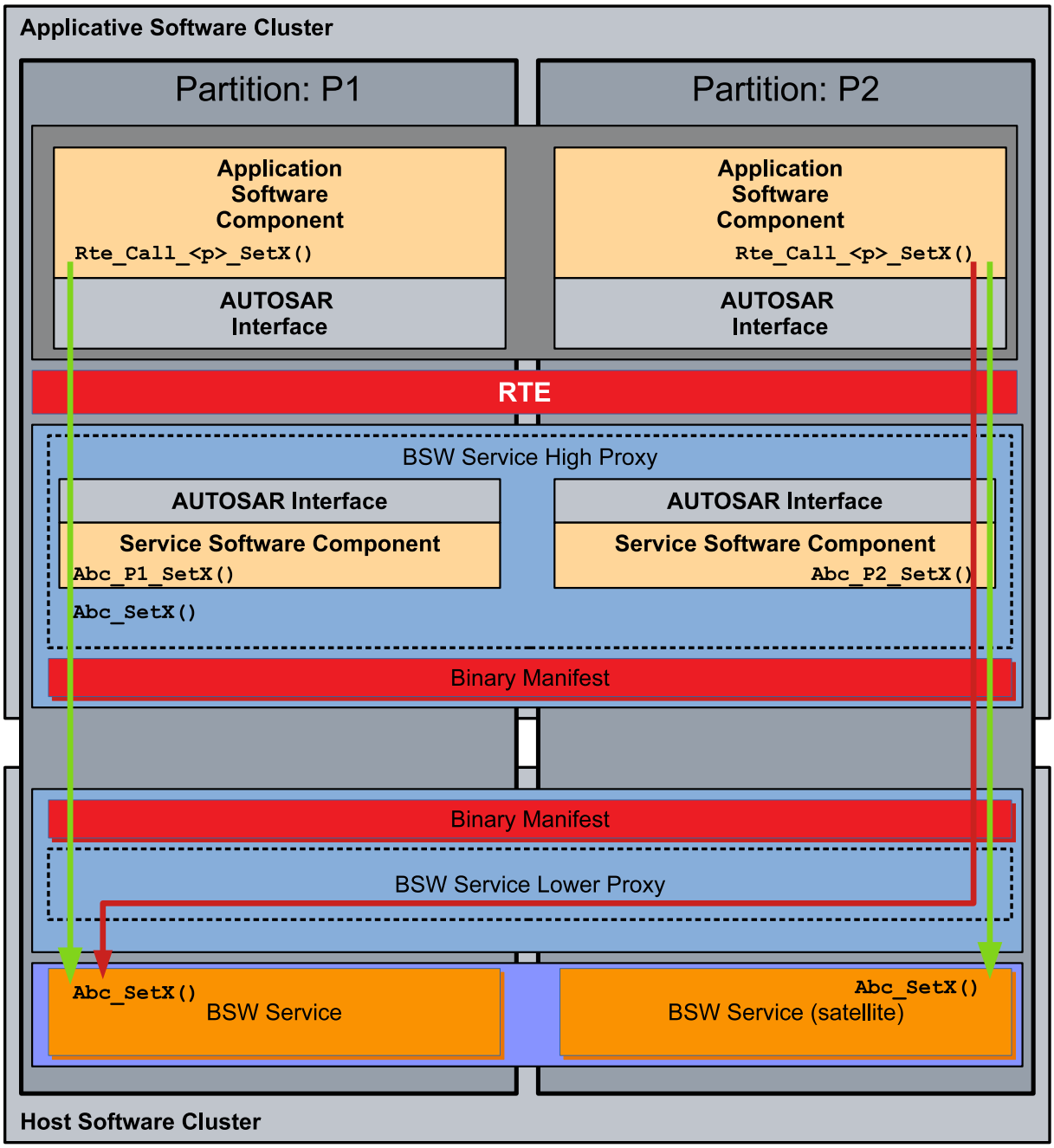

Figure 7.10: Proxy Modules and Partitions

Figure 7.10 illustrates this approach. The `AUTOSAR Interfaces` are offered in each partition, via a dedicated Service Software Component. The related `Standardized AUTOSAR Interface` is provided in exactly one Partition, in which the BSW module integration is supported.

A call from the Application Software Component is passed vertically through the partition, down to the BSW Module or its satellite (green arrow).

If such a satellite is not available, the Low Proxy Module can implement (e.g. via `SchM`) a cross partition call (red arrow).

[SWS_SwCluC_02001]{DRAFT} ⌈The High Proxy shall provide the applicable Standardized AUTOSAR Interfaces once per configured EcucPartition. In doing so, the APIs are provided with the orignal Mip, at the EcucPartition where the SwCluCNativeBswApi is set to true. For all the other BSW API instances, the Mip is replaced by <Mip>_<shortName of EcucPartition>.⌋*()*

[SWS_SwCluC_02002]{DRAFT} ⌈The High Proxy shall create one partition specific Service Software Component per configured EcucPartition, with the name <Mip>_<EcucPartition shortName>.⌋*()*

[SWS_SwCluC_02003]{DRAFT} ⌈The High Proxy shall provide the Ports belonging to a specific CpSoftwareClusterServiceResource, at the partition specific Service Software Component, to which the using SwComponentPrototype is mapped to.⌋*()*

7.4.2.3 Unconnected Service Resources

[SWS_SwCluC_02004]{DRAFT} ⌈The High Proxy shall implement a 'reasonable' behavior for a service resource, if the connection to the resource provider does not exist. In any case, the OUT arguments of functions shall return a 'neutral' value, or the value which can be assumed after a reset. The functional behavior should be like the state directly after a reset, before further activity occured.⌋*()*

Note: To decide what is 'reasonable', a functional understanding of the service resource is required. Therefore, [SWS_SwCluC_02004] can only give some rough expectations.

7.4.2.4 Ecu Configuration Principles

[SWS_SwCluC_02005]{DRAFT} ⌈To resolve the ECU Configuration dependency, the Proxy Module implementation shall define a Vendor Specific Module Definition [10], where the ECU Configuration container and parameters are used to configure the interface towards the user of the substituted BSW Module.⌋*()*

Typically, those are the containers and parameters configuring the variation (e.g. existence) of Standardized AUTOSAR Interfaces, or parameters relevant for symbolic name values.

In addition, these containers need to have a relationship to the software cluster resource pool, in order to manage the configuration needs exchange between the different Software Cluster providers. Furthermore, the CpSoftwareClusterServiceResource element defines the globalResourceId, which is required for the resource entrys in the Binary Manifest.

[SWS_SwCluC_02006]{DRAFT} ⌈The Proxy Module implementation shall define an EcucContainerDef, which defines a mapping between

- the EcucContainerDef(s) identifying the module user channel in the StMD of substituted BSW Module

  AND

- the CpSoftwareClusterServiceResource representing the user channel in the software cluster resource pool. The EcucForeignReferenceDef shall be named SwCluCResourceRef.

⌋()

Note that this EcucContainerDef of [SWS_SwCluC_02006] can also be used to add additional Parameters, to control the connection behavior between High Proxy and Low Proxy. For an example, see: SwCluCNvMProxyNvBlock.

[SWS_SwCluC_02007]{DRAFT} ⌈The Proxy Module implementation shall define, for each different Proxy Module, an EcucEnumerationParamDef named SwCluCProxyGeneration<Mip>, with the literals

- PROXY_DISABLED, to disable the according Proxy Module code and AUTOSAR model generation
- HIGH_PROXY, to enable the according High Proxy Module code and AUTOSAR model generation
- LOW_PROXY, to enable the according Low Proxy Module code and AUTOSAR model generation

⌋()

7.4.2.5 Proxy Modules and Binary Manifest

It is the responsibility of the Proxy Module implementation to put its Resource Entrys into the Binary Manifest of the respective Software Cluster. This includes service user specific channels (e.g. for a specific NV block), or some general links - typically named base socket - to link some generic infrastructure between Low and High Proxy (e.g. the set of BSW APIs).

[SWS_SwCluC_02008]{DRAFT} ⌈The Proxy Module implementation of the Software Cluster Connection shall put one Resource Entry, for each associated CpSoftwareClusterServiceResource, into the Binary Manifest.⌋()

7.4.3 Specific Proxy Module Requirements

7.4.3.1 OS Proxy

7.4.3.1.1 Enable OS Proxy Generation

[SWS_SwCluC_02200]{DRAFT} ⌈The OS High Proxy code, and related AUTOSAR model descriptions, shall only be created, if the configuration parameter `SwCluCProxyGenerationOs` is set to `HIGH_PROXY`.⌋*()*

[SWS_SwCluC_02201]{DRAFT} ⌈The OS LOW Proxy code, and related AUTOSAR model descriptions, shall only be created, if the configuration parameter `SwCluCProxyGenerationOs` is set to `LOW_PROXY`.⌋*()*

7.4.3.1.2 General OS Proxy functionality

7.4.3.1.3 Overview

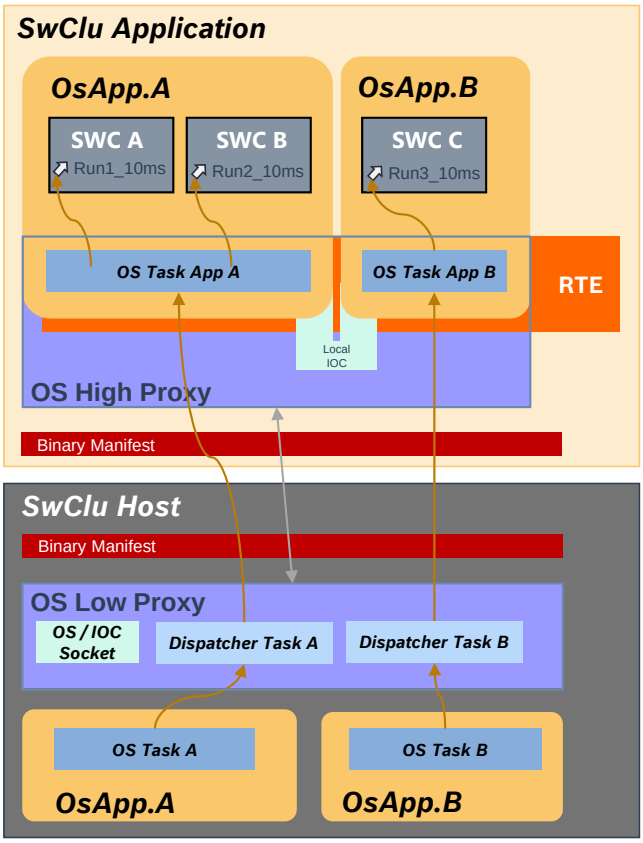

Figure 7.11: Overview of Os Proxy

As explained in chapter 7.1.1 and 7.4.1, the concept of Software Clusters also has to provide a way to divide some parts of an AUTOSAR Os into smaller units (see document [12] as reference for Os). Basically, runnables of a Software Cluster shall be executed on a target machine. Since the system is divided into a machine dependent part (Host Software Cluster), and several machine independent parts (Applicative Software Clusters), an interface layer in between is required. This layer will be realized using the proxy concept.

The Applicative Software Cluster contains proxies for those parts of the OS implementation that are needed to complete the runtime environment (RTE). This is called Os High Proxy.

The Host Software Cluster will implement the AUTOSAR Os, as well as the overall configuration of the machine. The Os Low Proxy makes this AUTOSAR Os accessible by the Os High Proxy, via the Binary Manifest. Both are linked together via the information held in the Binary Manifest, which enables an API exchange between Software Clusters.

The OS features (e.g. tasks, resources, spinlocks) supported by the Os High Proxy are configured with EcucModuleConfigurationValues for an Os EcucModuleDef, with according OsTask, OsResource, OsSpinlock, etc. containers. In the Applicative Software Cluster and in the Host Software Cluster, the configuration of SwCluCOsProxyOsTask, SwCluCOsProxyOsResource, SwCluCOsProxyOsSpinlock, etc. containers map the OS configuration containers to CpSoftwareClusterServiceResources, defining each the globalResourceId and the isMandatory attribute, relevant for the corresponding Resource Entry in the Binary Manifest.

The Os Low Proxy provides an Os- and IOC-socket for each Applicative Software Cluster. It also provides a dispatcher, which is responsible for calling each Os-Task of an Applicative Software Cluster. The Os- and IOC-socket provides basic hooks to the Applicative Software Cluster. These are used, for example, to do basic initialization of local Os and IOC channels, to forward general APIs like spinlock, or to do a context switch. The dispatcher controls and calls the Os-Tasks from the Applicative Software Cluster. This could, for example, be implemented as runnables, which are indirectly calling the Os High Proxy Task, depending on the entry found in the Binary Manifest.

The Host Software Cluster provides the implementation of Os Applications, and an abstraction of them in the Os Low Proxy. The Os-Application context inside an Applicative Software Cluster matches to the corresponding Os Application in the Host Software Cluster. The Os High Proxy task is executed in the same context as the low proxy task, and has the same level of priority, trust and access.

As mentioned before, a connection between Os Low Proxy and Os High Proxy is required. It will be established with the mechanism described in chapter 7.1.5.2 (Connecting Resources). In this chapter, the metaclasses for Os service resources are

defined: CpSoftwareClusterServiceResources. Esential to run a Applicative Software Cluster are resources for the Os- and IOC-socket and the Os-ProxyTasks. These resources are part of the system's resource pool (CpSoftwareClusterResourcePool), which includes the description of their relevant attributes and a unique ResourceID (CpSoftwareClusterResource.globalResourceId). A valid connection can only be established, if all attributes configured for those resources match during the connection process.

To achieve a robust system, and to avoid erroneous behavior, it is recommended to mark Os service resources as mandatory.

Detailed requirements can be found below.

7.4.3.1.4 General requirements

[SWS_SwCluC_02202]{DRAFT} ⌈The Os Proxy shall support the execution of OsTasks from an Applicative Software Cluster, in the context of the Host Software Cluster.⌋*()*

[SWS_SwCluC_CONSTR_02203]{DRAFT} **Constraint for Interrupts** ⌈Interrupts are not supported for an Applicative Software Cluster, since it does not have access to the hardware or peripherials.⌋*()*

[SWS_SwCluC_02204]{DRAFT} ⌈The Os Proxy shall support a socket mechanism to hook several Applicative Software Clusters onto the Host Software Cluster.⌋*()*

For example, this can be used to support Applicative Software Cluster local IOC communication paths.

[SWS_SwCluC_02205]{DRAFT} ⌈The Os High Proxy interfaces shall be described with the parameters, and according containers, as defined in Os EcucModuleDef.⌋*()*

[SWS_SwCluC_02206]{DRAFT} ⌈The Os Proxy interfaces shall be configured as a service resource, with the parameters as defined in the SwCluCOsProxy container.⌋*()*

7.4.3.1.5 OS High Proxy

[SWS_SwCluC_02210]{DRAFT} ⌈The Os High Proxy shall provide the header file Os.h and Os_Cfg.h .⌋*()*

[SWS_SwCluC_02211]{DRAFT} ⌈The Os High Proxy shall provide a proxy OsTask, to which events from the Applicative Software Cluster can be mapped, during the design of an Applicative Software Cluster.⌋*()*

[SWS_SwCluC_02230]{DRAFT} ⌈The Os High Proxy shall support the implementation of a task body of the proxy OsTask, with the TASK() macro.⌋*()*

[SWS_SwCluC_02212]{DRAFT} ⌈The Os High Proxy shall provide a set of OsTask specific APIs that are required by the Applicative Software Cluster:

- ActivateTask
- ChainTask
- TerminateTask

⌋()

[SWS_SwCluC_02213]{DRAFT} ⌈If SwCluCOsProxyOsTask.SwCluCOsProxyTaskActivation .PASS_TASK_ACTIVATION is set for a specific OsTask, the API ActivateTask called for this OS task shall trigger a task activation in the Host Software Cluster.⌋()

[SWS_SwCluC_02229]{DRAFT} ⌈If SwCluCOsProxyOsTask.SwCluCOsProxyTaskActivation .OMIT_TASK_ACTIVATION is set for a specific OsTask, the API ActivateTask called for this OS task returns without effect.⌋()

[SWS_SwCluC_CONSTR_02228]{DRAFT} **Constraint for ActivateTask** ⌈ActivateTask is only allowed to be called in the task body of the Applicative Software Cluster.⌋()

[SWS_SwCluC_02214]{DRAFT} ⌈The API for ChainTask shall be implemented as macro with a return statement in the Applicative Software Cluster, to be used in the context of the task body, and will end the execution of the proxy task.⌋()

Please note: The OS task chaining can only be handled in the OS implementation in the Host Software Cluster.

[SWS_SwCluC_CONSTR_02229]{DRAFT} **Constraint for ChainTask** ⌈ChainTask is only allowed to be called in the task body of the Applicative Software Cluster.⌋()

[SWS_SwCluC_02215]{DRAFT} ⌈The API for TerminateTask shall be implemented as macro with a return statement in the Applicative Software Cluster, to be used in the context of the task body, to be able to end the proxy tasks execution.⌋()

Please note: The real OS task termination can only be handled in the OS implementation in the Host Software Cluster.

[SWS_SwCluC_CONSTR_02230]{DRAFT} **Constraint for TerminateTask** ⌈TerminateTask is only allowed to be called in the task body of the Applicative Software Cluster.⌋()

[SWS_SwCluC_02216]{DRAFT} ⌈The Os High Proxy shall provide default handles for resources, in case a connection to the host system could not be established successfully between Applicative Software Cluster and the Host Software Cluster. The default implementation shall be a neutral behavior, with the return value E_OK, to support "standard status" from OS.⌋()

See the section 7.4.3.1.6 'Os Low Proxy', for an explanation of how a missing connection is handled in the Host Software Cluster.

[SWS_SwCluC_02217]{DRAFT} ⌈The Os High Proxy shall have a service resource SwCluCOsProxyOsTask, for each OsTask required in the Applicative Software Cluster. For this resource, an entry in the Binary Manifest shall be created, to be able to link the task to its OsLowProxy implementation in the Host Software Cluster. Since the content of the OsTask is created by the RTE, this API shall be used and registered as service resource. It is recommended that this resource is marked with the mandatory flag (SwCluCBManifIsMandatory).⌋*()*

[SWS_SwCluC_02218]{DRAFT} ⌈The Os High Proxy shall contain a required service resource SwCluCOsProxyOsTaskDispatcher. The Proxy-OsTask shall be called in that context. For this resource, an entry in the Binary Manifest shall be created, to be able to link it to the OsLowProxy dispatcher in the Host Software Cluster. It is recommended that this resource is marked with the mandatory flag (SwCluCBManifIsMandatory).⌋*()*

[SWS_SwCluC_02219]{DRAFT} ⌈The Os High Proxy shall contain a required service resource SwCluCOsProxyFunctionDispatcher. This function dispatcher is used to call functions in context of the Host Software Cluster. For this resource, an entry in the Binary Manifest shall be created, to be able to link it to the OsLowProxy dispatcher in the Host Software Cluster.⌋*()*

For example, to call an Applicative Software Cluster local Bsw-Module Init function, in dedicated context of the Host Software Cluster.

[SWS_SwCluC_02220]{DRAFT} ⌈The Os High Proxy shall contain a required service resource SwCluCOsProxyOsBaseSocket. This collects all hooks needed to be linked to the Host Software Cluster. It also contains Applicative Software Cluster local Os specific functions for StartOs, initialize and shutdown. It should also cover the local IOC, if present. It is recommended that this resource is marked with the mandatory flag (SwCluCBManifIsMandatory).⌋*()*

Please note: This is highly vendor specific.

[SWS_SwCluC_02221]{DRAFT} ⌈Since a Applicative Software Cluster can contain several OS Applications, the Os High Proxy shall be able to provide a local IOC implementation for the cluster internal communication paths between OS Applications. The link to the Host Software Cluster shall be established as part of the SwCluCOsProxyOsBaseSocket service resource.⌋*()*

[SWS_SwCluC_02222]{DRAFT} ⌈The Os High Proxy shall require a service resource SwCluCOsProxyOsApplication, which is used in the Applicative Software Cluster, and needs to be matched to the Host Software Cluster.⌋*()*

This service resource represents the actual configuration of the Applicative Software Cluster's local OS Application, with its parameters and linked local OS objects.

[SWS_SwCluC_02223]{DRAFT} ⌈The Os High Proxy shall require a service resource SwCluCOsProxyOsResource, which is used in the Applicative Software Cluster, and needs to be matched to the Host Software Cluster.⌋*()*

[SWS_SwCluC_02224]{DRAFT} ⌈The Os High Proxy shall require a service resource SwCluCOsProxyOsSpinlock, which is used in the Applicative Software Cluster, and needs to be matched to the Host Software Cluster.⌋*()*

[SWS_SwCluC_02225]{DRAFT} ⌈The Os High Proxy shall provide the following APIs for the Applicative Software Cluster:

- GetResource
- ReleaseResource
- SuspendOSInterrupts
- ResumeOSInterrupts
- GetSpinlock
- ReleaseSpinlock
- GetApplicationID

Each of these APIs corresponds to a service resource in the Os Low Proxy. There will be an entry in the Binary Manifest, to require these service resources, which are described here: SwCluCOsProxy.⌋*()*

[SWS_SwCluC_02226]{DRAFT} ⌈The Os High Proxy shall NOT support the following APIs for the Applicative Software Cluster.

- StartScheduleTable
- StopScheduleTable
- Schedule
- DisableAllInterrupts
- EnableAllInterrupts
- SuspendAllInterrupts
- ResumeAllInterrupts

⌋*()*

[SWS_SwCluC_02227]{DRAFT} ⌈The configuration of the Os High Proxy for a complete Applicative Software Cluster shall be specified with a set of ECUC-parameters defined here: SwCluCOsProxy.⌋*()*

7.4.3.1.6 OS Low Proxy

[SWS_SwCluC_02250]{DRAFT} ⌈The Os Low Proxy shall only be provided by the Host Software Cluster.⌋*()*

[SWS_SwCluC_02251]{DRAFT} ⌈The configuration of the Os Low Proxy for a complete Host Software Cluster shall be specified with a set of ECUC-parameters defined here: SwCluCOsProxy. The configuration of the Os Low Proxy shall be derived from the implemented configuration for the Os in the Host Software Cluster.⌋*()*

[SWS_SwCluC_02252]{DRAFT} ⌈The Os Low Proxy shall provide a socket for the Os High Proxys of the Applicative Software Clusters. For each Applicative Software Cluster, a corresponding service resource SwCluCOsProxyOsBaseSocket and configuration shall be provided.⌋*()*

[SWS_SwCluC_02253]{DRAFT} ⌈The Os Low Proxy shall provide for each SwCluCOsProxyOsBaseSocket a SwCluC_OsProxy_Init API, to initialize the Os High Proxy of an Applicative Software Cluster. This shall be one element of the service resource SwCluCOsProxyOsBaseSocket.⌋*()*

For example, an Init-Callback could be provided for each Applicative Software Cluster, and called from the Os Low Proxy during initialization phase of the Host Software Cluster.

[SWS_SwCluC_02254]{DRAFT} ⌈The Os Low Proxy shall provide an API to start the Os High Proxy of an Applicative Software Cluster. This shall be one element of the service resource SwCluCOsProxyOsBaseSocket.⌋*()*

For example, a Startup hook could be provided for each Applicative Software Cluster.

[SWS_SwCluC_02255]{DRAFT} ⌈The Os Low Proxy shall provide an API to shutdown the Os High Proxy of an Applicative Software Cluster. This shall be one element of the service resource SwCluCOsProxyOsBaseSocket.⌋*()*

For example, a shutdown hook could be provided for each Applicative Software Cluster.

[SWS_SwCluC_02256]{DRAFT} ⌈The Os Low Proxy shall be able to provide an API to connect IOC from Os High Proxy of an Applicative Software Cluster. This shall be one element of the service resource SwCluCOsProxyOsBaseSocket.⌋*()*

For example, an IOC-Init callback could be provided for each Applicative Software Cluster, and called from the Os Low Proxy, if a connection is established.

[SWS_SwCluC_02257]{DRAFT} ⌈The Os Low Proxy shall provide a task dispatcher, to call the connected Os High Proxy from Applicative Software Cluster.⌋*()*

Please note section 'Overview of Os Proxy' 7.11.

Example: The Host Software Cluster has a full configuration of the Os, and is prepared to host serveral Applicative Software Cluster, with their local RTE, to be able to execute the runnables of the Applicative Software Clusters Software components. This preconfiguration will preserve positions in an OsTask, like for events of runnables. Instead of executing a runnable, a so called dispatcher will be executed. This dispatcher acts as a link to Applicative Software Clusters task container, which will then execute all mapped runnables for this specific part of the task.

A connection between Software Clusters is established, when ResourceIDs in the Binary Manifests matches. This is necessary for service resources of the task dispatcher and the task itself. It is advised to check the connection status of these, in the implementation of the task dispatcher.

[SWS_SwCluC_02258]{DRAFT} ⌈The Os Low Proxy shall provide a service resource SwCluCOsProxyOsTaskDispatcher, for each Applicative Software Clusters configuration that the system shall be prepared for.⌋*()*

[SWS_SwCluC_02259]{DRAFT} ⌈The Os Low Proxy shall provide a service resource SwCluCOsProxyOsTask, for each Applicative Software Clusters task configuration that the system shall be prepared for.⌋*()*

[SWS_SwCluC_02260]{DRAFT} ⌈The Os Low Proxy shall provide a function dispatcher, to call the connected Os High Proxy from Applicative Software Cluster, to trigger general function calls.⌋*()*

Example: The Host Software Cluster has a full configuration of the BSW. To initialize all modules properly, a call to Applicative Software Cluster might be necessary. For example, to initialize a local RAM area there. Since this function or runnable might not be known by the RTE as runnable of a Software Component, this function can be called via the Os proxy function dispatcher. This dispatcher can be configured in the Host Software Cluster, to be called from ECUM or an Ini-Task. Each dispatcher provides its own service resource in the Binary Manifest. If a successful link to another Software Cluster is established, the call can be executed from the Host Software Cluster into the Applicative Software Cluster.

[SWS_SwCluC_02261]{DRAFT} ⌈The Os Low Proxy shall provide a service resource SwCluCOsProxyOsFunctionDispatcher, for each Applicative Software Clusters configuration that the system shall be prepared for.⌋*()*

[SWS_SwCluC_02262]{DRAFT} ⌈The Os Low Proxy shall provide a service resource SwCluCOsProxyOsApplication, for each OsApplication that is configured in the Host Software Cluster, and prepared for use in Applicative Software Clusters.⌋*()*

[SWS_SwCluC_02263]{DRAFT} ⌈The Os Low Proxy shall provide a service resource SwCluCOsProxyOsResource, for each OsResource used in Applicative Software Clusters.⌋*()*

[SWS_SwCluC_02264]{DRAFT} ⌈The Os Low Proxy shall provide a service resource SwCluCOsProxyOsSpinlock, for each OsSpinlock used in Applicative Software Clusters.⌋*()*

[SWS_SwCluC_02265]{DRAFT} ⌈The Os Low Proxy shall provide a basic service implementation for the following APIs, to be used and connected with an Applicative Software Cluster:

- GetResource
- ReleaseResource
- SuspendOSInterrupts
- ResumeOSInterrupts
- ReleaseResource
- GetSpinlock
- ReleaseSpinlock
- GetApplicationID

These APIs shall be provided with a corresponding service resource from SwCluCOsProxy.⌋*()*

[SWS_SwCluC_02266]{DRAFT} ⌈The Os Low Proxy shall provide default handles for resources, in case a connection could not be established successfully between Applicative Software Cluster and the Host Software Cluster.⌋*()*

For example, if an Applicative Software Cluster is missing, the task-dispatcher in the host calls an empty function.

Please note: One property of a clustered system is that parts shall be independently buildable. Another property is that some parts can be absent. Therefore, "default stubs" are needed for non-local APIs. But in case of missing interfaces to Os, Applicative Software Clusters cannot run successfully. The usage of a suitable mechanism to detect this case is advised. To achieve a robust system, the Host Software Cluster is still running in such a state.

7.4.3.2 NvM Proxy

The specified NvM Proxy has the following underlying design principle.

In the Applicative Software Cluster, NV Blocks supported by the NvM High Proxy are configured with EcucModuleConfigurationValues for a NvM EcucModuleDef, with according NvMBlockDescriptors containers. In the Applicative Software Cluster and in the Host Software Cluster, the configuration of SwCluCNvMProxyNvBlock containers map the NvMBlockDescriptors to CpSoftwareClusterServiceResources, defining each the globalResourceId

and the isMandatory attribute, relevant for the corresponding Resource Entry in the Binary Manifest.

The connection between the NvM High Proxy and the NvM Low Proxy utilizes the explicit synchronization, as specified in document [13]. The explicit synchronization has the advantage that the addresses of RAM Blocks and ROM blocks of Applicative Software Cluster do not have to be known by the NVRAM Manager in the Host Software Cluster. In addition, the function interface can be used to introduce additional functionality, like the NV block length adjustment ([SWS_SwCluC_02105]). On the other hand, this design principle cannot support temporary RAM blocks, since the NvM configuration in the Host Software Cluster may hold a different size than the NvM configuration in the Applicative Software Cluster.

[SWS_SwCluC_CONSTR_02134]{DRAFT} ⌈The NvM High Proxy shall reject configurations of temporary RAM blocks.⌋*(SRS_SwCluC_00206)*

Note: this means only NVRAM blocks configured with a permanent RAM block or explicit synchronization callbacks are supported in the NvM High Proxy.

In addition, the Host Software Cluster should not have a functional dependency to the content stored in NV Blocks of the Applicative Software Cluster. Furthermore, the NvM High Proxy shall be able to initialize the RAM Block, even if the related NV Block is not connected to a Host Software Cluster. This requires the existence of ROM block default values.

[SWS_SwCluC_CONSTR_02135]{DRAFT} ⌈The NvM High Proxy shall reject configurations, where no ROM block is configured via the parameter NvMRomBlockDataAddress, or the parameter NvMInitBlockCallback.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_CONSTR_02141]{DRAFT} ⌈The NvM High Proxy shall reject configurations, where the NV Block length given by NvMNvBlockLength is larger than the maximum size foreseen in the Host Software Cluster, given by SwCluCNvMProxyNvBlockMaxLength.⌋*(SRS_SwCluC_00206)*

Please note that SwCluCNvMProxyNvBlockMaxLength is also part of the guard value, and therefore protected from unilateral changes.

7.4.3.2.1 Enable NvM Proxy Generation

[SWS_SwCluC_02101]{DRAFT} ⌈The NvM High Proxy code, and related AUTOSAR model descriptions, shall only be created, if the configuration parameter SwCluCProxyGenerationNvM is set to HIGH_PROXY.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02144]{DRAFT} ⌈The NvM Low Proxy code, and related AUTOSAR model descriptions, shall only be created, if the configuration parameter SwCluCProxyGenerationNvM is set to LOW_PROXY.⌋*(SRS_SwCluC_00206)*

7.4.3.2.2 General NvM Proxy functionality

[SWS_SwCluC_02104]{DRAFT} ⌈The NvM Proxy shall support that the BlockIds of the NvM of Host Software Cluster can change, without reconfiguration or rebuild of the Applicative Software Cluster using these NV blocks.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02105]{DRAFT} ⌈The NvM Proxy shall support that the NV block length used in the Applicative Software Cluster can be <= the NV block length configured in the NvM of the Host Software Cluster. In case NV block length in Applicative Software Cluster is smaller, the NV block is filled with default values. The default fill value of a single byte is defined by the configuration parameter SwCluCNvMProxyNvBlockFillValue.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02106]{DRAFT} ⌈The NvM Proxy shall support that the NV block length used in the Applicative Software Cluster can be changed (in the range 0 <= NV block length configured in NvM), without reconfiguration or rebuild of the Host Software Cluster.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02112]{DRAFT} ⌈In case it is configured in the NvM High Proxy, the NvM Proxy shall invoke the NvM_SingleBlockCallbackFunction or JobFinished operation, when the according instance of the NvM Low Proxy is called.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02122]{DRAFT} ⌈In case it is configured in the NvM High Proxy, the NvM Proxy shall invoke the NvM_InitBlockCallbackFunction or InitBlock operation, when the according instance of the NvM Low Proxy is called.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02123]{DRAFT} ⌈In case it is configured in the NvM High Proxy, the NvM Proxy shall invoke the NvM_ReadRamBlockFromNvm function or ReadRamBlockFromNvM operation, when the according instance of the NvM Low Proxy is called.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02124]{DRAFT} ⌈In case it is configured in the NvM High Proxy, the NvM Proxy shall invoke the NvM_WriteRamBlockToNvm function or WriteRamBlockToNvM operation, when the according instance of the NvM Low Proxy is called.⌋*(SRS_SwCluC_00206)*

7.4.3.2.3 NvM High Proxy

[SWS_SwCluC_02102]{DRAFT} ⌈The NvM High Proxy shall provide the header file NvM.h.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02103]{DRAFT} ⌈The NvM High Proxy shall provide the symbolic name values for NvBlocks, for NvM users in the Applicative Software Cluster.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02125]{DRAFT} ⌈The NvM High Proxy shall provide the functions for single block requests

- NvM_SetDataIndex
- NvM_GetDataIndex
- NvM_SetBlockProtection
- NvM_GetErrorStatus
- NvM_SetRamBlockStatus
- NvM_SetBlockLockStatus
- NvM_CancelJobs
- NvM_ReadBlock
- NvM_WriteBlock
- NvM_RestoreBlockDefaults
- NvM_EraseNvBlock
- NvM_InvalidateNvBlock
- NvM_ReadPRAMBlock
- NvM_WritePRAMBlock
- NvM_RestorePRAMBlockDefaults

for NvM users in the Applicative Software Cluster.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02126]{DRAFT} ⌈The NvM High Proxy shall provide the functions for single block requests, according to [SWS_SwCluC_02103], once per configured SwCluCNvMBaseSocket. In doing so, the APIs are provided with the original Mip NvM at the EcucPartition, where the SwCluCNativeBswApi is set to true. For all the other SwCluCNvMBaseSocket instances, the Mip NvM is replaced by NvM_<shortName of EcucPartition>.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02136]{DRAFT} ⌈The functions

- NvM_ReadBlock
- NvM_WriteBlock
- NvM_RestoreBlockDefaults

of the NvM High Proxy shall only accept single block requests, where the pointer to the RAM data block is set to NULL_PTR. Otherwise, E_NOT_OK is returned, and the request is discarded.⌋*(SRS_SwCluC_00206)*

For example, the function NvM_GetErrorStatus can be provided as

```
1 /* EcucPartition "Core0_QM", SwCluCNativeBswApi == true */
2 Std_ReturnType NvM_GetErrorStatus (
```

```
    NvM_BlockIdType BlockId,
    NvM_RequestResultType * RequestResultPtr
)
...

/* EcucPartition "Core1_QM", SwCluCNativeBswApi == false */
Std_ReturnType NvM_Core1_QM_GetErrorStatus (
    NvM_BlockIdType BlockId,
    NvM_RequestResultType * RequestResultPtr
)
...
```

Note: SwCluCNativeBswApi can only be set to true, for at most one SwCluCNvM-BaseSocket container in an Applicative Software Cluster configuration.

Callback Functions

[SWS_SwCluC_02109]{DRAFT} ⌈The NvM High Proxy shall support the optional configuration of a NvM_SingleBlockCallbackFunction, according to [ECUC_NvM_00562], [ECUC_NvM_00506] and [ECUC_NvM_00559].⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02110]{DRAFT} ⌈The NvM High Proxy shall support the optional configuration of a NvM_PNJF_{Block} port for NvMNotifyJobFinished notification.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02115]{DRAFT} ⌈The NvM High Proxy shall support the optional configuration of a NvM_InitBlockCallbackFunction, according to [ECUC_NvM_00561], [ECUC_NvM_00116] and [ECUC_NvM_00559].⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02116]{DRAFT} ⌈The NvM High Proxy shall support the optional configuration of a NvM_PNIB_{Block} port for NvMNotifyInitBlock notification.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02117]{DRAFT} ⌈The NvM High Proxy shall support the optional configuration of a NvM_ReadRamBlockFromNvm, according to [ECUC_NvM_00521].⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02118]{DRAFT} ⌈The NvM High Proxy shall support the optional configuration of a NvM_WriteRamBlockToNvm, according to [ECUC_NvM_00520].⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02119]{DRAFT} ⌈The NvM High Proxy shall support the optional configuration of a NvM_PM_{Block} port for NvMMirror notification.⌋*(SRS_SwCluC_00206)*

Multiple Partitions

[SWS_SwCluC_02120]{DRAFT} ⌈The NvM High Proxy shall create one partition specific Service Software Component per configured EcucPartition, with the name NvM_<EcucPartition shortName>.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02121]{DRAFT} ⌈The NvM High Proxy shall provide the Ports belonging to a specific Nv Block

- NvM_PS_{Block}
- NvM_PAdmin_{Block}
- NvM_PNIB_{Block}
- NvM_PNJF_{Block}
- NvM_PM_{Block}

at the partition specific Service Software Component, to which the using SwComponentPrototype is mapped.⌋(*SRS_SwCluC_00206*)

Handling of unconnected blocks

[SWS_SwCluC_02107]{DRAFT} ⌈In case a NV block configured in the NvM High Proxy is not connected to any NvM Low Proxy, the NvM High Proxy shall initialize the Nv RAM Block, with the default values given by NvMRomBlockDataAddress and / or NvMInitBlockCallback.⌋(*SRS_SwCluC_00206*)

[SWS_SwCluC_02113]{DRAFT} ⌈In case a NV block configured in the NvM High Proxy is not connected to any NvM Low Proxy, the function NvM_GetErrorStatus of the NvM High Proxy shall still return E_OK, and the request result is set to NVM_REQ_RESTORED_FROM_ROM.⌋(*SRS_SwCluC_00206*)

[SWS_SwCluC_02114]{DRAFT} ⌈In case a NV block configured in the NvM High Proxy is not connected to any NvM Low Proxy, the functions

- NvM_SetDataIndex
- NvM_GetDataIndex
- NvM_SetBlockProtection
- NvM_SetRamBlockStatus
- NvM_SetBlockLockStatus
- NvM_CancelJobs
- NvM_ReadBlock
- NvM_WriteBlock
- NvM_RestoreBlockDefaults
- NvM_EraseNvBlock
- NvM_InvalidateNvBlock
- NvM_ReadPRAMBlock
- NvM_WritePRAMBlock
- NvM_RestorePRAMBlockDefaults

of the NvM High Proxy shall return E_NOT_OK.⌋(*SRS_SwCluC_00206*)

7.4.3.2.4 NvM Low Proxy

As shown in figure 7.10, the NvM High Proxy shall be able to invoke APIs of the BSW modules in the Host Software Cluster, without a partition change in the Applicative Software Clusters. This partition change is only needed, if the BSW module does not offer a satellite on this partition. The availability of the BSW modules on dedicated EcucPartitions is a property of the Host Software Cluster.

[SWS_SwCluC_02132]{DRAFT} ⌈The NvM Low Proxy shall provide the entry functions for single block requests, according to [SWS_SwCluC_02103], once per configured SwCluCNvMBaseSocket at the EcucPartition configured by SwCluCNvMBaseSocket.SwCluCProxyEcucPartitionRef.
In case the SwCluCNvMProxyUsedSatelliteRef is set, the EcucPartition shall be changed, before the NVM API is called.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02111]{DRAFT} ⌈The NvM Low Proxy shall provide an instance of a NvM_SingleBlockCallbackFunction per configured NV block (SwCluCNvMProxyNvBlock) in the NvM Low Proxy.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02127]{DRAFT} ⌈The NvM Low Proxy shall provide an instance of a NvM_InitBlockCallbackFunction per configured NV block (SwCluCNvMProxyNvBlock) in the NvM Low Proxy.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02128]{DRAFT} ⌈The NvM Low Proxy shall provide an instance of a NvM_ReadRamBlockFromNvm per configured NV block (SwCluCNvMProxyNvBlock) in the NvM Low Proxy.⌋*(SRS_SwCluC_00206)*

[SWS_SwCluC_02129]{DRAFT} ⌈The NvM Low Proxy shall provide an instance of a NvM_WriteRamBlockToNvm per configured NV block (SwCluCNvMProxyNvBlock) in the NvM Low Proxy.⌋*(SRS_SwCluC_00206)*

Contrary to the call of BSW module's APIs from an Applicative Software Cluster, the invoked callback notifications need to change the partition in the Host Software Cluster, before the calls are ending up in the Applicative Software Cluster, if the BSW module does not offer a satellite on this partition.

[SWS_SwCluC_02133]{DRAFT} ⌈The callback instances according to [SWS_SwCluC_02111], [SWS_SwCluC_02127], [SWS_SwCluC_02128], and [SWS_SwCluC_02129], are provided

- at the EcucPartition referenced by SwCluCNvMProxyNvBlock.SwCluCNvMProxyNvBlockEcucPartitionRef, if a SwCluCNvMBaseSocket is directly available at this EcucPartition

  OR

- at the EcucPartition of the SwCluCNvMBaseSocket referenced by the SwCluCNvMBaseSocket - via SwCluCNvMProxyUsedSatelliteRef, which is on the EcucPartition given by SwCluCNvMProxyNvBlock.SwCluCNvMProxyNvBlockEcucPartitionRef. In this case, the EcucPartition shall be changed, before the callback in the High Proxy is called.

⌋(*SRS_SwCluC_00206*)

[SWS_SwCluC_02108]{DRAFT} ⌈In case a NV block configured in the NvM Low Proxy is not connected to any NvM High Proxy, the NvM Low Proxy implementation shall ensure that the content of the NV block is not modified by the NvM Low Proxy.⌋ (*SRS_SwCluC_00206*)

Note: An access by Dcm might still change the NV block content in this case.

[SWS_SwCluC_02142]{DRAFT} ⌈The NvM Low Proxy implementation shall describe the NV blocks provided to NvM High Proxies via `NvBlockNeeds`, derived from the `CpSoftwareClusterServiceResource.resourceNeeds`.⌋(*SRS_SwCluC_00206*)

Note: [SWS_SwCluC_02142] ensures an automated configuration of the NVRAM Manager.

[SWS_SwCluC_02143]{DRAFT} ⌈The `NvM Low Proxy` shall provide, for each `SwCluCNvMBaseSocket`, a `SwCluC_NvMProxy_Init` API, to initialize the `NvM High Proxy` of an `Applicative Software Cluster`. This shall be one element of the service resource `SwCluCNvMBaseSocket`.⌋()

7.4.3.2.5 Error detection

[SWS_SwCluC_02137]{DRAFT} ⌈If development error detection is enabled for NvM High Proxy, the APIs

- `NvM_SetDataIndex`
- `NvM_GetDataIndex`
- `NvM_SetBlockProtection`
- `NvM_GetErrorStatus`
- `NvM_SetRamBlockStatus`
- `NvM_SetBlockLockStatus`
- `NvM_CancelJobs`
- `NvM_ReadBlock`
- `NvM_WriteBlock`
- `NvM_RestoreBlockDefaults`
- `NvM_EraseNvBlock`
- `NvM_InvalidateNvBlock`
- `NvM_ReadPRAMBlock`
- `NvM_WritePRAMBlock`

- NvM_RestorePRAMBlockDefaults

shall report the DET error NVM_E_UNINIT, when NVM High Proxy is not yet initialized.⌋ *(SRS_SwCluC_00206, SRS_BSW_00369, SRS_BSW_00323)*

[SWS_SwCluC_02138]{DRAFT} ⌈If development error detection is enabled for NvM High Proxy, the APIs

- NvM_SetDataIndex
- NvM_GetDataIndex
- NvM_SetBlockProtection
- NvM_GetErrorStatus
- NvM_SetRamBlockStatus
- NvM_SetBlockLockStatus
- NvM_CancelJobs
- NvM_ReadBlock
- NvM_WriteBlock
- NvM_RestoreBlockDefaults
- NvM_EraseNvBlock
- NvM_InvalidateNvBlock
- NvM_ReadPRAMBlock
- NvM_WritePRAMBlock
- NvM_RestorePRAMBlockDefaults

shall report the DET error NVM_E_PARAM_BLOCK_ID, when the passed BlockId is out of range of the BlockIds configured in the NvM High Proxy.⌋ *(SRS_SwCluC_00206, SRS_BSW_00369, SRS_BSW_00323)*

[SWS_SwCluC_02139]{DRAFT} ⌈If development error detection is enabled for NvM High Proxy, the APIs

- NvM_ReadBlock
- NvM_WriteBlock
- NvM_RestoreBlockDefaults

shall report the DET error NVM_E_PARAM_ADDRESS, when a RAM block address different than NULL_PTR is passed.⌋ *(SRS_SwCluC_00206, SRS_BSW_00369, SRS_BSW_00323)*

7.4.4 Error Classification

7.4.4.1 Development Errors

[SWS_SwCluC_02140]{DRAFT} **Development Error Types of SwCluC NvM Proxy** ⌈The development error types defined for the SwCluC NvM Proxy are listed in table Table 7.3.⌋*(SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00327, SRS_BSW_00480, SRS_BSW_00487)*

Type of error	*Related error code*	*Value [hex]*
API service called with wrong parameter	NVM_E_PARAM_BLOCK_ID	0x0A
API is called with wrong parameter pointer	NVM_E_PARAM_POINTER	0x0F
API is called when NvM Proxy is not initialized yet	NVM_E_UNINIT	0x14

Table 7.3: Development Error Types for SwCluC NvM Proxy

7.4.4.2 Runtime Errors

Runtime errors are not applicable for the `Cross Cluster Communication` of the `Software Cluster Connection`.

7.4.4.3 Transient Faults

Transient Faults are not applicable for the `Cross Cluster Communication` of the `Software Cluster Connection`.

7.4.4.4 Production Errors

Production Errors are not applicable for the `Cross Cluster Communication` of the `Software Cluster Connection`.

7.4.4.5 Extended Production Errors

Extended Production Errors are not applicable for the `Cross Cluster Communication` of the `Software Cluster Connection`.

7.5 Standardized Service Resources

This section list the standardized definitions of CpSoftwareClusterServiceResource with the defined categorys and the applicable resourceNeeds.

Table 7.4 defines the meaning of the tables for CpSoftwareClusterServiceResource in this section.

Column	***Description***	
Category	category of the CpSoftwareClusterServiceResource referencing hte related EcucContainerValues via the resourceNeeds reference.	
Context and ***Container / Parameter***	path to the standardized definition	
Mult.	Describes the multiplicity, which applies for the Container, Parameter, or Reference when it is used for the description of a CpSoftwareClusterServiceResource. resourceNeeds. This multicity can deviate from the multiplicity defind for the Standardized Module Defintion	
Description / Restrictions	Describes the multiplicity, which applies for the Container, Parameter, or Reference when it is used for the description of a CpSoftwareClusterServiceResource. resourceNeeds	
Guard Value relevant	Defines whether and how the Container, Parameter, or Reference is treated for the calculation of the Guard Value of a Resource Entry. Following list of standard definitions apply:	
	No	The Container, Parameter, or Reference is only used for information or to express a configuration request which is optionally to be fulfilled and does not impact the calculated Guard Value
	Value	The value of the Parameter is considered for the Guard Value calculation
	Choice	Applicable in case the mentioned Container is child of a choice container. The selected kind chosen Container is part of the Guard Value calculation
	from ref	Take the attributes from the referenced Container. The related Container, Parameter, or Reference will be given additionally in this table

Table 7.4: How to read Standardized Service Resource tables

7.5.1 Software Cluster Base Configuration Check

For a CpSoftwareClusterServiceResource defined for a Software Cluster Base Configuration Check [SWS_SwCluC_90000] applies.

[SWS_SwCluC_90000]{DRAFT} **SWCLUSTER_RES_BASE_CNF** ⌈

Category	SWCLUSTER_RES_BASE_CNF			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.5: SWCLUSTER_RES_BASE_CNF

⌋()

7.5.2 Cross Cluster Communication

For a CpSoftwareClusterServiceResource defined for a Cross Cluster Communication Base Socket (see section 7.3.2.3) [SWS_SwCluC_90008] applies.

[SWS_SwCluC_90008]{DRAFT} **SWCLUSTER_RES_XCC_BASE_SOCKET** ⌈

Category	SWCLUSTER_RES_XCC_BASE_SOCKET			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.6: SWCLUSTER_RES_XCC_BASE_SOCKET

⌋()

7.5.3 OS Proxy

For a CpSoftwareClusterServiceResource defined for a Os Base Socket (see section 7.4.3.1.3) [SWS_SwCluC_90002] applies.

[SWS_SwCluC_90002]{DRAFT} **SWCLUSTER_RES_OS_BASE_SOCKET** ⌈

Category	SWCLUSTER_RES_OS_BASE_SOCKET			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.7: SWCLUSTER_RES_OS_BASE_SOCKET

⌋()

For a `CpSoftwareClusterServiceResource` defined for a `OsApplication` (see section 7.4.3.1.5) [SWS_SwCluC_90003] applies.

[SWS_SwCluC_90003]{DRAFT} **SWCLUSTER_RES_OS_APPLICATION** ⌈

Category	SWCLUSTER_RES_OS_APPLICATION			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.8: SWCLUSTER_RES_OS_APPLICATION

⌋()

For a `CpSoftwareClusterServiceResource` defined for a `OsTask` (see section 7.4.3.1.5) [SWS_SwCluC_90004] applies.

[SWS_SwCluC_90004]{DRAFT} **SWCLUSTER_RES_OS_TASK** ⌈

Category	SWCLUSTER_RES_OS_TASK			
Context	*Container / Parameter*	*Mult.*	*Description / Restrictions*	*Guard Value relevant*
no standardized resourceNeeds				

Table 7.9: SWCLUSTER_RES_OS_TASK

⌋()

For a `CpSoftwareClusterServiceResource` defined for a `OsResource` (see section 7.4.3.1.5) [SWS_SwCluC_90005] applies.

[SWS_SwCluC_90005]{DRAFT} **SWCLUSTER_RES_OS_RESOURCE** ⌈

Category	SWCLUSTER_RES_OS_RESOURCE			
Context	**Container / Parameter**	**Mult.**	**Description / Restrictions**	**Guard Value relevant**
no standardized resourceNeeds				

Table 7.10: SWCLUSTER_RES_OS_RESOURCE

⌋()

For a CpSoftwareClusterServiceResource defined for a OsSpinlock (see section 7.4.3.1.5) [SWS_SwCluC_90006] applies.

[SWS_SwCluC_90006]{DRAFT} **SWCLUSTER_RES_OS_SPINLOCK** ⌈

Category	SWCLUSTER_RES_OS_SPINLOCK			
Context	**Container / Parameter**	**Mult.**	**Description / Restrictions**	**Guard Value relevant**
no standardized resourceNeeds				

Table 7.11: SWCLUSTER_RES_OS_SPINLOCK

⌋()

For a CpSoftwareClusterServiceResource defined for a task dispatcher (see section 7.4.3.1.5) [SWS_SwCluC_90007] applies.

[SWS_SwCluC_90007]{DRAFT} **SWCLUSTER_RES_OS_TASK_DISPATCHER** ⌈

Category	SWCLUSTER_RES_OS_TASK_DISPATCHER			
Context	**Container / Parameter**	**Mult.**	**Description / Restrictions**	**Guard Value relevant**
Os/OsTask	OsTaskPriority	1		Value
Os/OsTask	OsTaskSchedule	1		Value

Table 7.12: SWCLUSTER_RES_OS_TASK_DISPATCHER

⌋()

For a CpSoftwareClusterServiceResource defined for a function dispatcher (see section 7.4.3.1.5) [SWS_SwCluC_90008] applies.

[SWS_SwCluC_90009]{DRAFT} **SWCLUSTER_RES_OS_FNC_DISPATCHER** ⌈

Category	SWCLUSTER_RES_OS_FNC_DISPATCHER			
Context	**Container / Parameter**	**Mult.**	**Description / Restrictions**	**Guard Value relevant**
no standardized resourceNeeds				

Table 7.13: SWCLUSTER_RES_OS_FNC_DISPATCHER

⌋()

7.5.4 NvM Proxy

For a `CpSoftwareClusterServiceResource` defined for a `NvM Base Socket` (see section 7.4.3.2.4) [SWS_SwCluC_90010] applies.

[SWS_SwCluC_90010]{DRAFT} **SWCLUSTER_RES_NVM_BASE_SOCKET** ⌈

Category	SWCLUSTER_RES_NVM_BASE_SOCKET			
Context	**Container / Parameter**	**Mult.**	**Description / Restrictions**	**Guard Value relevant**
no standardized resourceNeeds				

Table 7.14: SWCLUSTER_RES_NVM_BASE_SOCKET

⌋()

For a `CpSoftwareClusterServiceResource` defined for a `Nv Block` (see section 7.4.3.2) [SWS_SwCluC_90001] applies.

[SWS_SwCluC_90001]{DRAFT} **SWCLUSTER_RES_NV_BLOCK** ⌈

Category	SWCLUSTER_RES_NV_BLOCK			
Context	**Container / Parameter**	**Mult.**	**Description / Restrictions**	**Guard Value relevant**
NvM/NvMBlockDescriptor	NvMBlockManagementType	1		Value
NvM/NvMBlockDescriptor	NvMBlockUseAutoValidation	1		Value
NvM/NvMBlockDescriptor	NvMBlockUseCrc	1		Value
NvM/NvMBlockDescriptor	NvMBlockUseSetRamBlock Status	1		Value
NvM/NvMBlockDescriptor	NvMBlockWriteProt	1		Value
NvM/NvMBlockDescriptor	NvMCalcRamBlockCrc	1		Value

▽

△

Category	SWCLUSTER_RES_NV_BLOCK			
NvM/NvMBlockDescriptor	NvMResistantToChangedSw	1		Value
NvM/NvMBlockDescriptor	NvMRomBlockNum	1		Value
NvM/NvMBlockDescriptor	NvMSelectBlockForReadAll	1		Value
NvM/NvMBlockDescriptor	NvMSelectBlockForWriteAll	1		Value
NvM/NvMBlockDescriptor	NvMWriteBlockOnce	1		Value
NvM/NvMCommon	NvMSetRamBlockStatusApi	1		Value
SwCluC/SwCluCProxies/SwCluCNvMProxy/SwCluCNvMProxyNvBlock	SwCluCNvMProxyNvBlockMaxLength	1		Value

Table 7.15: SWCLUSTER_RES_NV_BLOCK

⌋()

8 API specification

8.1 Imported types

In this chapter all types included from the following files are listed.

[] ⌈

Module	*Header File*	*Imported Type*
There are no imported types.		

⌋()

Please note that the `Software Cluster Connection` additionally utilizes the Platform Types (as defined in document [14]) and Standard Types (as defined in document [15]).

8.2 Type definitions

8.2.1 Binary Manifest

8.2.1.1 SwCluC_BManif_SwClusterIdType

[SWS_SwCluC_10000]{DRAFT} ⌈

Name	SwCluC_BManif_SwClusterIdType (draft)
Kind	Type
Derived from	uint8
Description	Type of the Software Cluster's ID value **Tags:**atp.Status=draft
Available via	SwCluC_BManif.h

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

8.2.1.2 SwCluC_BManif_MachineIdType

[SWS_SwCluC_10001]{DRAFT} ⌈

Name	SwCluC_BManif_MachineIdType (draft)
Kind	Type
Derived from	uint8
Description	Type for the global ID value of a resource **Tags:**atp.Status=draft
Available via	SwCluC_BManif.h

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

8.2.1.3 SwCluC_BManif_ConCtrlType

[SWS_SwCluC_10002]{DRAFT} ⌈

Name	SwCluC_BManif_ConCtrlType (draft)		
Kind	Type		
Derived from	uint16		
Range	SWCLUC_BMANIF_DISABLE_ON_ECU_CONNECTION	0x8000	The connection of Software Clusters on ECU is disabled for this machine.
Description	Type to code control flags for the connection process. Bit 0..14 are reserved. **Tags:**atp.Status=draft		
Available via	SwCluC_BManif.h		

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

8.2.1.4 SwCluC_BManif_ResourcePropertiesType

[SWS_SwCluC_10003]{DRAFT} ⌈

Name	SwCluC_BManif_ResourcePropertiesType (draft)		
Kind	Type		
Derived from	uint8		
Range	SWCLUC_BMANIF_MANDATORY_RESOURCE	0x40	the required resource is mandatory
	SWCLUC_BMANIF_PROVIDED_RESOURCE	0x80	the resource is provided by the Software Cluster
Description	Type to code the properties of a resource. Bit 0..5 are reserved. **Tags:**atp.Status=draft		
Available via	SwCluC_BManif.h		

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

8.2.1.5 SwCluC_BManif_ResourceTypeIdType

[SWS_SwCluC_10004]{DRAFT} ⌈

Name	SwCluC_BManif_ResourceTypeIdType (draft)
Kind	Type
Derived from	uint8
Description	Type to code the type of a resource **Tags:**atp.Status=draft
Available via	SwCluC_BManif.h

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

8.2.1.6 SwCluC_BManif_GlobalResourceIdType

[SWS_SwCluC_10005]{DRAFT} ⌈

Name	SwCluC_BManif_GlobalResourceIdType (draft)
Kind	Type
Derived from	uint32
Description	Type for the global ID value of a resource **Tags:**atp.Status=draft
Available via	SwCluC_BManif.h

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

8.2.1.7 SwCluC_BManif_ResourceGuardValueType

[SWS_SwCluC_10006]{DRAFT} ⌈

Name	SwCluC_BManif_ResourceGuardValueType (draft)
Kind	Type
Derived from	uint32
Description	Type for the guard value of a resource **Tags:**atp.Status=draft
Available via	SwCluC_BManif.h

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

8.2.1.8 SwCluC_BManif_TableIndexType

[SWS_SwCluC_10007]{DRAFT} ⌈

Name	SwCluC_BManif_TableIndexType (draft)
Kind	Type
Derived from	uint16
Description	Index type to address the immutable and modifiable interface table **Tags:**atp.Status=draft
Available via	SwCluC_BManif.h

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

8.2.1.9 SwCluC_BManif_HandleIndexType

[SWS_SwCluC_10008]{DRAFT} ⌈

Name	SwCluC_BManif_HandleIndexType (draft)
Kind	Type
Derived from	uint8
Description	Index type to address a set of handles in the immutable and modifiable interface table **Tags:**atp.Status=draft
Available via	SwCluC_BManif.h

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

8.2.1.10 SwCluC_BManif_VoidFncPtrType

[SWS_SwCluC_10009]{DRAFT} ⌈

Name	SwCluC_BManif_VoidFncPtrType (draft)
Kind	Pointer
Type	int (*SwCluC_BManif_VoidFncPtrType)(void)
Description	Generic function pointer type according C default type for functions **Tags:**atp.Status=draft
Available via	SwCluC_BManif.h

⌋(*SRS_SwCluC_00006*)

8.2.1.11 SwCluC_BManif_HandleType

[SWS_SwCluC_10010]{DRAFT} ⌈

Name	SwCluC_BManifHandleType (draft)	
Kind	Union	
Elements	dptr	
	Type	void*
	Comment	Handle as pointer to variable
	val	
	Type	uint32
	Comment	Handle as value
	fptr	
	Type	SwCluC_BManif_VoidFncPtrType
	Comment	Handle as pointer to function
Description	Type of a handle in the interface table. **Tags:**atp.Status=draft	
Available via	SwCluC_BManif.h	

⌋(*SRS_SwCluC_00006*)

8.2.1.12 SwCluC_BManif_HeaderType

[SWS_SwCluC_10011]{DRAFT} ⌈

Name	SwCluC_BManif_HeaderType (draft)	
Kind	Structure	
	Preamble	
	Type	uint64
	Comment	magic pattern of Manifest Header begin
	ManifestMajorVersion	

▽

△

	Type	uint8
	Comment	version of the Binary Manifest Layout
	ManifestMinorVersion	
	Type	uint8
	Comment	version of the Binary Manifest Layout
	SwClusterId	
	Type	SwCluC_BManif_SwClusterIdType
	Comment	Software Cluster Id
	MachineId	
	Type	SwCluC_BManif_MachineIdType
	Comment	Machine Id
	SwClusterType	
	Type	uint8
	Comment	kind of Software Cluster
	Reserved1	
	Type	uint8
	Comment	reserved
	Reserved2	
	Type	uint8
	Comment	reserved
	Reserved3	
	Type	uint8
	Comment	reserved
	ConnectorControlFlags	
	Type	SwCluC_BManif_ConCtrlType
	Comment	Control flags for Software Cluster connector
	NoOfInterfaceDescriptors	
	Type	uint16
	Comment	Number of Interface Descriptors
	NoOfOfferedInterfaceHandles	
	Type	uint16
	Comment	Number of Handles in the Offered Interface
	NoOfSubscribedInterfaceHandles	
	Type	uint16
	Comment	Number of Handles in the Subscribed Interface
	ImmutableTablesChecksumPtr	
	Type	const uint32*
	Comment	Pointer to checksum about immutable descriptor and interface tables
	TotalManifestChecksumPtr	
	Type	const uint32*
	Comment	Pointer to checksum about complete Binary Manifest
	SubscribedTablesValidityMarkerPtr	
	Type	const uint32*

▽

△

	Comment	Pointer to validity marker indicating whether all subscribed tables are written
	ResourcePropertiesDescriptorColumnPtr	
	Type	const SwCluC_BManif_ResourcePropertiesType*
	Comment	Pointer to descriptor column for resource properties
	ResourceTypeDescriptorColumnPtr	
	Type	const SwCluC_BManif_ResourceTypeIdType*
	Comment	Pointer to descriptor column for resource type Ids
	GlobalResourceIdDescriptorColumnPtr	
	Type	const SwCluC_BManif_GlobalResourceIdType*
	Comment	Pointer to descriptor column for global resource Ids
	ResourceGuardValueDescriptorColumnPtr	
	Type	const SwCluC_BManif_ResourceGuardValueType*
	Comment	Pointer to descriptor column for guard values
	OfferedInterfaceIndexDescriptorColumnPtr	
	Type	const SwCluC_BManif_TableIndexType*
	Comment	Pointer to descriptor column for offered interface column index
	OfferedInterfaceNoOfHandlesDescriptorColumnPtr	
	Type	const SwCluC_BManif_HandleIndexType*
	Comment	Pointer to descriptor column for number of handles in offered interface column
	SubscribedInterfaceIndexDescriptorColumnPtr	
	Type	const SwCluC_BManif_TableIndexType*
	Comment	Pointer to descriptor column for subscribed interface column index
	SubscribedInterfaceNoOfHandlesDescriptorColumnPtr	
	Type	const SwCluC_BManif_HandleIndexType*
	Comment	Pointer to descriptor column for number of handles in subscribed interface column
	SubscribedInterfaceNoOfHandleSetsDescriptorColumnPtr	
	Type	const SwCluC_BManif_HandleIndexType*
	Comment	Pointer to descriptor column for number of handle sets in subscribed interface column
	OfferedInterfaceHandleColumnPtr	
	Type	const SwCluC_BManif_HandleType*
	Comment	Pointer to interface column for offered handles
	SubscribedInterfaceDefaultHandleColumnPtr	
	Type	const SwCluC_BManif_HandleType*
	Comment	Pointer to default interface column for subscribed handles
	SubscribedInterfaceHandleColumnPtr	
	Type	const SwCluC_BManif_HandleType*
	Comment	Pointer to interface column for subscribed handles
	SubscribedInterfaceConnectedSwClusterIdColumnPtr	
	Type	const SwCluC_BManif_SwClusterIdType*
	Comment	Pointer to interface column for connected cluster Id values

▽

△

Description	– **Tags:**atp.Status=draft
Available via	

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

8.2.2 Cross Cluster Connection

The `Cross Cluster Connection` implements the types of a `Cross Software Cluster Communication Plug-In` as specified in document [9].

8.2.3 ProxyModules

The `Proxy Modules` partly implement the types of the according AUTOSAR BSW modules. Those are listed in appendix D.

8.3 Function definitions

8.3.1 BinaryManifest

8.3.1.1 API Principles

None of the functions defined for the `Binary Manifest` access are utilizing the `Std_ReturnType` to provide runtime errors. Since the `Binary Manifest` itself is implemented by a set of const variables, which need to be located to Flash ROM, all the failure situations caused by a temporary unavailability of the `Binary Manifest` can be excluded. Instead, it needs to be considered that the `Binary Manifest` is involved very frequently, since it is needed to access any resource from other `Software Clusters`, including the `Host Software Cluster`.

When an implementation of the Software Cluster Connection or a CDD utilizes the `Binary Manifest`, it is not expected that the usage of the `SwCluC_BManif_GetHandle`, `SwCluC_BManif_GetConSwClusterId`, and `SwCluC_BManif_GetNoOfHandleSets` APIs result in real function calls. From the implementation perspective of the `Binary Manifest`, this should never be required, since accesses to handles or Software Cluster Ids are accesses to primitive constant variables.

[SWS_SwCluC_00057]{DRAFT} ⌈The functions `SwCluC_BManif_GetHandle`, `SwCluC_BManif_GetConSwClusterId`, and `SwCluC_BManif_GetNoOfHandleSets`, according to [SWS_SwCluC_10020], [SWS_SwCluC_10021], [SWS_SwCluC_10022], [SWS_SwCluC_10023], [SWS_SwCluC_10032], and

[SWS_SwCluC_10033], shall each be implemented as a C-function-like macro.⌋ *(SRS_SwCluC_00012)*

[SWS_SwCluC_00058]{DRAFT} ⌈<Resource Entry Group> is the shortName of either the SwCluCBManifProvideResourceEntryGroup or SwCluCBManifRequireResourceEntryGroup.⌋*(SRS_SwCluC_00012)*

[SWS_SwCluC_00059]{DRAFT} ⌈<Resource Entry> is the shortName of either the SwCluCBManifProvideResourceEntry or SwCluCBManifRequireResourceEntry.⌋*(SRS_SwCluC_00012)*

[SWS_SwCluC_00060]{DRAFT} ⌈<Handle> is the shortName of either the SwCluCBManifProvideHandle or SwCluCBManifNotifierHandle of the SwCluCBManifResourceType, referenced by SwCluCBManifProvideResourceEntryGroup or SwCluCBManifRequireResourceEntryGroup.⌋*(SRS_SwCluC_00012)*

[SWS_SwCluC_00061]{DRAFT} ⌈<handleType> is either

- the referenced ImplementationDataType, if the SwCluCBManifHandleImpleTypeRef is defined for a Provide Handle of Notifier Handle. OR
- uint32, if the SwCluCBManifHandleImpleTypeRef is not defined, and SwCluCBManifNativeHandleType is set to VALUE OR
- void *, if the SwCluCBManifHandleImpleTypeRef is not defined, and SwCluCBManifNativeHandleType is set to DATA_REFERENCE OR
- SwCluC_BManif_VoidFncPtrType, if the SwCluCBManifHandleImpleTypeRef is not defined, and SwCluCBManifNativeHandleType is set to FUNCTION_REFERENCE

⌋*(SRS_SwCluC_00012)*

8.3.1.2 SwCluC_BManif_GetHandle

[SWS_SwCluC_10020]{DRAFT} ⌈

Service Name	SwCluC_BManif_GetHandle_<ResourceEntryGroup>_<ResourceEntry>_<Handle> (draft)
Syntax	<handleType> SwCluC_BManif_GetHandle_<ResourceEntryGroup>_<ResourceEntry>_<Handle> (SwCluC_BManif_HandleIndexType notifierSetIndex)
Service ID [hex]	0x10
Sync/Async	Synchronous
Reentrancy	Reentrant

▽

△

Parameters (in)	notifierSetIndex	Optional parameter for the notifier set index in the range 0.. SWCLUC_BMANIF_MAX_NO_OF_NOTIFIER_ SETS_<Resource Entry Group>_<Resource Entry> -1 It exists if for a notifier handle of a provided resource multiple notifier sets are supported
Parameters (inout)	None	
Parameters (out)	None	
Return value	<handleType>	Pointer or value stored in Binary Manifest for this handle
Description	Returns a handle of a Resource Entry in a Resource Entry Group **Tags:**atp.Status=draft	
Available via	SwCluC_BManif.h	

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

[SWS_SwCluC_00062]{DRAFT} ⌈The function `SwCluC_BManif_GetHandle` according to [SWS_SwCluC_10020] shall exist for each `SwCluCBManifProvideHandle` and `SwCluCBManifNotifierHandle` defined for each `SwCluCBManifProvideResourceEntry` and `SwCluCBManifRequireResourceEntry` in a `SwCluCBManifProvideResourceEntryGroup` and `SwCluCBManifRequireResourceEntryGroup`.⌋(*SRS_SwCluC_00006*)

8.3.1.3 SwCluC_BManif_GetConSwClusterId

[SWS_SwCluC_10021]{DRAFT} ⌈

Service Name	SwCluC_BManif_GetConSwClusterId_<ResourceEntryGroup>_<ResourceEntry>_<Handle> (draft)	
Syntax	`SwCluC_BManif_SwClusterIdType SwCluC_BManif_GetConSwClusterId_< ResourceEntryGroup>_<ResourceEntry>_<Handle> ( SwCluC_BManif_HandleIndexType notifierSetIndex )`	
Service ID [hex]	0x11	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	notifierSetIndex	Optional parameter for the notifier set index in the range 0.. SWCLUC_BMANIF_MAX_NO_OF_NOTIFIER_ SETS_<Resource Entry Group>_<Resource Entry> -1 It exists if for a notifier handle of a provided resource multiple notifier sets are supported
Parameters (inout)	None	
Parameters (out)	None	
Return value	SwCluC_BManif_SwClusterIdType	Id of the connected Software Cluster

▽

△

Description	Returns the Id of the connected Software Cluster for a Notifier Handle of a Provide Resource Entry or a Provide Handle of a Require Resource Entry **Tags:**atp.Status=draft
Available via	SwCluC_BManif.h

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

[SWS_SwCluC_00063]{DRAFT} ⌈The function `SwCluC_BManif_GetConSwClusterId` according to [SWS_SwCluC_10021] shall exist

- for each `SwCluCBManifNotifierHandle` defined for each `SwCluCBManifProvideResourceEntry` in a `SwCluCBManifProvideResourceEntryGroup`

  AND

- for each `SwCluCBManifProvideHandle` defined for each `SwCluCBManifRequireResourceEntry` in a `SwCluCBManifRequireResourceEntryGroup`.

⌋(*SRS_SwCluC_00006*)

8.3.1.4 SwCluC_BManif_GetHandle

[SWS_SwCluC_10022]{DRAFT} ⌈

Service Name	SwCluC_BManif_GetHandle_<ResourceEntryGroup>_<Handle> (draft)	
Syntax	`SwCluC_BManif_SwClusterIdType SwCluC_BManif_GetHandle_<ResourceEntryGroup>_<Handle> ( SwCluC_BManif_TableIndexType resIndex, SwCluC_BManif_HandleIndexType notifierSetIndex )`	
Service ID [hex]	0x12	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	resIndex	Index of the resource entry in a Resource Entry Group in the range 0.. SWCLUC_BMANIF_NO_OF_ENTRIES_<Resource Entry Group> -1.
	notifierSetIndex	Optional parameter for the notifier set index in the range 0.. SWCLUC_BMANIF_MAX_NO_OF_NOTIFIER_SETS_<Resource Entry Group>_<Resource Entry> -1 It exists if for a notifier handle of a provided resource multiple notifier sets are supported
Parameters (inout)	None	
Parameters (out)	None	
Return value	SwCluC_BManif_SwClusterIdType	Id of the connected Software Cluster

▽

△

Description	Returns the Id of the connected Software Cluster for a Notifier Handle of a Provide Resource Entry or a Provide Handle of a Require Resource Entry **Tags:**atp.Status=draft
Available via	SwCluC_BManif.h

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

[SWS_SwCluC_00064]{DRAFT} ⌈The function `SwCluC_BManif_GetHandle` according to [SWS_SwCluC_10022] shall exist for each `SwCluCBManifProvideHandle` and `SwCluCBManifNotifierHandle` defined for each `SwCluCBManifProvideResourceEntryGroup` and `SwCluCBManifRequireResourceEntryGroup`.⌋(*SRS_SwCluC_00006*)

8.3.1.5 SwCluC_BManif_GetConSwClusterId

[SWS_SwCluC_10023]{DRAFT} ⌈

Service Name	SwCluC_BManif_GetConSwClusterId_<ResourceEntryGroup>_<Handle> (draft)	
Syntax	`SwCluC_BManif_SwClusterIdType SwCluC_BManif_GetConSwClusterId_< ResourceEntryGroup>_<Handle> ( SwCluC_BManif_TableIndexType resIndex, SwCluC_BManif_HandleIndexType notifierSetIndex )`	
Service ID [hex]	0x13	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	resIndex	Index of the resource entry in a Resource Entry Group in the range 0.. SWCLUC_BMANIF_NO_OF_ENTRIES_<Resource Entry Group> -1.
	notifierSetIndex	Optional parameter for the notifier set index in the range 0.. SWCLUC_BMANIF_MAX_NO_OF_NOTIFIER_ SETS_<Resource Entry Group>_<Resource Entry> -1 It exists if for a notifier handle of a provided resource multiple notifier sets are supported.
Parameters (inout)	None	
Parameters (out)	None	
Return value	SwCluC_BManif_Sw ClusterIdType	Id of the connected Software Cluster
Description	Returns the Id of the connected Software Cluster for a Notifier Handle of a Provide Resource Entry or Provide Handle of a Require Resource Entry **Tags:**atp.Status=draft	
Available via	SwCluC_BManif.h	

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

[SWS_SwCluC_00065]{DRAFT} ⌈The function `SwCluC_BManif_GetConSwClusterId` according to [SWS_SwCluC_10023] shall exist

- for each SwCluCBManifNotifierHandle defined for a SwCluCBManifProvideResourceEntryGroup

  AND

- for each SwCluCBManifProvideHandle defined for a SwCluCBManifRequireResourceEntryGroup.

⌋(*SRS_SwCluC_00006*)

8.3.1.6 SwCluC_BManif_GetNoOfHandleSets

[SWS_SwCluC_10032]{DRAFT} ⌈

Service Name	SwCluC_BManif_GetNoOfHandleSets_<Resource Entry Group>_<Resource Entry> (draft)	
Syntax	`SwCluC_BManif_HandleIndexType SwCluC_BManif_GetNoOfHandleSets_<Resource Entry Group>_<Resource Entry> ( void )`	
Service ID [hex]	0x14	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	SwCluC_BManif_HandleIndexType	Number of actually used - and thereby connected - handle sets
Description	Returns the number of actually used - and thereby connected - handle sets **Tags:**atp.Status=draft	
Available via	SwCluC_BManif.h	

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

[SWS_SwCluC_00066]{DRAFT} ⌈The function SwCluC_BManif_GetNoOfHandleSets according to [SWS_SwCluC_10032] shall exist for each SwCluCBManifProvideResourceEntry in a SwCluCBManifProvideResourceEntryGroup where the parameter SwCluCBManifMultipleNotifierSupport is set to MULTIPLE_NOTIFIER_SETS.⌋(*SRS_SwCluC_00006*)

8.3.1.7 SwCluC_BManif_GetNoOfHandleSets

[SWS_SwCluC_10033]{DRAFT} ⌈

Service Name	SwCluC_BManif_GetNoOfHandleSets_<Resource Entry Group> (draft)	
Syntax	`SwCluC_BManif_HandleIndexType SwCluC_BManif_GetNoOfHandleSets_< Resource Entry Group> ( SwCluC_BManif_TableIndexType resIndex )`	
Service ID [hex]	0x15	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	resIndex	Index of the resource entry in a Resource Entry Group in the range 0.. SWCLUC_BMANIF_NO_OF_ENTRIES_<Resource Entry Group> -1.
Parameters (inout)	None	
Parameters (out)	None	
Return value	SwCluC_BManif_HandleIndexType	Number of actually used - and thereby connected - handle sets
Description	Returns the number of actually used - and thereby connected - handle sets. **Tags:**atp.Status=draft	
Available via	SwCluC_BManif.h	

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

[SWS_SwCluC_00067]{DRAFT} ⌈The function `SwCluC_BManif_GetNoOfHandleSets` according to [SWS_SwCluC_10033] shall exist for each `SwCluCBManifProvideResourceEntryGroup` where the parameter `SwCluCBManifMultipleNotifierSupport` is set to `MULTIPLE_NOTIFIER_SETS`.⌋(*SRS_SwCluC_00006*)

8.3.1.8 SwCluC_BManif_GetValidityMarker

[SWS_SwCluC_10034]{DRAFT} ⌈

Service Name	SwCluC_BManif_GetValidityMarker (draft)	
Syntax	`uint32 SwCluC_BManif_GetValidityMarker ( void )`	
Service ID [hex]	0x16	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint32	Value of the Subscribed Interface Validity Marker
Description	Returns the value of the Subscribed Interface Validity Marker. **Tags:**atp.Status=draft	

▽

△

Available via	SwCluC_BMManif.h

⌋(*SRS_SwCluC_00006*, *SRS_BSW_00310*)

8.3.1.9 SWCLUC_BMANIF_NO_OF_ENTRIES

[SWS_SwCluC_01000]{DRAFT} ⌈
Symbolic name: SWCLUC_BMANIF_NO_OF_ENTRIES_<Resource Entry Group>
Value: Number of Resource Entrys in the according Resource Entry Group
Comments: n.a.⌋(*SRS_SwCluC_00006*, *SRS_BSW_00441*)

[SWS_SwCluC_00068]{DRAFT} ⌈The definition SWCLUC_BMANIF_NO_OF_ENTRIES according to [SWS_SwCluC_01000] shall exist for each SwCluCBManifProvideResourceEntryGroup.⌋(*SRS_SwCluC_00006*)

8.3.1.10 SWCLUC_BMANIF_NO_OF_NOTIFIER_SETS

[SWS_SwCluC_01002]{DRAFT} ⌈
Symbolic name: SWCLUC_BMANIF_MAX_NO_OF_NOTIFIER_SETS_
<Resource Entry Group>_<Resource Entry>
Value: Maximum number of notifier sets for the according Provide Resource Entry
Comments: n.a.⌋(*SRS_SwCluC_00006*, *SRS_BSW_00441*)

[SWS_SwCluC_00069]{DRAFT} ⌈The definition SWCLUC_BMANIF_NO_OF_NOTIFIER_SETS according to [SWS_SwCluC_01002] shall exist for each SwCluCBManifProvideResourceEntry in a SwCluCBManifProvideResourceEntryGroup where the parameter SwCluCBManifMultipleNotifierSupport is set to MULTIPLE_NOTIFIER_SETS.⌋(*SRS_SwCluC_00006*)

8.3.2 Cross Cluster Communication

In addition to the interfaces listed in this section, the Cross Cluster Connection implements the interfaces of a Cross Software Cluster Communication Plug-In as specified in document [9].

8.3.2.1 SwCluC_Xcc_Init1

[SWS_SwCluC_11000]{DRAFT} ⌈

Service Name	SwCluC_Xcc_Init1 (draft)
Syntax	`void SwCluC_Xcc_Init1 ( void )`
Service ID [hex]	0x20
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	SwCluC_Xcc_Init is intended to allocate and initialize system resources used by the Software Cluster Connection. It initializes interface memory which might be needed to be read by other Software Clusters but it is not allowed to do any read access to memory of any another Software Cluster. **Tags:**atp.Status=draft
Available via	SwCluC.h

⌋(*SRS_SwCluC_00100*, *SRS_SwCluC_00108*, *SRS_BSW_00101*, *SRS_BSW_00358*, *SRS_BSW_00310*)

8.3.2.2 SwCluC_Xcc_Init2

[SWS_SwCluC_11001]{DRAFT} ⌈

Service Name	SwCluC_Xcc_Init2 (draft)
Syntax	`void SwCluC_Xcc_Init2 ( void )`
Service ID [hex]	0x21
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	SwCluC_Xcc_Init2 is intended to allocate and initialize system resources used by the Software Cluster Connection. It can execute initializations which require a read access to interface memory of another Software Clusters. **Tags:**atp.Status=draft
Available via	SwCluC.h

⌋(*SRS_SwCluC_00100*, *SRS_SwCluC_00108*, *SRS_BSW_00101*, *SRS_BSW_00358*, *SRS_BSW_00310*)

8.3.3 Proxy Modules

The Proxy Modules partly implement the interfaces of the according AUTOSAR BSW modules. Those are listed in appendix D.

Following functions are provided in addition:

8.3.3.1 SwCluC_OsProxy_Init

[SWS_SwCluC_12000]{DRAFT} ⌈

Service Name	SwCluC_OsProxy_Init_<SwCluCOsProxyOsBaseSocket> (draft)

Syntax	`void SwCluC_OsProxy_Init_<SwCluCOsProxyOsBaseSocket> (`
	`  void`
	`)`

Service ID [hex]	0x30
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	SwCluC_OsProxy_Init is intended to allocate and initialize system resources used by the Os Proxy linked to this specific socked **Tags:**atp.Status=draft
Available via	SwCluC.h

⌋(*SRS_BSW_00101*, *SRS_BSW_00358*, *SRS_BSW_00310*)

The name part <SwCluCOsProxyOsBaseSocket> is the shortName of the SwCluCOsProxyOsBaseSocket.

8.3.3.2 SwCluC_NvMProxy_Init

[SWS_SwCluC_12100]{DRAFT} ⌈

Service Name	SwCluC_NvMProxy_Init_<SwCluCNvMBaseSocket> (draft)

Syntax	`void SwCluC_NvMProxy_Init_<SwCluCNvMBaseSocket> (`
	`  void`
	`)`

Service ID [hex]	0x31
Sync/Async	Synchronous
Reentrancy	Non Reentrant

▽

△

Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	SwCluC_NvMProxy_Init is intended to allocate and initialize system resources used by the Nv MProxy linked to this specific socked. **Tags:**atp.Status=draft
Available via	SwCluC.h

⌋(*SRS_BSW_00101*, *SRS_BSW_00358*, *SRS_BSW_00310*)

The name part <SwCluCNvMBaseSocket> is the shortName of the SwCluCNvMBaseSocket.

8.4 Callback notifications

This is a list of functions provided for other modules.

8.4.1 Binary Manifest

The Binary Manifest has no callback notifications.

8.4.2 Cross Cluster Connection

The Cross Cluster Connection implements the interfaces of a Cross Software Cluster Communication Plug-In as specified in document [9].

8.4.3 Proxy Modules

The Proxy Modules partly implement the callback notifications of the according AUTOSAR BSW modules. Those are listed in appendix D.

8.5 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

8.5.1 Binary Manifest

The `Binary Manifest` has no scheduled functions.

8.5.2 Cross Cluster Communication Scheduled functions

The `Cross Cluster Communications` has no standardized scheduled functions but is free to implement some if functionally needed.

8.5.3 Proxy Modules Scheduled functions

The `Proxy Modules` have no standardized scheduled functions but are free to implement some if functionally needed.

8.6 Expected interfaces

In this chapter, all interfaces required from other modules are listed.

8.6.1 Mandatory interfaces

Note: This section defines all interfaces, which are required to fulfill the core functionality of the module.

8.6.1.1 Binary Manifest

The `Binary Manifest` has no mandatory interfaces.

8.6.1.2 Cross Cluster Connection

The `Cross Cluster Connection` has no mandatory interfaces.

8.6.1.3 Proxy Modules

The `Proxy Modules` partly implement the mandatory interfaces of the according AUTOSAR BSW modules. Those are listed in appendix D.

8.6.2 Optional interfaces

This section defines all interfaces, which are required to fulfill an optional functionality of the module.

8.6.2.1 Binary Manifest

The `Binary Manifest` has no optional interfaces.

8.6.2.2 Cross Cluster Connection

The `Cross Cluster Connection` has no optional interfaces.

8.6.2.3 Proxy Modules

The `Proxy Modules` partly implement the optional interfaces of the according AUTOSAR BSW modules. Those are listed in appendix D.

8.6.3 Configurable interfaces

In this section, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of this kind of interfaces are not fixed because they are configurable.

8.6.3.1 Binary Manifest

The `Binary Manifest` has no configurable interfaces.

8.6.3.2 Cross Cluster Connection

8.6.3.3 Proxy Modules

The `Proxy Modules` partly implement the configurable interfaces of the according AUTOSAR BSW modules. Those are listed in appendix D.

8.7 Service Interfaces

8.7.1 Binary Manifest

The `Binary Manifest` has no Service Interfaces.

8.7.2 Cross Cluster Connection

The `Cross Cluster Connection` has no Service Interfaces.

8.7.3 Proxy Modules

The `Proxy Modules` only implement Service Interfaces, which are already defined by the original AUTOSAR Service. Those are listed in appendix E.

9 Sequence diagrams

No sequence diagrams are contained in this section.

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module `SwCluC`.

Chapter 10.3 specifies published information of the module `SwCluC`.

10.1 How to read this chapter

For details, refer to the chapter 10.1 “Introduction to configuration specification” in SWS_BSWGeneral.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 7 and Chapter 8.

10.2.1 Module Configuration

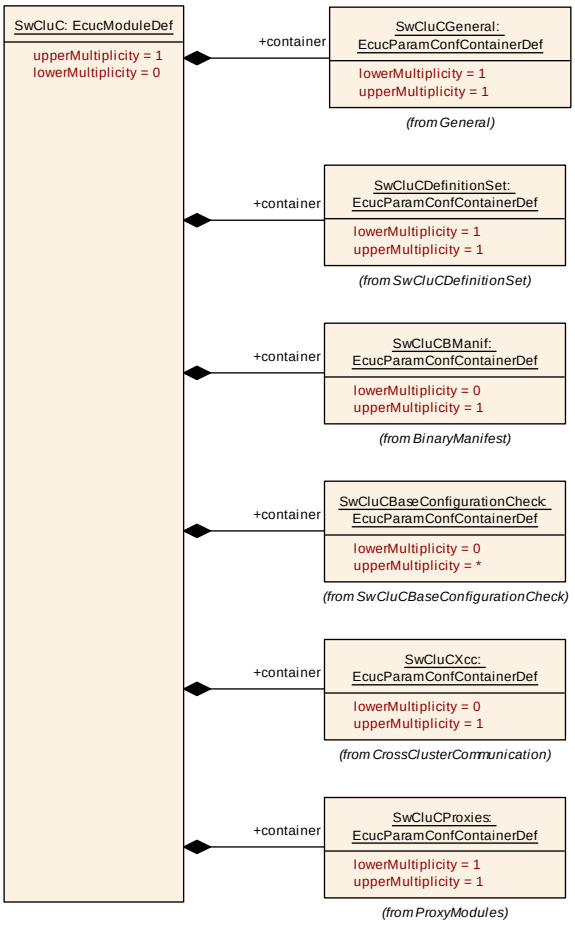

Figure 10.1: SwCluC configuration overview

Module SWS Item	ECUC_SwCluC_00001	
Module Name	SwCluC	
Module Description	Module to collect Software Cluster Connection specific configuration information.	
Post-Build Variant Support	true	
Supported Config Variants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE	
Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**

Container Name	Multiplicity	Scope / Dependency
SwCluCBManif	0..1	Configuration of the Binary Manifest of the Software Cluster Connection **Tags:** atp.Status=draft
SwCluCBaseConfiguration Check	0..*	Configuration of the base configuration check of Software Cluster Connection. This configuration places a mandatory entry in the Binary Manifest where the guard value is used to proof if the general setup of the Applicative Software Cluster is compatible to the general setup of the Host Software Cluster. For each Applicative Software Cluster an individual base configuration check should be defined. **Tags:** atp.Status=draft
SwCluCDefinitionSet	1	Definition of a set of Software Clusters **Tags:** atp.Status=draft
SwCluCGeneral	1	General configuration of the Software Cluster Connection **Tags:** atp.Status=draft
SwCluCProxies	1	General configuration of the Proxy Modules of Software Cluster Connection **Tags:** atp.Status=draft
SwCluCXcc	0..1	Configuration of the Binary Manifest of the Software Cluster Connection **Tags:** atp.Status=draft

10.2.2 General configuration parameters

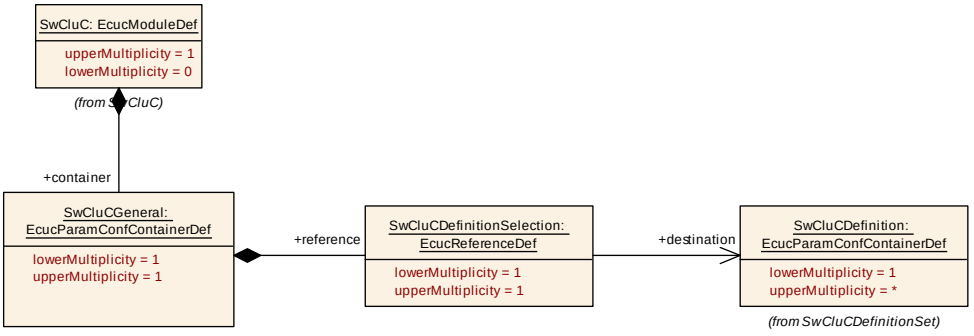

Figure 10.2: SwCluC General configuration parameters

SWS Item	[ECUC_SwCluC_00002]

Container Name	SwCluCGeneral		
Parent Container	SwCluC		
Description	General configuration of the Software Cluster Connection **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCDefinitionSelection [ECUC_SwCluC_00005]		
Parent Container	SwCluCGeneral		
Description	This reference selects the Software Cluster Definition which is applied for this Software Cluster. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Reference to SwCluCDefinition		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.2.3 Software Cluster Definition

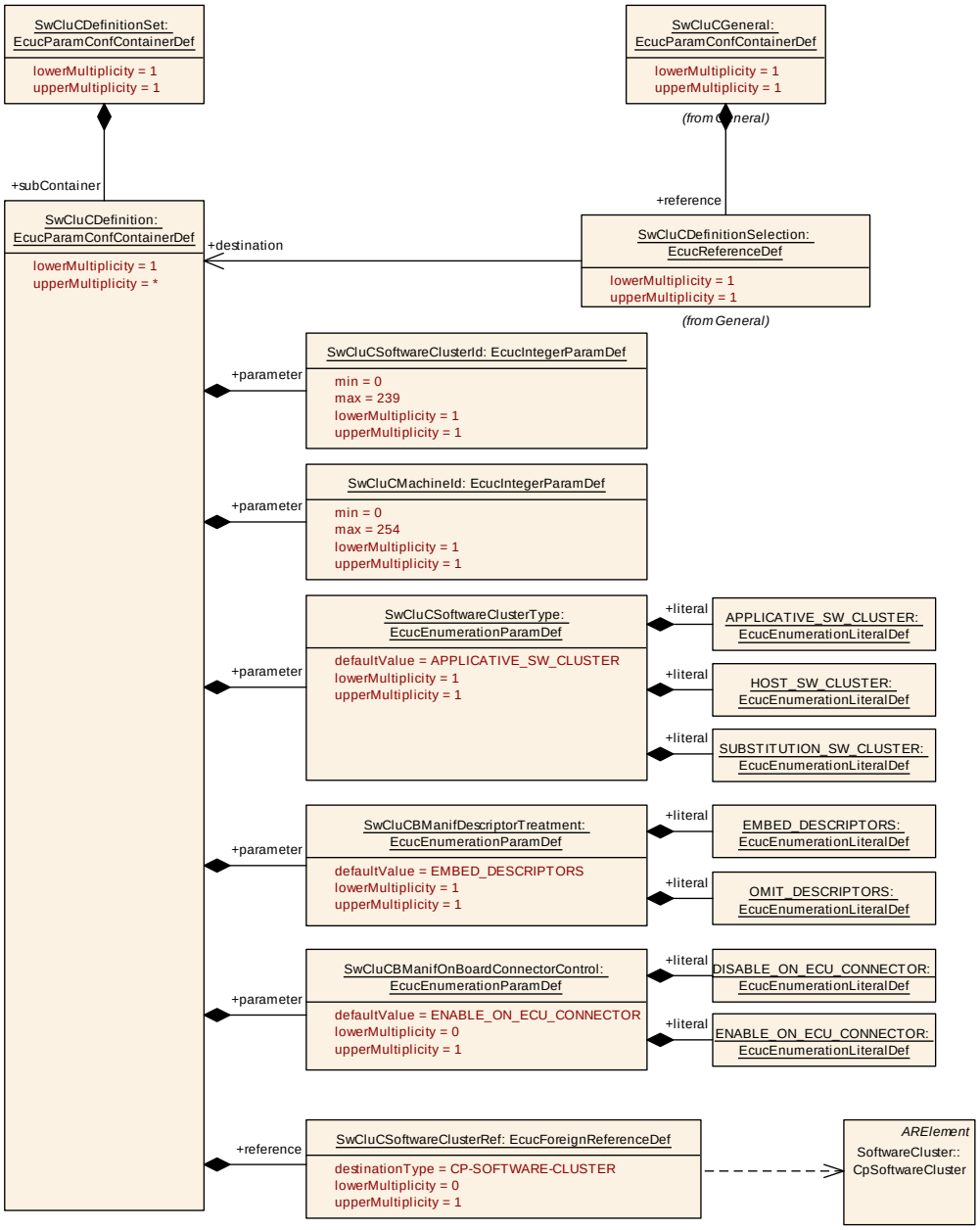

Figure 10.3: SwCluC Software Cluster Definition

SWS Item	[ECUC_SwCluC_00004]
Container Name	SwCluCDefinitionSet
Parent Container	SwCluC
Description	Definition of a set of Software Clusters **Tags:** atp.Status=draft
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
SwCluCDefinition	1..*	Definition of the general properties of the Software Cluster. The defintions are separated in a set of defintions in order to enable an holistic view about all exsiting Software Cluster Definitions. **Tags:** atp.Status=draft

SWS Item	[ECUC_SwCluC_00006]		
Container Name	SwCluCDefinition		
Parent Container	SwCluCDefinitionSet		
Description	Definition of the general properties of the Software Cluster. The defintions are separated in a set of defintions in order to enable an holistic view about all exsiting Software Cluster Definitions. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCBManifDescriptorTreatment [ECUC_SwCluC_00010]	
Parent Container	SwCluCDefinition	
Description	Configures the existence of the Interface Descriptor Table in the Binary Object of the Software Cluster. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	EMBED_DESCRIPTORS	The Interface Descriptor Table of the Binary Manifest is embed into the Binary Object. **Tags:** atp.Status=draft
	OMIT_DESCRIPTORS	The Interface Descriptor Table of the Binary Manifest is omitted from the Binary Object. The information needs to be delivered as CpSoftwareClusterBinaryManifest-Descriptor. **Tags:** atp.Status=draft
Default Value	EMBED_DESCRIPTORS	

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCBManifOnBoardConnectorControl [ECUC_SwCluC_00011]		
Parent Container	SwCluCDefinition		
Description	SwCluCBManifOnBoardConnectorControl enables or disables the onECU execution of the Software Cluster Connector. This setting is only applicable for the Host Software Cluster. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	DISABLE_ON_ECU_CONNECTOR	The connection of Software Clusters OnECU is disabled for this machine. **Tags:** atp.Status=draft	
	ENABLE_ON_ECU_CONNECTOR	The connection of Software Clusters OnECU is enabled for this machine. **Tags:** atp.Status=draft	
Default Value	ENABLE_ON_ECU_CONNECTOR		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCMachineId [ECUC_SwCluC_00008]		
Parent Container	SwCluCDefinition		
Description	Unique number of the (virtual or physical) machine to which the Software Cluster belongs. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 254		
Default Value			
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCSoftwareClusterId [ECUC_SwCluC_00007]		
Parent Container	SwCluCDefinition		
Description	Unique number of the Software Cluster Numbers >= 0xFE are reserved to indicate special values or the unconnected state. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 239		
Default Value			
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCSoftwareClusterType [ECUC_SwCluC_00009]		
Parent Container	SwCluCDefinition		
Description	The type of the Software Cluster **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	APPLICATIVE_SW_CLUSTER	This Software Cluster is an Applicative Software Cluster. **Tags:** atp.Status=draft	
	HOST_SW_CLUSTER	This Software Cluster is a Host Software Cluster. **Tags:** atp.Status=draft	
	SUBSTITUTION_SW_CLUSTER	This Software Cluster is a Substitution Software Cluster. **Tags:** atp.Status=draft	
Default Value	APPLICATIVE_SW_CLUSTER		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCSoftwareClusterRef [ECUC_SwCluC_00034]		
Parent Container	SwCluCDefinition		
Description	Reference to the CpSoftwareCluster to define the link to the Software Cluster description in the System. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Foreign reference to CP-SOFTWARE-CLUSTER		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

10.2.4 Software Cluster Base Configuration Check

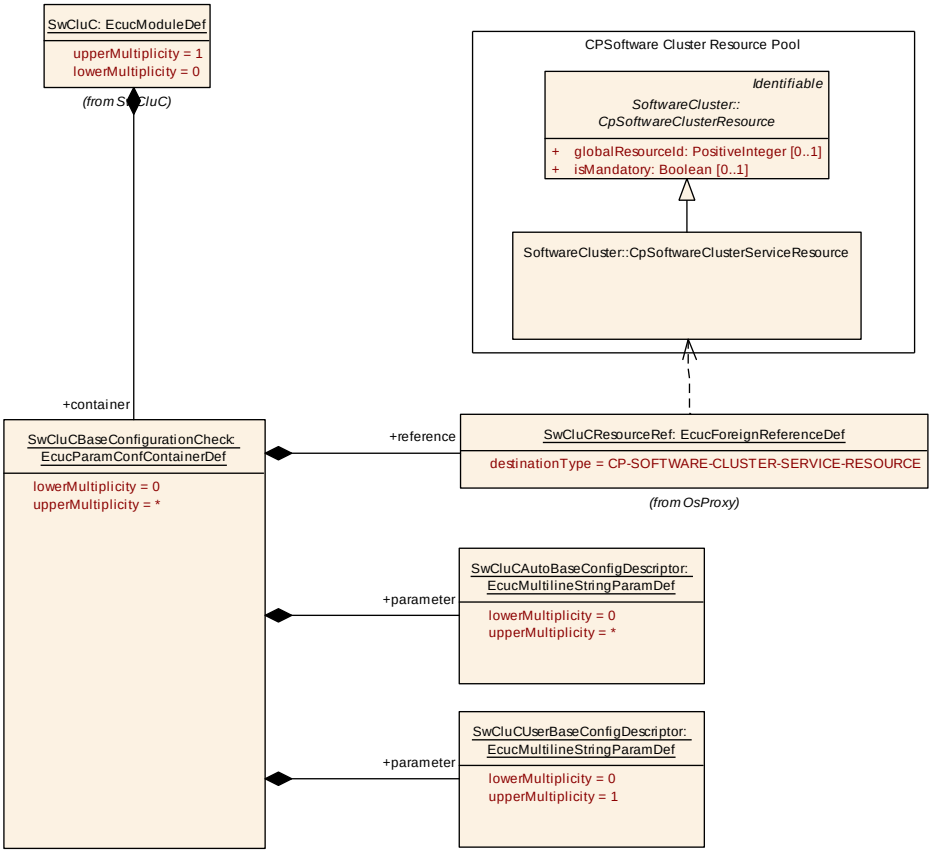

Figure 10.4: Software Cluster Base Configuration Check

SWS Item	[ECUC_SwCluC_00079]		
Container Name	SwCluCBaseConfigurationCheck		
Parent Container	SwCluC		
Description	Configuration of the base configuration check of Software Cluster Connection. This configuration places a mandatory entry in the Binary Manifest where the guard value is used to proof if the general setup of the Applicative Software Cluster is compatible to the general setup of the Host Software Cluster. For each Applicative Software Cluster an individual base configuration check should be defined. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCAutoBaseConfigDescriptor [ECUC_SwCluC_00080]		
Parent Container	SwCluCBaseConfigurationCheck		
Description	A machine determined string which represent the basic configuration assumption. Multiple SwCluCAutoBaseConfigDescriptor values can be added to support that different tools contribute to the checksum. For instance Software Cluster Connection Tool, Compiler, Link, or Locate tooling. isAuto shall be set to true since tool chain adds its collected values automatically. **Tags:** atp.Status=draft		
Multiplicity	0..*		
Type	EcucMultilineStringParamDef		
Default Value			
Regular Expression			
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCUserBaseConfigDescriptor [ECUC_SwCluC_00081]		
Parent Container	SwCluCBaseConfigurationCheck		
Description	A user determined string which represent the basic configuration assumption. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	EcucMultilineStringParamDef		
Default Value			
Regular Expression			
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCResourceRef [ECUC_SwCluC_00093]
Parent Container	SwCluCBaseConfigurationCheck

Description	Reference to the CpSoftwareClusterServiceResource. **Tags:** atp.Status=draft
Multiplicity	1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-SERVICE-RESOURCE
Scope / Dependency	scope: ECU

No Included Containers

[SWS_SwCluC_CONSTR_00078]{DRAFT} **SwCluCBaseConfigurationCheck relates only to a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_BASE_CNF** ⌈The SwCluCBaseConfigurationCheck.SwCluCResourceRef shall only reference a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_BASE_CNF.⌋*()*

10.2.5 Binary Manifest

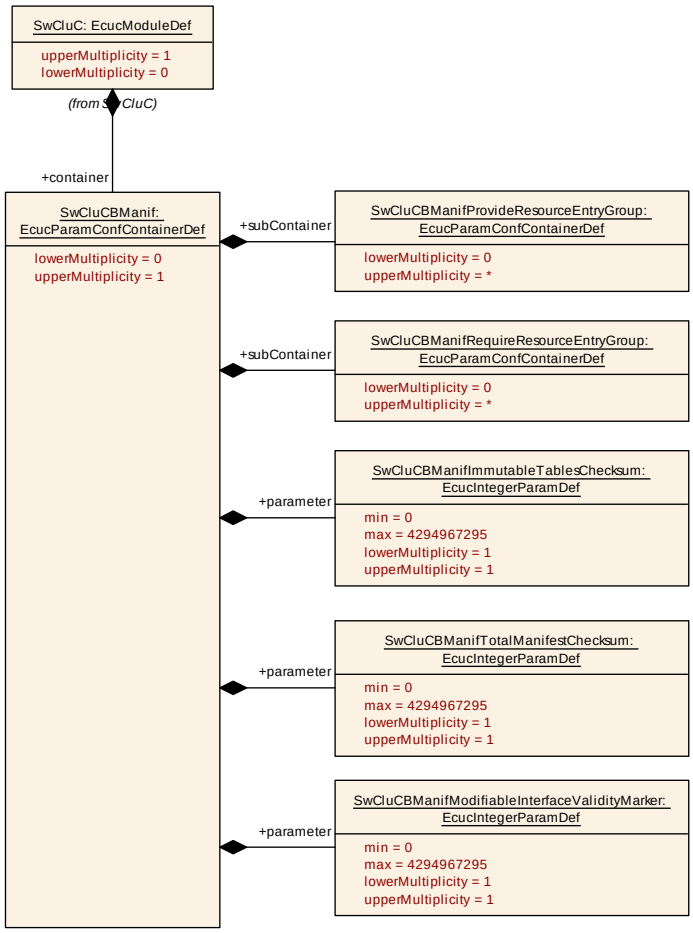

Figure 10.5: SwCluC Binary Manifest

SWS Item	[ECUC_SwCluC_00003]		
Container Name	SwCluCBManif		
Parent Container	SwCluC		
Description	Configuration of the Binary Manifest of the Software Cluster Connection **Tags:** atp.Status=draft		
Configuration Parameters			

Name	SwCluCBManifImmutableTablesChecksum [ECUC_SwCluC_00020]		
Parent Container	SwCluCBManif		
Description	This parameter defines the initialization value of the Immutable Tables Checksum. If the Binary Manifest code generator already calculates the checksum, the value can be stored here (withAuto == true). **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default Value			
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCBManifModifiableInterfaceValidityMarker [ECUC_SwCluC_00022]		
Parent Container	SwCluCBManif		
Description	This parameter defines the initialization value of the Modifiable Interface Validity Marker. The init value should be set to the erased value of the flash memory. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default Value			
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCBManifTotalManifestChecksum [ECUC_SwCluC_00021]		
Parent Container	SwCluCBManif		
Description	This parameter defines the initialization value of the Total Manifest Checksum. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default Value			
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
SwCluCBManifProvide ResourceEntryGroup	0..*	Describes a group of provided resources in the Binary Manifest. The belonging handles are put into the Binary Manifest's tables into an consecutive order which supports an array based access. **Tags:** atp.Status=draft
SwCluCBManifRequire ResourceEntryGroup	0..*	Describes a group of required resources in the Binary Manifest. The belonging handles are put into the Binary Manifest's tables into an consecutive order which supports an array based access. **Tags:** atp.Status=draft
SwCluCBManifResource Type	0..*	This container defines the structure of a resource type in the Binary Manifest. **Tags:** atp.Status=draft

10.2.5.1 Provide Resource Entry Group

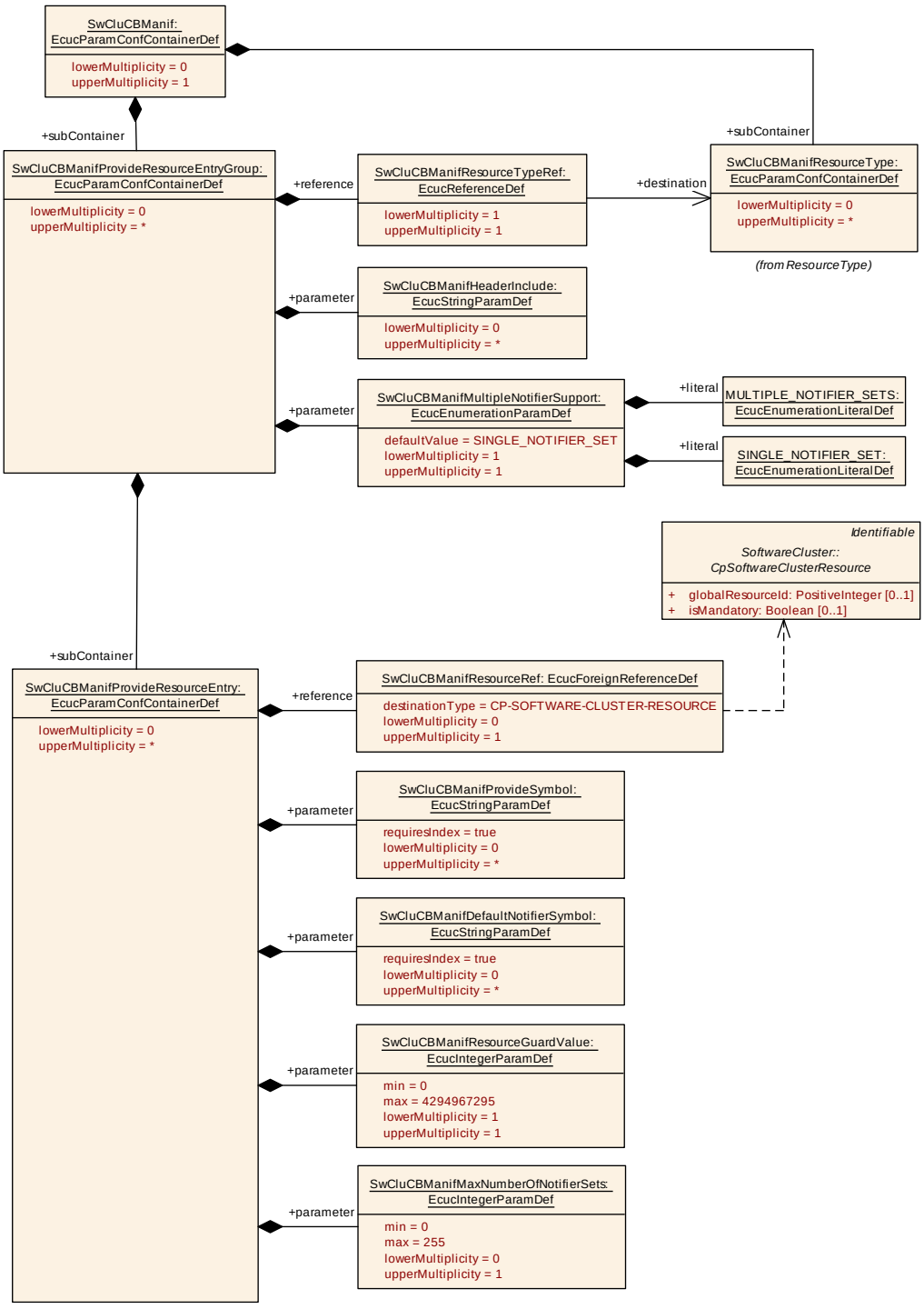

Figure 10.6: SwCluC Provide Resource Entry Group

SWS Item	[ECUC_SwCluC_00018]
Container Name	SwCluCBManifProvideResourceEntryGroup
Parent Container	SwCluCBManif

Description	Describes a group of provided resources in the Binary Manifest. The belonging handles are put into the Binary Manifest's tables into an consecutive order which supports an array based access. **Tags:** atp.Status=draft
Configuration Parameters	

Name	SwCluCBManifHeaderInclude [ECUC_SwCluC_00024]		
Parent Container	SwCluCBManifProvideResourceEntryGroup		
Description	Defines the header file(s) where the owner of the SwCluCBManifProvideResourceEntryGroup / SwCluCBManifRequireResourceEntryGroup has the declarations of the symbols for the handle initialization. **Tags:** atp.Status=draft		
Multiplicity	0..*		
Type	EcucStringParamDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	SwCluCBManifMultipleNotifierSupport [ECUC_SwCluC_00016]	
Parent Container	SwCluCBManifProvideResourceEntryGroup	
Description	SwCluCBManifMultipleNotifierSupport defines whether multiple Software Cluster can register on the notifier handles. For each possible connection a set of notifier handles is allocated. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	MULTIPLE_NOTIFIER_SETS	The provided resource supports the co-resident notifier connections to multiple required resources. **Tags:** atp.Status=draft

	SINGLE_NOTIFIER_SET	The provided resource supports at most one connection of notifiers to a required resource.	
Default Value	SINGLE_NOTIFIER_SET		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	SwCluCBManifResourceTypeRef [ECUC_SwCluC_00023]
Parent Container	SwCluCBManifProvideResourceEntryGroup
Description	Defines the resource type for all SwCluCBManifProvideResourceEntry / SwCluCBManifRequireResourceEntry **Tags:** atp.Status=draft
Multiplicity	1
Type	Reference to SwCluCBManifResourceType
Scope / Dependency	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
SwCluCBManifProvide ResourceEntry	0..*	Describes a single entry for a provided resource in the Binary Manifest. **Tags:** atp.Status=draft

10.2.5.2 Provided Resource Entry

SWS Item	[ECUC_SwclC_0009]
Container Name	SwCluCBManifProvideResourceEntry
Parent Container	SwCluCBManifProvideResourceEntryGroup
Description	Describes a single entry for a provided resource in the Binary Manifest. **Tags:** atp.Status=draft
Configuration Parameters	

Name	SwCluCBManifDefaultNotifierSymbol [ECUC_SwCluC_00027]
Parent Container	SwCluCBManifProvideResourceEntry
Description	SwCluCBManifDefaultNotifierSymbol set the default value of a Notifier Handle put into the Modifiable Interface. The number and order of SwCluCBManifDefaultNotifierSymbol needs to match the number and order of SwCluCBManifNotifierHandle in the referenced SwCluCBManifResourceType. The parameter is defined as string in order to support the usage of symbols as initialize. Values have to be set as string in a C supported number format, e.g. 0xC001CAFE. In case the provide handle holds a pointer the address operator is part of the string, e.g. &myExampleFunction **Tags:** atp.Status=draft **Attributes:** requiresIndex=true
Multiplicity	0..*
Type	EcucStringParamDef
Default Value	
Regular Expression	
Scope / Dependency	

Name	SwCluCBManifMaxNumberOfNotifierSets [ECUC_SwCluC_00029]		
Parent Container	SwCluCBManifProvideResourceEntry		
Description	SwCluCBManifMaxNumberOfNotifierSets defines the maximum number of possible required resources using notifier handles. It is required in case SwCluCBManifMultipleNotifierSupport is set to MULTIPLE_NOTIFIER_SETS. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

Name	SwCluCBManifProvideSymbol [ECUC_SwCluC_00026]
Parent Container	SwCluCBManifProvideResourceEntry
Description	SwCluCBManifProvideSymbol set the value of a Provide Handle put into the Immutable Interface. The number and order of SwCluCBManifProvideSymbols needs to match the number and order of SwCluCBManifProvideHandle in the referenced SwCluCBManifResourceType. The parameter is defined as string in order to support the usage of symbols as initialize. Values have to be set as string in a C supported number format, e.g. 0xC001CAFE. In case the provide handle holds a pointer the address operator is part of the string, e.g. &myExampleFunction **Tags:** atp.Status=draft **Attributes:** requiresIndex=true
Multiplicity	0..*
Type	EcucStringParamDef
Default Value	
Regular Expression	
Scope / Dependency	

Name	SwCluCBManifResourceGuardValue [ECUC_SwCluC_00028]	
Parent Container	SwCluCBManifProvideResourceEntry	
Description	SwCluCBManifResourceGuardValue provides guarding information checked by the Software Cluster connector. A provided resources and required resources of Software Clusters can only be connected, if the resource guarding values of provider and requester are equal. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default Value		
Scope / Dependency		

Name	SwCluCBManifResourceRef [ECUC_SwCluC_00025]
Parent Container	SwCluCBManifProvideResourceEntry
Description	Reference to the CpSoftwareClusterResource determining the global resource Id. If the reference is not set, the global resource Id in the Binary Manifest will be set to 0 for this resource entry in order to indicate an invalid resource. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-RESOURCE
Scope / Dependency	

No Included Containers

10.2.5.3 Require Resource Entry Group

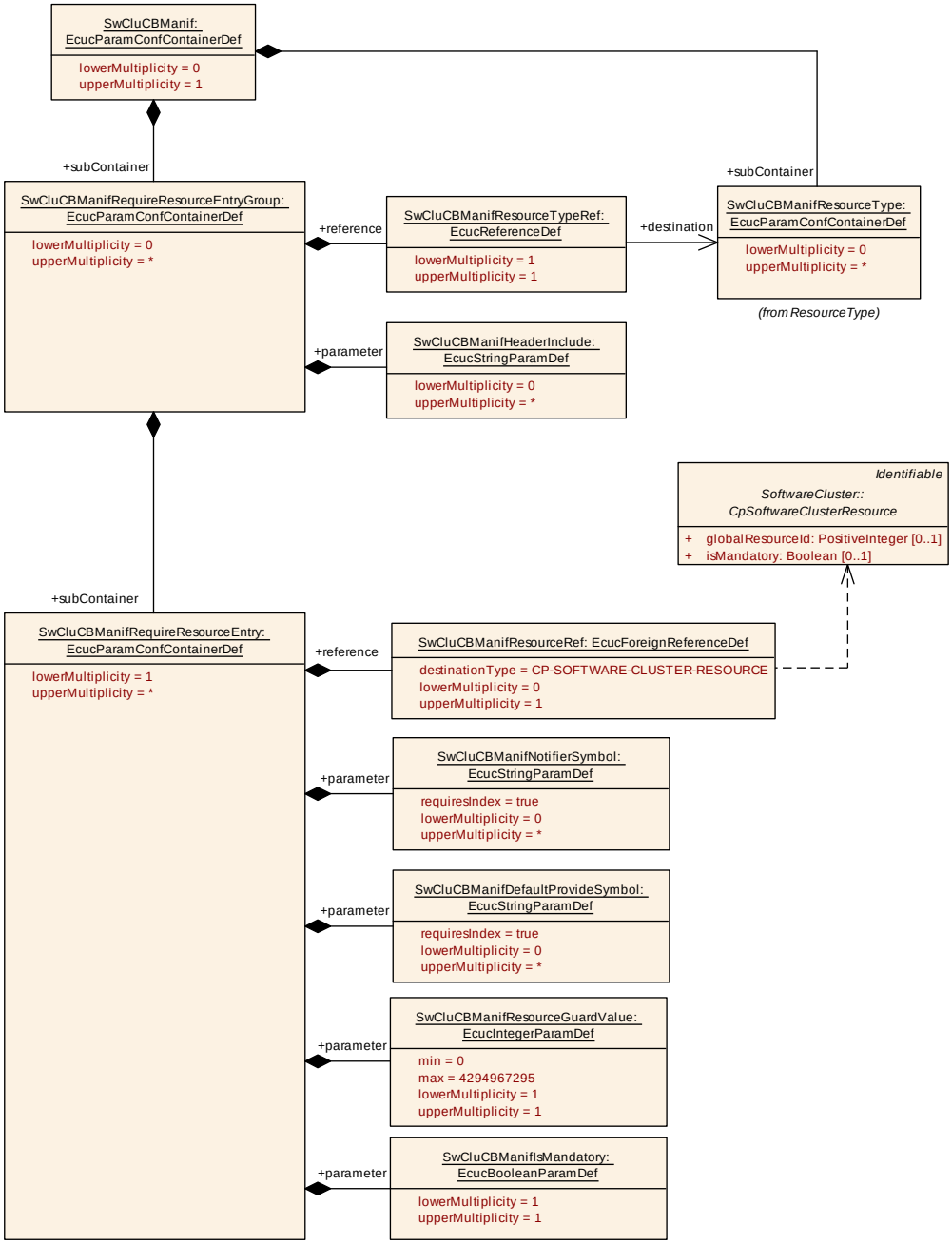

Figure 10.7: SwCluC Require Resource Entry Group

SWS Item	[ECUC_SwCluC_00019]
Container Name	SwCluCBManifRequireResourceEntryGroup
Parent Container	SwCluCBManif

Description	Describes a group of required resources in the Binary Manifest. The belonging handles are put into the Binary Manifest's tables into an consecutive order which supports an array based access. **Tags:** atp.Status=draft
Configuration Parameters	

Name	SwCluCBManifHeaderInclude [ECUC_SwCluC_00024]		
Parent Container	SwCluCBManifRequireResourceEntryGroup		
Description	Defines the header file(s) where the owner of the SwCluCBManifProvideResourceEntryGroup / SwCluCBManifRequireResourceEntryGroup has the declarations of the symbols for the handle initialization. **Tags:** atp.Status=draft		
Multiplicity	0..*		
Type	EcucStringParamDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	SwCluCBManifResourceTypeRef [ECUC_SwCluC_00023]
Parent Container	SwCluCBManifRequireResourceEntryGroup
Description	Defines the resource type for all SwCluCBManifProvideResourceEntry / SwCluCBManifRequireResourceEntry **Tags:** atp.Status=draft
Multiplicity	1
Type	Reference to SwCluCBManifResourceType
Scope / Dependency	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
SwCluCBManifRequireResourceEntry	1..*	Describes a single entry for a required resource in the Binary Manifest. **Tags:** atp.Status=draft

10.2.5.4 Require Resource Entry

SWS Item	[ECUC_SwclC_00021]
Container Name	SwCluCBManifRequireResourceEntry
Parent Container	SwCluCBManifRequireResourceEntryGroup
Description	Describes a single entry for a required resource in the Binary Manifest. **Tags:** atp.Status=draft
Configuration Parameters	

Name	SwCluCBManifDefaultProvideSymbol [ECUC_SwCluC_00031]
Parent Container	SwCluCBManifRequireResourceEntry
Description	SwCluCBManifDefaultProvideSymbol set the default value of a Provide Handle put into the Modifiable Interface. The number and order of SwCluCBManifDefaultProvideSymbols needs to match the number and order of SwclCBManifProvideHandles in the referenced SwCluCBManifResourceType. The parameter is defined as string in order to support the usage of symbols as initialize. Values have to be set as string in a C supported number format, e.g. 0xC001CAFE. In case the provide handle holds a pointer the address operator is part of the string, e.g. &myExampleFunction **Tags:** atp.Status=draft **Attributes:** requiresIndex=true
Multiplicity	0..*
Type	EcucStringParamDef
Default Value	
Regular Expression	
Scope / Dependency	

Name	SwCluCBManifIsMandatory [ECUC_SwCluC_00032]
Parent Container	SwCluCBManifRequireResourceEntry
Description	SwCluCBManifIsMandatoryindicates, that the resource is mandatory to operate this Software Cluster. If the resource is not provided on the machine the connection process for this Software Cluster requiring this resource gets aborted. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucBooleanParamDef
Default Value	
Scope / Dependency	

Name	SwCluCBManifNotifierSymbol [ECUC_SwCluC_00030]
Parent Container	SwCluCBManifRequireResourceEntry
Description	SwCluCBManifNotifierSymbol set the value of a Notifier Handle put into the Immutable Interface. The number and order of SwCluCBManifNotifierSymbol needs to match the number and order of SwclCBManifNotifierHandles in the referenced SwCluCBManifResourceType. The parameter is defined as string in order to support the usage of symbols as initialize. Values have to be set as string in a C supported number format, e.g. 0xC001CAFE. In case the provide handle holds a pointer the address operator is part of the string, e.g. &myExampleFunction **Tags:** atp.Status=draft **Attributes:** requiresIndex=true
Multiplicity	0..*
Type	EcucStringParamDef
Default Value	
Regular Expression	
Scope / Dependency	

Name	SwCluCBManifResourceGuardValue [ECUC_SwCluC_00028]
Parent Container	SwCluCBManifRequireResourceEntry
Description	SwCluCBManifResourceGuardValue provides guarding information checked by the Software Cluster connector. A provided resources and required resources of Software Clusters can only be connected, if the resource guarding values of provider and requester are equal. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 4294967295
Default Value	

Scope / Dependency	

Name	SwCluCBManifResourceRef [ECUC_SwCluC_00025]
Parent Container	SwCluCBManifRequireResourceEntry
Description	Reference to the CpSoftwareClusterResource determining the global resource Id. If the reference is not set, the global resource Id in the Binary Manifest will be set to 0 for this resource entry in order to indicate an invalid resource. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-RESOURCE
Scope / Dependency	

No Included Containers

10.2.5.5 Resource Type

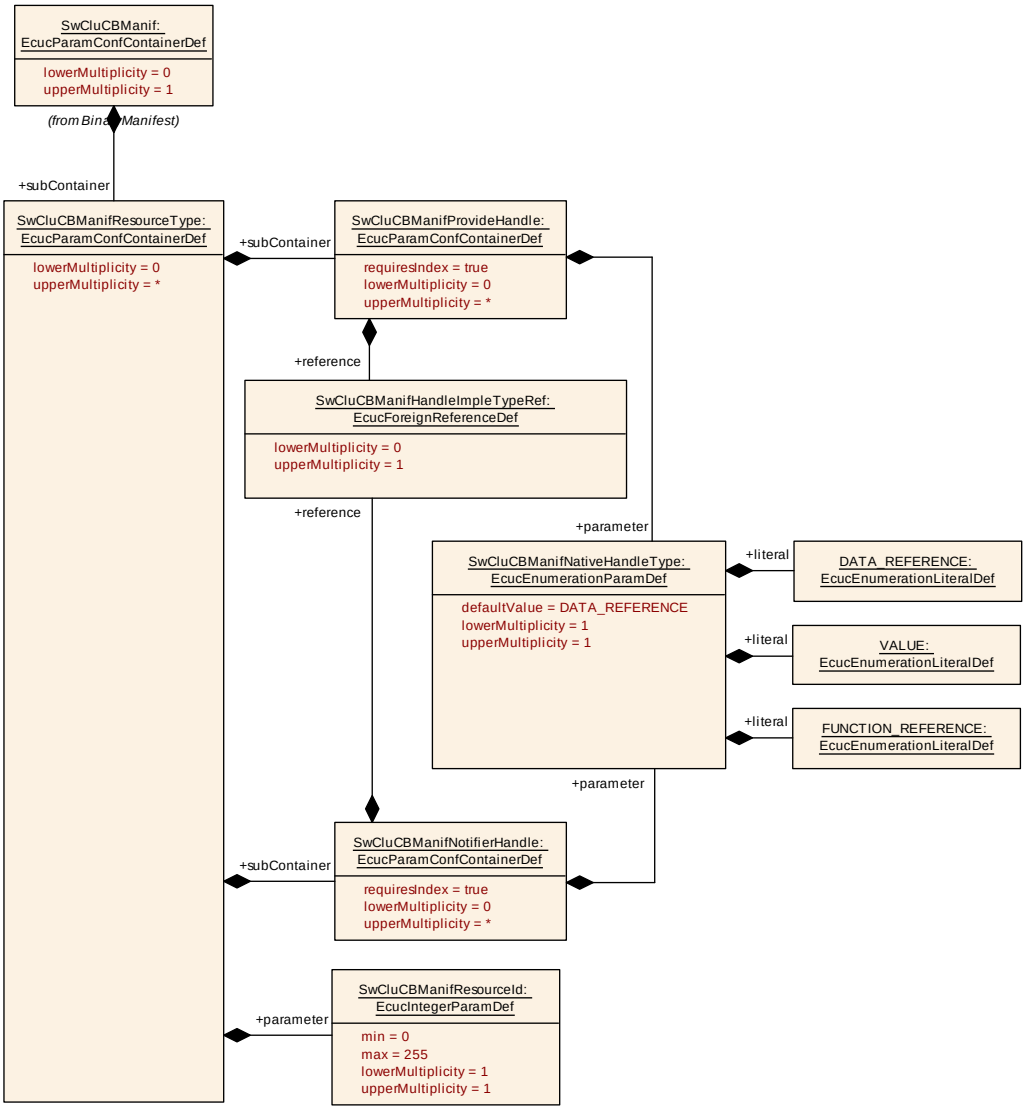

Figure 10.8: SwCluC resource type

SWS Item	[ECUC_SwCluC_00012]		
Container Name	SwCluCBManifResourceType		
Parent Container	SwCluCBManif		
Description	This container defines the structure of a resource type in the Binary Manifest. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCBManifResourceId [ECUC_SwCluC_00017]		
Parent Container	SwCluCBManifResourceType		
Description	Unique number of the resource type. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
SwCluCBManifNotifierHandle	0..*	Defines a Notifier Handle and its properties for this resource type. **Tags:** atp.Status=draft
SwCluCBManifProvideHandle	0..*	Defines a Provide Handle and its properties for this resource type. **Tags:** atp.Status=draft

SWS Item	[ECUC_SwCluC_00013]
Container Name	SwCluCBManifProvideHandle
Parent Container	SwCluCBManifResourceType
Description	Defines a Provide Handle and its properties for this resource type. **Tags:** atp.Status=draft **Attributes:** requiresIndex=true
Configuration Parameters	

Name	SwCluCBManifNativeHandleType [ECUC_SwCluC_00033]		
Parent Container	SwCluCBManifProvideHandle		
Description	This paramter determines the unterlying native type of the handle which can be • pointer to variable • value • pointer to function This information is required to correctly initialize the handle in the Immutable Interface or Modifiable Interface. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DATA_REFERENCE	Handle is pointer to variable	
	FUNCTION_REFERENCE	Handle is pointer to function	
	VALUE	Handle is a value **Tags:** atp.Status=draft	
Default Value	DATA_REFERENCE		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	SwCluCBManifHandleImpleTypeRef [ECUC_SwCluC_00015]		
Parent Container	SwCluCBManifProvideHandle		
Description	Reference to the ImplementationDataType of the handle. This type can be additionally configured to get a correct casting of the native handle type to the returned handel type. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Foreign reference to		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Scope / Dependency	scope: local

No Included Containers

SWS Item	[ECUC_SwCluC_00014]
Container Name	SwCluCBManifNotifierHandle
Parent Container	SwCluCBManifResourceType
Description	Defines a Notifier Handle and its properties for this resource type. **Tags:** atp.Status=draft **Attributes:** requiresIndex=true
Configuration Parameters	

Name	SwCluCBManifNativeHandleType [ECUC_SwCluC_00033]		
Parent Container	SwCluCBManifNotifierHandle		
Description	This paramter determines the unterlying native type of the handle which can be • pointer to variable • value • pointer to function This information is required to correctly initialize the handle in the Immutable Interface or Modifiable Interface. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DATA_REFERENCE	Handle is pointer to variable	
	FUNCTION_REFERENCE	Handle is pointer to function	
	VALUE	Handle is a value **Tags:** atp.Status=draft	
Default Value	DATA_REFERENCE		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	SwCluCBManifHandleImpleTypeRef [ECUC_SwCluC_00015]		
Parent Container	SwCluCBManifNotifierHandle		
Description	Reference to the ImplementationDataType of the handle. This type can be additionally configured to get a correct casting of the native handle type to the returned handel type. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Foreign reference to		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.2.6 Cross Cluster Communication

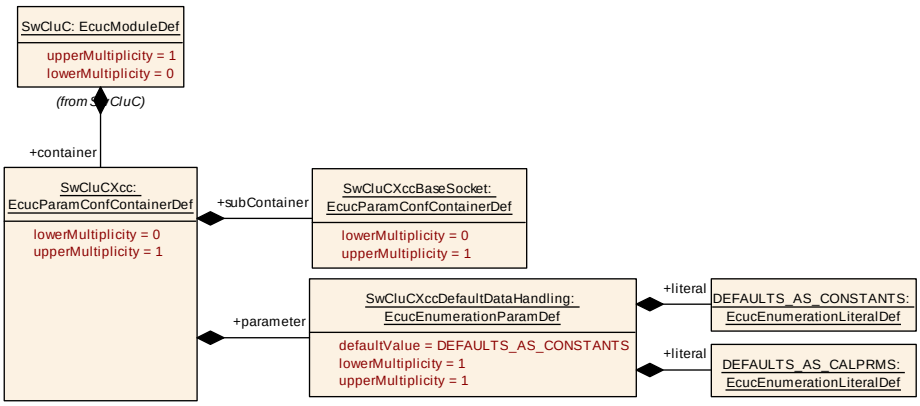

Figure 10.9: General Cross Cluster Communication Parameters

SWS Item	[ECUC_SwCluC_00075]
Container Name	SwCluCXcc
Parent Container	SwCluC
Description	Configuration of the Binary Manifest of the Software Cluster Connection **Tags:** atp.Status=draft
Configuration Parameters	

Name	SwCluCXccDefaultDataHandling [ECUC_SwCluC_00084]		
Parent Container	SwCluCXcc		
Description	SwCluCXccDefaultDataHandling defines whether the default data for unconnected RPortPrototypes of a Software Cluster are instantiated as fixed constants or as calibration parameter. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DEFAULTS_AS_CALPRMS	Default data are implemented as calibration parameter. **Tags:** atp.Status=draft	
	DEFAULTS_AS_CONSTANTS	Default data are implemented as fixed constants. **Tags:** atp.Status=draft	
Default Value	DEFAULTS_AS_CONSTANTS		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
SwCluCXccBaseSocket	0..1	Maps a XccBaseSocket to the CpSoftwareClusterServiceResource describing the XccBaseSocket resource in the clustered system. Maps a XccBaseSocket to a specific EcucPartition on which the XccBaseSocket is working. **Tags:** atp.Status=draft

10.2.6.1 Cross Cluster Communication Base Socket

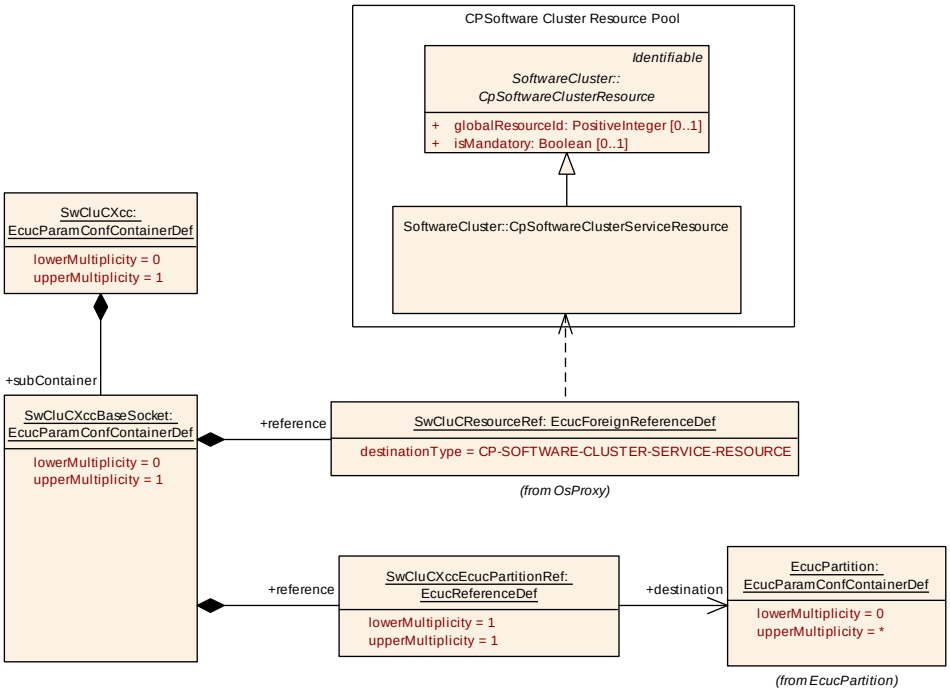

Figure 10.10: Cross Cluster Communication Base Socket

SWS Item	[ECUC_SwCluC_00076]
Container Name	SwCluCXccBaseSocket
Parent Container	SwCluCXcc
Description	Maps a XccBaseSocket to the CpSoftwareClusterServiceResource describing the XccBaseSocket resource in the clustered system. Maps a XccBaseSocket to a specific EcucPartition on which the XccBaseSocket is working. **Tags:** atp.Status=draft
Configuration Parameters	

Name	SwCluCResourceRef [ECUC_SwCluC_00096]
Parent Container	SwCluCXccBaseSocket
Description	Reference to the CpSoftwareClusterServiceResource. **Tags:** atp.Status=draft
Multiplicity	1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-SERVICE-RESOURCE
Scope / Dependency	scope: ECU

Name	SwCluCXccEcucPartitionRef [ECUC_SwCluC_00077]
Parent Container	SwCluCXccBaseSocket

Description	Reference to the EcucPartition on which the XccBaseSocket is available. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Reference to EcucPartition		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

[SWS_SwCluC_CONSTR_03063]{DRAFT} **SwCluCXccBaseSocket relates only to a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_XCC_BASE_SOCKET** ⌈The SwCluCXccBaseSocket.SwCluCResourceRef shall only reference a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_XCC_BASE_SOCKET.⌋*()*

10.2.7 Proxy Modules

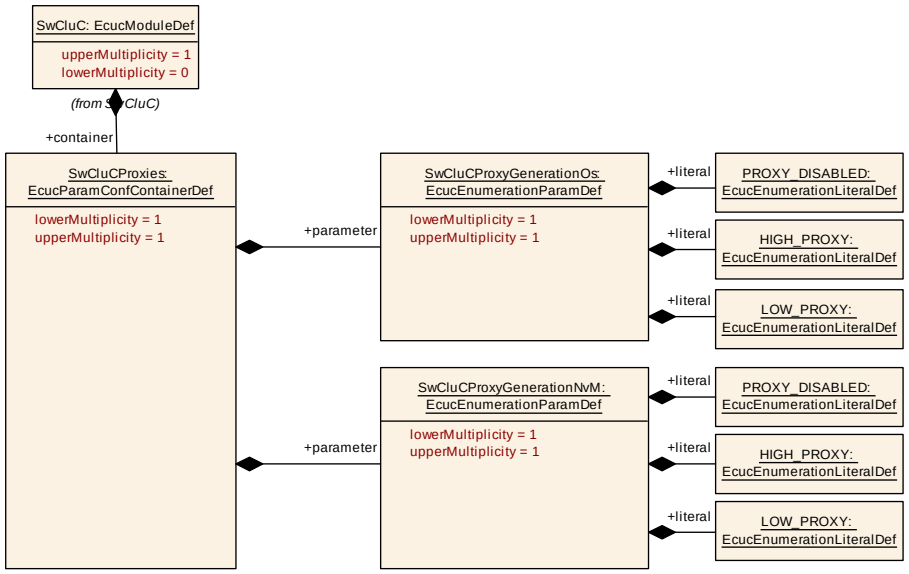

Figure 10.11: General Proxies Parameters

SWS Item	[ECUC_SwCluC_00036]
Container Name	SwCluCProxies

Parent Container	SwCluC		
Description	General configuration of the Proxy Modules of Software Cluster Connection **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCProxyGenerationNvM [ECUC_SwCluC_00038]		
Parent Container	SwCluCProxies		
Description	Defines whether the NvM Proxy code and models are generated. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	HIGH_PROXY	Enables the according High Proxy Module code and AUTOSAR model generation.	
	LOW_PROXY	Enables the according Low Proxy Module code and AUTOSAR model generation.	
	PROXY_DISABLED	Disables the Proxy Module code and AUTOSAR model generation.	
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCProxyGenerationOs [ECUC_SwCluC_00037]	
Parent Container	SwCluCProxies	
Description	Defines whether the Os Proxy code and models are generated. **Tags:** atp.Status=draft	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	HIGH_PROXY	Enables the according High Proxy Module code and AUTOSAR model generation.

	LOW_PROXY		Enables the according Low Proxy Module code and AUTOSAR model generation.
	PROXY_DISABLED		Disables the Proxy Module code and AUTOSAR model generation
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
SwCluCNvMProxy	1	NvM Proxy specific parameters. **Tags:** atp.Status=draft
SwCluCOsProxy	1	Os Proxy specific parameters. **Tags:** atp.Status=draft

10.2.7.1 NvM Proxy

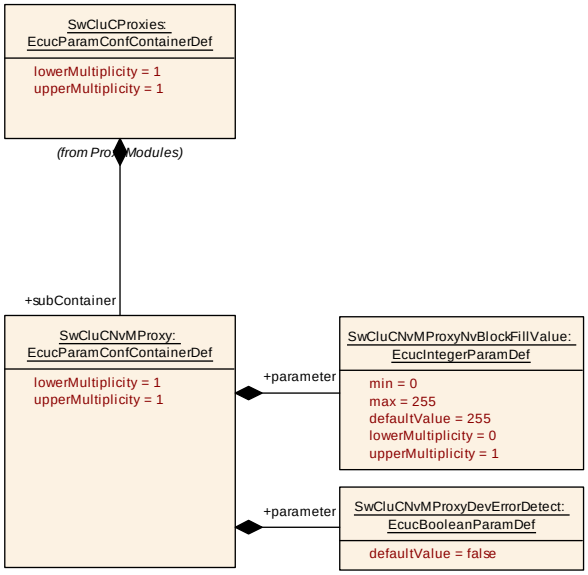

Figure 10.12: NvM Proxy Parameter

SWS Item	[ECUC_SwCluC_00039]

Container Name	SwCluCNvMProxy		
Parent Container	SwCluCProxies		
Description	NvM Proxy specific parameters. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCNvMProxyDevErrorDetect [ECUC_SwCluC_00083]		
Parent Container	SwCluCNvMProxy		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	SwCluCNvMProxyNvBlockFillValue [ECUC_SwCluC_00055]	
Parent Container	SwCluCNvMProxy	
Description	Defines the default value used to fill unused bytes in NvBlock space in case NvBlock length in Applicative Software Cluster < NvBlock length configured in NvM of Host Software Cluster. Dependent on NvM Proxy implementation this parameter is only relevant for NvM Low Proxy or NvM High Proxy **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default Value	255	
Post-Build Variant Value	false	

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
SwCluCNvMBaseSocket	1..*	This container configures how many EcucPartitions specific API links are required for the NvM Proxy and to which cluster resource the API set belongs. In the NvM Low Proxy all offered API sets needs to be configured. In the NvM High Proxy only the ones are configured which are offered inside this Applicative Software Cluster. The reference SwCluCProxyEcucPartitionRef denotes the EcucPartition on which the API set is provided. **Tags:** atp.Status=draft
SwCluCNvMProxyNvBlock	1..*	Maps a NvMBlockDescriptor to the CpSoftwareClusterServiceResource describing the NvBlock resource in the clustered system. **Tags:** atp.Status=draft

10.2.7.1.1 NvM Base Socket

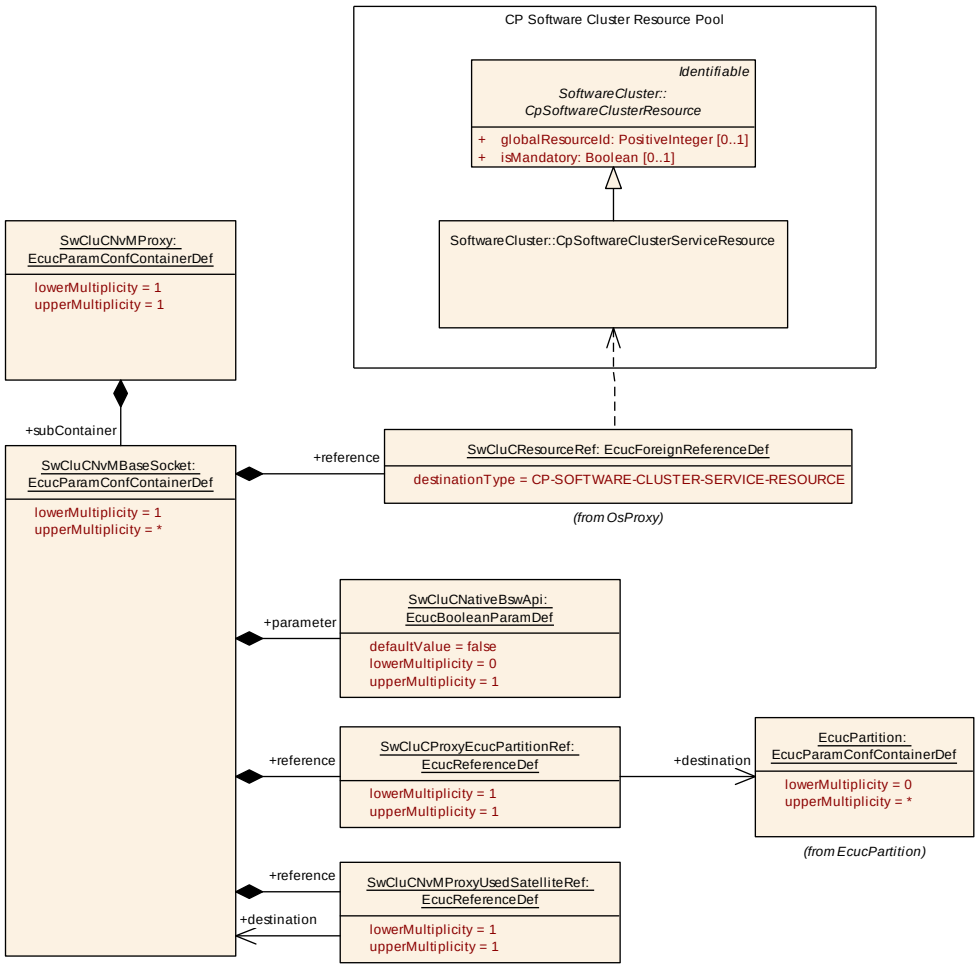

Figure 10.13: NvM Proxy Base Socket

SWS Item	[ECUC_SwCluC_00056]
Container Name	SwCluCNvMBaseSocket
Parent Container	SwCluCNvMProxy
Description	This container configures how many EcucPartitions specific API links are required for the NvM Proxy and to which cluster resource the API set belongs. In the NvM Low Proxy all offered API sets needs to be configured. In the NvM High Proxy only the ones are configured which are offered inside this Applicative Software Cluster. The reference SwCluCProxyEcucPartitionRef denotes the EcucPartition on which the API set is provided. **Tags:** atp.Status=draft

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCNativeBswApi [ECUC_SwCluC_00058]		
Parent Container	SwCluCNvMBaseSocket		
Description	Defines if the native C-API without pre or suffixes is offered in the Applicative Software Cluster on this EcucPartition. This Parameter is only relevant for the NvM High Proxy. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCNvMProxyUsedSatelliteRef [ECUC_SwCluC_00082]		
Parent Container	SwCluCNvMBaseSocket		
Description	Reference to the SwCluCNvMBaseSocket which has access to a satellite of the NvM. The owning SwCluCNvMBaseSocket uses the NvM satellite of the referenced SwCluCNvMBaseSocket. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Reference to SwCluCNvMBaseSocket		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCProxyEcucPartitionRef [ECUC_SwCluC_00059]		
Parent Container	SwCluCNvMBaseSocket		
Description	Reference to the EcucPartition. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Reference to EcucPartition		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCResourceRef [ECUC_SwCluC_00092]
Parent Container	SwCluCNvMBaseSocket
Description	Reference to the CpSoftwareClusterServiceResource. **Tags:** atp.Status=draft
Multiplicity	1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-SERVICE-RESOURCE
Scope / Dependency	scope: ECU

No Included Containers

[SWS_SwCluC_CONSTR_02130]{DRAFT} **SwCluCNvMBaseSocket relates only to a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_NVM_BASE_SOCKET** ⌈The SwCluCNvMBaseSocket.SwCluCResourceRef shall only reference a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_NVM_BASE_SOCKET.⌋*()*

10.2.7.1.2 NvM Proxy NvBlock configuration

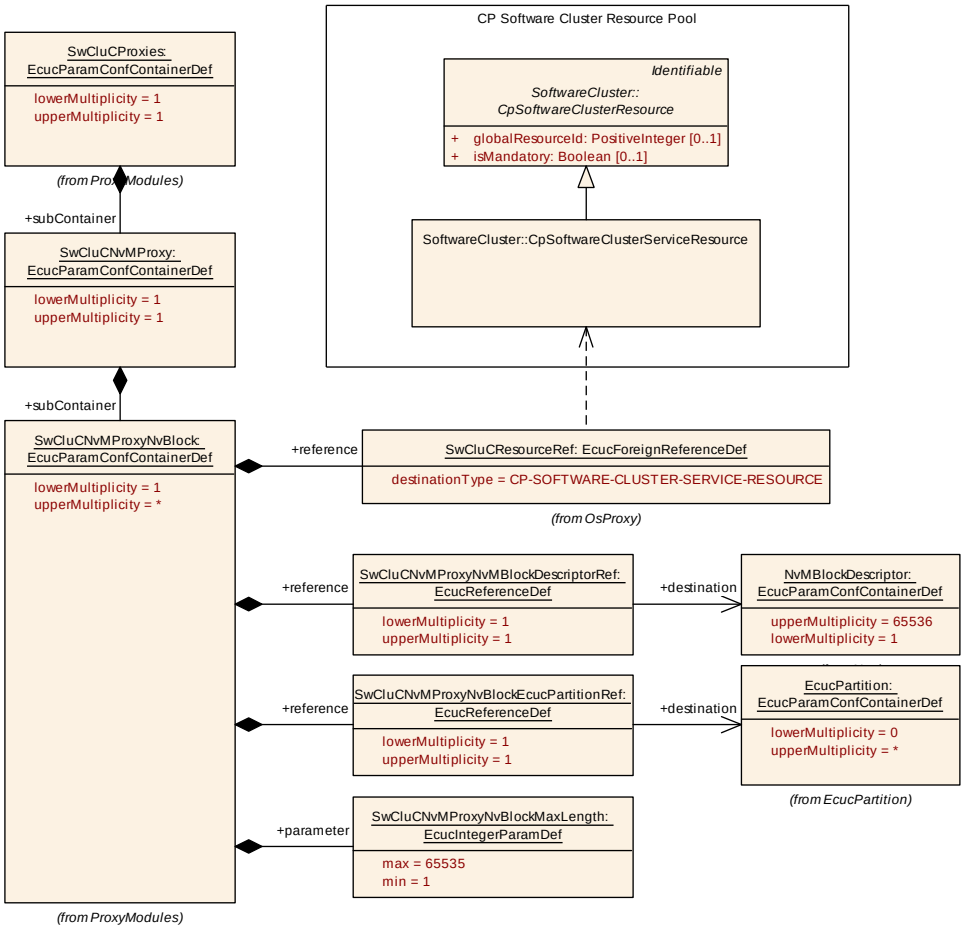

Figure 10.14: NvM Proxy NvBlock

SWS Item	[ECUC_SwCluC_00040]		
Container Name	SwCluCNvMProxyNvBlock		
Parent Container	SwCluCNvMProxy		
Description	Maps a NvMBlockDescriptor to the CpSoftwareClusterServiceResource describing the NvBlock resource in the clustered system. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCNvMProxyNvBlockMaxLength [ECUC_SwCluC_00043]		
Parent Container	SwCluCNvMProxyNvBlock		
Description	Defines the maximum NV block data length in bytes. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default Value			
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCNvMProxyNvBlockEcucPartitionRef [ECUC_SwCluC_00060]		
Parent Container	SwCluCNvMProxyNvBlock		
Description	Reference to the EcucPartition on which the NvM Proxy offers the NvBlock access (call of APIfunctions, callback functions and Ports on the Service Software Component) **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Reference to EcucPartition		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCNvMProxyNvMBlockDescriptorRef [ECUC_SwCluC_00042]
Parent Container	SwCluCNvMProxyNvBlock
Description	Reference to the NvMBlockDescriptor **Tags:** atp.Status=draft
Multiplicity	1
Type	Reference to NvMBlockDescriptor

Scope / Dependency	scope: ECU

Name	SwCluCResourceRef [ECUC_SwCluC_00091]
Parent Container	SwCluCNvMProxyNvBlock
Description	Reference to the CpSoftwareClusterServiceResource. **Tags:** atp.Status=draft
Multiplicity	1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-SERVICE-RESOURCE
Scope / Dependency	scope: ECU

No Included Containers

[SWS_SwCluC_CONSTR_02131]{DRAFT} **SwCluCNvMProxyNvBlock relates only to a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_NV_BLOCK** ⌈The SwCluCNvMProxyNvBlock.SwCluCResourceRef shall only reference a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_NV_BLOCK.⌋*()*

10.2.7.2 Os Proxy

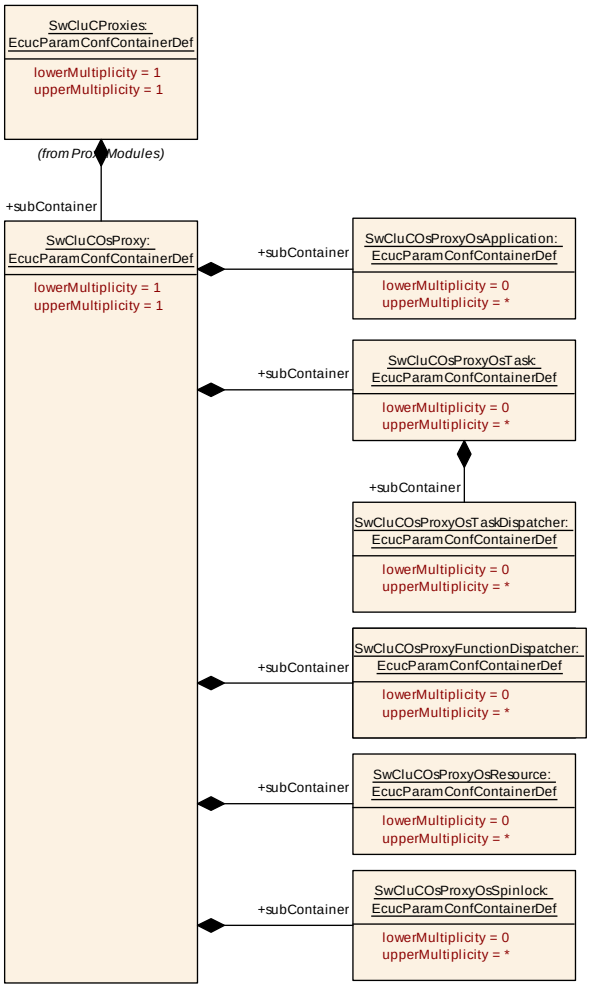

Figure 10.15: Os Proxy Parameter

SWS Item	[ECUC_SwCluC_00085]		
Container Name	SwCluCOsProxy		
Parent Container	SwCluCProxies		
Description	Os Proxy specific parameters. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
SwCluCOsProxyFunction Dispatcher	0..*	This container configures a function dispatcher. In the Os High Proxy this configuration puts a function as a Dispatch Entry Point in the Binary Manifest which can be called by the Os Low Proxy. In the Os Low Proxy this configuration provides the Dispatcher Function (and according resource Entry in the Binary Manifest) which is able to call the MaxNumberOfCallee Dispatch Entry Points of Os High Proxies. **Tags:** atp.Status=draft
SwCluCOsProxyOs Application	0..*	Maps a OsApplication to the CpSoftwareClusterServiceResource describing the OsApplication resource in the clustered system. **Tags:** atp.Status=draft
SwCluCOsProxyOsBase Socket	0..*	Maps a OsBaseSocket to the CpSoftwareClusterServiceResource describing the OsBaseSocket resource in the clustered system. **Tags:** atp.Status=draft
SwCluCOsProxyOs Resource	0..*	Maps a OsResource to the CpSoftwareClusterServiceResource describing the OsResource resource in the clustered system. **Tags:** atp.Status=draft
SwCluCOsProxyOs Spinlock	0..*	Maps a OsSpinlock to the CpSoftwareClusterServiceResource describing the OsSpinlock resource in the clustered system. **Tags:** atp.Status=draft
SwCluCOsProxyOsTask	0..*	Maps a OsTask to the CpSoftwareClusterServiceResource describing the OsTask resource in the clustered system. **Tags:** atp.Status=draft

10.2.7.2.1 Os Proxy Base Socket configuration

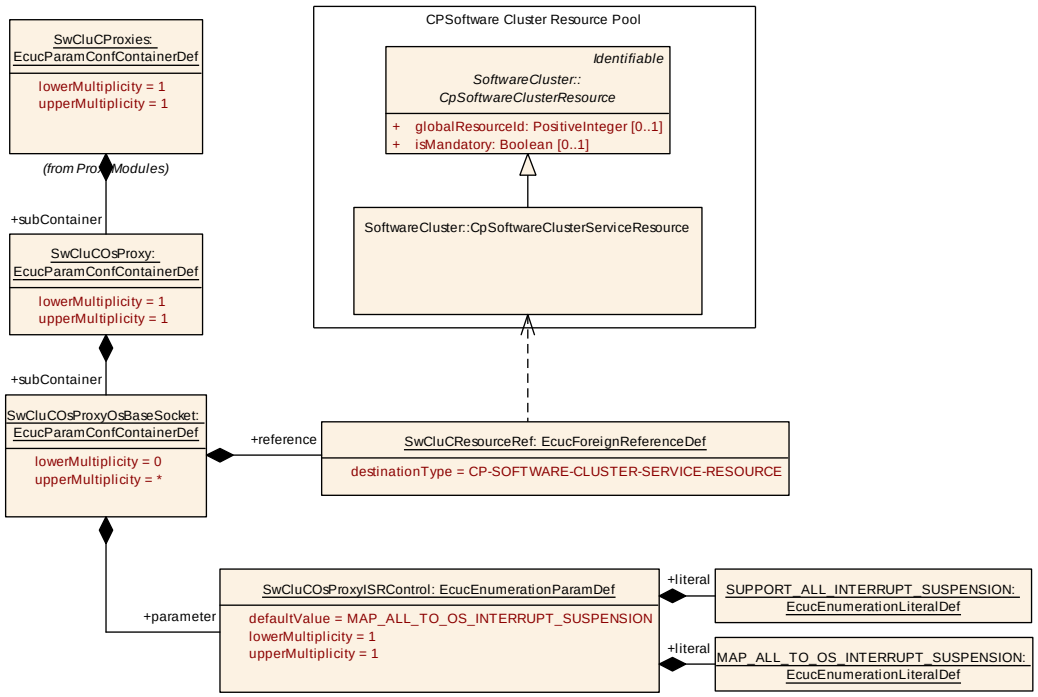

Figure 10.16: Os Proxy Base Socket Parameter

SWS Item	[ECUC_SwCluC_00053]		
Container Name	SwCluCOsProxyOsBaseSocket		
Parent Container	SwCluCOsProxy		
Description	Maps a OsBaseSocket to the CpSoftwareClusterServiceResource describing the OsBaseSocket resource in the clustered system. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCOsProxyISRControl [ECUC_SwCluC_00054]		
Parent Container	SwCluCOsProxyOsBaseSocket		
Description	Configures whether the ISR handling in the Os Base Socket maps the Suspend / ResumeAllInterrupt to Suspend / ResumeOsInterrupt **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	MAP_ALL_TO_OS_INTERRUPT_SUSPENSION	ActivateTask is NOT passed to the Host Software Cluster. **Tags:** atp.Status=draft	
	SUPPORT_ALL_INTERRUPT_SUSPENSION	ActivateTask is passed to the Host Software Cluster. **Tags:** atp.Status=draft	
Default Value	MAP_ALL_TO_OS_INTERRUPT_SUSPENSION		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCResourceRef [ECUC_SwCluC_00090]
Parent Container	SwCluCOsProxyOsBaseSocket
Description	Reference to the CpSoftwareClusterServiceResource. **Tags:** atp.Status=draft
Multiplicity	1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-SERVICE-RESOURCE
Scope / Dependency	scope: ECU

No Included Containers

[SWS_SwCluC_CONSTR_02231]{DRAFT} **SwCluCOsProxyOsBaseSocket relates only to a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_BASE_SOCKET** ⌈The SwCluCOsProxyOsBaseSocket.SwCluCResourceRef shall only reference a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_BASE_SOCKET.⌋*()*

10.2.7.2.2 Os Proxy OsApplication configuration

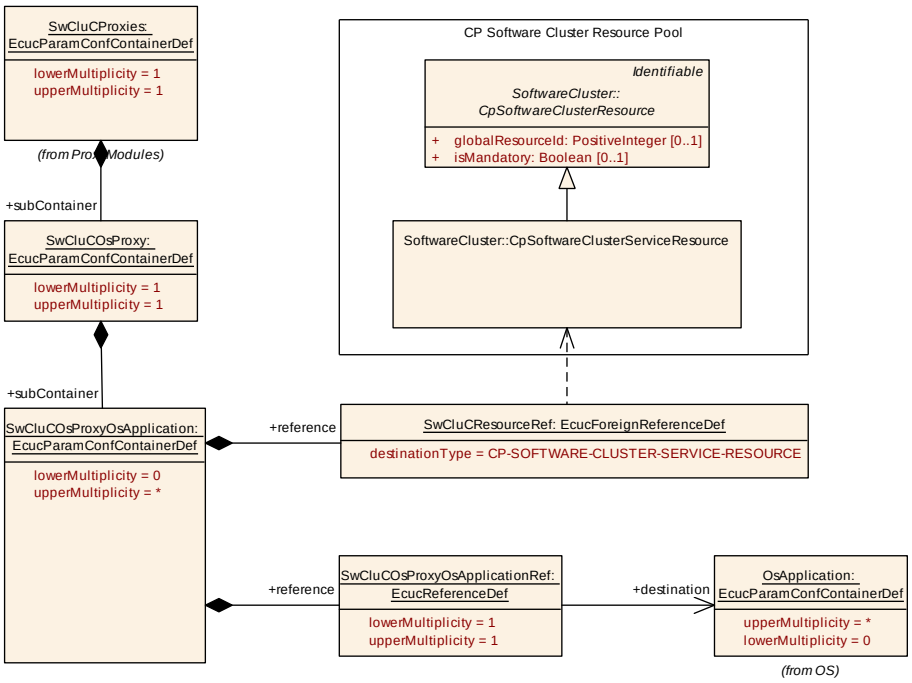

Figure 10.17: Os Proxy OsApplication Parameter

SWS Item	[ECUC_SwCluC_00046]		
Container Name	SwCluCOsProxyOsApplication		
Parent Container	SwCluCOsProxy		
Description	Maps a OsApplication to the CpSoftwareClusterServiceResource describing the OsApplication resource in the clustered system. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCOsProxyOsApplicationRef [ECUC_SwCluC_00050]
Parent Container	SwCluCOsProxyOsApplication
Description	Reference to the OsApplication **Tags:** atp.Status=draft
Multiplicity	1
Type	Reference to OsApplication
Scope / Dependency	scope: ECU

Name	SwCluCResourceRef [ECUC_SwCluC_00088]

Parent Container	SwCluCOsProxyOsApplication
Description	Reference to the CpSoftwareClusterServiceResource. **Tags:** atp.Status=draft
Multiplicity	1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-SERVICE-RESOURCE
Scope / Dependency	scope: ECU

No Included Containers

[SWS_SwCluC_CONSTR_02232]{DRAFT} **SwCluCOsProxyOsApplication relates only to a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_APPLICATION** ⌈The SwCluCOsProxyOsApplication.SwCluCResourceRef shall only reference a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_APPLICATION.⌋*()*

10.2.7.2.3 Os Proxy OsTask configuration

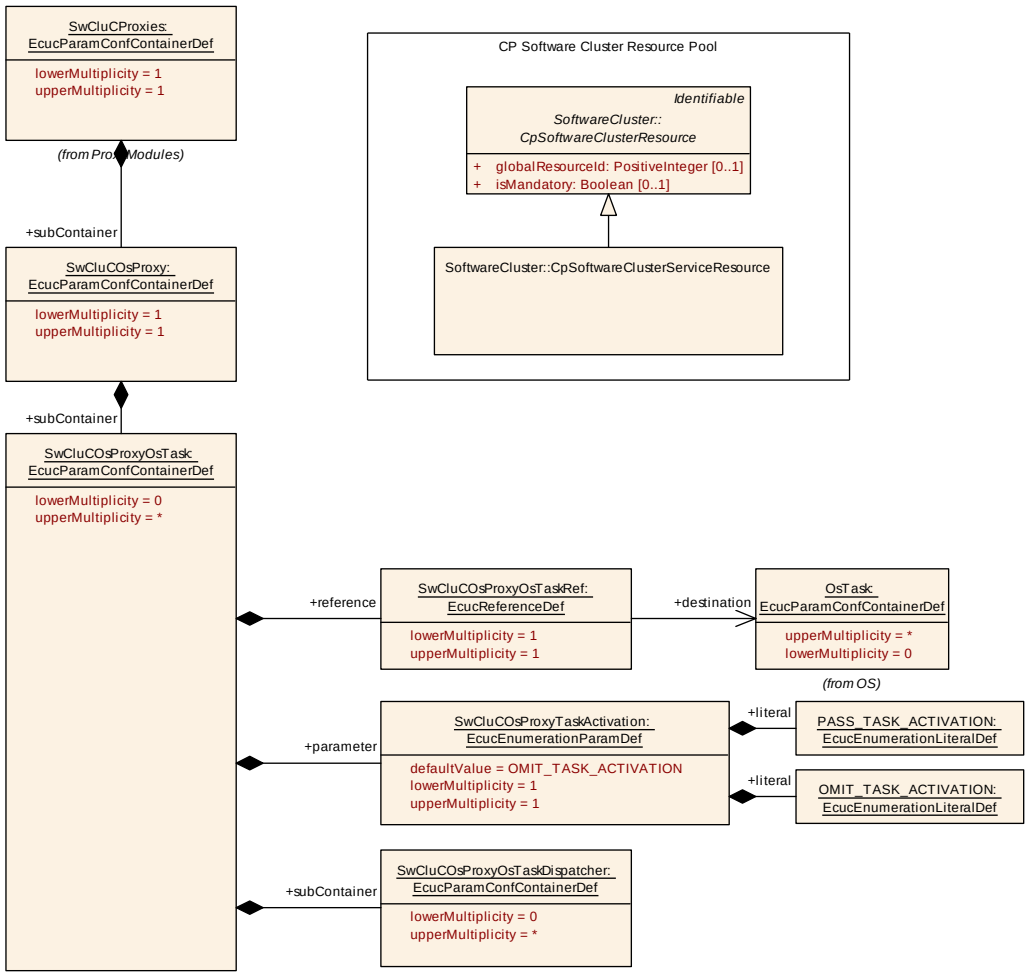

Figure 10.18: Os Proxy OsTask Parameter

SWS Item	[ECUC_SwCluC_00044]		
Container Name	SwCluCOsProxyOsTask		
Parent Container	SwCluCOsProxy		
Description	Maps a OsTask to the CpSoftwareClusterServiceResource describing the OsTask resource in the clustered system. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCOsProxyTaskActivation [ECUC_SwCluC_00048]		
Parent Container	SwCluCOsProxyOsTask		
Description	Configures for this particular OsTask whether ActivateTask is passed to the Host Software Cluster or not. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	OMIT_TASK_ACTIVATION	ActivateTask is NOT passed to the Host Software Cluster. **Tags:** atp.Status=draft	
	PASS_TASK_ACTIVATION	ActivateTask is passed to the Host Software Cluster. **Tags:** atp.Status=draft	
Default Value	OMIT_TASK_ACTIVATION		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCOsProxyOsTaskRef [ECUC_SwCluC_00045]
Parent Container	SwCluCOsProxyOsTask
Description	Reference to the OsTask **Tags:** atp.Status=draft
Multiplicity	1
Type	Reference to OsTask
Scope / Dependency	scope: ECU

Name	SwCluCResourceRef [ECUC_SwCluC_00094]
Parent Container	SwCluCOsProxyOsTask
Description	Reference to the CpSoftwareClusterServiceResource. **Tags:** atp.Status=draft
Multiplicity	1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-SERVICE-RESOURCE
Scope / Dependency	scope: ECU

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
SwCluCOsProxyOsTask Dispatcher	0..*	This container defines a task dispatcher for the owning proxy task. In the Os High Proxy this configuration provides the OS Task body function as a Dispatch Entry Point in the Binary manifest which can be called by the Os Low Proxy. In the HIGH_PROXY configuration at most one SwCluCOsProxyOsTaskDispatcher can be owned by a SwCluCOsProxyOsTask In the Os High Proxy this configuration provides the Dispatcher Runnable (and according resource Entry in the Binary Manifest) which is able to call the MaxNumberOfCallee Dispatch Entry Points of Os High Proxies. In the LOW_PROXY configuration multiple SwCluCOsProxyOsTaskDispatcher can be owned by a SwCluCOsProxyOsTask. If none of the SwCluCOsProxyOsTaskDispatcher<xxx>Event container is configured it's up on the integrator to model the appropriate RTEEvent. **Tags:** atp.Status=draft

[SWS_SwCluC_CONSTR_02233]{DRAFT} **SwCluCOsProxyOsTask relates only to a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_TASK** ⌈The SwCluCOsProxyOsTask.SwCluCResourceRef shall only reference a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_TASK.⌋*()*

10.2.7.2.4 Os Proxy OsResorce configuration

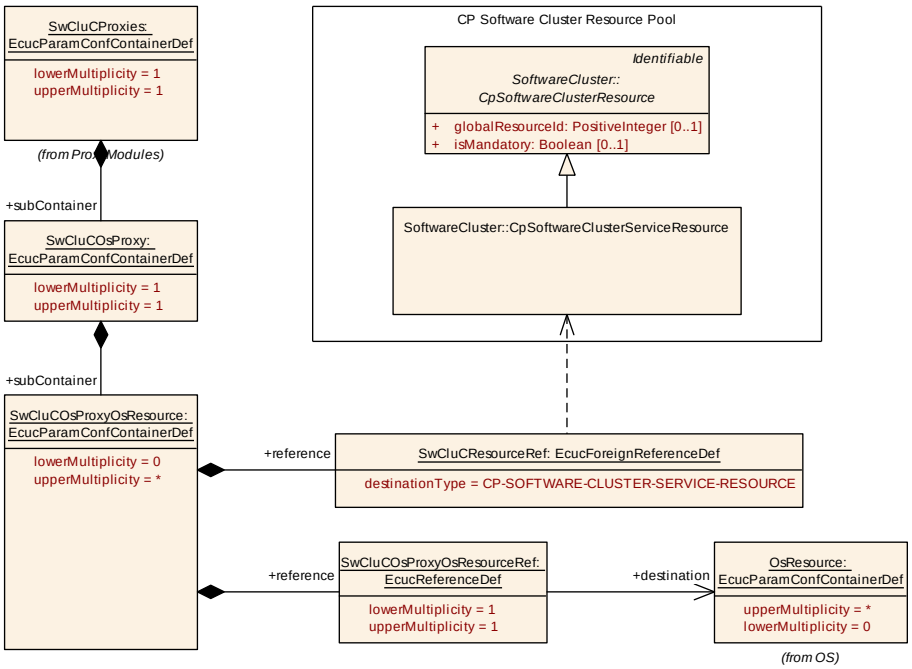

Figure 10.19: Os Proxy OsResource Parameter

SWS Item	[ECUC_SwCluC_00049]		
Container Name	SwCluCOsProxyOsResource		
Parent Container	SwCluCOsProxy		
Description	Maps a OsResource to the CpSoftwareClusterServiceResource describing the OsResource resource in the clustered system. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCOsProxyOsResourceRef [ECUC_SwCluC_00047]
Parent Container	SwCluCOsProxyOsResource
Description	Reference to the OsResource **Tags:** atp.Status=draft
Multiplicity	1
Type	Reference to OsResource
Scope / Dependency	scope: ECU

Name	SwCluCResourceRef [ECUC_SwCluC_00087]
Parent Container	SwCluCOsProxyOsResource

Description	Reference to the CpSoftwareClusterServiceResource. **Tags:** atp.Status=draft
Multiplicity	1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-SERVICE-RESOURCE
Scope / Dependency	scope: ECU

No Included Containers

[SWS_SwCluC_CONSTR_02234]{DRAFT} **SwCluCOsProxyOsResource relates only to a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_RESOURCE** ⌈The SwCluCOsProxyOsResource.SwCluCResourceRef shall only reference a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_RESOURCE.⌋()

10.2.7.2.5 Os Proxy OsSpinlock configuration

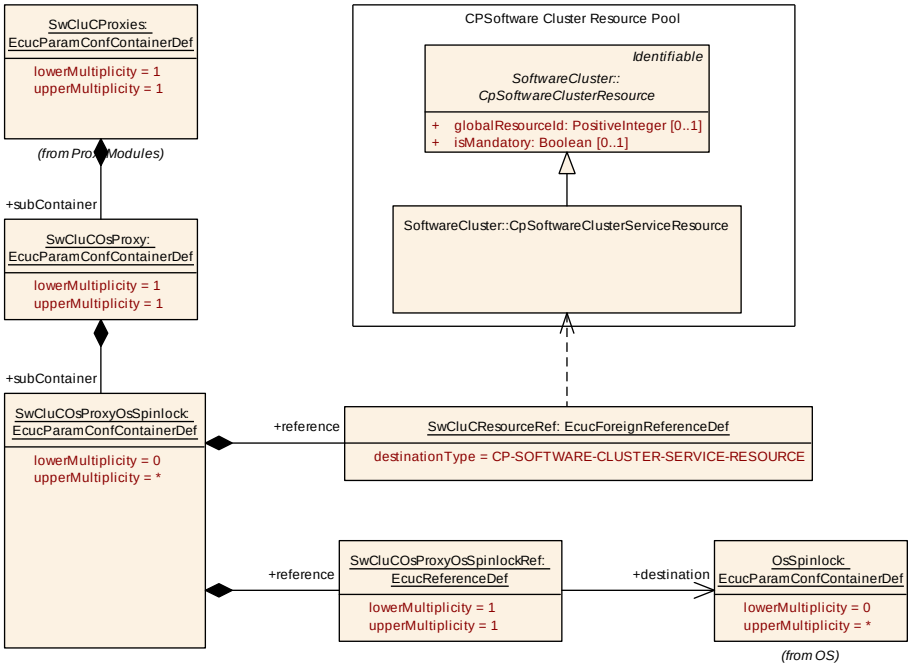

Figure 10.20: Os Proxy OsSpinlock Parameter

SWS Item	[ECUC_SwCluC_00051]
Container Name	SwCluCOsProxyOsSpinlock
Parent Container	SwCluCOsProxy

Description	Maps a OsSpinlock to the CpSoftwareClusterServiceResource describing the OsSpinlock resource in the clustered system. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCOsProxyOsSpinlockRef [ECUC_SwCluC_00052]
Parent Container	SwCluCOsProxyOsSpinlock
Description	Reference to the OsSpinlock **Tags:** atp.Status=draft
Multiplicity	1
Type	Reference to OsSpinlock
Scope / Dependency	scope: ECU

Name	SwCluCResourceRef [ECUC_SwCluC_00089]
Parent Container	SwCluCOsProxyOsSpinlock
Description	Reference to the CpSoftwareClusterServiceResource. **Tags:** atp.Status=draft
Multiplicity	1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-SERVICE-RESOURCE
Scope / Dependency	scope: ECU

No Included Containers

[SWS_SwCluC_CONSTR_02235]{DRAFT} **SwCluCOsProxyOsSpinlock relates only to a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_SPINLOCK** ⌈The SwCluCOsProxyOsSpinlock.SwCluCResourceRef shall only reference a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_SPINLOCK.⌋*()*

10.2.7.2.6 Os Proxy Task Dispatcher configuration

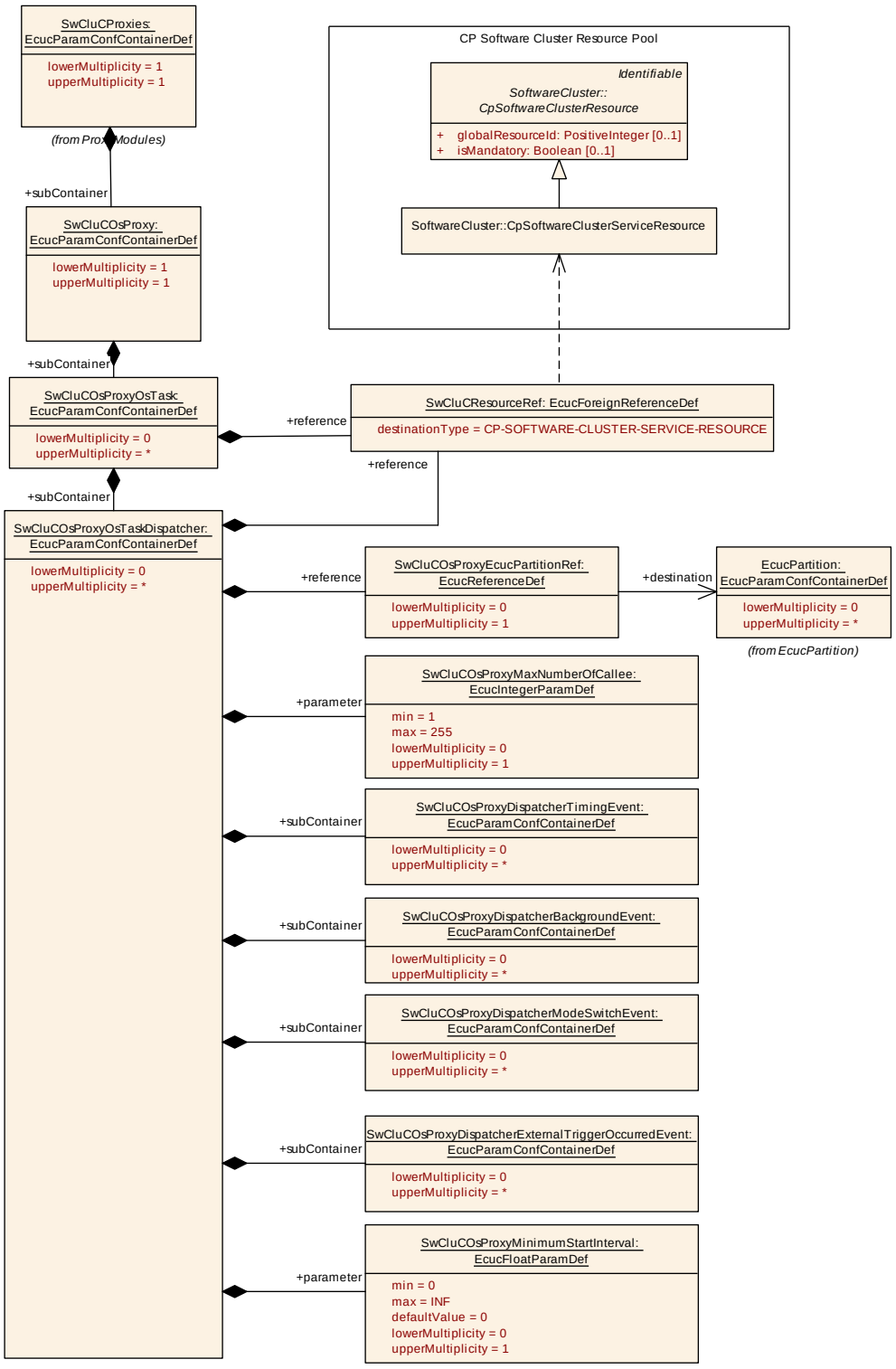

Figure 10.21: Os Proxy Task Dispatcher Parameter

SWS Item	[ECUC_SwCluC_00061]
Container Name	SwCluCOsProxyOsTaskDispatcher
Parent Container	SwCluCOsProxyOsTask

Description	This container defines a task dispatcher for the owning proxy task. In the Os High Proxy this configuration provides the OS Task body function as a Dispatch Entry Point in the Binary manifest which can be called by the Os Low Proxy. In the HIGH_PROXY configuration at most one SwCluCOsProxyOsTaskDispatcher can be owned by a SwCluCOsProxyOsTask In the Os High Proxy this configuration provides the Dispatcher Runnable (and according resource Entry in the Binary Manifest) which is able to call the MaxNumberOfCallee Dispatch Entry Points of Os High Proxies. In the LOW_PROXY configuration multiple SwCluCOsProxyOsTaskDispatcher can be owned by a SwCluCOsProxyOsTask. If none of the SwCluCOsProxyOsTaskDispatcher<xxx>Event container is configured it's up on the integrator to model the appropriate RTEEvent. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCOsProxyMaxNumberOfCallee [ECUC_SwCluC_00062]	
Parent Container	SwCluCOsProxyOsTaskDispatcher	
Description	Defines how many Dispatch Entry Points the dispatcher of the Os Low Proxy is able to call at most. In the Os Low Proxy this configuration parameter is mandatory. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 255	
Default Value		
Scope / Dependency	scope: ECU	

Name	SwCluCOsProxyMinimumStartInterval [ECUC_SwCluC_00070]	
Parent Container	SwCluCOsProxyOsTaskDispatcher	
Description	Specifies the time in seconds by which two consecutive starts of an dispatcher Runnable are guaranteed to be separated. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default Value	0	
Scope / Dependency	scope: ECU	

Name	SwCluCOsProxyEcucPartitionRef [ECUC_SwCluC_00078]		
Parent Container	SwCluCOsProxyOsTaskDispatcher		
Description	If the reference is not provided, the task dispatcher is provided on the EcuPartition to which the OsApplication of the OsTask related to the owning SwCluCOsProxyOsTask belongs. If the reference is provided, the task dispatcher is provided on the referenced EcuPartition. In this case the task dispatcher is scheduled in the OsTask related to the owning SwCluCOsProxyOsTask but is changing its partition before the task dispatcher schedules proxy tasks in the Applicative Software Cluster. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Reference to EcucPartition		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCResourceRef [ECUC_SwCluC_00095]
Parent Container	SwCluCOsProxyOsTaskDispatcher
Description	Reference to the CpSoftwareClusterServiceResource. **Tags:** atp.Status=draft
Multiplicity	1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-SERVICE-RESOURCE
Scope / Dependency	scope: ECU

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
SwCluCOsProxyDispatcherBackgroundEvent	0..*	Configures a BackgroundEvent starting the owing Dispatcher Runnable. **Tags:** atp.Status=draft
SwCluCOsProxyDispatcherExternalTriggerOccurredEvent	0..*	Configures a ExternalTriggerOccurredEvent starting the owing Dispatcher Runnable. **Tags:** atp.Status=draft

SwCluCOsProxy DispatcherModeSwitch Event	0..*	Configures a ModeSwitchEvent starting the owing Dispatcher Runnable. **Tags:** atp.Status=draft
SwCluCOsProxy DispatcherTimingEvent	0..*	Configures a TimingEvent starting the owing Dispatcher Runnable. **Tags:** atp.Status=draft

[SWS_SwCluC_CONSTR_02236]{DRAFT} **SwCluCOsProxyOsTaskDispatcher relates only to a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_TASK_DISPATCHER** ⌈The SwCluCOsProxyOsTaskDispatcher.SwCluCResourceRef shall only reference a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_TASK_DISPATCHER.⌋()

10.2.7.2.6.1 Os Proxy Task Dispatcher Timing Event configuration

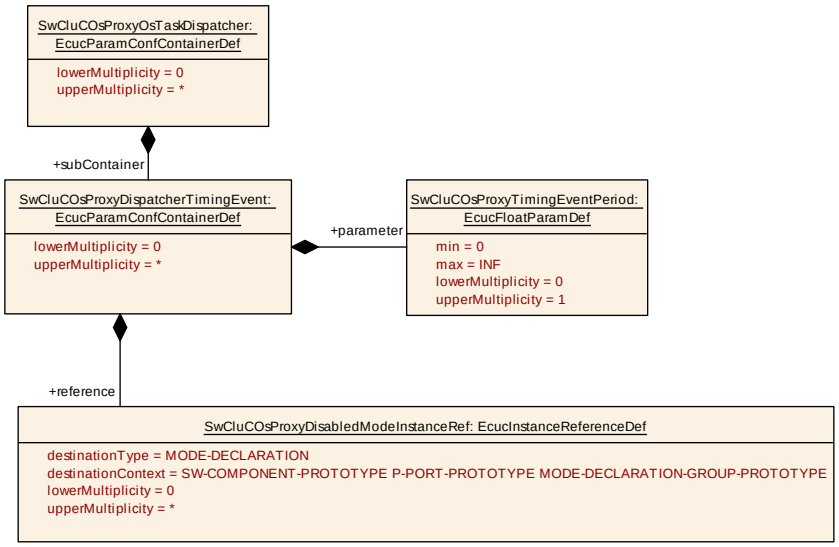

Figure 10.22: Os Proxy Task Dispatcher Timing Event Parameter

SWS Item	[ECUC_SwCluC_00066]		
Container Name	SwCluCOsProxyDispatcherTimingEvent		
Parent Container	SwCluCOsProxyOsTaskDispatcher		
Description	Configures a TimingEvent starting the owing Dispatcher Runnable. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Configuration Parameters

Name	SwCluCOsProxyTimingEventPeriod [ECUC_SwCluC_00065]	
Parent Container	SwCluCOsProxyDispatcherTimingEvent	
Description	Period of timing event in seconds. The value of this attribute shall be greater than zero. **Tags:** atp.Status=draft	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default Value		
Scope / Dependency	scope: ECU	

Name	SwCluCOsProxyDisabledModeInstanceRef [ECUC_SwCluC_00071]
Parent Container	SwCluCOsProxyDispatcherTimingEvent
Description	Reference to the Mode instance in a PPortPrototype that disable the Event. **Tags:** atp.Status=draft
Multiplicity	0..*
Type	Instance reference to MODE-DECLARATION context: SW-COMPONENT-PROTOTYPE P-PORT-PROTOTYPE MODE-DECLARATION-GROUP-PROTOTYPE
Scope / Dependency	scope: ECU

No Included Containers

10.2.7.2.6.2 Os Proxy Task Dispatcher Background Event configuration

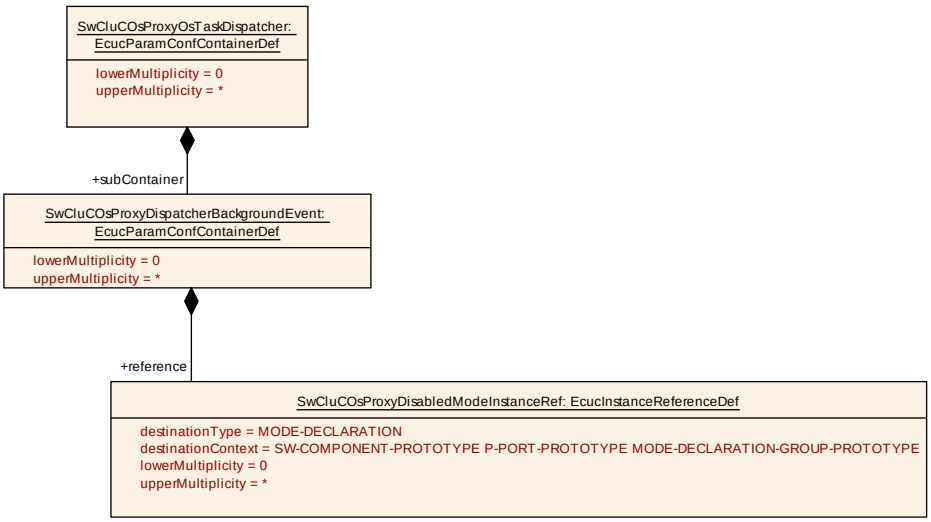

Figure 10.23: Os Proxy Task Dispatcher Background Event Parameter

SWS Item	[ECUC_SwCluC_00067]		
Container Name	SwCluCOsProxyDispatcherBackgroundEvent		
Parent Container	SwCluCOsProxyOsTaskDispatcher		
Description	Configures a BackgroundEvent starting the owing Dispatcher Runnable. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCOsProxyDisabledModeInstanceRef [ECUC_SwCluC_00071]
Parent Container	SwCluCOsProxyDispatcherBackgroundEvent
Description	Reference to the Mode instance in a PPortPrototype that disable the Event. **Tags:** atp.Status=draft
Multiplicity	0..*
Type	Instance reference to MODE-DECLARATION context: SW-COMPONENT-PROTOTYPE P-PORT-PROTOTYPE MODE-DECLARATION-GROUP-PROTOTYPE
Scope / Dependency	scope: ECU

No Included Containers

10.2.7.2.6.3 Os Proxy Task Dispatcher Mode Switch Event configuration

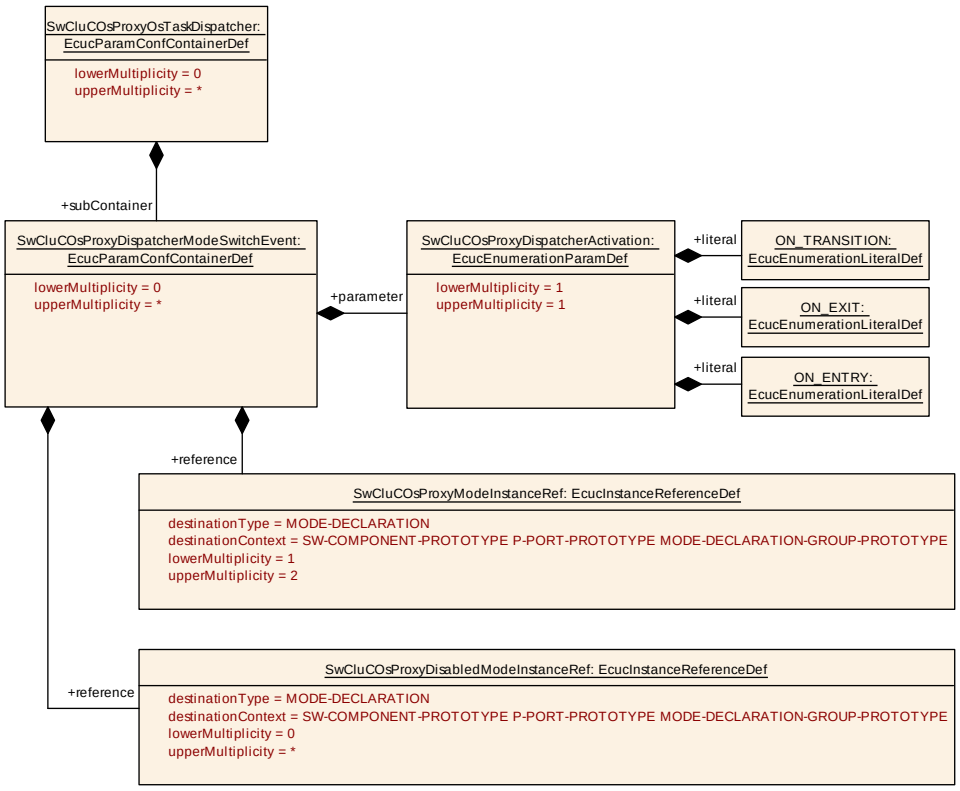

Figure 10.24: Os Proxy Task Dispatcher Mode Switch Event Parameter

SWS Item	[ECUC_SwCluC_00068]		
Container Name	SwCluCOsProxyDispatcherModeSwitchEvent		
Parent Container	SwCluCOsProxyOsTaskDispatcher		
Description	Configures a ModeSwitchEvent starting the owing Dispatcher Runnable. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCOsProxyDispatcherActivation [ECUC_SwCluC_00072]		
Parent Container	SwCluCOsProxyDispatcherModeSwitchEvent		
Description	Specifies if the event is activated on entering or exiting the referenced Mode instance. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	ON_ENTRY	On entering the referred mode.	
	ON_EXIT	On exiting the referred mode.	
	ON_TRANSITION	On transition of the 1st referred mode to the 2nd referred mode.	
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCOsProxyDisabledModeInstanceRef [ECUC_SwCluC_00071]
Parent Container	SwCluCOsProxyDispatcherModeSwitchEvent
Description	Reference to the Mode instance in a PPortPrototype that disable the Event. **Tags:** atp.Status=draft
Multiplicity	0..*
Type	Instance reference to MODE-DECLARATION context: SW-COMPONENT-PROTOTYPE P-PORT-PROTOTYPE MODE-DECLARATION-GROUP-PROTOTYPE
Scope / Dependency	scope: ECU

Name	SwCluCOsProxyModeInstanceRef [ECUC_SwCluC_00073]
Parent Container	SwCluCOsProxyDispatcherModeSwitchEvent
Description	Reference to one or two Mode instances in a PPortPrototype that that initiate the mode switch. **Tags:** atp.Status=draft
Multiplicity	1..2
Type	Instance reference to MODE-DECLARATION context: SW-COMPONENT-PROTOTYPE P-PORT-PROTOTYPE MODE-DECLARATION-GROUP-PROTOTYPE
Scope / Dependency	scope: ECU

No Included Containers

10.2.7.2.6.4 Os Proxy Task Dispatcher External Trigger Occurred Event configuration

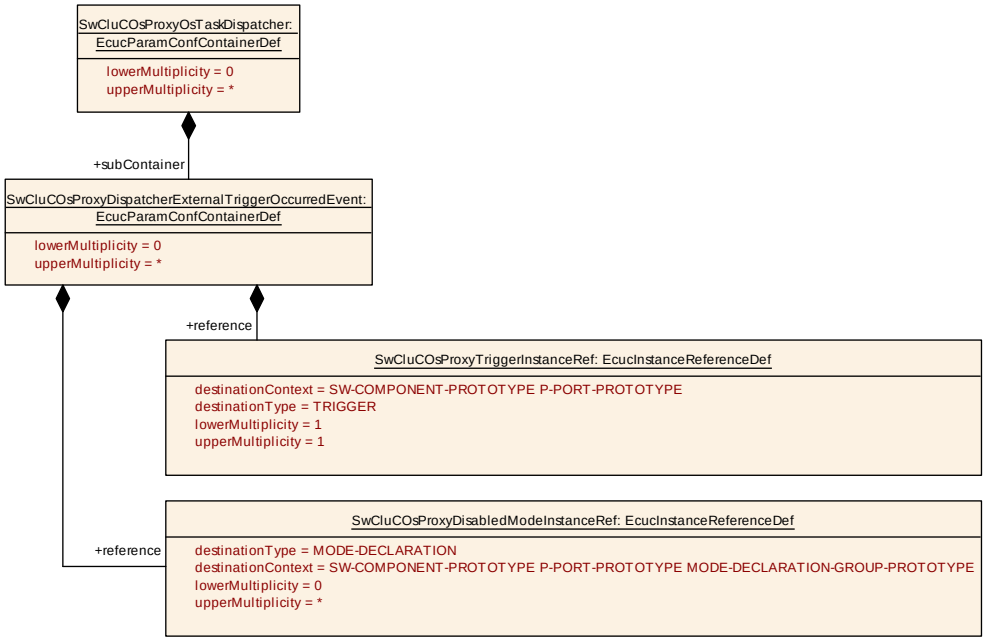

Figure 10.25: Os Proxy Task Dispatcher External Trigger Occurred Event Parameter

SWS Item	[ECUC_SwCluC_00069]		
Container Name	SwCluCOsProxyDispatcherExternalTriggerOccurredEvent		
Parent Container	SwCluCOsProxyOsTaskDispatcher		
Description	Configures a ExternalTriggerOccurredEvent starting the owing Dispatcher Runnable. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCOsProxyDisabledModeInstanceRef [ECUC_SwCluC_00071]
Parent Container	SwCluCOsProxyDispatcherExternalTriggerOccurredEvent
Description	Reference to the Mode instance in a PPortPrototype that disable the Event. **Tags:** atp.Status=draft
Multiplicity	0..*

Type	Instance reference to MODE-DECLARATION context: SW-COMPONENT-PROTOTYPE P-PORT-PROTOTYPE MODE-DECLARATION-GROUP-PROTOTYPE
Scope / Dependency	scope: ECU

Name	SwCluCOsProxyTriggerInstanceRef [ECUC_SwCluC_00074]
Parent Container	SwCluCOsProxyDispatcherExternalTriggerOccurredEvent
Description	Reference to the Trigger instances in a PPortPrototype that raise the trigger. **Tags:** atp.Status=draft
Multiplicity	1
Type	Instance reference to TRIGGER context: SW-COMPONENT-PROTOTYPE P-PORT-PROTOTYPE
Scope / Dependency	scope: ECU

No Included Containers

10.2.7.2.7 Os Proxy Function Dispatcher configuration

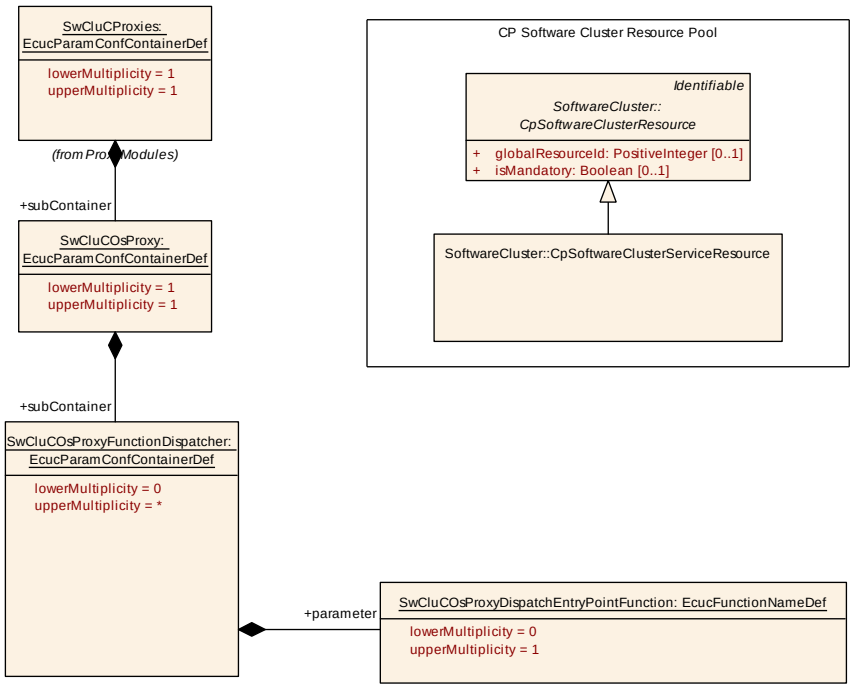

Figure 10.26: Os Proxy Function Dispatcher Parameter

SWS Item	[ECUC_SwCluC_00063]
Container Name	SwCluCOsProxyFunctionDispatcher
Parent Container	SwCluCOsProxy

Description	This container configures a function dispatcher. In the Os High Proxy this configuration puts a function as a Dispatch Entry Point in the Binary Manifest which can be called by the Os Low Proxy. In the Os Low Proxy this configuration provides the Dispatcher Function (and according resource Entry in the Binary Manifest) which is able to call the MaxNumberOfCallee Dispatch Entry Points of Os High Proxies. **Tags:** atp.Status=draft		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	SwCluCOsProxyDispatchEntryPointFunction [ECUC_SwCluC_00064]		
Parent Container	SwCluCOsProxyFunctionDispatcher		
Description	Name of the function which is put into the Binary Manifest as Dispatch Entry Point. In the Os High Proxy this configuration parameter is mandatory **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	SwCluCResourceRef [ECUC_SwCluC_00097]
Parent Container	SwCluCOsProxyFunctionDispatcher
Description	Reference to the CpSoftwareClusterServiceResource. **Tags:** atp.Status=draft
Multiplicity	1
Type	Foreign reference to CP-SOFTWARE-CLUSTER-SERVICE-RESOURCE
Scope / Dependency	scope: ECU

No Included Containers

[SWS_SwCluC_CONSTR_02237]{DRAFT} **SwCluCOsProxyFunctionDispatcher relates only to a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_FNC_DISPATCHER** ⌈The SwCluCOsProxyFunctionDispatcher.SwCluCResourceRef shall only reference a CpSoftwareClusterServiceResource of category SWCLUSTER_RES_OS_FNC_DISPATCHER.⌋*()*

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in SWS_BSWGeneral.

A Not applicable requirements

[SWS_SwCluC_00999]{DRAFT} ⌈These requirements are not applicable to this specification.⌋*(SRS_BSW_00344, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00170, SRS_BSW_00380, SRS_BSW_00419, SRS_BSW_00438, SRS_BSW_00375, SRS_BSW_00416, SRS_BSW_00406, SRS_BSW_00467, SRS_BSW_00437, SRS_BSW_00168, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00450, SRS_BSW_00461, SRS_BSW_00451, SRS_BSW_00478, SRS_BSW_00339, SRS_BSW_00422, SRS_BSW_00417, SRS_BSW_00409, SRS_BSW_00452, SRS_BSW_00458, SRS_BSW_00466, SRS_BSW_00488, SRS_BSW_00469, SRS_BSW_00470, SRS_BSW_00471, SRS_BSW_00472, SRS_BSW_00473, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00415, SRS_BSW_00164, SRS_BSW_00325, SRS_BSW_00342, SRS_BSW_00343, SRS_BSW_00160, SRS_BSW_00453, SRS_BSW_00456, SRS_BSW_00457, SRS_BSW_00479, SRS_BSW_00483, SRS_BSW_00007, SRS_BSW_00413, SRS_BSW_00347, SRS_BSW_00307, SRS_BSW_00373, SRS_BSW_00335, SRS_BSW_00410, SRS_BSW_00411, SRS_BSW_00463, SRS_BSW_00481, SRS_BSW_00346, SRS_BSW_00314, SRS_BSW_00447, SRS_BSW_00348, SRS_BSW_00353, SRS_BSW_00361, SRS_BSW_00301, SRS_BSW_00302, SRS_BSW_00328, SRS_BSW_00312, SRS_BSW_00006, SRS_BSW_00439, SRS_BSW_00448, SRS_BSW_00449, SRS_BSW_00357, SRS_BSW_00377, SRS_BSW_00304, SRS_BSW_00378, SRS_BSW_00306, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00484, SRS_BSW_00485, SRS_BSW_00486, SRS_BSW_00371, SRS_BSW_00414, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00440, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00454, SRS_BSW_00477, SRS_BSW_00459, SRS_BSW_00460)*

B Referenced Meta Classes

Class	***AbstractAccessPoint*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::AccessCount			
Note	Abstract class indicating an access point from an ExecutableEntity.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AsynchronousServerCallResultPoint, ExternalTriggeringPointIdent, InternalTriggeringPoint, ModeAccessPointIdent, ModeSwitchPoint, ParameterAccess, *ServerCallPoint*, VariableAccess			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
returnValue Provision	RteApiReturnValue ProvisionEnum	0..1	attr	This attribute controls the provision of return values for RTE APIs that correspond to the enclosing access point.

Table B.1: AbstractAccessPoint

Class	**ApplicationArrayDataType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	An application data type which is an array, each element is of the same application data type. **Tags:**atp.recommendedPackage=ApplicationDataTypes			
Base	*ARElement*, *ARObject*, *ApplicationCompositeDataType*, *ApplicationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dynamicArray SizeProfile	String	0..1	attr	Specifies the profile which the array will follow if it is a variable size array.
element	ApplicationArray Element	0..1	aggr	This association implements the concept of an array element. That is, in some cases it is necessary to be able to identify single array elements, e.g. as input values for an interpolation routine.

Table B.2: ApplicationArrayDataType

Class	**ApplicationArrayElement**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Describes the properties of the elements of an application array data type.			
Base	*ARObject*, *ApplicationCompositeElementDataPrototype*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arraySize Handling	ArraySizeHandling Enum	0..1	attr	The way how the size of the array is handled.
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls how the information about the array size shall be interpreted.
indexDataType	ApplicationPrimitive DataType	0..1	ref	This reference can be taken to assign a CompuMethod of category TEXTTABLE to the array. The texttable entries associate a textual value to an index number such that the element with that index number is represented by a symbolic name.
maxNumberOf Elements	PositiveInteger	0..1	attr	The maximum number of elements that the array can contain. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime

Table B.3: ApplicationArrayElement

Class	***ApplicationCompositeElementDataPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	This class represents a data prototype which is aggregated within a composite application data type (record or array). It is introduced to provide a better distinction between target and context in instance Refs.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ApplicationArrayElement, ApplicationRecordElement			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
type	ApplicationDataType	0..1	tref	This represents the corresponding data type. **Stereotypes:** isOfType

Table B.4: ApplicationCompositeElementDataPrototype

Class	***ApplicationDataType*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	ApplicationDataType defines a data type from the application point of view. Especially it should be used whenever something "physical" is at stake. An ApplicationDataType represents a set of values as seen in the application model, such as measurement units. It does not consider implementation details such as bit-size, endianess, etc. It should be possible to model the application level aspects of a VFB system by using ApplicationData Types only.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	*ApplicationCompositeDataType*, ApplicationPrimitiveDataType			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.5: ApplicationDataType

Class	**ApplicationError**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	This is a user-defined error that is associated with an element of an AUTOSAR interface. It is specific for the particular functionality or service provided by the AUTOSAR software component.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
errorCode	Integer	0..1	attr	The RTE generator is forced to assign this value to the corresponding error symbol. Note that for error codes certain ranges are predefined (see RTE specification).

Table B.6: ApplicationError

Class	**ApplicationPrimitiveDataType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	A primitive data type defines a set of allowed values. **Tags:**atp.recommendedPackage=ApplicationDataTypes			
Base	*ARElement*, *ARObject*, *ApplicationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
–	–	–	–	–

Table B.7: ApplicationPrimitiveDataType

Class	ApplicationRecordDataType			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	An application data type which can be decomposed into prototypes of other application data types. **Tags:**atp.recommendedPackage=ApplicationDataTypes			
Base	*ARElement*, *ARObject*, *ApplicationCompositeDataType*, *ApplicationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
element (ordered)	ApplicationRecordElement	*	aggr	Specifies an element of a record. The aggregation of ApplicationRecordElement is subject to variability with the purpose to support the conditional existence of elements inside a ApplicationrecordDataType. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime

Table B.8: ApplicationRecordDataType

Class	ApplicationRecordElement			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Describes the properties of one particular element of an application record data type.			
Base	*ARObject*, *ApplicationCompositeElementDataPrototype*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing ApplicationRecordElement as optional. This means the that, at runtime, the ApplicationRecordElement may or may not have a valid value and shall therefore be ignored. The underlying runtime software provides means to set the ApplicationRecordElement as not valid at the sending end of a communication and determine its validity at the receiving end.

Table B.9: ApplicationRecordElement

Class	ArgumentDataPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	An argument of an operation, much like a data element, but also carries direction information and is owned by a particular ClientServerOperation.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
direction	ArgumentDirectionEnum	0..1	attr	This attribute specifies the direction of the argument prototype.
serverArgumentImplPolicy	ServerArgumentImplPolicyEnum	0..1	attr	This defines how the argument type of the servers RunnableEntity is implemented. If the attribute is not defined this has the same semantics as if the attribute is set to the value useArgumentType for primitive arguments and structures.

Table B.10: ArgumentDataPrototype

Enumeration	**ArraySizeSemanticsEnum**
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes
Note	This type controls how the information about the number of elements in an ApplicationArrayDataType is to be interpreted.
Literal	***Description***
fixedSize	This means that the ApplicationArrayDataType will always have a fixed number of elements. **Tags:**atp.EnumerationLiteralIndex=0
variableSize	This implies that the actual number of elements in the ApplicationArrayDataType might vary at run-time. The value of arraySize represents the maximum number of elements in the array. **Tags:**atp.EnumerationLiteralIndex=1

Table B.11: ArraySizeSemanticsEnum

Class	**AssemblySwConnector**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	AssemblySwConnectors are exclusively used to connect SwComponentPrototypes in the context of a CompositionSwComponentType.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *SwConnector*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
provider	AbstractProvidedPort Prototype	0..1	iref	Instance of providing port. **InstanceRef implemented by:**PPortInComposition InstanceRef
requester	AbstractRequiredPort Prototype	0..1	iref	Instance of requiring port. **InstanceRef implemented by:**RPortInComposition InstanceRef

Table B.12: AssemblySwConnector

Class	**AsynchronousServerCallReturnsEvent**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
Note	This event is raised when an asynchronous server call is finished.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *Multilanguage Referrable*, *RTEEvent*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
eventSource	AsynchronousServer CallResultPoint	0..1	ref	The referenced AsynchronousServerCallResultPoint which is raises the RTEEvent in case of returning asynchronous server call.

Table B.13: AsynchronousServerCallReturnsEvent

Class	***AutosarDataPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Base class for prototypical roles of an AutosarDataType.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ArgumentDataPrototype, ParameterDataPrototype, VariableDataPrototype			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
type	AutosarDataType	0..1	tref	This represents the corresponding data type. **Stereotypes:** isOfType

Table B.14: AutosarDataPrototype

Class	***AutosarDataType*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes			
Note	Abstract base class for user defined AUTOSAR data types for software.			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *Multilanguage Referrable*, *PackageableElement*, *Referrable*			
Subclasses	*AbstractImplementationDataType*, *ApplicationDataType*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of this AutosarDataType.

Table B.15: AutosarDataType

Class	***BaseType*** (abstract)			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This abstract meta-class represents the ability to specify a platform dependant base type.			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Subclasses	SwBaseType			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
baseType Definition	BaseTypeDefinition	1	aggr	This is the actual definition of the base type. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

Table B.16: BaseType

Class	**BaseTypeDirectDefinition**			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This BaseType is defined directly (as opposite to a derived BaseType)			
Base	*ARObject*, *BaseTypeDefinition*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
baseType Encoding	BaseTypeEncoding String	0..1	attr	This specifies, how an object of the current BaseType is encoded, e.g. in an ECU within a message sequence. **Tags:**xml.sequenceOffset=90
baseTypeSize	PositiveInteger	0..1	attr	Describes the length of the data type specified in the container in bits. **Tags:**xml.sequenceOffset=70
byteOrder	ByteOrderEnum	0..1	attr	This attribute specifies the byte order of the base type. **Tags:**xml.sequenceOffset=110
memAlignment	PositiveInteger	0..1	attr	This attribute describes the alignment of the memory object in bits. E.g. "8" specifies, that the object in question is aligned to a byte while "32" specifies that it is aligned four byte. If the value is set to "0" the meaning shall be interpreted as "unspecified". **Tags:**xml.sequenceOffset=100

△

Class	BaseTypeDirectDefinition			
native Declaration	NativeDeclarationString	0..1	attr	This attribute describes the declaration of such a base type in the native programming language, primarily in the Programming language C. This can then be used by a code generator to include the necessary declarations into a header file. For example BaseType with shortName: "MyUnsignedInt" native Declaration: "unsigned short" Results in typedef unsigned short MyUnsignedInt; If the attribute is not defined the referring Implementation DataTypes will not be generated as a typedef by RTE. If a nativeDeclaration type is given it shall fulfill the characteristic given by basetypeEncoding and baseType Size. This is required to ensure the consistent handling and interpretation by software components, RTE, COM and MCM systems. **Tags:**xml.sequenceOffset=120

Table B.17: BaseTypeDirectDefinition

Class	BswExternalTriggerOccurredEvent			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	A BswEvent resulting from a trigger released by another module or cluster.			
Base	*ARObject*, *AbstractEvent*, *BswEvent*, *BswScheduleEvent*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
trigger	Trigger	1	ref	The trigger associated with this event. The trigger is external to this module.

Table B.18: BswExternalTriggerOccurredEvent

Class	BswInternalTriggerOccurredEvent			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	A BswEvent, which can happen sporadically. The event is activated by explicit calls from the module to the BSW Scheduler. The main purpose for such an event is to cause a context switch, e.g. from an ISR context into a task context. Activation and switching are handled within the same module or cluster only.			
Base	*ARObject*, *AbstractEvent*, *BswEvent*, *BswScheduleEvent*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
eventSource	BswInternalTriggering Point	1	ref	The activation point is the source of this event.

Table B.19: BswInternalTriggerOccurredEvent

Class	BswModeSwitchEvent			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	A BswEvent resulting from a mode switch.			
Base	*ARObject*, *AbstractEvent*, *BswEvent*, *BswScheduleEvent*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	**BswModeSwitchEvent**			
activation	ModeActivationKind	1	attr	Kind of activation w.r.t. to the referred mode.
mode (ordered)	ModeDeclaration	0..2	iref	Reference to one or two Modes that initiate the Mode Switch Event. **InstanceRef implemented by:**ModeInBswModule DescriptionInstanceRef

Table B.20: BswModeSwitchEvent

Class	**BswModuleDescription**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswOverview			
Note	Root element for the description of a single BSW module or BSW cluster. In case it describes a BSW module, the short name of this element equals the name of the BSW module. **Tags:**atp.recommendedPackage=BswModuleDescriptions			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
bswModule Dependency	BswModuleDependency	*	aggr	Describes the dependency to another BSW module. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=bswModuleDependency.shortName, bsw ModuleDependency.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
bswModule Documentation	SwComponent Documentation	0..1	aggr	This adds a documentation to the BSW module. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=bswModuleDocumentation, bswModule Documentation.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=6
expectedEntry	BswModuleEntry	*	ref	Indicates an entry which is required by this module. Replacement of outgoingCallback / requiredEntry. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=expectedEntry.bswModuleEntry, expected Entry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
implemented Entry	BswModuleEntry	*	ref	Specifies an entry provided by this module which can be called by other modules. This includes "main" functions, interrupt routines, and callbacks. Replacement of providedEntry / expectedCallback. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=implementedEntry.bswModuleEntry, implementedEntry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
internalBehavior	BswInternalBehavior	*	aggr	The various BswInternalBehaviors associated with a Bsw ModuleDescription can be distributed over several physical files. Therefore the aggregation is <<atp Splitable>>. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=internalBehavior.shortName xml.sequenceOffset=65

▽

△

Class	**BswModuleDescription**			
moduleId	PositiveInteger	0..1	attr	Refers to the BSW Module Identifier defined by the AUTOSAR standard. For non-standardized modules, a proprietary identifier can be optionally chosen. **Tags:**xml.sequenceOffset=5
providedClient ServerEntry	BswModuleClientServer Entry	*	aggr	Specifies that this module provides a client server entry which can be called from another parition or core.This entry is declared locally to this context and will be connected to the requiredClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=providedClientServerEntry.shortName, providedClientServerEntry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=45
providedData	VariableDataPrototype	*	aggr	Specifies a data prototype provided by this module in order to be read from another partition or core.The providedData is declared locally to this context and will be connected to the requiredData of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=providedData.shortName, provided Data.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=55
providedMode Group	ModeDeclarationGroup Prototype	*	aggr	A set of modes which is owned and provided by this module or cluster. It can be connected to the required ModeGroups of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with modes provided via ports by an associated ServiceSwComponentType, EcuAbstraction SwComponentType or ComplexDeviceDriverSw ComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=providedModeGroup.shortName, provided ModeGroup.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=25
releasedTrigger	Trigger	*	aggr	A Trigger released by this module or cluster. It can be connected to the requiredTriggers of other modules or clusters via the configuration of the BswScheduler. It can also be synchronized with Triggers provided via ports by an associated ServiceSwComponentType, Ecu AbstractionSwComponentType or ComplexDeviceDriver SwComponentType. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=releasedTrigger.shortName, released Trigger.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=35

▽

△

Class	**BswModuleDescription**			
requiredClient ServerEntry	BswModuleClientServer Entry	*	aggr	Specifies that this module requires a client server entry which can be implemented on another parition or core.This entry is declared locally to this context and will be connected to the providedClientServerEntry of another or the same module via the configuration of the BSW Scheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=requiredClientServerEntry.shortName, requiredClientServerEntry.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=50
requiredData	VariableDataPrototype	*	aggr	Specifies a data prototype required by this module in oder to be provided from another partition or core.The required Data is declared locally to this context and will be connected to the providedData of another or the same module via the configuration of the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=requiredData.shortName, required Data.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=60
requiredMode Group	ModeDeclarationGroup Prototype	*	aggr	Specifies that this module or cluster depends on a certain mode group. The requiredModeGroup is local to this context and will be connected to the providedModeGroup of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=requiredModeGroup.shortName, required ModeGroup.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=30
requiredTrigger	Trigger	*	aggr	Specifies that this module or cluster reacts upon an external trigger.This requiredTrigger is declared locally to this context and will be connected to the providedTrigger of another module or cluster via the configuration of the BswScheduler. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=requiredTrigger.shortName, required Trigger.variationPoint.shortLabel vh.latestBindingTime=preCompileTime xml.sequenceOffset=40

Table B.21: BswModuleDescription

Class	***BswModuleEntity*** (abstract)			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	Specifies the smallest code fragment which can be described for a BSW module or cluster within AUTOSAR.			
Base	*ARObject*, *ExecutableEntity*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	BswCalledEntity, BswInterruptEntity, BswSchedulableEntity			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	***BswModuleEntity*** (abstract)			
accessedMode Group	ModeDeclarationGroup Prototype	*	ref	A mode group which is accessed via API call by this entity. It shall be a ModeDeclarationGroupPrototype required by this module or cluster. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
activationPoint	BswInternalTriggering Point	*	ref	Activation point used by the module entity to activate one or more internal triggers. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
callPoint	BswModuleCallPoint	*	aggr	A call point used in the code of this entitiy. The variablity of this association is especially targeted at debug scenarios: It is possible to have one variant calling into the AUTOSAR debug module and another one which doesn't. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
dataReceive Point	BswVariableAccess	*	aggr	The data is received via the BSW Scheduler. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
dataSendPoint	BswVariableAccess	*	aggr	The data is sent via the BSW Scheduler. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
implemented Entry	BswModuleEntry	1	ref	The entry which is implemented by this module entity.
issuedTrigger	Trigger	*	ref	A trigger issued by this entity via BSW Scheduler API call. It shall be a BswTrigger released (i.e. owned) by this module or cluster. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
managedMode Group	ModeDeclarationGroup Prototype	*	ref	A mode group which is managed by this entity. It shall be a ModeDeclarationGroupPrototype provided by this module or cluster. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
schedulerName Prefix	BswSchedulerName Prefix	0..1	ref	A prefix to be used in generated names for the Bsw ModuleScheduler in the context of this BswModuleEntity, for example entry point prototypes, macros for dealing with exclusive areas, header file names. Details are defined in the SWS RTE. The prefix supersedes default rules for the prefix of those names.

Table B.22: BswModuleEntity

Class	**BswSchedulableEntity**			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	BSW module entity, which is designed for control by the BSW Scheduler. It may for example implement a so-called "main" function.			
Base	*ARObject*, *BswModuleEntity*, *ExecutableEntity*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	BswSchedulableEntity			
–	–	–	–	–

Table B.23: BswSchedulableEntity

Class	BswVariableAccess			
Package	M2::AUTOSARTemplates::BswModuleTemplate::BswBehavior			
Note	The presence of a BswVariableAccess implies that a BswModuleEntity needs access to a VariableData Prototype via the BSW Scheduler. The kind of access is specified by the role in which the class is used.			
Base	*ARObject*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
accessed Variable	VariableDataPrototype	1	ref	The data accessed via the BSW Scheduler.
context Limitation	BswDistinguished Partition	*	ref	The existence of this reference indicates that the variable is recevied resp. sent only in the context of the referred BswDistinguishedPartitions.

Table B.24: BswVariableAccess

Class	ClientServerApplicationErrorMapping			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	This meta-class represents the ability to map ApplicationErrors onto each other.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
firstApplication Error	ApplicationError	0..1	ref	This represents the first ApplicationError in the context of the ClientServerApplicationErrorMapping.
second ApplicationError	ApplicationError	0..1	ref	This represents the second ApplicationError in the context of the ClientServerApplicationErrorMapping.

Table B.25: ClientServerApplicationErrorMapping

Class	ClientServerInterface			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A client/server interface declares a number of operations that can be invoked on a server by a client. **Tags:**atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
operation	ClientServerOperation	*	aggr	ClientServerOperation(s) of this ClientServerInterface. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=blueprintDerivationTime
possibleError	ApplicationError	*	aggr	Application errors that are defined as part of this interface.

Table B.26: ClientServerInterface

Class	ClientServerOperation
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface
Note	An operation declared within the scope of a client/server interface.
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	**ClientServerOperation**			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument (ordered)	ArgumentDataPrototype	*	aggr	An argument of this ClientServerOperation **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=blueprintDerivationTime
diagArgIntegrity	Boolean	0..1	attr	This attribute shall only be used in the implementation of diagnostic routines to support the case where input and output arguments are allocated in a shared buffer and might unintentionally overwrite input arguments by tentative write operations to output arguments. This situation can happen during sliced execution or while output parameters are arrays (call by reference). The value true means that the ClientServerOperation is aware of the usage of a shared buffer and takes precautions to avoid unintentional overwrite of input arguments. If the attribute does not exist or is set to false the Client ServerOperation does not have to consider the usage of a shared buffer.
possibleError	ApplicationError	*	ref	Possible errors that may by raised by the referring operation.

Table B.27: ClientServerOperation

Class	**ClientServerOperationMapping**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Defines the mapping of two particular ClientServerOperations in context of two different ClientServer Interfaces.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument Mapping	DataPrototypeMapping	*	aggr	Defines the mapping of two particular ArgumentData Prototypes with unequal names or unequal semantic (resolution or range) in context of Operations.
firstOperation	ClientServerOperation	0..1	ref	First to-be-mapped ClientServerOperation of a Client ServerInterface.
firstToSecond Data Transformation	DataTransformation	0..1	ref	This reference indicates that a DataTransformation is intended in the context of the ClientServerOperation Mapping.
second Operation	ClientServerOperation	0..1	ref	Second to-be-mapped ClientServerOperation of a Client ServerInterface.

Table B.28: ClientServerOperationMapping

Class	**CompositionSwComponentType**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	A CompositionSwComponentType aggregates SwComponentPrototypes (that in turn are typed by Sw ComponentTypes) as well as SwConnectors for primarily connecting SwComponentPrototypes among each others and towards the surface of the CompositionSwComponentType. By this means hierarchical structures of software-components can be created. **Tags:**atp.recommendedPackage=SwComponentTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*, *SwComponentType*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	**CompositionSwComponentType**			
component	SwComponent Prototype	*	aggr	The instantiated components that are part of this composition. The aggregation of SwComponentPrototype is subject to variability with the purpose to support the conditional existence of a SwComponentPrototype. Please be aware: if the conditional existence of Sw ComponentPrototypes is resolved post-build the deselected SwComponentPrototypes are still contained in the ECUs build but the instances are inactive in in that they are not scheduled by the RTE. The aggregation is marked as atpSplitable in order to allow the addition of service components to the ECU extract during the ECU integration. The use case for having 0 components owned by the CompositionSwComponentType could be to deliver an empty CompositionSwComponentType to e.g. a supplier for filling the internal structure. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=component.shortName, component.variation Point.shortLabel vh.latestBindingTime=postBuild
connector	SwConnector	*	aggr	SwConnectors have the principal ability to establish a connection among PortPrototypes. They can have many roles in the context of a CompositionSwComponentType. Details are refined by subclasses. The aggregation of SwConnectors is subject to variability with the purpose to support variant data flow. The aggregation is marked as atpSplitable in order to allow the extension of the ECU extract with AssemblySw Connectors between ApplicationSwComponentTypes and ServiceSwComponentTypes during the ECU integration. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=connector.shortName, connector.variation Point.shortLabel vh.latestBindingTime=postBuild
constantValue Mapping	ConstantSpecification MappingSet	*	ref	Reference to the ConstantSpecificationMapping to be applied for initValues of PPortComSpecs and RPortCom Spec. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=constantValueMapping
dataType Mapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMapping to be applied for the used ApplicationDataTypes in PortInterfaces. Background: when developing subsystems it may happen that ApplicationDataTypes are used on the surface of CompositionSwComponentTypes. In this case it would be reasonable to be able to also provide the intended mapping to the ImplementationDataTypes. However, this mapping shall be informal and not technically binding for the implementors mainly because the RTE generator is not concerned about the CompositionSwComponent Types. Rationale: if the mapping of ApplicationDataTypes on the delegated and inner PortPrototype matches then the ▽

▽

△

Class	CompositionSwComponentType			
				△ mapping to ImplementationDataTypes is not impacting compatibility. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=dataTypeMapping
instantiation RTEEventProps	InstantiationRTEEvent Props	*	aggr	This allows to define instantiation specific properties for RTE Events, in particular for instance specific scheduling. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=instantiationRTEEventProps.shortLabel, instantiationRTEEventProps.variationPoint.shortLabel vh.latestBindingTime=codeGenerationTime

Table B.29: CompositionSwComponentType

Class	Compu			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to express one particular computation.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
compuContent	CompuContent	0..1	aggr	This specifies the details of the computation. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false
compuDefault Value	CompuConst	0..1	aggr	This property can be used to specify an output value for a conversion formula, if the value to be converted lies outside the plausibility limit. Although this is possible for all conversion formulae, it is especially valid for variables with tabular conversion formulae. **Tags:**xml.sequenceOffset=70

Table B.30: Compu

Class	CompuConstFormulaContent			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the fact that the constant value of the computation method is represented by a variation point. This difference is due to compatibility with ASAM HDO.			
Base	*ARObject*, *CompuConstContent*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
vf	Numerical	1	attr	Value calculated via a system constant. This element is included in every case where parameters should be generated from numerical values during compile time (not runtime!). Thus for example, the influence of the cylinder number on conversion formulae can be introduced in a repeatable manner. ▽

▽

△

Class	CompuConstFormulaContent			
				△ **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=codeGenerationTime xml.sequenceOffset=30

Table B.31: CompuConstFormulaContent

Class	CompuConstTextContent			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the textual content of a scale.			
Base	*ARObject*, *CompuConstContent*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
vt	VerbatimString	0..1	attr	This represents a textual constant in the computation method.

Table B.32: CompuConstTextContent

Class	CompuMethod			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to express the relationship between a physical value and the mathematical representation. Note that this is still independent of the technical implementation in data types. It only specifies the formula how the internal value corresponds to its physical pendant. **Tags:**atp.recommendedPackage=CompuMethods			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
compuInternalToPhys	Compu	0..1	aggr	This specifies the computation from internal values to physical values. **Tags:**xml.sequenceOffset=80
compuPhysToInternal	Compu	0..1	aggr	This represents the computation from physical values to the internal values. **Tags:**xml.sequenceOffset=90
displayFormat	DisplayFormatString	0..1	attr	This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools. **Tags:**xml.sequenceOffset=20
unit	Unit	0..1	ref	This is the physical unit of the Physical values for which the CompuMethod applies. **Tags:**xml.sequenceOffset=30

Table B.33: CompuMethod

Class	CompuNominatorDenominator			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This class represents the ability to express a polynomial either as Nominator or as Denominator.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**

▽

△

Class	CompuNominatorDenominator			
v (ordered)	Numerical	*	attr	this is the list of polynomial factors. Note that the first vf represents the power=0. The polynomial is v[0] * x^0 + v[1] * x^1 ... **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.roleElement=true xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

Table B.34: CompuNominatorDenominator

Class	CompuRationalCoeffs			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to express a rational function by specifying the coefficients of nominator and denominator.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
compu Denominator	CompuNominator Denominator	0..1	aggr	This is the denominator of the expression. **Tags:**xml.sequenceOffset=30
compu Numerator	CompuNominator Denominator	0..1	aggr	This is the numerator of the rational expression. **Tags:**xml.sequenceOffset=20

Table B.35: CompuRationalCoeffs

Class	CompuScale			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to specify one segment of a segmented computation method.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
compuInverse Value	CompuConst	0..1	aggr	This is the inverse value of the constraint. This supports the case that the scale is not reversible per se. **Tags:**xml.sequenceOffset=60
compuScale Contents	CompuScaleContents	0..1	aggr	This represents the computation details of the scale. **Tags:** xml.roleElement=false xml.roleWrapperElement=false xml.sequenceOffset=70 xml.typeElement=false xml.typeWrapperElement=false
desc	MultiLanguageOverview Paragraph	0..1	aggr	<desc> represents a general but brief description of the object in question. **Tags:**xml.sequenceOffset=30
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=40

▽

△

Class	CompuScale			
mask	PositiveInteger	0..1	attr	In difference to all the other computational methods every COMPU-SCALE will be applied including the bit MASK. Therefore it is allowed for this type of COMPU-METHOD, that COMPU-SCALES overlap. To calculate the string reverse to a value, the string has to be split and the according value for each substring has to be summed up. The sum is finally transmitted. The processing has to be done in order of the COMPU-SCALE elements. **Tags:**xml.sequenceOffset=35
shortLabel	Identifier	0..1	attr	This element specifies a short name for the particular scale. The name can for example be used to derive a programming language identifier. **Tags:**xml.sequenceOffset=20
symbol	CIdentifier	0..1	attr	The symbol, if provided, is used by code generators to get a C identifier for the CompuScale. The name will be used as is for the code generation, therefore it needs to be unique within the generation context. **Tags:**xml.sequenceOffset=25
upperLimit	Limit	0..1	attr	This specifies the upper limit of a of the scale. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=50

Table B.36: CompuScale

Class	CompuScaleConstantContents			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the fact that a particular scale of the computation method is constant.			
Base	*ARObject*, *CompuScaleContents*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
compuConst	CompuConst	0..1	aggr	This represents the fact that the scale is a constant. The use case is mainly a non interpolated scale. It is a simplification of the fact that a constant scale can also be expressed as rational function of order 0. **Tags:**xml.sequenceOffset=90

Table B.37: CompuScaleConstantContents

Class	CompuScales			
Package	M2::MSR::AsamHdo::ComputationMethod			
Note	This meta-class represents the ability to stepwise express a computation method.			
Base	*ARObject*, *CompuContent*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
compuScale (ordered)	CompuScale	*	aggr	This represents one scale within the compu method. Note that it contains a Variationpoint in order to support blueprints of enumerations. **Stereotypes:** atpVariation **Tags:** ▽

▽

△

Class	CompuScales			
				△ vh.latestBindingTime=blueprintDerivationTime xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=40 xml.typeElement=false xml.typeWrapperElement=false

Table B.38: CompuScales

Class	CpSoftwareCluster			
Package	M2::AUTOSARTemplates::SystemTemplate::SoftwareCluster			
Note	This meta class provides the ability to define a CP Software Cluster. Each CP Software Cluster can be integrated and build individually. It defines the sub-set of hierarchical tree(s) of Software Components belonging to this CP Software Cluster. Resources required or provided by this CP Software Cluster are given in the according mappings. **Tags:** atp.Status=draft atp.recommendedPackage=CpSoftwareClusters			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
swComponent Assignment	SwComponent PrototypeAssignment	*	aggr	This is the colllection of SwComponentPrototype Assignments **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=swComponentAssignment, swComponent Assignment.variationPoint.shortLabel atp.Status=draft vh.latestBindingTime=postBuild
swComposition	CompositionSw ComponentType	*	ref	Software Components in the context of a CompositionSw ComponentType belonging to this CP Software Cluster. This reference can be used to describe the belonging SWCs when the CP Software Cluster is described out of the context of a System, e.g. reusable CP Software Cluster. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=swComposition.compositionSwComponent Type, swComposition.variationPoint.shortLabel atp.Status=draft vh.latestBindingTime=systemDesignTime

Table B.39: CpSoftwareCluster

Class	CpSoftwareClusterBinaryManifestDescriptor
Package	M2::AUTOSARTemplates::SystemTemplate::SoftwareCluster::BinaryManifest
Note	This meta-class has the ability to act as a hub for all information related to the binary manifest of a given CP software cluster. The manifest is subject to integrator work and therefore not a part of the definition of the CP software cluster itself. **Tags:** atp.Status=draft atp.recommendedPackage=CpSoftwareClusterBinaryManifestDescriptors
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*

▽

△

Class	CpSoftwareClusterBinaryManifestDescriptor			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
cpSoftware Cluster	CpSoftwareCluster	0..1	ref	This reference identifies the CpSoftwareCluster to which the enclosing CpSoftwareClusterBinaryManifest Descriptor belongs, The CpSoftwareClusterBinaryManifestDescriptor is defined in an integration phase while the refrenced Cp SoftwareCluster represents a design element. Therefore, it makes sense to use a reference rather than an aggregation in the relation of the two meta-classes. **Tags:**atp.Status=draft
metaDataField	BinaryManifestMeta DataField	*	aggr	This aggregation identifies the collection of meta-data contained in the enclosing binary manifest. **Tags:**atp.Status=draft
provide Resource	BinaryManifestProvide Resource	*	aggr	This aggregation represents the collection of provided resources in the enclosing binary manifest. **Tags:**atp.Status=draft
require Resource	BinaryManifestRequire Resource	*	aggr	This aggregation represents the collection of required resources in the enclosing binary manifest. **Tags:**atp.Status=draft
resource Definition	BinaryManifest ResourceDefinition	*	aggr	This aggregation represents the collection of binary manifest resource definitions that belong to the enclosing CpSoftwareClusterBinaryManifestDescriptor. **Tags:**atp.Status=draft
softwareCluster Id	PositiveInteger	0..1	attr	This attribute represents the value of the id of the corresponding CP software cluster. This id is only assigned by an integrator and can therefore not be part of the description of the CP software cluster itself.

Table B.40: CpSoftwareClusterBinaryManifestDescriptor

Class	CpSoftwareClusterCommunicationResource			
Package	M2::AUTOSARTemplates::SystemTemplate::SoftwareCluster			
Note	Represents a single resource required or provided by a CP Software Cluster which relates to the port based communication on VFB level. **Tags:**atp.Status=draft			
Base	*ARObject*, *CpSoftwareClusterResource*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
comProps	CpSoftwareCluster Communication ResourceProps	0..1	aggr	This aggregation supports the further qualification of the enclosing CpSoftwareClusterCommunicationRecource by means of additional attributes depending on the nature of the CpSoftwareClusterCommunicationRecource.

Table B.41: CpSoftwareClusterCommunicationResource

Class	***CpSoftwareClusterResource*** (abstract)
Package	M2::AUTOSARTemplates::SystemTemplate::SoftwareCluster
Note	Represents a single resource required or provided by a CP Software Cluster. **Tags:** atp.Status=draft atp.recommendedPackage=Resources

▽

△

Class	***CpSoftwareClusterResource*** (abstract)			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	CpSoftwareClusterCommunicationResource, CpSoftwareClusterServiceResource			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dependent Resource	RoleBasedResource Dependency	*	aggr	Link to a resource which depends on this resource to implement them.
globalResource Id	PositiveInteger	0..1	attr	A unique identifiers per resource used for the connection process. The identifier is required to be unique in the scope of a single machine. If software clusters are designed to be reused on multiple machines the uniqueness requirements applies for all the intended machines.
isMandatory	Boolean	0..1	attr	This attribute indicates, that the resource is mandatory to operate the Software Cluster. If the resource is not provided on the machine the connection process of any Software Cluster requiring this resource gets aborted.

Table B.42: CpSoftwareClusterResource

Class	**CpSoftwareClusterResourcePool**			
Package	M2::AUTOSARTemplates::SystemTemplate::SoftwareCluster			
Note	Represents the pool of resources which can be provided or required by CP Software Clusters. **Tags:** atp.Status=draft atp.recommendedPackage=CpSoftwareClusterResourcePools			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ecuScope	EcuInstance	*	ref	This reference identifies the EcuInstance in which the resource pool is defined. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=ecuScope
resource	CpSoftwareCluster Resource	*	aggr	This aggregation represents the collection of resources in the enclosing resource pool. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=resource.shortName atp.Status=draft

Table B.43: CpSoftwareClusterResourcePool

Class	**CpSoftwareClusterServiceResource**			
Package	M2::AUTOSARTemplates::SystemTemplate::SoftwareCluster			
Note	Represents a single resource required or provided by a CP Software Cluster which relates to the BSW. **Tags:**atp.Status=draft			
Base	*ARObject*, *CpSoftwareClusterResource*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
resourceNeeds	EcucContainerValue	*	ref	Refernce(s) to one or multiple EcucContainerValue(s) qualifying the characteristics of the resource. **Tags:**atp.Status=draft

Table B.44: CpSoftwareClusterServiceResource

Class	**DataComProps**			
Package	M2::AUTOSARTemplates::SystemTemplate::SoftwareCluster			
Note	Represents a single resource required or provided by a CP Software Cluster which relates to the port based communication on VFB level. **Tags:**atp.Status=draft			
Base	*ARObject*, *CpSoftwareClusterCommunicationResourceProps*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
sendIndication	SendIndicationEnum	0..1	attr	Send indication behavior for last-is-the best data communciation.

Table B.45: DataComProps

Class	**DataConstr**			
Package	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
Note	This meta-class represents the ability to specify constraints on data. **Tags:**atp.recommendedPackage=DataConstrs			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataConstrRule	DataConstrRule	*	aggr	This is one particular rule within the data constraints. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=30 xml.typeElement=false xml.typeWrapperElement=false

Table B.46: DataConstr

Class	**DataConstrRule**			
Package	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
Note	This meta-class represents the ability to express one specific data constraint rule.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
constrLevel	Integer	0..1	attr	This attribute describes the category of a constraint. One of its functions is in the area of constraint violation, where it can be used from a certain level, to produce error messages. The lower the level, the more stringent the check. Used to distinguish hard or soft limits. **Tags:**xml.sequenceOffset=20
internalConstrs	InternalConstrs	0..1	aggr	Describes the limitations applicable on the internal domain (as opposed to the physical domain). **Tags:**xml.sequenceOffset=40
physConstrs	PhysConstrs	0..1	aggr	Describes the limitations applicable on the physical domain (as opposed to the internal domain). **Tags:**xml.sequenceOffset=30

Table B.47: DataConstrRule

Class	***DataMapping*** (abstract)			
Package	M2::AUTOSARTemplates::SystemTemplate::DataMapping			
Note	Mapping of port elements (data elements and parameters) to frames and signals.			
Base	*ARObject*			
Subclasses	ClientServerToSignalMapping, SenderReceiverCompositeElementToSignalMapping, SenderReceiverToSignalGroupMapping, SenderReceiverToSignalMapping, TriggerToSignalMapping			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
communicationDirection	CommunicationDirectionType	0..1	attr	This attribute controls the direction into which the mapped SystemSignal is communicated with respect to the kind of PortPrototype used as the context element of the Data Mapping.
introduction	DocumentationBlock	0..1	aggr	This represents introductory documentation about the data mapping.

Table B.48: DataMapping

Class	***DataPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	Base class for prototypical roles of any data type.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*ApplicationCompositeElementDataPrototype*, *AutosarDataPrototype*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
swDataDefProps	SwDataDefProps	0..1	aggr	This property allows to specify data definition properties which apply on data prototype level.

Table B.49: DataPrototype

Class	**DataReceivedEvent**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
Note	The event is raised when the referenced data elements are received.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
data	VariableDataPrototype	0..1	iref	Data element referenced by event **InstanceRef implemented by:**RVariableInAtomicSwcInstanceRef

Table B.50: DataReceivedEvent

Class	**EcuInstance**			
Package	M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology			
Note	ECUInstances are used to define the ECUs used in the topology. The type of the ECU is defined by a reference to an ECU specified with the ECU resource description. **Tags:**atp.recommendedPackage=EcuInstances			
Base	*ARObject*, *CollectableElement*, *FibexElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	EcuInstance			
associatedComIPduGroup	ISignalIPduGroup	*	ref	With this reference it is possible to identify which ISignal IPduGroups are applicable for which Communication Connector/ ECU. Only top level ISignalIPduGroups shall be referenced by an EcuInstance. If an ISignalIPduGroup contains other ISignalIPduGroups than these contained ISignalIPdu Groups shall not be referenced by the EcuInstance. Contained ISignalIPduGroups are associated to an Ecu Instance via the top level ISignalIPduGroup.
associated ConsumedProvided ServiceInstanceGroup	ConsumedProvided ServiceInstanceGroup	*	ref	With this reference it is possible to identify which ConsumedProvidedServiceInstanceGroups are applicable for which ECUInstance. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
associatedPdurIPduGroup	PdurIPduGroup	*	ref	With this reference it is possible to identify which PduR IPdu Groups are applicable for which Communication Connector/ ECU.
clientIdRange	ClientIdRange	0..1	aggr	Restriction of the Client Identifier for this Ecu to an allowed range of numerical values. The Client Identifier of the transaction handle is generated by the client RTE for inter-Ecu Client/Server communication.
com ConfigurationGwTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_MainFunctionRouteSignals of the AUTOSAR COM module in seconds.
com ConfigurationRxTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_MainFunctionRx of the AUTOSAR COM module in seconds.
com ConfigurationTxTimeBase	TimeValue	0..1	attr	The period between successive calls to Com_MainFunctionTx of the AUTOSAR COM module in seconds.
comEnableMDTForCyclicTransmission	Boolean	0..1	attr	Enables for the Com module of this EcuInstance the minimum delay time monitoring for cyclic and repeated transmissions (TransmissionModeTiming has cyclic Timing assigned or eventControlledTiming with numberOfRepetitions > 0).
commController	CommunicationController	1..*	aggr	CommunicationControllers of the ECU. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
connector	CommunicationConnector	*	aggr	All channels controlled by a single controller. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
dltConfig	DltConfig	0..1	aggr	Describes the Dlt configuration on this EcuInstance.
doIpConfig	DoIpConfig	0..1	aggr	DoIp configuration on this EcuInstance. **Tags:**atp.Status=draft
ethSwitchPortGroupDerivation	Boolean	0..1	attr	Defines whether the derivation of SwitchPortGroups based on VLAN and/or CouplingPort.pncMapping shall be performed for this EcuInstance. If not defined the derivation shall not be done.
partition	EcuPartition	*	aggr	Optional definition of Partitions within an Ecu.
pncPrepareSleepTimer	TimeValue	0..1	attr	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.

▽

△

Class	EcuInstance			
pnc Synchronous Wakeup	Boolean	0..1	attr	If this parameter is available and set to true then all available PNCs will be woken up as soon as a channel wakeup occurs. This is ensured by adding all PNCs to all channel wakeup sources during upstream mapping.
pnResetTime	TimeValue	0..1	attr	Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests in the EIRA and in the ERA.
sleepMode Supported	Boolean	1	attr	Specifies whether the ECU instance may be put to a "low power mode" • true: sleep mode is supported • false: sleep mode is not supported Note: This flag may only be set to "true" if the feature is supported by both hardware and basic software.
tcpIpIcmpProps	EthTcpIpIcmpProps	0..1	ref	EcuInstance specific ICMP (Internet Control Message Protocol) attributes
tcpIpProps	EthTcpIpProps	0..1	ref	EcuInstance specific TcpIp Stack attributes.
v2xSupported	V2xSupportEnum	0..1	attr	This attribute is used to control the existence of the V2X stack on the given EcuInstance.
wakeUpOver BusSupported	Boolean	1	attr	Driver support for wakeup over Bus.

Table B.51: EcuInstance

Class	*EcucContainerDef* (abstract)			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Base class used to gather common attributes of configuration container definitions.			
Base	*ARObject*, *AtpDefinition*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	EcucChoiceContainerDef, EcucParamConfContainerDef			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
destinationUri	EcucDestinationUriDef	*	ref	Several destinationUris can be defined for an Ecuc ContainerDef. With such destinationUris an Ecuc ContainerDef is applicable for several EcucUriReference Defs. **Stereotypes:** atpUriDef
multiplicity ConfigClass	EcucMultiplicity ConfigurationClass	*	aggr	Specifies which MultiplicityConfigurationClass this container is available for which ConfigurationVariant. This aggregation is optional if the surrounding EcucModuleDef has the Category STANDARDIZED_MODULE_ DEFINITION. If the category attribute of the EcucModule Def is set to VENDOR_SPECIFIC_MODULE_ DEFINITION and if the upperMultiplicity is greater than the lowerMultiplicity then this aggregation is mandatory. **Tags:**xml.name Plural=MULTIPLICITY-CONFIG-CLASSES
postBuildVariant Multiplicity	Boolean	0..1	attr	Indicates if a container may have different number of instances in different post-build variants (previously known as post-build selectable configuration sets). TRUE means yes, FALSE means no.
requiresIndex	Boolean	0..1	attr	Used to define whether the value element for this definition shall be provided with an index.

Table B.52: EcucContainerDef

Class	**EcucContainerValue**			
Package	M2::AUTOSARTemplates::ECUCDescriptionTemplate			
Note	Represents a Container definition in the ECU Configuration Description.			
Base	*ARObject*, *EcucIndexableValue*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
definition	EcucContainerDef	0..1	ref	Reference to the definition of this Container in the ECU Configuration Parameter Definition. **Stereotypes:** atpIdentityContributor **Tags:**xml.sequenceOffset=-10
parameterValue	EcucParameterValue	*	aggr	Aggregates all ECU Configuration Values within this Container. atpVariation: [RS_ECUC_00079] **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=parameterValue.definition, parameter Value.variationPoint.shortLabel vh.latestBindingTime=postBuild
referenceValue	EcucAbstractReference Value	*	aggr	Aggregates all References with this container. atpVariation: [RS_ECUC_00079] **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=referenceValue.definition, reference Value.variationPoint.shortLabel vh.latestBindingTime=postBuild
subContainer	EcucContainerValue	*	aggr	Aggregates all sub-containers within this container. atpVariation: [RS_ECUC_00078] **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=subContainer.definition, subContainer.short Name, subContainer.variationPoint.shortLabel vh.latestBindingTime=postBuild

Table B.53: EcucContainerValue

Class	**EcucEnumerationParamDef**			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Configuration parameter type for Enumeration. **Tags:**xml.sequenceOffset=0			
Base	*ARObject*, *AtpDefinition*, *EcucCommonAttributes*, *EcucDefinitionElement*, *EcucParameterDef*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
defaultValue	Identifier	0..1	attr	Default value of the enumeration configuration parameter. This string needs to be one of the literals specified for this enumeration.
literal	EcucEnumerationLiteral Def	*	aggr	Aggregation on the literals used to define this enumeration parameter. This aggregation is optional if the surrounding EcucModuleDef has the category STANDARDIZED_MODULE_DEFINITION. If the category attribute of the EcucModuleDef is set to VENDOR_SPECIFIC_MODULE_DEFINITION then this aggregation is mandatory. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=literal.shortName

Table B.54: EcucEnumerationParamDef

Class	EcucForeignReferenceDef			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Specify a reference to an XML description of an entity described in another AUTOSAR template.			
Base	*ARObject*, *AtpDefinition*, *EcucAbstractExternalReferenceDef*, *EcucAbstractReferenceDef*, *EcucCommonAttributes*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
destinationType	String	0..1	attr	The type in the AUTOSAR Metamodel to which instance this reference is allowed to point to.

Table B.55: EcucForeignReferenceDef

Class	EcucModuleConfigurationValues			
Package	M2::AUTOSARTemplates::ECUCDescriptionTemplate			
Note	Head of the configuration of one Module. A Module can be a BSW module as well as the RTE and ECU Infrastructure. As part of the BSW module description, the EcucModuleConfigurationValues element has two different roles: The recommendedConfiguration contains parameter values recommended by the BSW module vendor. The preconfiguredConfiguration contains values for those parameters which are fixed by the implementation and cannot be changed. These two EcucModuleConfigurationValues are used when the base EcucModuleConfigurationValues (as part of the base ECU configuration) is created to fill parameters with initial values. **Tags:**atp.recommendedPackage=EcucModuleConfigurationValuess			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
container	EcucContainerValue	*	aggr	Aggregates all containers that belong to this module configuration. atpVariation: [RS_ECUC_00078] **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=container.definition, container.shortName, container.variationPoint.shortLabel vh.latestBindingTime=postBuild xml.sequenceOffset=10
definition	EcucModuleDef	0..1	ref	Reference to the definition of this EcucModule ConfigurationValues element. Typically, this is a vendor specific module configuration. **Stereotypes:** atpIdentityContributor **Tags:**xml.sequenceOffset=-10
ecucDefEdition	RevisionLabelString	0..1	attr	This is the version info of the ModuleDef ECUC Parameter definition to which this values conform to / are based on. For the Definition of ModuleDef ECUC Parameters the AdminData shall be used to express the semantic changes. The compatibility rules between the definition and value revision labels is up to the module's vendor.
implementation ConfigVariant	EcucConfiguration VariantEnum	0..1	attr	Specifies the kind of deliverable this EcucModule ConfigurationValues element provides. If this element is not used in a particular role (e.g. preconfigured Configuration or recommendedConfiguration) then the value shall be one of VariantPreCompile, VariantLink Time, VariantPostBuild.

▽

△

Class	EcucModuleConfigurationValues			
module Description	BswImplementation	0..1	ref	Referencing the BSW module description, which this EcucModuleConfigurationValues element is configuring. This is optional because the EcucModuleConfiguration Values element is also used to configure the ECU infrastructure (memory map) or Application SW-Cs. However in case the EcucModuleConfigurationValues are used to configure the module, the reference is mandatory in order to fetch module specific "common" published information.
postBuildVariant Used	Boolean	0..1	attr	Indicates whether a module implementation has or plans to have (i.e., introduced at link or post-build time) new post-build variation points. TRUE means yes, FALSE means no. If the attribute is not defined, FALSE semantics shall be assumed.

Table B.56: EcucModuleConfigurationValues

Class	EcucModuleDef			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Used as the top-level element for configuration definition for Software Modules, including BSW and RTE as well as ECU Infrastructure. **Tags:**atp.recommendedPackage=EcucModuleDefs			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpDefinition*, *CollectableElement*, *EcucDefinitionElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
apiServicePrefix	CIdentifier	0..1	attr	For CDD modules this attribute holds the apiService Prefix. The shortName of the module definition of a Complex Driver is always "Cdd". Therefore for CDD modules the module apiServicePrefix is described with this attribute.
container	EcucContainerDef	*	aggr	Aggregates the top-level container definitions of this specific module definition. **Stereotypes:** atpSplitable **Tags:** atp.Splitkey=container.shortName xml.sequenceOffset=11
postBuildVariant Support	Boolean	0..1	attr	Indicates if a module supports different post-build variants (previously known as post-build selectable configuration sets). TRUE means yes, FALSE means no.
refinedModule Def	EcucModuleDef	0..1	ref	Optional reference from the Vendor Specific Module Definition to the Standardized Module Definition it refines. In case this EcucModuleDef has the category STANDARDIZED_MODULE_DEFINITION this reference shall not be provided. In case this EcucModuleDef has the category VENDOR_SPECIFIC_MODULE_ DEFINITION this reference is mandatory. **Stereotypes:** atpUriDef
supported ConfigVariant	EcucConfiguration VariantEnum	*	attr	Specifies which ConfigurationVariants are supported by this software module. This attribute is optional if the Ecuc ModuleDef has the category STANDARDIZED_ MODULE_DEFINITION. If the category attribute of the EcucModuleDef is set to VENDOR_SPECIFIC_ MODULE_DEFINITION then this attribute is mandatory.

Table B.57: EcucModuleDef

Class	EcucParamConfContainerDef			
Package	M2::AUTOSARTemplates::ECUCParameterDefTemplate			
Note	Used to define configuration containers that can hierarchically contain other containers and/or parameter definitions.			
Base	*ARObject*, *AtpDefinition*, *EcucContainerDef*, *EcucDefinitionElement*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
parameter	EcucParameterDef	*	aggr	The parameters defined within the EcucParamConf ContainerDef. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=parameter.shortName
reference	EcucAbstractReference Def	*	aggr	The references defined within the EcucParamConf ContainerDef. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=reference.shortName
subContainer	EcucContainerDef	*	aggr	The containers defined within the EcucParamConf ContainerDef. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=subContainer.shortName

Table B.58: EcucParamConfContainerDef

Class	***ExecutableEntity*** (abstract)			
Package	M2::AUTOSARTemplates::CommonStructure::InternalBehavior			
Note	Abstraction of executable code.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*BswModuleEntity*, RunnableEntity			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
activation Reason	ExecutableEntity ActivationReason	*	aggr	If the ExecutableEntity provides at least one activation Reason element the RTE resp. BSW Scheduler shall provide means to read the activation vector of this executable entity execution. If no activationReason element is provided the feature of being able to determine the activating RTEEvent is disabled for this ExecutableEntity.
canEnter ExclusiveArea	ExclusiveArea	*	ref	This means that the executable entity can enter/leave the referenced exclusive area through explicit API calls.
exclusiveArea NestingOrder	ExclusiveAreaNesting Order	*	ref	This represents the set of ExclusiveAreaNestingOrders recognized by this ExecutableEntity.
minimumStart Interval	TimeValue	0..1	attr	Specifies the time in seconds by which two consecutive starts of an ExecutableEntity are guaranteed to be separated.
reentrancyLevel	ReentrancyLevelEnum	0..1	attr	The reentrancy level of this ExecutableEntity. See the documentation of the enumeration type ReentrancyLevel Enum for details. Please note that nonReentrant interfaces can have also reentrant or multicoreReentrant implementations, and reentrant interfaces can also have multicoreReentrant implementations.
runsInside ExclusiveArea	ExclusiveArea	*	ref	The executable entity runs completely inside the referenced exclusive area.

△

Class	***ExecutableEntity*** (abstract)			
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this code entity. Via an association to the same SwAddrMethod, it can be specified that several code entities (even of different modules or components) shall be located in the same memory without already specifying the memory section itself.

Table B.59: ExecutableEntity

Class	**ExternalTriggerOccurredEvent**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
Note	The event is raised when the referenced trigger have been occurred.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *Multilanguage Referrable*, *RTEEvent*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
trigger	Trigger	0..1	iref	Reference to the applicable Trigger. **InstanceRef implemented by:**RTriggerInAtomicSwc InstanceRef

Table B.60: ExternalTriggerOccurredEvent

Class	**ExternalTriggeringPoint**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::Trigger			
Note	If a RunnableEntity owns an ExternalTriggeringPoint it is entitled to raise an ExternalTriggerOccurred Event.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ident	ExternalTriggeringPoint Ident	0..1	aggr	The aggregation in the role ident provides the ability to make the ExternalTriggeringPoint identifiable. From the semantical point of view, the ExternalTriggering Point is considered a first-class Identifiable and therefore the aggregation in the role ident shall always exist (until it may be possible to let ModeAccessPoint directly inherit from Identifiable). **Stereotypes:** atpIdentityContributor **Tags:** atp.Status=shallBecomeMandatory xml.sequenceOffset=-100
trigger	Trigger	0..1	iref	The trigger taken for the ExternalTriggeringPoint. **Tags:** xml.namePlural=TRIGGER-IREF xml.roleElement=false xml.roleWrapperElement=true xml.typeElement=true xml.typeWrapperElement=false **InstanceRef implemented by:**PTriggerInAtomicSwcType InstanceRef

Table B.61: ExternalTriggeringPoint

Class	**FlatInstanceDescriptor**			
Package	M2::AUTOSARTemplates::CommonStructure::FlatMap			
Note	Represents exactly one node (e.g. a component instance or data element) of the instance tree of a software system. The purpose of this element is to map the various nested representations of this instance to a flat representation and assign a unique name (shortName) to it. Use cases: • Specify unique names of measurable data to be used by MCD tools • Specify unique names of calibration data to be used by MCD tool • Specify a unique name for an instance of a component prototype in the ECU extract of the system description Note that in addition it is possible to assign alias names via AliasNameAssignment.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
ecuExtract Reference	AtpFeature	0..1	iref	Refers to the instance in the ECU extract. This is valid only, if the FlatMap is used in the context of an ECU extract. The reference shall be such that it uniquely defines the object instance. For example, if a data prototype is declared as a role within an SwcInternalBehavior, it is not enough to state the SwcInternalBehavior as context and the aggregated data prototype as target. In addition, the reference shall also include the complete path identifying instance of the component prototype and the Atomic SoftwareComponentType, which is refered by the particular SwcInternalBehavior. **Tags:**xml.sequenceOffset=40 **InstanceRef implemented by:**AnyInstanceRef
role	Identifier	0..1	attr	The role denotes the particular role of the downstream memory location described by this FlatInstanceDescriptor. It applies to use case where one upstream object results in multiple downstream objects, e.g. ModeDeclaration GroupPrototypes which are measurable. In this case the RTE will provide locations for current mode, previous mode and next mode.
rtePluginProps	RtePluginProps	0..1	aggr	The properties of a communication graph with respect to the utilization of RTE Implementation Plug-in. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=rtePluginProps
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of this FlatInstanceDescriptor.
upstream Reference	AtpFeature	0..1	iref	Refers to the instance in the context of an "upstream" descriptions, wich could be the system or system extract description, the basic software module description or (if a flat map is used in preliminary context) a description of an atomic component or composition. This reference is optional in case the flat map is used in ECU context. The reference shall be such that it uniquely defines the object instance in the given context. For example, if a data prototype is declared as a role within an SwcInternal Behavior, it is not enough to state the SwcInternal Behavior as context and the aggregated data prototype as target. In addition, the reference shall also include the complete path identifying the instance of the component ▽

▽

△

Class	**FlatInstanceDescriptor**			
				△ prototype that contains the particular instance of Swc InternalBehavior. **Tags:**xml.sequenceOffset=20 **InstanceRef implemented by:**AnyInstanceRef

Table B.62: FlatInstanceDescriptor

Enumeration	**HandleInvalidEnum**
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication
Note	Strategies of handling the reception of invalidValue.
Literal	***Description***
dontInvalidate	Invalidation is switched off. **Tags:**atp.EnumerationLiteralIndex=0
external Replacement	Replace a received invalidValue. The replacement value is sourced from the externalReplacement. **Tags:**atp.EnumerationLiteralIndex=1
keep	The application software is supposed to handle signal invalidation on RTE API level either by Data ReceiveErrorEvent or check of error code on read access. **Tags:**atp.EnumerationLiteralIndex=2
replace	Replace a received invalidValue. The replacement value is specified by the initValue. **Tags:**atp.EnumerationLiteralIndex=3

Table B.63: HandleInvalidEnum

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*, Application Endpoint, ApplicationError, ApplicationPartitionToEcuPartitionMapping, AsynchronousServerCallResult Point, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpFeature*, AutosarOperationArgumentInstance, AutosarVariableInstance, *BinaryManifestAddressableObject*, BinaryManifestItemDefinition, *Binary ManifestResource*, BinaryManifestResourceDefinition, BlockState, BswInternalTriggeringPoint, Bsw ModuleDependency, *BuildActionEntity*, BuildActionEnvironment, CanTpAddress, CanTpChannel, CanTp Node, Chapter, ClassContentConditional, ClientIdDefinition, ClientServerOperation, Code, *Collectable Element*, ComManagementMapping, *CommConnectorPort*, *CommunicationConnector*, *Communication Controller*, Compiler, ConsistencyNeeds, ConsumedEventGroup, CouplingPort, *CouplingPortStructural Element*, *CpSoftwareClusterResource*, CpSoftwareClusterResourceToApplicationPartitionMapping, Cp SoftwareClusterToEcuInstanceMapping, CpSoftwareClusterToResourceMapping, *CryptoService Mapping*, DataPrototypeGroup, DataTransformation, DependencyOnArtifact, *DiagEventDebounce Algorithm*, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticFunctionInhibitSource, *DiagnosticRoutineSubfunction*, DltArgument, DltLogChannel, DltMessage, DoIpInterface, DoIpLogic Address, DoIpRoutingActivation, ECUMapping, *EOCExecutableEntityRefAbstract*, EcuPartition, Ecuc ContainerValue, *EcucDefinitionElement*, EcucDestinationUriDef, EcucEnumerationLiteralDef, Ecuc Query, EcucValidationCondition, EndToEndProtection, EthernetWakeupSleepOnDatalineConfig, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeatureMapAssertion, FMFeature MapCondition, FMFeatureMapElement, FMFeatureRelation, FMFeatureRestriction, FMFeatureSelection, FlatInstanceDescriptor, FlexrayArTpNode, FlexrayTpConnectionControl, FlexrayTpNode, FlexrayTpPdu Pool, *FrameTriggering*, GeneralParameter, GlobalTimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, ▽

▽

△

Class	***Identifiable*** (abstract)			
	△ *HeapUsage*, HwAttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, IPSecRule, IPv6ExtHeader FilterList, ISignalToIPduMapping, ISignalTriggering, *IdentCaption*, InternalTriggeringPoint, J1939Shared AddressCluster, J1939TpNode, Keyword, LifeCycleState, LinScheduleTable, LinTpNode, Linker, Mac MulticastGroup, McDataInstance, MemorySection, ModeDeclaration, ModeDeclarationMapping, Mode SwitchPoint, NetworkEndpoint, *NmCluster*, NmEcu, *NmNode*, NvBlockDescriptor, *PackageableElement*, ParameterAccess, PduToFrameMapping, PduTriggering, PerInstanceMemory, *PhysicalChannel*, Port ElementToCommunicationResourceMapping, PortGroup, *PortInterfaceMapping*, PossibleErrorReaction, ResourceConsumption, RootSwCompositionPrototype, RptComponent, RptContainer, RptExecutable Entity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptServicePoint, RunnableEntity Group, *SdgAttribute*, SdgClass, SecureCommunicationAuthenticationProps, SecureCommunication FreshnessProps, SecurityEventContextProps, *ServerCallPoint*, *ServiceNeeds*, SignalServiceTranslation ElementProps, SignalServiceTranslationEventProps, SignalServiceTranslationProps, SocketAddress, SomeipTpChannel, *SpecElementReference*, *StackUsage*, StaticSocketConnection, StructuredReq, Sw GenericAxisParamType, SwServiceArg, SwcServiceDependency, SwcToApplicationPartitionMapping, SwcToEcuMapping, SwcToImplMapping, SystemMapping, TDCpSoftwareClusterMapping, TDCp SoftwareClusterResourceMapping, TcpOptionFilterList, TimingCondition, *TimingConstraint*, *Timing Description*, TimingExtensionResource, TimingModeInstance, TlsCryptoCipherSuite, Topic1, TpAddress, TraceableTable, TraceableText, *TracedFailure*, *TransformationProps*, TransformationTechnology, Trigger, VariableAccess, VariationPointProxy, ViewMap, VlanConfig, WaitPoint			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
adminData	AdminData	0..1	aggr	This represents the administrative data for the identifiable object. **Tags:**xml.sequenceOffset=-40
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 ▽

▽

△

Class	Identifiable (abstract)			
				△ 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 B.64: Identifiable

Class	ImplementationDataType			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code. **Tags:**atp.recommendedPackage=ImplementationDataTypes			
Base	*ARElement*, *ARObject*, *AbstractImplementationDataType*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *AutosarDataType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dynamicArray SizeProfile	String	0..1	attr	Specifies the profile which the array will follow in case this data type is a variable size array.
isStructWith Optional Element	Boolean	0..1	attr	This attribute is only valid if the attribute category is set to STRUCTURE. If set to True, this attribute indicates that the ImplementationDataType has been created with the intention to define at least one element of the structure as optional.
subElement (ordered)	ImplementationData TypeElement	*	aggr	Specifies an element of an array, struct, or union data type. The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a Implementation DataType representing a structure. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
symbolProps	SymbolProps	0..1	aggr	This represents the SymbolProps for the Implementation DataType. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=symbolProps.shortName
typeEmitter	NameToken	0..1	attr	This attribute is used to control which part of the AUTOSAR toolchain is supposed to trigger data type definitions.

Table B.65: ImplementationDataType

Class	**ImplementationDataTypeElement**			
Package	M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes			
Note	Declares a data object which is locally aggregated. Such an element can only be used within the scope where it is aggregated. This element either consists of further subElements or it is further defined via its swDataDefProps. There are several use cases within the system of ImplementationDataTypes fur such a local declaration: • It can represent the elements of an array, defining the element type and array size • It can represent an element of a struct, defining its type • It can be the local declaration of a debug element.			
Base	*ARObject*, *AbstractImplementationDataTypeElement*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
arrayImplPolicy	ArrayImplPolicyEnum	0..1	attr	This attribute controls the implementation of the payload of an array. It shall only be used if the enclosing ImplementationDataType constitutes an array.
arraySize	PositiveInteger	0..1	attr	The existence of this attributes (if bigger than 0) defines the size of an array and declares that this Implementation DataTypeElement represents the type of each single array element. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
arraySize Handling	ArraySizeHandling Enum	0..1	attr	The way how the size of the array is handled in case of a variable size array.
arraySize Semantics	ArraySizeSemantics Enum	0..1	attr	This attribute controls the meaning of the value of the array size.
isOptional	Boolean	0..1	attr	This attribute represents the ability to declare the enclosing ImplementationDataTypeElement as optional. This means that, at runtime, the ImplementationDataType Element may or may not have a valid value and shall therefore be ignored. The underlying runtime software provides means to set the CppImplementationDataTypeElement as not valid at the sending end of a communication and determine its validity at the receiving end.
subElement (ordered)	ImplementationData TypeElement	*	aggr	Element of an array, struct, or union in case of a nested declaration (i.e. without using "typedefs"). The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a Implementation DataType representing a structure. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
swDataDef Props	SwDataDefProps	0..1	aggr	The properties of this ImplementationDataTypeElement.

Table B.66: ImplementationDataTypeElement

Class	***InternalBehavior*** (abstract)
Package	M2::AUTOSARTemplates::CommonStructure::InternalBehavior
Note	Common base class (abstract) for the internal behavior of both software components and basic software modules/clusters.
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*

▽

△

Class	***InternalBehavior*** (abstract)			
Subclasses	BswInternalBehavior, SwcInternalBehavior			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
constant Memory	ParameterData Prototype	*	aggr	Describes a read only memory object containing characteristic value(s) implemented by this Internal Behavior. The shortName of ParameterDataPrototype has to be equal to the ”C’ identifier of the described constant. The characteristic value(s) might be shared between Sw ComponentPrototypes of the same SwComponentType. The aggregation of constantMemory is subject to variability with the purpose to support variability in the software component or module implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=constantMemory.shortName, constant Memory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
constantValue Mapping	ConstantSpecification MappingSet	*	ref	Reference to the ConstanSpecificationMapping to be applied for the particular InternalBehavior **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=constantValueMapping
dataType Mapping	DataTypeMappingSet	*	ref	Reference to the DataTypeMapping to be applied for the particular InternalBehavior **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=dataTypeMapping
exclusiveArea	ExclusiveArea	*	aggr	This specifies an ExclusiveArea for this InternalBehavior. The exclusiveArea is local to the component resp. module. The aggregation of ExclusiveAreas is subject to variability. Note: the number of ExclusiveAreas might vary due to the conditional existence of RunnableEntities or BswModuleEntities. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=exclusiveArea.shortName, exclusive Area.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
exclusiveArea NestingOrder	ExclusiveAreaNesting Order	*	aggr	This represents the set of ExclusiveAreaNestingOrder owned by the InternalBehavior. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=exclusiveAreaNestingOrder.shortName, exclusiveAreaNestingOrder.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
staticMemory	VariableDataPrototype	*	aggr	Describes a read and writeable static memory object representing measurerment variables implemented by this software component. The term "static" is used in the meaning of "non-temporary" and does not necessarily specify a linker encapsulation. This kind of memory is only supported if supportsMultipleInstantiation is FALSE. The shortName of the VariableDataPrototype has to be equal with the ”C’ identifier of the described variable. ▽

▽

△

Class	InternalBehavior (abstract)			
				△ The aggregation of staticMemory is subject to variability with the purpose to support variability in the software component's implementations. Typically different algorithms in the implementation are requiring different number of memory objects. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=staticMemory.shortName, staticMemory.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table B.67: InternalBehavior

Class	InternalConstrs			
Package	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
Note	This meta-class represents the ability to express internal constraints.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the constraint. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
maxDiff	Numerical	0..1	attr	Maximum difference that is permitted between two consecutive values if the constraint is applied to an axis. **Tags:**xml.sequenceOffset=60
maxGradient	Numerical	0..1	attr	This element specifies the maximum slope that may be used in maps and curves. **Tags:**xml.sequenceOffset=50
monotony	MonotonyEnum	0..1	attr	This element specifies the monotony characteristics of the current internal or physical limits. The following table shows the monotony characteristics which are to be filled through the corresponding values. If the element has no contents or if it is omitted, "no Monotony" is the default content. **Tags:**xml.sequenceOffset=70
scaleConstr (ordered)	ScaleConstr	*	aggr	This is one particular scale which contributes to the data constraints. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=40 xml.typeElement=false xml.typeWrapperElement=false
upperLimit	Limit	0..1	attr	This specifies the upper limit defined by the constraint. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=30

Table B.68: InternalConstrs

Class	InternalTriggerOccurredEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
Note	The event is raised when the referenced internal trigger have been occurred.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
eventSource	InternalTriggeringPoint	0..1	ref	Internal Triggering Point that triggers the event.

Table B.69: InternalTriggerOccurredEvent

Class	InvalidationPolicy			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Specifies whether the component can actively invalidate a particular dataElement. If no invalidationPolicy points to a dataElement this is considered to yield the identical result as if the handleInvalid attribute was set to dontInvalidate.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
dataElement	VariableDataPrototype	0..1	ref	Reference to the dataElement for which the Invalidation Policy applies.
handleInvalid	HandleInvalidEnum	0..1	attr	This attribute controls how invalidation is applied to the dataElement.

Table B.70: InvalidationPolicy

Class	McDataInstance			
Package	M2::AUTOSARTemplates::CommonStructure::MeasurementCalibrationSupport			
Note	Describes the specific properties of one data instance in order to support measurement and/or calibration of this data instance. The most important attributes are: • Its shortName is copied from the ECU Flat map (if applicable) and will be used as identifier and for display by the MC system. • The category is copied from the corresponding data type (ApplicationDataType if defined, otherwise ImplementationDataType) as far as applicable. • The symbol is the one used in the programming language. It will be used to find out the actual memory address by the final generation tool with the help of linker generated information. It is assumed that in the M1 model this part and all the aggregated and referred elements (with the exception of the Flat Map and the references from ImplementationElementInParameterInstanceRef and McAccessDetails) are completely generated from "upstream" information. This means, that even if an element like e.g. a CompuMethod is only used via reference here, it will be copied into the M1 artifact which holds the complete McSupportData for a given Implementation.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
arraySize	PositiveInteger	0..1	attr	The existence of this attribute turns the data instance into an array of data. The attribute determines the size of the array in terms of number of elements.
displayIdentifier	McdIdentifier	0..1	attr	An optional attribute to be used to set the ASAM ASAP2 DISPLAY_IDENTIFIER attribute.

▽

△

Class	**McDataInstance**			
flatMapEntry	FlatInstanceDescriptor	0..1	ref	Reference to the corresponding entry in the ECU Flat Map. This allows to trace back to the original specification of the generated data instance. This link shall be added by the RTE generator mainly for documentation purposes. The reference is optional because • The McDataInstance may represent an array or struct in which only the subElements correspond to FlatMap entries. • The McDataInstance may represent a task local buffer for rapid prototyping access which is different from the "main instance" used for measurement access.
instanceIn Memory	ImplementationElement InParameterInstance Ref	0..1	aggr	Reference to the corresponding data instance in the description of calibration data structures published by the RTE generator. This is used to support emulation methods inside the ECU, it is not required for A2L generation.
mcDataAccess Details	McDataAccessDetails	0..1	aggr	Refers to "upstream" information on how the RTE uses this data instance. Use Case: Rapid Prototyping
mcData Assignment	RoleBasedMcData Assignment	*	aggr	An assignment between McDataInstances. This supports the indication of related McDataElement implementing the of "RP global buffer", "RP global measurement buffer", "RP enabler flag".
resulting Properties	SwDataDefProps	0..1	aggr	These are the generated properties resulting from decisions taken by the RTE generator for the actually implemented data instance. Only those properties are relevant here, which are needed for the measurement and calibration system.
resultingRptSw Prototyping Access	RptSwPrototyping Access	0..1	aggr	Describes the implemented accessibility of data and modes by the rapid prototyping tooling.
role	Identifier	0..1	attr	An optional attribute to be used for additional information on the role of this data instance, for example in the context of rapid prototyping.
rptImplPolicy	RptImplPolicy	0..1	aggr	Describes the implemented code preparation for rapid prototyping at data accesses for a hook based bypassing.
subElement (ordered)	McDataInstance	*	aggr	This relation indicates, that the target element is part of a "struct" which is given by the source element. This information will be used by the final generator to set up the correct addressing scheme. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
symbol	SymbolString	0..1	attr	This String is used to determine the memory address during final generation of the MC configuration data (e.g. "A2L" file) . It shall be the name of the element in the programming language such that it can be identified in linker generated information. In case the McDataInstance is part of composite data in the programming language, the symbol String may include parts denoting the element context, unless the context is given by the symbol attribute of an enclosing McDataInstance. This means in particular for the C language that the "." character shall be used as a separator between the name of a "struct" variable the name of one of its elements. ▽

▽

△

Class	McDataInstance			
				△ The symbol can differ from the shortName in case of generated C data declarations. It is an optional attribute since it may be missing in case the instance represents an element (e.g. a single array element) which has no name in the linker map. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=symbol

Table B.71: McDataInstance

Class	McSupportData			
Package	M2::AUTOSARTemplates::CommonStructure::MeasurementCalibrationSupport			
Note	Root element for all measurement and calibration support data related to one Implementation artifact on an ECU. There shall be one such element related to the RTE implementation (if it owns MC data) and a separate one for each module or component, which owns private MC data.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
emulation Support	McSwEmulationMethod Support	*	aggr	Describes the calibration method used by the RTE. This information is not needed for A2L generation, but to setup software emulation in the ECU. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
mcParameter Instance	McDataInstance	*	aggr	A data instance to be used for calibration. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=mcParameterInstance.shortName, mc ParameterInstance.variationPoint.shortLabel vh.latestBindingTime=postBuild
mcVariable Instance	McDataInstance	*	aggr	A data instance to be used for measurement. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=mcVariableInstance.shortName, mcVariable Instance.variationPoint.shortLabel vh.latestBindingTime=postBuild
measurable System ConstantValues	SwSystemconstant ValueSet	*	ref	Sets of system constant values to be transferred to the MCD system, because the system constants have been specified with "swCalibrationAccess" = readonly.
rptSupportData	RptSupportData	0..1	aggr	The rapid prototyping support data belonging to this implementation. The aggregtion is <<atpSplitable>> because in case of an already exisiting BSW Implementation model, this description will be added later in the process, namely at code generation time. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=rptSupportData

Table B.72: McSupportData

Class	ModeAccessPoint
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ModeDeclarationGroup
Note	A ModeAccessPoint is required by a RunnableEntity owned by a Mode Manager or Mode User. Its semantics implies the ability to access the current mode (provided by the RTE) of a ModeDeclaration GroupPrototype's ModeDeclarationGroup.

▽

△

Class	ModeAccessPoint			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
ident	ModeAccessPointIdent	0..1	aggr	The aggregation in the role ident provides the ability to make the ModeAccessPoint identifiable. From the semantical point of view, the ModeAccessPoint is considered a first-class Identifiable and therefore the aggregation in the role ident shall always exist (until it may be possible to let ModeAccessPoint directly inherit from Identifiable). **Stereotypes:** atpIdentityContributor **Tags:** atp.Status=shallBecomeMandatory xml.sequenceOffset=-100
modeGroup	ModeDeclarationGroup Prototype	0..1	iref	The mode declaration group that is accessed by this runnable. **Tags:**xml.typeElement=true **InstanceRef implemented by:**ModeGroupInAtomicSwc InstanceRef

Table B.73: ModeAccessPoint

Class	ModeDeclaration			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
value	PositiveInteger	0..1	attr	The RTE shall take the value of this attribute for generating the source code representation of this Mode Declaration.

Table B.74: ModeDeclaration

Class	ModeDeclarationGroup			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	A collection of Mode Declarations. Also, the initial mode is explicitly identified. **Tags:**atp.recommendedPackage=ModeDeclarationGroups			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
initialMode	ModeDeclaration	0..1	ref	The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred.
mode Declaration	ModeDeclaration	*	aggr	The ModeDeclarations collected in this ModeDeclaration Group. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=blueprintDerivationTime
modeManager ErrorBehavior	ModeErrorBehavior	0..1	aggr	This represents the ability to define the error behavior expected by the mode manager in case of errors on the mode user side (e.g. terminated mode user).
modeTransition	ModeTransition	*	aggr	This represents the avaliable ModeTransitions of the ModeDeclarationGroup

▽

△

Class	ModeDeclarationGroup			
modeUserError Behavior	ModeErrorBehavior	0..1	aggr	This represents the definition of the error behavior expected by the mode user in case of errors on the mode manager side (e.g. terminated mode manager).
onTransition Value	PositiveInteger	0..1	attr	The value of this attribute shall be taken into account by the RTE generator for programmatically representing a value used for the transition between two statuses.

Table B.75: ModeDeclarationGroup

Class	ModeDeclarationGroupPrototype			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	The ModeDeclarationGroupPrototype specifies a set of Modes (ModeDeclarationGroup) which is provided or required in the given context.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
swCalibration Access	SwCalibrationAccess Enum	0..1	attr	This allows for specifying whether or not the enclosing ModeDeclarationGroupPrototype can be measured at run-time.
type	ModeDeclarationGroup	0..1	tref	The "collection of ModeDeclarations" (= ModeDeclaration Group) supported by a component **Stereotypes:** isOfType

Table B.76: ModeDeclarationGroupPrototype

Class	ModeDeclarationMappingSet			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	This meta-class implements a container for ModeDeclarationGroupMappings **Tags:**atp.recommendedPackage=PortInterfaceMappingSets			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
mode Declaration Mapping	ModeDeclaration Mapping	*	aggr	This represents the collection of ModeDeclaration Mappings owned by the enclosing ModeDeclaration MappingSet.

Table B.77: ModeDeclarationMappingSet

Class	ModeErrorBehavior			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	This represents the ability to define the error behavior in the context of mode handling.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
defaultMode	ModeDeclaration	0..1	ref	This represents the ModeDeclaration that is considered the error mode in the context of the enclosing Mode DeclarationGroup.
errorReaction Policy	ModeErrorReaction PolicyEnum	0..1	attr	This represents the ability to define the policy in terms of which default model shall apply in case an error occurs.

Table B.78: ModeErrorBehavior

Class	**ModeSwitchInterface**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A mode switch interface declares a ModeDeclarationGroupPrototype to be sent and received. **Tags:**atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
modeGroup	ModeDeclarationGroup Prototype	0..1	aggr	The ModeDeclarationGroupPrototype of this mode interface.

Table B.79: ModeSwitchInterface

Class	**ModeSwitchPoint**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ModeDeclarationGroup			
Note	A ModeSwitchPoint is required by a RunnableEntity owned a Mode Manager. Its semantics implies the ability to initiate a mode switch.			
Base	*ARObject*, *AbstractAccessPoint*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
modeGroup	ModeDeclarationGroup Prototype	0..1	iref	The mode declaration group that is switched by this runnable. **InstanceRef implemented by:**PModeGroupInAtomic SwcInstanceRef

Table B.80: ModeSwitchPoint

Class	**ModeTransition**			
Package	M2::AUTOSARTemplates::CommonStructure::ModeDeclaration			
Note	This meta-class represents the ability to describe possible ModeTransitions in the context of a Mode DeclarationGroup.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
enteredMode	ModeDeclaration	0..1	ref	This represents the entered model of the ModeTransition.
exitedMode	ModeDeclaration	0..1	ref	This represents the exited mode of the ModeTransition

Table B.81: ModeTransition

Enumeration	**MonotonyEnum**
Package	M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::PrimitiveTypes
Note	This enumerator denotes the values for specification of monotony for e.g. curves.
Literal	***Description***
decreasing	This indicates that the related curve needs to be monotony decreasing. **Tags:**atp.EnumerationLiteralIndex=0
increasing	This indicates that the related curve needs to be monotony increasing. **Tags:**atp.EnumerationLiteralIndex=1
monotonous	This indicates that the values shall be monotonously decreasing or increasing, depending on the trend set by the first values of the series. **Tags:**atp.EnumerationLiteralIndex=2

▽

△

Enumeration	**MonotonyEnum**
noMonotony	This indicates that the related curve needs not to be monotony. **Tags:**atp.EnumerationLiteralIndex=3
strictlyDecreasing	This indicates that the related curve needs to be strictly monotony decreasing. **Tags:**atp.EnumerationLiteralIndex=4
strictlyIncreasing	This indicates that the related curve needs to be strictly monotony increasing. **Tags:**atp.EnumerationLiteralIndex=5
strictMonotonous	This indicates that the values shall be strict monotonously decreasing or increasing, depending on the trend set by the first values of the series. **Tags:**atp.EnumerationLiteralIndex=6

Table B.82: MonotonyEnum

Class	**NonqueuedReceiverComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes specific to non-queued receiving.			
Base	*ARObject*, *RPortComSpec*, *ReceiverComSpec*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
aliveTimeout	TimeValue	0..1	attr	Specify the amount of time (in seconds) after which the software component (via the RTE) needs to be notified if the corresponding data item have not been received according to the specified timing description. If the aliveTimeout attribute is 0 no timeout monitoring shall be performed.
enableUpdate	Boolean	0..1	attr	This attribute controls whether application code is entitled to check whether the value of the corresponding Variable DataPrototype has been updated.
filter	DataFilter	0..1	aggr	The applicable filter algorithm for filtering the value of the corresponding dataElement.
handleData Status	Boolean	0..1	attr	If this attribute is set to true than the Rte_IStatus API shall exist. If the attribute does not exist or is set to false then the Rte_IStatus API may still exist in response to the existence of further conditions.
handleNever Received	Boolean	0..1	attr	This attribute specifies whether for the corresponding VariableDataPrototype the "never received" flag is available. If yes, the RTE is supposed to assume that initially the VariableDataPrototype has not been received before. After the first reception of the corresponding VariableDataPrototype the flag is cleared. • If the value of this attribute is set to "true" the flag is required. • If set to "false", the RTE shall not support the "never received" functionality for the corresponding VariableDataPrototype.
handleTimeout Type	HandleTimeoutEnum	0..1	attr	This attribute controls the behavior with respect to the handling of timeouts.
initValue	ValueSpecification	0..1	aggr	Initial value to be used in case the sending component is not yet initialized. If the sender also specifies an initial value the receiver's value will be used.
timeout Substitution Value	ValueSpecification	0..1	aggr	This attribute represents the substitution value applicable in the case of a timeout.

Table B.83: NonqueuedReceiverComSpec

Class	**NonqueuedSenderComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes for non-queued sender/receiver communication (sender side)			
Base	*ARObject*, *PPortComSpec*, *SenderComSpec*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
dataFilter	DataFilter	0..1	aggr	The applicable filter algorithm for filtering the value of the corresponding dataElement.
initValue	ValueSpecification	0..1	aggr	Initial value to be sent if sender component is not yet fully initialized, but receiver needs data already.

Table B.84: NonqueuedSenderComSpec

Class	**NvBlockNeeds**			
Package	M2::AUTOSARTemplates::CommonStructure::ServiceNeeds			
Note	Specifies the abstract needs on the configuration of a single NVRAM Block.			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServiceNeeds*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
calcRamBlock Crc	Boolean	0..1	attr	Defines if CRC (re)calculation for the permanent RAM Block is required.
checkStatic BlockId	Boolean	0..1	attr	Defines if the Static Block Id check shall be enabled.
cyclicWriting Period	TimeValue	0..1	attr	This represents the period for cyclic writing of NvData to store the associated RAM Block.
nDataSets	PositiveInteger	0..1	attr	Number of data sets to be provided by the NVRAM manager for this block. This is the total number of ROM Blocks and RAM Blocks.
nRomBlocks	PositiveInteger	0..1	attr	Number of ROM Blocks to be provided by the NVRAM manager for this block. Please note that these multiple ROM Blocks are given in a contiguous area.
ramBlockStatus Control	RamBlockStatusControl Enum	0..1	attr	This attribute defines how the management of the RAM Block status is controlled.
readonly	Boolean	0..1	attr	True: data of this NVRAM Block are write protected for normal operation (but protection can be disabled) false: no restriction
reliability	NvBlockNeeds ReliabilityEnum	0..1	attr	Reliability against data loss on the non-volatile medium.
resistantTo ChangedSw	Boolean	0..1	attr	Defines whether an NVRAM Block shall be treated resistant to configuration changes (true) or not (false). For details how to handle initialization in the latter case, please refer to the NVRAM specification.
restoreAtStart	Boolean	0..1	attr	Defines whether the associated RAM Block shall be implicitly restored during startup by the basic software.
selectBlockFor FirstInitAll	Boolean	0..1	attr	If this attribute is set to true the NvM shall process this block in the NvM_FirstInitAll() function.
storeAt Shutdown	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored during shutdown by the basic software.
storeCyclic	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored periodically by the basic software.
store Emergency	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored in case of ECU failure (e.g. loss of power) by the basic software. If the attribute store Emergency is set to true the associated RAM Block shall be configured to have immediate priority.

△

Class	**NvBlockNeeds**			
storeImmediate	Boolean	0..1	attr	Defines whether or not the associated RAM Block shall be implicitly stored immediately during or after execution of the according SW-C RunnableEntity by the basic software.
useAuto ValidationAt ShutDown	Boolean	0..1	attr	If set to true the RAM Block shall be auto validated during shutdown phase.
useCRCComp Mechanism	Boolean	0..1	attr	If set to true the CRC of the RAM Block shall be compared during a write job with the CRC which was calculated during the last successful read or write job in order to skip unnecessary NVRAM writings.
writeOnlyOnce	Boolean	0..1	attr	Defines write protection after first write: true: This block is prevented from being changed/erased or being replaced with the default ROM data after first initialization by the software-component. false: No such restriction.
writeVerification	Boolean	0..1	attr	Defines if Write Verification shall be enabled for this NVRAM Block.
writing Frequency	PositiveInteger	0..1	attr	Provides the amount of updates to this block from the application point of view. It has to be provided in "number of write access per year".
writingPriority	NvBlockNeedsWriting PriorityEnum	0..1	attr	Requires the priority of writing this block in case of concurrent requests to write other blocks.

Table B.85: NvBlockNeeds

Class	**NvDataInterface**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A non volatile data interface declares a number of VariableDataPrototypes to be exchanged between non volatile block components and atomic software components. **Tags:**atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DataInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
nvData	VariableDataPrototype	*	aggr	The VariableDataPrototype of this nv data interface.

Table B.86: NvDataInterface

Class	**OperationInvokedEvent**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
Note	The OperationInvokedEvent references the ClientServerOperation invoked by the client.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *Multilanguage Referrable*, *RTEEvent*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
operation	ClientServerOperation	0..1	iref	The operation to be executed as the consequence of the event. **InstanceRef implemented by:**POperationInAtomicSwc InstanceRef

Table B.87: OperationInvokedEvent

Class	OsTaskExecutionEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
Note	This RTEEvent is supposed to execute RunnableEntities which have to react on the execution of specific OsTasks. Therefore, this event is unconditionally raised whenever the OsTask on which it is mapped is executed. The main use case for this event is scheduling of Runnables of Complex Drivers which have to react on task executions. **Tags:**atp.Status=draft			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table B.88: OsTaskExecutionEvent

Class	PPortPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port providing a certain port interface.			
Base	*ARObject*, *AbstractProvidedPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
provided Interface	PortInterface	0..1	tref	The interface that this port provides. **Stereotypes:** isOfType

Table B.89: PPortPrototype

Class	ParameterAccess			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::DataElements			
Note	The presence of a ParameterAccess implies that a RunnableEntity needs access to a ParameterData Prototype.			
Base	*ARObject*, *AbstractAccessPoint*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
accessed Parameter	AutosarParameterRef	0..1	aggr	Refernce to the accessed calibration parameter.
swDataDef Props	SwDataDefProps	0..1	aggr	This allows denote instance and access specific properties, mainly input values and common axis.

Table B.90: ParameterAccess

Class	ParameterDataPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	A parameter element used for parameter interface and internal behavior, supporting signal like parameter and characteristic value communication patterns and parameter and characteristic value definition.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the ParameterDataPrototype

Table B.91: ParameterDataPrototype

Class	ParameterInterface			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A parameter interface declares a number of parameter and characteristic values to be exchanged between parameter components and software components. **Tags:**atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DataInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
parameter	ParameterData Prototype	*	aggr	The ParameterDataPrototype of this ParameterInterface.

Table B.92: ParameterInterface

Class	ParameterRequireComSpec			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	"Communication" specification that applies to parameters on the required side of a connection.			
Base	*ARObject*, *RPortComSpec*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
initValue	ValueSpecification	0..1	aggr	The initial value applicable for the corresponding ParameterDataPrototype.
parameter	ParameterData Prototype	0..1	ref	The ParameterDataPrototype to which the Parameter RequireComSpec applies.

Table B.93: ParameterRequireComSpec

Class	PhysConstrs			
Package	M2::MSR::AsamHdo::Constraints::GlobalConstraints			
Note	This meta-class represents the ability to express physical constraints. Therefore it has (in opposite to InternalConstrs) a reference to a Unit.			
Base	*ARObject*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
lowerLimit	Limit	0..1	attr	This specifies the lower limit of the constraint. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=20
maxDiff	Numerical	0..1	attr	Maximum difference that is permitted between two consecutive values if the constraint is applied to an axis. **Tags:**xml.sequenceOffset=60
maxGradient	Numerical	0..1	attr	This element specifies the maximum slope that may be used in curves and maps. **Tags:**xml.sequenceOffset=50
monotony	MonotonyEnum	0..1	attr	This specifies the monotony constraints on the data object. Note that this applies only to curves and maps. **Tags:**xml.sequenceOffset=70

△

Class	**PhysConstrs**			
scaleConstr (ordered)	ScaleConstr	*	aggr	This is one particular scale which contributes to the data constraints. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=40 xml.typeElement=false xml.typeWrapperElement=false
unit	Unit	0..1	ref	This is the unit to which the physical constraints relate to. In particular, it is the physical unit of the specified limits. **Tags:**xml.sequenceOffset=80
upperLimit	Limit	0..1	attr	This specifies the upper limit of the constraint. **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=30

Table B.94: PhysConstrs

Class	**PhysicalDimension**			
Package	M2::MSR::AsamHdo::Units			
Note	This class represents a physical dimension. If the physical dimension of two units is identical, then a conversion between them is possible. The conversion between units is related to the definition of the physical dimension. Note that the equivalence of the exponents does not per se define the convertibility. For example Energy and Torque share the same exponents (Nm). Please note further the value of an exponent does not necessarily have to be an integer number. It is also possible that the value yields a rational number, e.g. to compute the square root of a given physical quantity. In this case the exponent value would be a rational number where the numerator value is 1 and the denominator value is 2. **Tags:**atp.recommendedPackage=PhysicalDimensions			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
currentExp	Numerical	0..1	attr	This attribute represents the exponent of the physical dimension "electric current". **Tags:**xml.sequenceOffset=50
lengthExp	Numerical	0..1	attr	The exponent of the physical dimension "length". **Tags:**xml.sequenceOffset=20
luminous IntensityExp	Numerical	0..1	attr	The exponent of the physical dimension "luminous intensity". **Tags:**xml.sequenceOffset=80
massExp	Numerical	0..1	attr	The exponent of the physical dimension "mass". **Tags:**xml.sequenceOffset=30
molarAmount Exp	Numerical	0..1	attr	The exponent of the physical dimension "quantity of substance". **Tags:**xml.sequenceOffset=70
temperatureExp	Numerical	0..1	attr	The exponent of the physical dimension "temperature". **Tags:**xml.sequenceOffset=60

▽

△

Class	**PhysicalDimension**			
timeExp	Numerical	0..1	attr	The exponent of the physical dimension "time". **Tags:**xml.sequenceOffset=40

Table B.95: PhysicalDimension

Class	**PortElementToCommunicationResourceMapping**			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	This meta class maps a communication resource to CP Software Clusters. In this case the kind of Port Prototype specified whether the Software Cluster has to provide or to require the resource. **Tags:**atp.Status=draft			
Base	*ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
clientServer Operation	ClientServerOperation	0..1	iref	ClientServerOperation instance qualifying the communication resource **Tags:**atp.Status=draft **InstanceRef implemented by:**OperationInSystem InstanceRef
communication Resource	CpSoftwareCluster Communication Resource	0..1	ref	Communication resource for which the mapping applies. **Tags:**atp.Status=draft
mode Declaration GroupPrototype	ModeDeclarationGroup Prototype	0..1	iref	ModeDeclarationGroupPrototype instance qualifying the communication resource **Tags:**atp.Status=draft **InstanceRef implemented by:**ModeDeclarationGroup PrototypeInSystemInstanceRef
parameterData Prototype	ParameterData Prototype	0..1	iref	ParameterDataPrototype instance qualifying the communication resource. **Tags:**atp.Status=draft **InstanceRef implemented by:**ParameterDataPrototype InSystemInstanceRef
trigger	Trigger	0..1	iref	Trigger instance qualifying the communication resource. **Tags:**atp.Status=draft **InstanceRef implemented by:**TriggerInSystemInstance Ref
variableData Prototype	VariableDataPrototype	0..1	iref	VariableDataPrototype instance qualifying the communication resource **Tags:**atp.Status=draft **InstanceRef implemented by:**VariableDataPrototypeIn SystemInstanceRef

Table B.96: PortElementToCommunicationResourceMapping

Class	***PortInterface*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Abstract base class for an interface that is either provided or required by a port of a software component.			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Subclasses	ClientServerInterface, *DataInterface*, ModeSwitchInterface, TriggerInterface			
Attribute	***Type***	***Mult.***	***Kind***	***Note***

▽

△

Class	***PortInterface*** (abstract)			
isService	Boolean	0..1	attr	This flag is set if the PortInterface is to be used for communication between an • ApplicationSwComponentType or • ServiceProxySwComponentType or • SensorActuatorSwComponentType or • ComplexDeviceDriverSwComponentType • ServiceSwComponentType • EcuAbstractionSwComponentType and a ServiceSwComponentType (namely an AUTOSAR Service) located on the same ECU. Otherwise the flag is not set.
serviceKind	ServiceProviderEnum	0..1	attr	This attribute provides further details about the nature of the applied service.

Table B.97: PortInterface

Class	***PortInterfaceMapping*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	Specifies one PortInterfaceMapping to support the connection of Ports typed by two different Port Interfaces with PortInterface elements having unequal names and/or unequal semantic (resolution or range).			
Base	*ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	ClientServerInterfaceMapping, ModeInterfaceMapping, TriggerInterfaceMapping, VariableAndParameter InterfaceMapping			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
–	–	–	–	–

Table B.98: PortInterfaceMapping

Class	***PortPrototype*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Base class for the ports of an AUTOSAR software component. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.			
Base	*ARObject*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	*AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
clientServer Annotation	ClientServerAnnotation	*	aggr	Annotation of this PortPrototype with respect to client/ server communication.
delegatedPort Annotation	DelegatedPort Annotation	0..1	aggr	Annotations on this delegated port.
ioHwAbstraction Server Annotation	IoHwAbstractionServer Annotation	*	aggr	Annotations on this IO Hardware Abstraction port.
modePort Annotation	ModePortAnnotation	*	aggr	Annotations on this mode port.
nvDataPort Annotation	NvDataPortAnnotation	*	aggr	Annotations on this non voilatile data port.
parameterPort Annotation	ParameterPort Annotation	*	aggr	Annotations on this parameter port.

▽

△

Class	***PortPrototype*** (abstract)			
senderReceiver Annotation	SenderReceiver Annotation	*	aggr	Collection of annotations of this ports sender/receiver communication.
triggerPort Annotation	TriggerPortAnnotation	*	aggr	Annotations on this trigger port.

Table B.99: PortPrototype

Class	**QueuedReceiverComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes specific to queued receiving.			
Base	*ARObject*, *RPortComSpec*, *ReceiverComSpec*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
queueLength	PositiveInteger	0..1	attr	Length of queue for received events.

Table B.100: QueuedReceiverComSpec

Class	**RPortPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Components			
Note	Component port requiring a certain port interface.			
Base	*ARObject*, *AbstractRequiredPortPrototype*, *AtpBlueprintable*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *PortPrototype*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
required Interface	PortInterface	0..1	tref	The interface that this port requires. **Stereotypes:** isOfType

Table B.101: RPortPrototype

Class	***RTEEvent*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
Note	Abstract base class for all RTE-related events			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Subclasses	AsynchronousServerCallReturnsEvent, BackgroundEvent, DataReceiveErrorEvent, DataReceivedEvent, DataSendCompletedEvent, DataWriteCompletedEvent, ExternalTriggerOccurredEvent, InitEvent, InternalTriggerOccurredEvent, ModeSwitchedAckEvent, OperationInvokedEvent, OsTaskExecutionEvent, SwcModeManagerErrorEvent, SwcModeSwitchEvent, TimingEvent, TransformerHardErrorEvent			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
disabledMode	ModeDeclaration	*	iref	Reference to the Modes that disable the Event. **Stereotypes:** atpSplitable **Tags:**atp.Splitkey=disabledMode.contextModeDeclaration GroupPrototype, disabledMode.contextPort, disabled Mode.targetModeDeclaration **InstanceRef implemented by:**RModeInAtomicSwc InstanceRef
startOnEvent	RunnableEntity	0..1	ref	RunnableEntity starts when the corresponding RTEEvent occurs.

Table B.102: RTEEvent

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, Bsw VariableAccess, CouplingPortTrafficClassAssignment, DiagnosticDebounceAlgorithmProps, *Diagnostic EnvModeElement*, EthernetPriorityRegeneration, EventHandler, ExclusiveAreaNestingOrder, *Hw DescriptionEntity*, *ImplementationProps*, LinSlaveConfigIdent, ModeTransition, *MultilanguageReferrable*, PduActivationRoutingGroup, PncMappingIdent, *SingleLanguageReferrable*, SoConIPduIdentifier, Socket ConnectionBundle, 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
shortName Fragment	ShortNameFragment	*	aggr	This specifies how the Referrable.shortName is composed of several shortNameFragments. **Tags:**xml.sequenceOffset=-90

Table B.103: Referrable

Class	**RtePluginProps**			
Package	M2::AUTOSARTemplates::CommonStructure::FlatMap			
Note	The properties of a communication graph with respect to the utilization of RTE Implementation Plug-in.			
Base	*ARObject*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
associated CrossSwCluster ComRtePlugin	EcucContainerValue	0..1	ref	This associates a communication graph to a specific RTE Implementation Plug-in handling cross Software Cluster communication.
associatedRte Plugin	EcucContainerValue	0..1	ref	This associates a communication graph to a specific RTE Implementation Plug-in handling local Software Cluster communication or communication in a non-cluster ECU.

Table B.104: RtePluginProps

Class	**RunnableEntity**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior			
Note	A RunnableEntity represents the smallest code-fragment that is provided by an AtomicSwComponent Type and are executed under control of the RTE. RunnableEntities are for instance set up to respond to data reception or operation invocation on a server.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *ExecutableEntity*, *Identifiable*, *Multilanguage Referrable*, *Referrable*			
Attribute	***Type***	***Mult.***	***Kind***	***Note***
argument (ordered)	RunnableEntity Argument	*	aggr	This represents the formal definition of a an argument to a RunnableEntity.
asynchronous ServerCall ResultPoint	AsynchronousServer CallResultPoint	*	aggr	The server call result point admits a runnable to fetch the result of an asynchronous server call. The aggregation of AsynchronousServerCallResultPoint is subject to variability with the purpose to support the ▽

▽

△

Class	**RunnableEntity**			
				△ conditional existence of client server PortPrototypes and the variant existence of server call result points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=asynchronousServerCallResultPoint.shortName, asynchronousServerCallResultPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
canBeInvoked Concurrently	Boolean	0..1	attr	If the value of this attribute is set to "true" the enclosing RunnableEntity can be invoked concurrently (even for one instance of the corresponding AtomicSwComponentType). This implies that it is the responsibility of the implementation of the RunnableEntity to take care of this form of concurrency. Note that the default value of this attribute is set to "false".
dataRead Access	VariableAccess	*	aggr	RunnableEntity has implicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataReadAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataReadAccess in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataReadAccess.shortName, dataReadAccess.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataReceive PointBy Argument	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of an argument in the function signature. The aggregation of dataReceivePointByArgument is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data receive points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataReceivePointByArgument.shortName, dataReceivePointByArgument.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataReceive PointByValue	VariableAccess	*	aggr	RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of the return value. The aggregation of dataReceivePointByValue is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of data receive points in the implementation. ▽

▽

△

Class	**RunnableEntity**			
				△ **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataReceivePointByValue.shortName, data ReceivePointByValue.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataSendPoint	VariableAccess	*	aggr	RunnableEntity has explicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataSendPoint is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data send points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataSendPoint.shortName, dataSend Point.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
dataWrite Access	VariableAccess	*	aggr	RunnableEntity has implicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The aggregation of dataWriteAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataWriteAccess in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=dataWriteAccess.shortName, dataWrite Access.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
external TriggeringPoint	ExternalTriggeringPoint	*	aggr	The aggregation of ExternalTriggeringPoint is subject to variability with the purpose to support the conditional existence of trigger ports or the variant existence of external triggering points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=externalTriggeringPoint.ident.shortName, externalTriggeringPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
internal TriggeringPoint	InternalTriggeringPoint	*	aggr	The aggregation of InternalTriggeringPoint is subject to variability with the purpose to support the variant existence of internal triggering points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=internalTriggeringPoint.shortName, internal TriggeringPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
modeAccess Point	ModeAccessPoint	*	aggr	The runnable has a mode access point. The aggregation of ModeAccessPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode access points in the implementation. ▽

▽

△

Class	**RunnableEntity**			
				△ **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeAccessPoint.ident.shortName, mode AccessPoint.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
modeSwitch Point	ModeSwitchPoint	*	aggr	The runnable has a mode switch point. The aggregation of ModeSwitchPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode switch points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=modeSwitchPoint.shortName, modeSwitch Point.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
parameter Access	ParameterAccess	*	aggr	The presence of a ParameterAccess implies that a RunnableEntity needs read only access to a Parameter DataPrototype which may either be local or within a Port Prototype. The aggregation of ParameterAccess is subject to variability with the purpose to support the conditional existence of parameter ports and component local parameters as well as the variant existence of Parameter Access (points) in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=parameterAccess.shortName, parameter Access.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
readLocal Variable	VariableAccess	*	aggr	The presence of a readLocalVariable implies that a RunnableEntity needs read access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable. The aggregation of readLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of read LocalVariable (points) in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=readLocalVariable.shortName, readLocal Variable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
serverCallPoint	ServerCallPoint	*	aggr	The RunnableEntity has a ServerCallPoint. The aggregation of ServerCallPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes or the variant existence of server call points in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=serverCallPoint.shortName, serverCall Point.variationPoint.shortLabel vh.latestBindingTime=preCompileTime
symbol	CIdentifier	0..1	attr	The symbol describing this RunnableEntity's entry point. This is considered the API of the RunnableEntity and is required during the RTE contract phase.

▽

△

Class	**RunnableEntity**			
waitPoint	WaitPoint	*	aggr	The WaitPoint associated with the RunnableEntity.
writtenLocal Variable	VariableAccess	*	aggr	The presence of a writtenLocalVariable implies that a RunnableEntity needs write access to a VariableData Prototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable. The aggregation of writtenLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicit InterRunnableVariable or the variant existence of written LocalVariable (points) in the implementation. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=writtenLocalVariable.shortName, written LocalVariable.variationPoint.shortLabel vh.latestBindingTime=preCompileTime

Table B.105: RunnableEntity

Class	**SenderReceiverInterface**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A sender/receiver interface declares a number of data elements to be sent and received. **Tags:**atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *DataInterface*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
dataElement	VariableDataPrototype	*	aggr	The data elements of this SenderReceiverInterface.
invalidation Policy	InvalidationPolicy	*	aggr	InvalidationPolicy for a particular dataElement
metaDataItem Set	MetaDataItemSet	*	aggr	This aggregation defines fixed sets of meta-data items associated with dataElements of the enclosing Sender ReceiverInterface

Table B.106: SenderReceiverInterface

Class	***ServerCallPoint*** (abstract)			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServerCall			
Note	If a RunnableEntity owns a ServerCallPoint it is entitled to invoke a particular ClientServerOperation of a specific RPortPrototype of the corresponding AtomicSwComponentType			
Base	*ARObject*, *AbstractAccessPoint*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Subclasses	AsynchronousServerCallPoint, SynchronousServerCallPoint			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
operation	ClientServerOperation	0..1	iref	The operation that is called by this runnable. **InstanceRef implemented by:**ROperationInAtomicSwc InstanceRef
timeout	TimeValue	0..1	attr	Time in seconds before the server call times out and returns with an error message. It depends on the call type (synchronous or asynchronous) how this is reported.

Table B.107: ServerCallPoint

Class	**ServerComSpec**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Communication			
Note	Communication attributes for a server port (PPortPrototype and ClientServerInterface).			
Base	*ARObject*, *PPortComSpec*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
operation	ClientServerOperation	0..1	ref	Operation these communication attributes apply to.
queueLength	PositiveInteger	0..1	attr	Length of call queue on the server side. The queue is implemented by the RTE. The value shall be greater or equal to 1. Setting the value of queueLength to 1 implies that incoming requests are rejected while another request that arrived earlier is being processed.
transformation ComSpecProps	TransformationCom SpecProps	*	aggr	This references the TransformationComSpecProps which define port-specific configuration for data transformation.

Table B.108: ServerComSpec

Class	**SubElementMapping**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	This meta-class allows for the definition of mappings of elements of a composite data type.			
Base	*ARObject*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
firstElement	SubElementRef	0..1	aggr	This represents the first element referenced in the scope of the mapping. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
secondElement	SubElementRef	0..1	aggr	This represents the second element referenced in the scope of the mapping. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
textTable Mapping	TextTableMapping	0..2	aggr	This allows for the text-table translation of individual elements of a composite data type.

Table B.109: SubElementMapping

Class	**SwBaseType**			
Package	M2::MSR::AsamHdo::BaseTypes			
Note	This meta-class represents a base type used within ECU software. **Tags:**atp.recommendedPackage=BaseTypes			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *BaseType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
–	–	–	–	–

Table B.110: SwBaseType

Enumeration	**SwCalibrationAccessEnum**
Package	M2::MSR::DataDictionary::DataDefProperties
Note	Determines the access rights to a data object w.r.t. measurement and calibration.
Literal	*Description*

▽

△

Enumeration	**SwCalibrationAccessEnum**
notAccessible	The element will not be accessible via MCD tools, i.e. will not appear in the ASAP file. **Tags:**atp.EnumerationLiteralIndex=0
readOnly	The element will only appear as read-only in an ASAP file. **Tags:**atp.EnumerationLiteralIndex=1
readWrite	The element will appear in the ASAP file with both read and write access. **Tags:**atp.EnumerationLiteralIndex=2

Table B.111: SwCalibrationAccessEnum

Class	**SwComponentPrototype**			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Composition			
Note	Role of a software component within a composition.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
type	SwComponentType	0..1	tref	Type of the instance. **Stereotypes:** isOfType

Table B.112: SwComponentPrototype

Class	<<atpVariation>> **SwDataDefProps**			
Package	M2::MSR::DataDictionary::DataDefProperties			
Note	This class is a collection of properties relevant for data objects under various aspects. One could consider this class as a "pattern of inheritance by aggregation". The properties can be applied to all objects of all classes in which SwDataDefProps is aggregated. Note that not all of the attributes or associated elements are useful all of the time. Hence, the process definition (e.g. expressed with an OCL or a Document Control Instance MSR-DCI) has the task of implementing limitations. SwDataDefProps covers various aspects: • Structure of the data element for calibration use cases: is it a single value, a curve, or a map, but also the recordLayouts which specify how such elements are mapped/converted to the Data Types in the programming language (or in AUTOSAR). This is mainly expressed by properties like swRecordLayout and swCalprmAxisSet • Implementation aspects, mainly expressed by swImplPolicy, swVariableAccessImplPolicy, sw AddrMethod, swPointerTagetProps, baseType, implementationDataType and additionalNative TypeQualifier • Access policy for the MCD system, mainly expressed by swCalibrationAccess • Semantics of the data element, mainly expressed by compuMethod and/or unit, dataConstr, invalidValue • Code generation policy provided by swRecordLayout **Tags:**vh.latestBindingTime=codeGenerationTime			
Base	*ARObject*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*

▽

△

Class	<<atpVariation>> **SwDataDefProps**			
additionalNative TypeQualifier	NativeDeclarationString	0..1	attr	This attribute is used to declare native qualifiers of the programming language which can neither be deduced from the baseType (e.g. because the data object describes a pointer) nor from other more abstract attributes. Examples are qualifiers like "volatile", "strict" or "enum" of the C-language. All such declarations have to be put into one string. **Tags:**xml.sequenceOffset=235
annotation	Annotation	*	aggr	This aggregation allows to add annotations (yellow pads ...) related to the current data object. **Tags:** xml.roleElement=true xml.roleWrapperElement=true xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false
baseType	SwBaseType	0..1	ref	Base type associated with the containing data object. **Tags:**xml.sequenceOffset=50
compuMethod	CompuMethod	0..1	ref	Computation method associated with the semantics of this data object. **Tags:**xml.sequenceOffset=180
dataConstr	DataConstr	0..1	ref	Data constraint for this data object. **Tags:**xml.sequenceOffset=190
displayFormat	DisplayFormatString	0..1	attr	This property describes how a number is to be rendered e.g. in documents or in a measurement and calibration system. **Tags:**xml.sequenceOffset=210
display Presentation	DisplayPresentation Enum	0..1	attr	This attribute controls the presentation of the related data for measurement and calibration tools.
implementation DataType	AbstractImplementation DataType	0..1	ref	This association denotes the ImplementationDataType of a data declaration via its aggregated SwDataDefProps. It is used whenever a data declaration is not directly referring to a base type. Especially • redefinition of an ImplementationDataType via a "typedef" to another ImplementationDatatype • the target type of a pointer (see SwPointerTarget Props), if it does not refer to a base type directly • the data type of an array or record element within an ImplementationDataType, if it does not refer to a base type directly • the data type of an SwServiceArg, if it does not refer to a base type directly **Tags:**xml.sequenceOffset=215
invalidValue	ValueSpecification	0..1	aggr	Optional value to express invalidity of the actual data element. **Tags:**xml.sequenceOffset=255
stepSize	Float	0..1	attr	This attribute can be used to define a value which is added to or subtracted from the value of a DataPrototype when using up/down keys while calibrating.

▽

△

Class	<<atpVariation>> **SwDataDefProps**			
swAddrMethod	SwAddrMethod	0..1	ref	Addressing method related to this data object. Via an association to the same SwAddrMethod it can be specified that several DataPrototypes shall be located in the same memory without already specifying the memory section itself. **Tags:**xml.sequenceOffset=30
swAlignment	AlignmentType	0..1	attr	The attribute describes the intended alignment of the DataPrototype. If the attribute is not defined the alignment is determined by the swBaseType size and the memory AllocationKeywordPolicy of the referenced SwAddr Method. **Tags:**xml.sequenceOffset=33
swBit Representation	SwBitRepresentation	0..1	aggr	Description of the binary representation in case of a bit variable. **Tags:**xml.sequenceOffset=60
swCalibration Access	SwCalibrationAccess Enum	0..1	attr	Specifies the read or write access by MCD tools for this data object. **Tags:**xml.sequenceOffset=70
swCalprmAxis Set	SwCalprmAxisSet	0..1	aggr	This specifies the properties of the axes in case of a curve or map etc. This is mainly applicable to calibration parameters. **Tags:**xml.sequenceOffset=90
swComparison Variable	SwVariableRefProxy	*	aggr	Variables used for comparison in an MCD process. **Tags:** xml.sequenceOffset=170 xml.typeElement=false
swData Dependency	SwDataDependency	0..1	aggr	Describes how the value of the data object has to be calculated from the value of another data object (by the MCD system). **Tags:**xml.sequenceOffset=200
swHostVariable	SwVariableRefProxy	0..1	aggr	Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects. **Tags:** xml.sequenceOffset=220 xml.typeElement=false
swImplPolicy	SwImplPolicyEnum	0..1	attr	Implementation policy for this data object. **Tags:**xml.sequenceOffset=230
swIntended Resolution	Numerical	0..1	attr	The purpose of this element is to describe the requested quantization of data objects early on in the design process. The resolution ultimately occurs via the conversion formula present (compuMethod), which specifies the transition from the physical world to the standardized world (and vice-versa) (here, "the slope per bit" is present implicitly in the conversion formula). In the case of a development phase without a fixed conversion formula, a pre-specification can occur through swIntendedResolution. The resolution is specified in the physical domain according to the property "unit". **Tags:**xml.sequenceOffset=240

▽

△

Class	<<atpVariation>> **SwDataDefProps**			
swInterpolation Method	Identifier	0..1	attr	This is a keyword identifying the mathematical method to be applied for interpolation. The keyword needs to be related to the interpolation routine which needs to be invoked. **Tags:**xml.sequenceOffset=250
swIsVirtual	Boolean	0..1	attr	This element distinguishes virtual objects. Virtual objects do not appear in the memory, their derivation is much more dependent on other objects and hence they shall have a swDataDependency . **Tags:**xml.sequenceOffset=260
swPointerTarget Props	SwPointerTargetProps	0..1	aggr	Specifies that the containing data object is a pointer to another data object. **Tags:**xml.sequenceOffset=280
swRecord Layout	SwRecordLayout	0..1	ref	Record layout for this data object. **Tags:**xml.sequenceOffset=290
swRefresh Timing	MultidimensionalTime	0..1	aggr	This element specifies the frequency in which the object involved shall be or is called or calculated. This timing can be collected from the task in which write access processes to the variable run. But this cannot be done by the MCD system. So this attribute can be used in an early phase to express the desired refresh timing and later on to specify the real refresh timing. **Tags:**xml.sequenceOffset=300
swTextProps	SwTextProps	0..1	aggr	the specific properties if the data object is a text object. **Tags:**xml.sequenceOffset=120
swValueBlock Size	Numerical	0..1	attr	This represents the size of a Value Block **Stereotypes:** atpVariation **Tags:** vh.latestBindingTime=preCompileTime xml.sequenceOffset=80
swValueBlock SizeMult (ordered)	Numerical	*	attr	This attribute is used to specify the dimensions of a value block (VAL_BLK) for the case that that value block has more than one dimension. The dimensions given in this attribute are ordered such that the first entry represents the first dimension, the second entry represents the second dimension, and so on. For one-dimensional value blocks the attribute swValue BlockSize shall be used and this attribute shall not exist. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=preCompileTime
unit	Unit	0..1	ref	Physical unit associated with the semantics of this data object. This attribute applies if no compuMethod is specified. If both units (this as well as via compuMethod) are specified the units shall be compatible. **Tags:**xml.sequenceOffset=350

▽

△

Class	<<atpVariation>> **SwDataDefProps**			
valueAxisData Type	ApplicationPrimitive DataType	0..1	ref	The referenced ApplicationPrimitiveDataType represents the primitive data type of the value axis within a compound primitive (e.g. curve, map). It supersedes CompuMethod, Unit, and BaseType. **Tags:**xml.sequenceOffset=355

Table B.113: SwDataDefProps

Enumeration	**SwImplPolicyEnum**
Package	M2::MSR::DataDictionary::DataDefProperties
Note	Specifies the implementation strategy with respect to consistency mechanisms of variables.
Literal	*Description*
const	forced implementation such that the running software within the ECU shall not modify it. For example implemented with the "const" modifier in C. This can be applied for parameters (not for those in NVRAM) as well as argument data prototypes. **Tags:**atp.EnumerationLiteralIndex=0
fixed	This data element is fixed. In particular this indicates, that it might also be implemented e.g. as in place data, (#DEFINE). **Tags:**atp.EnumerationLiteralIndex=1
measurementPoint	The data element is created for measurement purposes only. The data element is never read directly within the ECU software. In contrast to a "standard" data element in an unconnected provide port is, this unconnection is guaranteed for measurementPoint data elements. **Tags:**atp.EnumerationLiteralIndex=2
queued	The content of the data element is queued and the data element has 'event' semantics, i.e. data elements are stored in a queue and all data elements are processed in 'first in first out' order. The queuing is intended to be implemented by RTE Generator. This value is not applicable for parameters. **Tags:**atp.EnumerationLiteralIndex=3
standard	This is applicable for all kinds of data elements. For variable data prototypes the 'last is best' semantics applies. For parameter there is no specific implementation directive. **Tags:**atp.EnumerationLiteralIndex=4

Table B.114: SwImplPolicyEnum

Class	**SwRecordLayout**			
Package	M2::MSR::DataDictionary::RecordLayout			
Note	Defines how the data objects (variables, calibration parameters etc.) are to be stored in the ECU memory. As an example, this definition specifies the sequence of axis points in the ECU memory. Iterations through axis values are stored within the sub-elements swRecordLayoutGroup. **Tags:**atp.recommendedPackage=SwRecordLayouts			
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*			
Attribute	*Type*	*Mult.*	*Kind*	*Note*
swRecord LayoutGroup	SwRecordLayoutGroup	0..1	aggr	This is the top level record layout group. **Tags:** xml.roleElement=true xml.roleWrapperElement=false xml.sequenceOffset=20 xml.typeElement=false xml.typeWrapperElement=false

Table B.115: SwRecordLayout

Class	SwcModeSwitchEvent			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents			
Note	This event is raised upon a received mode change.			
Base	*ARObject*, *AbstractEvent*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *RTEEvent*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
activation	ModeActivationKind	0..1	attr	Specifies if the event is activated on entering or exiting the referenced Mode.
mode (ordered)	ModeDeclaration	0..2	iref	Reference to one or two Modes that initiate the SwcMode SwitchEvent. **InstanceRef implemented by:**RModeInAtomicSwcInstanceRef

Table B.116: SwcModeSwitchEvent

Class	SynchronousServerCallPoint			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::ServerCall			
Note	This means that the RunnableEntity is supposed to perform a blocking wait for a response from the server.			
Base	*ARObject*, *AbstractAccessPoint*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*, *ServerCallPoint*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
calledFrom WithinExclusive Area	ExclusiveAreaNesting Order	0..1	ref	This indicates that the call point is located at the deepest level inside one or more ExclusiveAreas that are nested in the given order.

Table B.117: SynchronousServerCallPoint

Class	System			
Package	M2::AUTOSARTemplates::SystemTemplate			
Note	The top level element of the System Description. The System description defines five major elements: Topology, Software, Communication, Mapping and Mapping Constraints. The System element directly aggregates the elements describing the Software, Mapping and Mapping Constraints; it contains a reference to an ASAM FIBEX description specifying Communication and Topology. **Tags:**atp.recommendedPackage=Systems			
Base	*ARElement*, *ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
clientId DefinitionSet	ClientIdDefinitionSet	*	ref	Set of Client Identifiers that are used for inter-ECU client-server communication in the System.
containerIPdu HeaderByte Order	ByteOrderEnum	0..1	attr	Defines the byteOrder of the header in ContainerIPdus.
ecuExtract Version	RevisionLabelString	0..1	attr	Version number of the Ecu Extract.
fibexElement	FibexElement	*	ref	Reference to ASAM FIBEX elements specifying Communication and Topology. All Fibex Elements used within a System Description shall be referenced from the System Element. ▽

▽

△

Class	**System**			
				△ atpVariation: In order to describe a product-line, all Fibex Elements can be optional. **Stereotypes:** atpVariation **Tags:**vh.latestBindingTime=postBuild
interpolation Routine MappingSet	InterpolationRoutine MappingSet	*	ref	This reference identifies the InterpolationRoutineMapping Sets that are relevant in the context of the enclosing System.
j1939Shared AddressCluster	J1939SharedAddress Cluster	*	aggr	Collection of J1939Clusters that share a common address space for the routing of messages. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=j1939SharedAddressCluster.shortName, j1939SharedAddressCluster.variationPoint.shortLabel vh.latestBindingTime=postBuild
mapping	SystemMapping	*	aggr	Aggregation of all mapping aspects (mapping of SW components to ECUs, mapping of data elements to signals, and mapping constraints). In order to support OEM / Tier 1 interaction and shared development for one common System this aggregation is atpSplitable and atpVariation. The content of System Mapping can be provided by several parties using different names for the SystemMapping. This element is not required when the System description is used for a network-only use-case. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=mapping.shortName, mapping.variation Point.shortLabel vh.latestBindingTime=postBuild
pncVector Length	PositiveInteger	0..1	attr	Length of the partial networking request release information vector (in bytes).
pncVectorOffset	PositiveInteger	0..1	attr	Absolute offset (with respect to the NM-PDU) of the partial networking request release information vector that is defined in bytes as an index starting with 0.
rootSoftware Composition	RootSwComposition Prototype	0..1	aggr	Aggregation of the root software composition, containing all software components in the System in a hierarchical structure. This element is not required when the System description is used for a network-only use-case. atpVariation: The RootSwCompositionPrototype can vary. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=rootSoftwareComposition.shortName, root SoftwareComposition.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime
swCluster	CpSoftwareCluster	*	ref	CP Software Clusters of this System **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=swCluster.cpSoftwareCluster, sw Cluster.variationPoint.shortLabel atp.Status=draft vh.latestBindingTime=systemDesignTime

▽

△

Class	System			
system Documentation	Chapter	*	aggr	Possibility to provide additional documentation while defining the System. The System documentation can be composed of several chapters. **Stereotypes:** atpSplitable; atpVariation **Tags:** atp.Splitkey=systemDocumentation.shortName, systemDocumentation.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime xml.sequenceOffset=-10
systemVersion	RevisionLabelString	1	attr	Version number of the System Description.

Table B.118: System

Class	Trigger			
Package	M2::AUTOSARTemplates::CommonStructure::TriggerDeclaration			
Note	A trigger which is provided (i.e. released) or required (i.e. used to activate something) in the given context.			
Base	*ARObject*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
swImplPolicy	SwImplPolicyEnum	0..1	attr	This attribute, when set to value queued, allows for a queued processing of Triggers.
triggerPeriod	MultidimensionalTime	0..1	aggr	Optional definition of a period in case of a periodically (time or angle) driven external trigger.

Table B.119: Trigger

Class	TriggerInterface			
Package	M2::AUTOSARTemplates::SWComponentTemplate::PortInterface			
Note	A trigger interface declares a number of triggers that can be sent by an trigger source. **Tags:**atp.recommendedPackage=PortInterfaces			
Base	*ARElement*, *ARObject*, *AtpBlueprint*, *AtpBlueprintable*, *AtpClassifier*, *AtpType*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *PortInterface*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
trigger	Trigger	*	aggr	The Trigger of this trigger interface.

Table B.120: TriggerInterface

Class	Unit
Package	M2::MSR::AsamHdo::Units
Note	This is a physical measurement unit. All units that might be defined should stem from SI units. In order to convert one unit into another factor and offset are defined. For the calculation from SI-unit to the defined unit the factor (factorSiToUnit) and the offset (offsetSiTo Unit) are applied as follows: x [{unit}] := y * [{siUnit}] * factorSiToUnit [[unit]/{siUnit}] + offsetSiToUnit [{unit}] For the calculation from a unit to SI-unit the reciprocal of the factor (factorSiToUnit) and the negation of the offset (offsetSiToUnit) are applied. y {siUnit} := (x*{unit} - offsetSiToUnit [{unit}]) / (factorSiToUnit [[unit]/{siUnit}] **Tags:**atp.recommendedPackage=Units
Base	*ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable*

▽

△

Class	Unit			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
displayName	SingleLanguageUnit Names	0..1	aggr	This specifies how the unit shall be displayed in documents or in user interfaces of tools.The displayName corresponds to the Unit.Display in an ASAM MCD-2MC file. **Tags:**xml.sequenceOffset=20
factorSiToUnit	Float	0..1	attr	This is the factor for the conversion from SI Units to units. The inverse is used for conversion from units to SI Units. **Tags:**xml.sequenceOffset=30
offsetSiToUnit	Float	0..1	attr	This is the offset for the conversion from and to siUnits. **Tags:**xml.sequenceOffset=40
physical Dimension	PhysicalDimension	0..1	ref	This association represents the physical dimension to which the unit belongs to. Note that only values with units of the same physical dimensions might be converted. **Tags:**xml.sequenceOffset=50

Table B.121: Unit

Class	VariableAccess			
Package	M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::DataElements			
Note	The presence of a VariableAccess implies that a RunnableEntity needs access to a VariableData Prototype. The kind of access is specified by the role in which the class is used.			
Base	*ARObject*, *AbstractAccessPoint*, *AtpClassifier*, *AtpFeature*, *AtpStructureElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
accessed Variable	AutosarVariableRef	0..1	aggr	This denotes the accessed variable.
scope	VariableAccessScope Enum	0..1	attr	This attribute allows for constraining the scope of the corresponding communication. For example, it possible to express whether the communication is intended to cross the boundary of an ECU or whether it is intended not to cross the boundary of a single partition.

Table B.122: VariableAccess

Class	VariableDataPrototype			
Package	M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes			
Note	A VariableDataPrototype is used to contain values in an ECU application. This means that most likely a VariableDataPrototype allocates "static" memory on the ECU. In some cases optimization strategies might lead to a situation where the memory allocation can be avoided. In particular, the value of a VariableDataPrototype is likely to change as the ECU on which it is used executes.			
Base	*ARObject*, *AtpFeature*, *AtpPrototype*, *AutosarDataPrototype*, *DataPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable*			
Attribute	**Type**	**Mult.**	**Kind**	**Note**
initValue	ValueSpecification	0..1	aggr	Specifies initial value(s) of the VariableDataPrototype

Table B.123: VariableDataPrototype

C Referenced ECUC Configuration Parameters

C.1 EcuC

C.1.1 EcucPartition

SWS Item	[ECUC_EcuC_00005]		
Container Name	EcucPartition		
Parent Container	EcucPartitionCollection		
Description	Definition of one Partition on this ECU. One Partition will be implemented using one Os-Application.		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE, VARIANT-POST-BUILD
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	EcucDefaultBswPartition [ECUC_EcuC_00037]		
Parent Container	EcucPartition		
Description	Denotes the default BSW partition. This partition will host all BSW Modules, which are not explicitly mapped to a different partition. For partitions other than the default BSW partition this parameter can be omitted.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

Name	PartitionCanBeRestarted [ECUC_EcuC_00006]		
Parent Container	EcucPartition		
Description	Specifies the requirement whether the Partition can be restarted. If set to true all software executing in this partition shall be capable of handling a restart.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

Name	EcucEcuPartitionRef [ECUC_EcuC_00083]		
Parent Container	EcucPartition		
Description	Reference to the EcuPartition to define the link to the partition described in the System description. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Foreign reference to ECU-PARTITION		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	EcucPartitionBswModuleDistinguishedPartition [ECUC_EcuC_00068]		
Parent Container	EcucPartition		
Description	This maps the abstract partition of the Bsw Module to a concrete Partition existing in the ECU.		
Multiplicity	0..*		
Type	Foreign reference to BSW-DISTINGUISHED-PARTITION		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

Name	EcucPartitionSoftwareComponentInstanceRef [ECUC_EcuC_00036]		
Parent Container	EcucPartition		
Description	References the SW Component instances from the Ecu Extract that shall be executed in this partition.		
Multiplicity	0..*		
Type	Instance reference to SW-COMPONENT-PROTOTYPE context: ROOT-SW-COMPOSITION-PROTOTYPE		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

No Included Containers

C.2 RTE

C.2.1 RteRipsPluginProps

SWS Item	[ECUC_Rte_09173]
Container Name	RteRipsPluginProps
Destination Uri Definition	RteRipsPlugin
Description	This container defines the identity of the Rte Implementation Plug-in and provides the RTE relevant parameters of the Rte Implementation Plug-in. The shortName of the container defines the name of the Rte Implementation Plug-in used for the API infixes.
Configuration Parameters	

Name	RtePluginSupportsIReadIWrite [ECUC_Rte_09169]		
Parent Container	RteRipsPluginProps		
Description	Denotes if or if not the plug-in supports the Rte_Rips_IRead/IWrite macros for primitive data.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteRipsGlobalCopyInstantiationPolicy [ECUC_Rte_09170]		
Parent Container	RteRipsPluginProps		
Description	Globally enables or disables the support for Rte Implementation Plug-Ins (RIPS)		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	RTE_RIPS_INSTANTIATION_BY_PLUGIN	The Rte Implementation Plug-In shall provide the global copy(s) for each Communication Graph.	
	RTE_RIPS_INSTANTIATION_BY_RTE	The RTE shall provide an individual global copy for each Communication Graph.	
Default Value	RTE_RIPS_INSTANTIATION_BY_RTE		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteRipsPluginCommunicationScope [ECUC_Rte_09171]		
Parent Container	RteRipsPluginProps		
Description	Defines the communication scope for which the Rte Implementation Plug-Ins (RIPS) serves. If this parameter is not set, the default behavior RTE_RIPS_LOCAL_SW_CLUSTER_COM applies.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	RTE_RIPS_CROSS_SW_CLUSTER_COM	The Rte Implementation Plug-In handles the Cross Software Cluster Communication.	
	RTE_RIPS_LOCAL_SW_CLUSTER_COM	The Rte Implementation Plug-In handles the Local Software Cluster Communication.	
Default Value	RTE_RIPS_LOCAL_SW_CLUSTER_COM		

Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

C.2.2 RteEventToTaskMapping

SWS Item	[ECUC_Rte_09020]
Container Name	RteEventToTaskMapping
Parent Container	RteSwComponentInstance
Description	Maps an instance of a RunnableEntity onto one OsTask based on the activating RTEEvent. In the case of a RunnableEntity executed via a direct or trusted function call this RteEventToTaskMapping is still specified but no RteMappedToTask element is included. The RtePositionInTask parameter is necessary to provide an ordering of events invoked by the same RTE API.
Configuration Parameters	

Name	RteActivationOffset [ECUC_Rte_09018]		
Parent Container	RteEventToTaskMapping		
Description	Activation offset in seconds.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteImmediateRestart [ECUC_Rte_09092]		
Parent Container	RteEventToTaskMapping		
Description	When RteImmediateRestart is set to true the RunnableEntitiy shall be immediately re-started after termination if it was activated by this RTEEvent while it was already started. This parameter shall not be set to true when the mapped RTEEvent refers to a RunnableEntity which minimumStartInterval attribute is > 0.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteOsSchedulePoint [ECUC_Rte_09022]	
Parent Container	RteEventToTaskMapping	
Description	Introduce a schedule point by explicitly calling Os Schedule service after the execution of the ExecutableEntity. The Rte generator is allowed to optimize several consecutive calls to Os schedule into one single call if the ExecutableEntity executions in between have been skipped. The absence of this parameter is interpreted as "NONE". It shall be considered an invalid configuration if the task is preemptable and the value of this parameter is not set to "NONE" or the parameter is absent.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	CONDITIONAL	A Schedule Point shall be introduced at the end of the execution of this ExecutableEntity. The Schedule Point can be skipped if several Schedule Points would be called without any ExecutableEntity execution in between.
	NONE	No Schedule Point shall be introduced at the end of the execution of this ExecutableEntity.
	UNCONDITIONAL	A Schedule Point shall always be introduced at the end of the execution of this ExecutableEntity.
Post-Build Variant Multiplicity	false	
Post-Build Variant Value	false	

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RtePositionInTask [ECUC_Rte_09023]		
Parent Container	RteEventToTaskMapping		
Description	Each RunnableEntity mapped to an OsTask has a specific position within the task execution. For periodic activation this is the order of execution. For event driver activation this is the order of evaluation which actual RunnableEntity has to be executed. In case of direct or trusted function calls this parameter is necessary to provide an ordering of events when several ExecutableEntities are invoked by the same RTE API.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteServerQueueLength [ECUC_Rte_09133]		
Parent Container	RteEventToTaskMapping		
Description	Specifies the length of the queue for the server call serialization. This value overwrites the queueLength specified at the ServerComSpec.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteEventPredecessorSyncPointRef [ECUC_Rte_09128]		
Parent Container	RteEventToTaskMapping		
Description	The RteEventPredecessorSyncPointRef is necessary to provide a cross core synchronization in case of RteEvents triggered by the same event source but mapped to tasks belonging to different partitions on different cores. The synchronization point must be reached by all referencing RteEvents before the execution in all related tasks is continued. In case of RteEventPredecessorSyncPointRef the RunnableEntity activated by the mapped RteEvent is executed after the synchronization point is passed.		
Multiplicity	0..1		
Type	Reference to RteSyncPoint		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteEventRef [ECUC_Rte_09019]
Parent Container	RteEventToTaskMapping
Description	Reference to the description of the RTEEvent which is pointing to the RunnableEntity being mapped. This allows a fine grained mapping of RunnableEntites based on the activating RTEEvent.
Multiplicity	1..*
Type	Foreign reference to RTE-EVENT
Post-Build Variant Value	false

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteEventSuccessorSyncPointRef [ECUC_Rte_09129]		
Parent Container	RteEventToTaskMapping		
Description	The RteEventSuccessorSyncPointRef is necessary to provide a cross core synchronization in case of RteEvents triggered by the same event source but mapped to tasks belonging to different partitions on different cores. The synchronization point must be reached by all referencing RteEvents before the execution in all related tasks is continued. In case of RteEventSuccessorSyncPointRef the RunnableEntity activated by the mapped RteEvent is executed before the synchronization point is entered.		
Multiplicity	0..1		
Type	Reference to RteSyncPoint		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteMappedToTaskRef [ECUC_Rte_09021]		
Parent Container	RteEventToTaskMapping		
Description	Reference to the OsTask the RunnableEntity activated by the RteEventRef is mapped to. If no reference to the OsTask is specified the RunnableEntity shall be executed via a direct or trusted function call. The fact that no reference to an OsTask is specified for a RunnableEntity does not necessarily imply that every RTE generator has to support the implementation of this RunnableEntity as a direct or trusted function call. The standard set of use cases for direct or trusted function calls that has to be supported by every RTE generator is explicitly stated as requirements in this document. For further optimization RTE vendors are free to support additional scenarios of direct or trusted function call implementations that are not explicitly required in this document.		
Multiplicity	0..1		
Type	Reference to OsTask		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteRipsFillRoutineRef [ECUC_Rte_89005]		
Parent Container	RteEventToTaskMapping		
Description	Reference to a Buffer-Fill Routine implemented by an RTE Implementation Plug-In. This routine gets invoked directly before the ExecutableEntity is started. **Attributes:** requiresIndex=true		
Multiplicity	0..*		
Type	Reference to destinationUri [RteRipsUriDefSet/RteRipsPluginFillFlush Routine]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteRipsFlushRoutineRef [ECUC_Rte_89006]		
Parent Container	RteEventToTaskMapping		
Description	Reference to a Buffer-Flush Routine implemented by an RTE Implementation Plug-In. This routine gets invoked directly after the ExecutableEntity has terminated. **Attributes:** requiresIndex=true		
Multiplicity	0..*		
Type	Reference to destinationUri [RteRipsUriDefSet/RteRipsPluginFillFlush Routine]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteRipsInvocationHandlerRef [ECUC_Rte_89008]		
Parent Container	RteEventToTaskMapping		
Description	Reference to a Buffer-Fill Routine implemented by an RTE Implementation Plug-In. This routine gets invoked directly before the ExecutableEntity is started.		
Multiplicity	0..1		
Type	Reference to destinationUri [RteRipsUriDefSet/RteRipsInvocation Handler]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteUsedInitFnc [ECUC_Rte_09116]		
Parent Container	RteEventToTaskMapping		
Description	The RunnableEntity is executed during initialization in the context of the Rte_Init_<InitContainer> function.		
Multiplicity	0..1		
Type	Reference to RteInitializationRunnableBatch		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteUsedOsAlarmRef [ECUC_Rte_09024]		
Parent Container	RteEventToTaskMapping		
Description	If an OsAlarm is used to activate the OsTask this RteEvent is mapped to it shall be referenced here.		
Multiplicity	0..1		
Type	Reference to OsAlarm		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteUsedOsEventRef [ECUC_Rte_09025]		
Parent Container	RteEventToTaskMapping		
Description	If an OsEvent is used to activate the OsTask this RteEvent is mapped to it shall be referenced here.		
Multiplicity	0..1		
Type	Reference to OsEvent		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteUsedOsSchTblExpiryPointRef [ECUC_Rte_09026]		
Parent Container	RteEventToTaskMapping		
Description	If an OsScheduleTableExpiryPoint is used to activate the OsTask this RteEvent is mapped to it shall be referenced here.		
Multiplicity	0..1		
Type	Reference to OsScheduleTableExpiryPoint		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	RteVirtuallyMappedToTaskRef [ECUC_Rte_09027]
Parent Container	RteEventToTaskMapping
Description	Optional reference to an OsTask where the activation of this RteEvent shall be evaluated. The actual execution of the Runnable Entity shall happen in the OsTask referenced by RteMappedToTaskRef.
Multiplicity	0..1
Type	Reference to OsTask
Post-Build Variant Multiplicity	false

Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

C.3 Os

C.3.1 OsAlarm

SWS Item	[ECUC_Os_00003]
Container Name	OsAlarm
Parent Container	Os
Description	An OsAlarm may be used to asynchronously inform or activate a specific task. It is possible to start alarms automatically at system start-up depending on the application mode.
Configuration Parameters	

Name	OsAlarmAccessingApplication [ECUC_Os_00004]		
Parent Container	OsAlarm		
Description	Reference to applications which have an access to this object.		
Multiplicity	0..*		
Type	Reference to OsApplication		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

Name	OsAlarmCounterRef [ECUC_Os_00005]		
Parent Container	OsAlarm		
Description	Reference to the assigned counter for that alarm		
Multiplicity	1		
Type	Reference to OsCounter		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
OsAlarmAction	1	This container defines which type of notification is used when the alarm expires.
OsAlarmAutostart	0..1	If present this container defines if an alarm is started automatically at system start-up depending on the application mode.

C.3.2 OsApplication

SWS Item	[ECUC_Os_00114]
Container Name	OsApplication
Parent Container	Os
Description	An AUTOSAR OS must be capable of supporting a collection of OS objects (tasks, interrupts, alarms, hooks etc.) that form a cohesive functional unit. This collection of objects is termed an OS-Application. All objects which belong to the same OS-Application have access to each other. Access means to allow to use these objects within API services. Access by other applications can be granted separately.
Configuration Parameters	

Name	OsTrusted [ECUC_Os_00115]
Parent Container	OsApplication
Description	Parameter to specify if an OS-Application is trusted or not. true: OS-Application is trusted false: OS-Application is not trusted (default)
Multiplicity	1
Type	EcucBooleanParamDef
Default Value	false
Post-Build Variant Value	false

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: Required for scalability class 3 and 4.		

Name	OsTrustedApplicationDelayTimingViolationCall [ECUC_Os_00395]		
Parent Container	OsApplication		
Description	Parameter to specify if a timing violation which occurs within an trusted OS-Application is raised immediately of if it is delayed until the current task returns to the calling OS-Application (return of CallTrustedFunction) true: violation / call to ProtectionHook() is delayed false: timing violation cause an immediate call to the ProtectionHook().		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	true		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	OsTrustedApplicationWithProtection [ECUC_Os_00394]		
Parent Container	OsApplication		
Description	Parameter to specify if a trusted OS-Application is executed with memory protection or not. true: OS-Application runs within a protected environment. This means that write access is limited. false: OS-Application has full write access (default)		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	OsAppAlarmRef [ECUC_Os_00231]		
Parent Container	OsApplication		
Description	Specifies the OsAlarms that belong to the OsApplication.		
Multiplicity	0..*		
Type	Reference to OsAlarm		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	OsAppCounterRef [ECUC_Os_00234]		
Parent Container	OsApplication		
Description	References the OsCounters that belong to the OsApplication.		
Multiplicity	0..*		
Type	Reference to OsCounter		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	OsAppEcucPartitionRef [ECUC_Os_00392]
Parent Container	OsApplication
Description	Denotes which "EcucPartition" is implemented by this "OSApplication".
Multiplicity	0..1
Type	Reference to EcucPartition
Post-Build Variant Multiplicity	false
Post-Build Variant Value	false

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

Name	OsAppIsrRef [ECUC_Os_00221]		
Parent Container	OsApplication		
Description	references which OsIsrs belong to the OsApplication		
Multiplicity	0..*		
Type	Reference to OsIsr		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	OsApplicationCoreRef [ECUC_Os_00393]		
Parent Container	OsApplication		
Description	Reference to the Core Definition in the Ecuc Module where the CoreId is defined. This reference is used to describe to which Core the OsApplication is bound.		
Multiplicity	0..1		
Type	Reference to EcucCoreDefinition		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	OsAppScheduleTableRef [ECUC_Os_00230]		
Parent Container	OsApplication		
Description	References the OsScheduleTables that belong to the OsApplication.		
Multiplicity	0..*		
Type	Reference to OsScheduleTable		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	OsAppTaskRef [ECUC_Os_00116]		
Parent Container	OsApplication		
Description	references which OsTasks belong to the OsApplication		
Multiplicity	0..*		
Type	Reference to OsTask		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	OsRestartTask [ECUC_Os_00120]		
Parent Container	OsApplication		
Description	Optionally one task of an OS-Application may be defined as Restart Task. Multiplicity = 1: Restart Task is activated by the Operating System if the protection hook requests it. Multiplicity = 0: No task is automatically started after a protection error happened.		
Multiplicity	0..1		
Type	Reference to OsTask		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU dependency: Required for scalability class 3 and 4.		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
OsApplicationHooks	1	Container to structure the OS-Application-specific hooks
OsApplicationTrusted Function	0..*	Container to structure the configuration parameters of trusted functions

C.3.3 OsCounter

SWS Item	[ECUC_Os_00026]
Container Name	OsCounter
Parent Container	Os
Description	Configuration information for the counters that belong to the OsApplication.
Configuration Parameters	

Name	OsCounterMaxAllowedValue [ECUC_Os_00027]	
Parent Container	OsCounter	
Description	Maximum possible allowed value of the system counter in ticks.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 18446744073709551615	
Default Value		

Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	OsCounterMinCycle [ECUC_Os_00028]		
Parent Container	OsCounter		
Description	The MINCYCLE attribute specifies the minimum allowed number of counter ticks for a cyclic alarm linked to the counter.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 18446744073709551615		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	OsCounterTicksPerBase [ECUC_Os_00029]		
Parent Container	OsCounter		
Description	The TICKSPERBASE attribute specifies the number of ticks required to reach a counterspecific unit. The interpretation is implementation-specific.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	OsCounterType [ECUC_Os_00255]		
Parent Container	OsCounter		
Description	This parameter contains the natural type or unit of the counter.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	HARDWARE	This counter is driven by some hardware e.g. a hardware timer unit.	
	SOFTWARE	The counter is driven by some software which calls the IncrementCounter service.	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	OsSecondsPerTick [ECUC_Os_00030]		
Parent Container	OsCounter		
Description	Time of one counter tick in seconds.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	OsCounterAccessingApplication [ECUC_Os_00031]
Parent Container	OsCounter
Description	Reference to applications which have an access to this object.
Multiplicity	0..*
Type	Reference to OsApplication
Post-Build Variant Multiplicity	false
Post-Build Variant Value	false

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
OsDriver	0..1	This Container contains the information who will drive the counter. This configuration is only valid if the counter has OsCounterType set to HARDWARE. If the container does not exist (multiplicity=0) the timer is managed by the OS internally (OSINTERNAL). If the container exists the OS can use the GPT interface to manage the timer. The user have to supply the GPT channel. If the counter is driven by some other (external to the OS) source (like a TPU for example) this must be described as a vendor specific extension.
OsTimeConstant	0..*	Allows the user to define constants which can be e.g. used to compare time values with timer tick values. `A time value will be converted to a timer tick value during generation and can later on accessed via the OsConstName. The conversation is done by rounding time values to the nearest fitting tick value.`

C.3.4 OsEvent

SWS Item	[ECUC_Os_00033]
Container Name	OsEvent
Parent Container	Os
Description	Representation of OS events in the configuration context. Adopted from the ISO 17356-6 specification.
Configuration Parameters	

Name	OsEventMask [ECUC_Os_00034]		
Parent Container	OsEvent		
Description	If event mask would be set to AUTO in OIL, this parameter should be omitted here.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

C.3.5 OsResource

SWS Item	[ECUC_Os_00252]
Container Name	OsResource
Parent Container	Os
Description	An OsResource object is used to co-ordinate the concurrent access by tasks and ISRs to a shared resource, e.g. the scheduler, any program sequence, memory or any hardware area.
Configuration Parameters	

Name	OsResourceProperty [ECUC_Os_00050]	
Parent Container	OsResource	
Description	This specifies the type of the resource.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	INTERNAL	The resource is an internal resource.
	LINKED	The resource is a linked resource (a second name for a existing resource).
	STANDARD	The resource is a standard resource.
Post-Build Variant Value	false	

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	OsResourceAccessingApplication [ECUC_Os_00051]		
Parent Container	OsResource		
Description	Reference to applications which have an access to this object.		
Multiplicity	0..*		
Type	Reference to OsApplication		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	OsResourceLinkedResourceRef [ECUC_Os_00052]		
Parent Container	OsResource		
Description	The link to the resource. Must be valid if OsResourceProperty is LINKED. If OsResourceProperty is not LINKED the value is ignored.		
Multiplicity	0..1		
Type	Reference to OsResource		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

C.3.6 OsScheduleTable

SWS Item	[ECUC_Os_00141]
Container Name	OsScheduleTable
Parent Container	Os
Description	An OsScheduleTable addresses the synchronization issue by providing an encapsulation of a statically defined set of alarms that cannot be modified at runtime.
Configuration Parameters	

Name	OsScheduleTableDuration [ECUC_Os_00053]		
Parent Container	OsScheduleTable		
Description	This parameter defines the modulus of the schedule table (in ticks).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	OsScheduleTableRepeating [ECUC_Os_00144]		
Parent Container	OsScheduleTable		
Description	true: first expiry point on the schedule table shall be processed at final expiry point delay ticks after the final expiry point is processed. false: the schedule table processing stops when the final expiry point is processed.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	OsScheduleTableCounterRef [ECUC_Os_00145]		
Parent Container	OsScheduleTable		
Description	This parameter contains a reference to the counter which drives the schedule table.		
Multiplicity	1		
Type	Reference to OsCounter		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	OsSchTblAccessingApplication [ECUC_Os_00054]		
Parent Container	OsScheduleTable		
Description	Reference to applications which have an access to this object.		
Multiplicity	0..*		
Type	Reference to OsApplication		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
OsScheduleTable Autostart	0..1	This container specifies if and how the schedule table is started on startup of the Operating System. The options to start a schedule table correspond to the API calls to start schedule tables during runtime.
OsScheduleTableExpiry Point	1..*	The point on a Schedule Table at which the OS activates tasks and/or sets events
OsScheduleTableSync	0..1	This container specifies the synchronization parameters of the schedule table.

C.3.7 OsScheduleTableExpiryPoint

SWS Item	[ECUC_Os_00143]
Container Name	OsScheduleTableExpiryPoint

Parent Container	OsScheduleTable
Description	The point on a Schedule Table at which the OS activates tasks and/or sets events
Configuration Parameters	

Name	OsScheduleTblExpPointOffset [ECUC_Os_00062]		
Parent Container	OsScheduleTableExpiryPoint		
Description	The offset from zero (in ticks) at which the expiry point is to be processed.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
OsScheduleTableEvent Setting	0..*	Event that is triggered by that schedule table.
OsScheduleTableTask Activation	0..*	Task that is triggered by that schedule table.
OsScheduleTbl AdjustableExpPoint	0..1	Adjustable expiry point

C.3.8 OsSpinlock

SWS Item	[ECUC_Os_00258]
Container Name	OsSpinlock
Parent Container	Os
Description	An OsSpinlock object is used to co-ordinate concurrent access by TASKs/ISR2s on different cores to a shared resource.
Configuration Parameters	

Name	OsSpinlockLockMethod [ECUC_Os_01038]		
Parent Container	OsSpinlock		
Description	Lock method which is used when a spinlock is taken. Note that it is possible that a user (e.g. a Task) might hold more than one spinlock. In this case the last lock taken is forced to use at least a lock methode which locks as strong as the current one.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	LOCK_ALL_INTERRUPTS		
	LOCK_CAT2_INTERRUPTS		
	LOCK_NOTHING		
	LOCK_WITH_RES_SCHEDULER		
Default Value	LOCK_NOTHING		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	OsSpinlockAccessingApplication [ECUC_Os_01021]		
Parent Container	OsSpinlock		
Description	Reference to OsApplications that have an access to this object.		
Multiplicity	1..*		
Type	Reference to OsApplication		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	OsSpinlockSuccessor [ECUC_Os_01022]		
Parent Container	OsSpinlock		
Description	Reference to OsApplications that have an access to this object. To check whether a spinlock can be occupied (in a nested way) without any danger of deadlock, a linked list of spinlocks can be defined. A spinlock can only be occupied in the order of the linked list. It is allowed to skip a spinlock. If no linked list is specified, spinlocks cannot be nested.		
Multiplicity	0..1		
Type	Reference to OsSpinlock		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

C.3.9 OsTask

SWS Item	[ECUC_Os_00073]
Container Name	OsTask
Parent Container	Os
Description	This container represents an ISO 17356 task.
Configuration Parameters	

Name	OsTaskActivation [ECUC_Os_00074]	
Parent Container	OsTask	
Description	This attribute defines the maximum number of queued activation requests for the task. A value equal to "1" means that at any time only a single activation is permitted for this task. Note that the value must be a natural number starting at 1.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	
Default Value		
Post-Build Variant Value	false	

Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	OsTaskPeriod [ECUC_Os_00404]		
Parent Container	OsTask		
Description	This parameter specifies the period in seconds of this task in case of a cyclically activated task. If this parameter is not given the task can be activated sporadicly or cyclically with a unknown period value. This value is information, e.g. for time base calculations in the RTE in case TimingEvents are mapped onto this OsTask.Be aware, that this parameter is not supposed to be relevant for the OS! This information is given as part of the OS configuration to support configuration work flows using a fixed set of OsTasks.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[-INF .. INF]		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	OsTaskPriority [ECUC_Os_00075]		
Parent Container	OsTask		
Description	The priority of a task is defined by the value of this attribute. This value has to be understood as a relative value, i.e. the values show only the relative ordering of the tasks. ISO 17356-3 defines the lowest priority as zero (0); larger values correspond to higher priorities.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default Value			

Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	OsTaskSchedule [ECUC_Os_00076]		
Parent Container	OsTask		
Description	The OsTaskSchedule attribute defines the preemptability of the task. If this attribute is set to NON, no internal resources may be assigned to this task.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FULL	Task is preemptable.	
	NON	Task is not preemptable.	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	OsMemoryMappingCodeLocationRef [ECUC_Os_00402]		
Parent Container	OsTask		
Description	Reference to the memory mapping containing details about the section where the code is placed.		
Multiplicity	0..1		
Type	Foreign reference to SW-ADDR-METHOD		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	OsTaskAccessingApplication [ECUC_Os_00077]
Parent Container	OsTask
Description	Reference to applications which have an access to this object.
Multiplicity	0..*
Type	Reference to OsApplication
Post-Build Variant Multiplicity	false

Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	OsTaskEventRef [ECUC_Os_00078]		
Parent Container	OsTask		
Description	This reference defines the list of events the extended task may react on.		
Multiplicity	0..*		
Type	Reference to OsEvent		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	OsTaskResourceRef [ECUC_Os_00079]		
Parent Container	OsTask		
Description	This reference defines a list of resources accessed by this task.		
Multiplicity	0..*		
Type	Reference to OsResource		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

Scope / Dependency	scope: local

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
OsTaskAutostart	0..1	This container determines whether the task is activated during the system start-up procedure or not for some specific application modes. If the task shall be activated during the system start-up, this container is present and holds the references to the application modes in which the task is auto-started.
OsTaskTimingProtection	0..1	This container contains all parameters regarding timing protection of the task.

C.4 NvM

C.4.1 NvMBlockDescriptor

SWS Item	[ECUC_NvM_00061]
Container Name	NvMBlockDescriptor
Parent Container	NvM
Description	Container for a management structure to configure the composition of a given NVRAM Block Management Type. Its multiplicity describes the number of configured NVRAM blocks, one block is required to be configured. The NVRAM block descriptors are condensed in the NVRAM block descriptor table.
Configuration Parameters	

Name	NvMBlockCrcType [ECUC_NvM_00476]		
Parent Container	NvMBlockDescriptor		
Description	Defines CRC data width for the NVRAM block. Default: NVM_CRC16, i.e. CRC16 will be used if NVM_BLOCK_USE_CRC==true		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	NVM_CRC16	(Default) CRC16 will be used if NVM_BLOCK_USE_CRC==true.	
	NVM_CRC32	CRC32 is selected for this NVRAM block if NVM_BLOCK_USE_CRC==true.	
	NVM_CRC8	CRC8 is selected for this NVRAM block if NVM_BLOCK_USE_CRC==true.	
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local dependency: NVM_BLOCK_USE_CRC, NVM_CALC_RAM_BLOCK_CRC		

Name	NvMBlockHeaderInclude [ECUC_NvM_00554]		
Parent Container	NvMBlockDescriptor		
Description	Defines the header file where the owner of the NVRAM block has the declarations of the permanent RAM data block, ROM data block (if configured) and the callback function prototype for each configured callback. If no permanent RAM block, ROM block or callback functions are configured then this configuration parameter shall be ignored.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMBlockJobPriority [ECUC_NvM_00477]		
Parent Container	NvMBlockDescriptor		
Description	Defines the job priority for a NVRAM block (0 = Immediate priority).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMBlockManagementType [ECUC_NvM_00062]		
Parent Container	NvMBlockDescriptor		
Description	Defines the block management type for the NVRAM block.[SWS_NvM_00137]		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	NVM_BLOCK_DATASET	NVRAM block is configured to be of dataset type.	
	NVM_BLOCK_NATIVE	NVRAM block is configured to be of native type.	
	NVM_BLOCK_REDUNDANT	NVRAM block is configured to be of redundant type.	
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMBlockUseAutoValidation [ECUC_NvM_00557]		
Parent Container	NvMBlockDescriptor		
Description	Defines whether the RAM Block shall be auto validated during shutdown phase. true: if auto validation mechanism is used, false: otherwise		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMBlockUseCompression [ECUC_NvM_00563]
Parent Container	NvMBlockDescriptor
Description	Defines whether the data is compressed before written. true: data compression activated (takes more time to read and write) false: no compression **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucBooleanParamDef
Default Value	false
Post-Build Variant Value	false

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMBlockUseCrc [ECUC_NvM_00036]		
Parent Container	NvMBlockDescriptor		
Description	Defines CRC usage for the NVRAM block, i.e. memory space for CRC is reserved in RAM and NV memory. true: CRC will be used for this NVRAM block. false: CRC will not be used for this NVRAM block.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMBlockUseCRCCompMechanism [ECUC_NvM_00556]		
Parent Container	NvMBlockDescriptor		
Description	Defines whether the CRC of the RAM Block shall be compared during a write job with the CRC which was calculated during the last successful read or write job. true: if compare mechanism is used, false: otherwise		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local dependency: False if NvMBlockUseCrc = False		

Name	NvMBlockUsePort [ECUC_NvM_00559]		
Parent Container	NvMBlockDescriptor		
Description	If this parameter is true it defines whether: • the port with interface 'NvMMirror' for synchronization mechanism callbacks are generated if the parameter NvMBlockUseSyncMechanism is configured TRUE; • the port with interface 'NvMNotifyInitBlock' for initialization block callback is generated if NvMInitBlockCallback parameter is configured (independent of the content); • the port with interface 'NvMNotifyJobFinished' for single block callback is generated if NvMSingleBlockCallback parameter is configured (independent of the content); • the port with interface 'NvMAdmin' for SetBlockProtection operation is generated.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMBlockUseSetRamBlockStatus [ECUC_NvM_00552]		
Parent Container	NvMBlockDescriptor		
Description	Defines if NvMSetRamBlockStatusApi shall be used for this block or not. Note: If NvMSetRamBlockStatusApi is disabled this configuration parameter shall be ignored. true: calling of NvMSetRamBlockStatus for this RAM block shall set the status of the RAM block. false: calling of NvMSetRamBlockStatus for this RAM block shall be ignored.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	

Scope / Dependency	scope: local

Name	NvMBlockUseSyncMechanism [ECUC_NvM_00519]		
Parent Container	NvMBlockDescriptor		
Description	Defines whether an explicit synchronization mechanism with a RAM mirror and callback routines for transferring data to and from NvM module's RAM mirror is used for NV block. true if synchronization mechanism is used, false otherwise.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMBlockWriteProt [ECUC_NvM_00033]		
Parent Container	NvMBlockDescriptor		
Description	Defines an initial write protection of the NV block true: Initial block write protection is enabled. false: Initial block write protection is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMBswMBlockStatusInformation [ECUC_NvM_00551]
Parent Container	NvMBlockDescriptor
Description	This parameter specifies whether BswM is informed about the current status of the specified block. True: Call BswM_NvM_CurrentBlockMode on changes False: Don't inform BswM at all
Multiplicity	1
Type	EcucBooleanParamDef
Default Value	false
Post-Build Variant Value	false

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMCalcRamBlockCrc [ECUC_NvM_00119]		
Parent Container	NvMBlockDescriptor		
Description	Defines CRC (re)calculation for the permanent RAM block or NVRAM blocks which are configured to use explicit synchronization mechanism. true: CRC will be (re)calculated for this permanent RAM block. false: CRC will not be (re)calculated for this permanent RAM block.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local dependency: NVM_BLOCK_USE_CRC		

Name	NvMMaxNumOfReadRetries [ECUC_NvM_00533]		
Parent Container	NvMBlockDescriptor		
Description	Defines the maximum number of read retries.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default Value	0		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMMaxNumOfWriteRetries [ECUC_NvM_00499]		
Parent Container	NvMBlockDescriptor		
Description	Defines the maximum number of write retries for a NVRAM block with [ECUC_NvM_00061]. Regardless of configuration a consistency check (and maybe write retries) are always forced for each block which is processed by the request NvM_WriteAll and NvM_WriteBlock.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 7		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMNvBlockBaseNumber [ECUC_NvM_00478]		
Parent Container	NvMBlockDescriptor		
Description	Configuration parameter to perform the link between the NVM_NVRAM_BLOCK_IDENTIFIER used by the SW-Cs and the FEE_BLOCK_NUMBER expected by the memory abstraction modules. The parameter value equals the FEE_BLOCK_NUMBER or EA_BLOCK_NUMBER shifted to the right by NvMDatasetSelectionBits bits. (ref. to chapter 7.1.2.1). Calculation Formula: value = TargetBlockReference.[Ea/Fee]BlockConfiguration.[Ea/Fee]BlockNumber » NvMDatasetSelectionBits		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 65534		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local dependency: FEE_BLOCK_NUMBER, EA_BLOCK_NUMBER		

Name	NvMNvBlockLength [ECUC_NvM_00479]		
Parent Container	NvMBlockDescriptor		
Description	Defines the NV block data length in bytes. Note: The implementer can add the attribute 'withAuto' to the parameter definition which indicates that the length can be calculated by the generator automatically (e.g. by using a parser that searches and analyzes the data structure corresponding to the block). When 'withAuto' is set to 'true' for this parameter definition the 'isAutoValue' can be set to 'true'. If 'isAutoValue' is set to 'true' the actual value will not be considered during ECU Configuration but will be (re-)calculated by the code generator and stored in the value attribute afterwards.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMNvBlockNum [ECUC_NvM_00480]		
Parent Container	NvMBlockDescriptor		
Description	Defines the number of multiple NV blocks in a contiguous area according to the given block management type. 1-255 For NVRAM blocks to be configured of block management type NVM_BLOCK_DATASET. The actual range is limited according to SWS_NvM_00444. 1 For NVRAM blocks to be configured of block management type NVM_BLOCK_NATIVE 2 For NVRAM blocks to be configured of block management type NVM_BLOCK_REDUNDANT		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 255		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local dependency: NVM_BLOCK_MANAGEMENT_TYPE		

Name	NvMNvramBlockIdentifier [ECUC_NvM_00481]		
Parent Container	NvMBlockDescriptor		
Description	Identification of a NVRAM block via a unique block identifier. Implementation Type: NvM_BlockIdType. min = 2 max = 2^ (16- NVM_DATASET_SELECTION_BITS)-1 Reserved NVRAM block IDs: 0 -> to derive multi block request results via NvM_GetErrorStatus 1 -> redundant NVRAM block which holds the configuration ID (generation tool should check that this block is correctly configured from type,CRC and size point of view)		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	2 .. 65535		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local dependency: NVM_DATASET_SELECTION_BITS		

Name	NvMNvramDeviceId [ECUC_NvM_00035]		
Parent Container	NvMBlockDescriptor		
Description	Defines the NVRAM device ID where the NVRAM block is located. Calculation Formula: value = TargetBlockReference.[Ea/Fee]BlockConfiguration.[Ea/Fee]DeviceIndex		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 1		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local dependency: EA_DEVICE_INDEX, FEE_DEVICE_INDEX		

Name	NvMRamBlockDataAddress [ECUC_NvM_00482]		
Parent Container	NvMBlockDescriptor		
Description	Defines the start address of the RAM block data. If this is not configured, no permanent RAM data block is available for the selected block management type.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMReadRamBlockFromNvCallback [ECUC_NvM_00521]		
Parent Container	NvMBlockDescriptor		
Description	Entry address of a block specific callback routine which shall be called in order to let the application copy data from the NvM module's mirror to RAM block. Implementation type: Std_ReturnType E_OK: copy was successful E_NOT_OK: copy was not successful, callback routine to be called again		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMResistantToChangedSw [ECUC_NvM_00483]		
Parent Container	NvMBlockDescriptor		
Description	Defines whether a NVRAM block shall be treated resistant to configuration changes or not. If there is no default data available at configuration time then the application shall be responsible for providing the default initialization data. In this case the application has to use NvM_GetErrorStatus()to be able to distinguish between first initialization and corrupted data. true: NVRAM block is resistant to changed software. false: NVRAM block is not resistant to changed software.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMRomBlockDataAddress [ECUC_NvM_00484]		
Parent Container	NvMBlockDescriptor		
Description	Defines the start address of the ROM block data. If not configured, no ROM block is available for the selected block management type.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMRomBlockNum [ECUC_NvM_00485]		
Parent Container	NvMBlockDescriptor		
Description	Defines the number of multiple ROM blocks in a contiguous area according to the given block management type. 0-254 For NVRAM blocks to be configured of block management type NVM_BLOCK_DATASET. The actual range is limited according to SWS_NvM_00444. 0-1 For NVRAM blocks to be configured of block management type NVM_BLOCK_NATIVE 0-1 For NVRAM blocks to be configured of block management type NVM_BLOCK_REDUNDANT		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 254		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local dependency: NVM_BLOCK_MANAGEMENT_TYPE, NVM_NV_BLOCK_NUM		

Name	NvMSelectBlockForFirstInitAll {NVM_SELECT_BLOCK_FOR_FIRST_INIT_ALL} [ECUC_NvM_00558]		
Parent Container	NvMBlockDescriptor		
Description	Defines whether a block will be processed or not by NvM_FirstInitAll. A block can be configured to be processed even if it doesn't have permanent RAM and/or explicit synchronization. TRUE: block will be processed by NvM_FirstInitAll FALSE: block will not be processed by NvM_FirstInitAll		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	

Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMSelectBlockForReadAll [ECUC_NvM_00117]		
Parent Container	NvMBlockDescriptor		
Description	Defines whether a NVRAM block shall be processed during NvM_ReadAll or not. This configuration parameter has only influence on those NVRAM blocks which are configured to have a permanent RAM block or which are configured to use explicit synchronization mechanism. true: NVRAM block shall be processed by NvM_ReadAll false: NVRAM block shall not be processed by NvM_ReadAll		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local dependency: NVM_RAM_BLOCK_DATA_ADDRESS		

Name	NvMSelectBlockForWriteAll [ECUC_NvM_00549]		
Parent Container	NvMBlockDescriptor		
Description	Defines whether a NVRAM block shall be processed during NvM_WriteAll or not. This configuration parameter has only influence on those NVRAM blocks which are configured to have a permanent RAM block or which are configured to use explicit synchronization mechanism. true: NVRAM block shall be processed by NvM_WriteAll false: NVRAM block shall not be processed by NvM_WriteAll		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Multiplicity	false		

Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local dependency: NVM_RAM_BLOCK_DATA_ADDRESS		

Name	NvMStaticBlockIDCheck [ECUC_NvM_00532]		
Parent Container	NvMBlockDescriptor		
Description	Defines if the Static Block ID check is enabled. false: Static Block ID check is disabled. true: Static Block ID check is enabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMWriteBlockOnce [ECUC_NvM_00072]		
Parent Container	NvMBlockDescriptor		
Description	Defines write protection after first write. The NVRAM manager sets the write protection bit either after the NV block was written the first time or if the block was already written and it is detected as valid and consistent during a read for it. true: Defines write protection after first write is enabled. false: Defines write protection after first write is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMWriteRamBlockToNvCallback [ECUC_NvM_00520]		
Parent Container	NvMBlockDescriptor		
Description	Entry address of a block specific callback routine which shall be called in order to let the application copy data from RAM block to NvM module's mirror. Implementation type: Std_ReturnType E_OK: copy was successful E_NOT_OK: copy was not successful, callback routine to be called again		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMWriteVerification [ECUC_NvM_00534]		
Parent Container	NvMBlockDescriptor		
Description	Defines if Write Verification is enabled. false: Write verification is disabled. true: Write Verification is enabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMWriteVerificationDataSize [ECUC_NvM_00538]	
Parent Container	NvMBlockDescriptor	
Description	Defines the number of bytes to compare in each step when comparing the content of a RAM Block and a block read back.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 65535	
Default Value		

Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

Name	NvMBlockCipheringRef [ECUC_NvM_00567]		
Parent Container	NvMBlockDescriptor		
Description	Reference to ciphering container. If configured, NvM encrypt the data before storage and decrypt the data after restoring. If empty, the NvM stores and restore the original user data. **Tags:** atp.Status=draft		
Multiplicity	0..1		
Type	Reference to NvMBlockCiphering		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local dependency: Key will be located in RAM if this configuration item is not present.		

Name	NvMBlockEcucPartitionRef [ECUC_NvM_00564]		
Parent Container	NvMBlockDescriptor		
Description	Maps the NV block to zero or one ECUC partition to limit the access to this NV block. The ECUC partition referenced is within the subset of the ECUC partitions where the NvM is mapped to.		
Multiplicity	0..1		
Type	Reference to EcucPartition		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
NvMInitBlockCallback	0..1	The presence of this container indicates, that a block specific callback routine is called if no ROM data is available for initialization of the NVRAM block. If the container is not present, no callback routine is called for initialization of the NVRAM block with default data. In case the container has a NvMInitBlockCallbackFnc, the NvM will call this function. In case there is no NvMInitBlockCallbackFnc, the NvM will have an port PNIB_{Block}.
NvMSingleBlockCallback	0..1	The presence of this container indicates, that the block specific callback routine which shall be invoked on termination of each asynchronous single block request [SWS_NvM_00113] If the container is not present, no callback routine is called.. In case the container has a NvMSingleBlockCallbackFnc, the NvM will call this function. In case there is no NvMSingleBlockCallbackFnc, the NvM will have an port PNJF_{Block}.
NvMTargetBlock Reference	1	This parameter is just a container for the parameters for EA and FEE

C.4.2 NvMInitBlockCallback

SWS Item	[ECUC_NvM_00561]		
Container Name	NvMInitBlockCallback		
Parent Container	NvMBlockDescriptor		
Description	The presence of this container indicates, that a block specific callback routine is called if no ROM data is available for initialization of the NVRAM block. If the container is not present, no callback routine is called for initialization of the NVRAM block with default data. In case the container has a NvMInitBlockCallbackFnc, the NvM will call this function. In case there is no NvMInitBlockCallbackFnc, the NvM will have an port PNIB_{Block}.		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Configuration Parameters			

Name	NvMInitBlockCallbackFnc [ECUC_NvM_00116]		
Parent Container	NvMInitBlockCallback		
Description	Entry address of a block specific callback routine which shall be called if no ROM data is available for initialization of the NVRAM block. If not configured, no specific callback routine shall be called for initialization of the NVRAM block with default data.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

C.4.3 NvMSingleBlockCallback

SWS Item	[ECUC_NvM_00562]		
Container Name	NvMSingleBlockCallback		
Parent Container	NvMBlockDescriptor		
Description	The presence of this container indicates, that the block specific callback routine which shall be invoked on termination of each asynchronous single block request [SWS_NvM_00113] If the container is not present, no callback routine is called.. In case the container has a NvMSingleBlockCallbackFnc, the NvM will call this function. In case there is no NvMSingleBlockCallbackFnc, the NvM will have an port PNJF_{Block}.		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Configuration Parameters			

Name	NvMSingleBlockCallbackFnc [ECUC_NvM_00506]		
Parent Container	NvMSingleBlockCallback		
Description	Entry address of the block specific callback routine which shall be invoked on termination of each asynchronous single block request [SWS_NvM_00113].		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

D Referenced C-API

D.1 RTE

D.1.1 Rte_Rips_DataIsUpdated

[SWS_Rte_91119]{DRAFT} ⌈

Service Name	Rte_Rips_<PlugIn>_DataIsUpdated_<SwcBswI>_<CGI> (draft)
Syntax	`boolean Rte_Rips_<PlugIn>_DataIsUpdated_<SwcBswI>_<CGI> ( void )`
Service ID [hex]	0xB4
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	None
Parameters (inout)	None

▽

△

Parameters (out)	None	
Return value	boolean	The return value is used to indicate if the data has been updated or not.
Description	The Rte_Rips_DataIsUpdated API provides access to the update flag for an explicit receiver **Tags:**atp.Status=draft	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

⌋(*SRS_Rte_00300*, *SRS_Rte_00301*, *SRS_Rte_00306*, *SRS_BSW_00310*)

D.1.2 Rte_Rips_DRead

[SWS_Rte_91122]{DRAFT} ⌈

Service Name	Rte_Rips_<PlugIn>_DRead_<SwcBswI>[Partition][_<ExE>]_<CGI>(draft)	

Syntax	`<return> Rte_Rips_<PlugIn>_DRead_<SwcBswI>[Partition][_<ExE>]_<CGI> (`
	`  [Std_TransformerError transformerError]`
	`)`	

Service ID [hex]	0xFF	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	transformerError	The OUT parameter transformerError contains the transformer error which occurred during execution of the transformer chain.
Return value	<return>	Return value provides access to the data value of the Variable DataPrototype. For details of the <return> value definition see section 5.2.6.6.
Description	Rte_Rips_DRead Performs an "explicit" read on a sender-receiver communication data element typed by a primitive data type.	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

⌋(*SRS_Rte_00300*, *SRS_Rte_00301*, *SRS_Rte_00306*, *SRS_BSW_00310*, *SRS_Rte_00183*)

D.1.3 Rte_Rips_DataIsUpdated_EventActivation

[SWS_Rte_91000]{DRAFT} ⌈

Service Name	Rte_Rips_<PlugIn>_DataIsUpdatedEventActivation_<SwcBswI>_<DR>_<CGI> (draft)

▽

△

Syntax	`boolean Rte_Rips_<PlugIn>_DataIsUpdatedEventActivation_<SwcBswI>_<DR>_<CGI> ( void )`	
Service ID [hex]	0xB5	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	boolean	The return value is used to indicate if the Runnable shall be activated or not.
Description	The Rte_Rips_DataIsUpdated_EventActivation API indicates the RTE to activate the Runnable Entity triggered by DataReceivedEvent when the related VariableDataPrototype has been updated **Tags:**atp.Status=draft	
Available via	Rte_Rips_<PlugIn>.h	

⌋(*SRS_Rte_00300*, *SRS_Rte_00301*, *SRS_Rte_00306*, *SRS_BSW_00310*)

D.1.4 Rte_Rips_Feedback

[SWS_Rte_91121]{DRAFT} ⌈

Service Name	Rte_Rips_<PlugIn>_Feedback_[<SwcBswI>][Partition]_<CGI> (draft)	
Syntax	`Std_ReturnType Rte_Rips_<PlugIn>_Feedback_[<SwcBswI>][Partition]_<CGI> ( void )`	
Service ID [hex]	0xB6	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	The return value is used to pass error notifications.
Description	The Rte_Rips_Feedback API provides access to acknowledgment notifications for explicit and implicit sender-receiver communication and to pass error notification to senders in a Software Cluster **Tags:**atp.Status=draft	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

⌋(*SRS_Rte_00319*, *SRS_Rte_00306*, *SRS_BSW_00310*)

D.1.5 Rte_Rips_Invoke

[SWS_Rte_89022]{DRAFT} ⌈

Service Name	Rte_Rips_<PlugIn>_Invoke_<SwcBswI>_<CGI>(draft)	

Syntax	`Std_ReturnType Rte_Rips_<PlugIn>_Invoke_<SwcBswI>_<CGI> (`
	`  [IN|IN/OUT|OUT] <data_1>,`
	`  [IN|IN/OUT|OUT] ...,`
	`  [IN|IN/OUT|OUT] <data_n>,`
	`  [Std_TransformerError transformerError]`
	`)`	

Service ID [hex]	0xEC	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	<data_1>	The Rte_Rips_Invoke API includes zero or more IN, IN/OUT and OUT parameters according SWS_Rte_01102 and none in case of triggers
Parameters (inout)	...	The Rte_Rips_Invoke API includes zero or more IN, IN/OUT and OUT parameters according SWS_Rte_01102 and none in case of triggers
Parameters (out)	<data_n>	The Rte_Rips_Invoke API includes zero or more IN, IN/OUT and OUT parameters according SWS_Rte_01102 and none in case of triggers
	transformerError	The OUT parameter transformerError contains the transformer error which occurred during execution of the transformer chain.
Return value	Std_ReturnType	The return value is used to indicate communication errors.
Description	Rte_Rips_Invoke performs a transformer or cross cluster invocation for clients or trigger sources.	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

⌋(*SRS_Rte_00312*, *SRS_Rte_00317*, *SRS_Rte_00306*, *SRS_BSW_00310*)

D.1.6 Rte_Rips_Prm

[SWS_Rte_91116]{DRAFT} ⌈

Service Name	Rte_Rips_<PlugIn>_Prm_<CGI> (draft)

Syntax	`<return> Rte_Rips_<PlugIn>_Prm_<CGI> (`
	`  void`
	`)`

Service ID [hex]	0x100
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None

▽

△

Return value	<return>	For primitive data types, the Rte_Rips_Prm API returns the parameter value. For composite data types, the Rte_Rips_Prm API returns a reference (in C, a pointer) to the constant parameter.
Description	The Rte_Rips_Prm API provides access to a parameter provided by another Software Cluster **Tags:**atp.Status=draft	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

⌋()

D.1.7 Rte_Rips_Read

[SWS_Rte_89020]{DRAFT} ⌈

Service Name	Rte_Rips_<PlugIn>_Read_[<SwcBswI>][Partition][_<ExE>]_<CGI>(draft)	

Syntax	`Std_ReturnType Rte_Rips_<PlugIn>_Read_[<SwcBswI>][Partition][_<ExE>]_<CGI> (`
	`  OUT <data>,`
	`  [Std_TransformerError transformerError]`
	`)`	

Service ID [hex]	0xEA	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	<data>	The OUT parameter <data> pass back the received data.
	transformerError	The OUT parameter transformerError contains the transformer error which occurred during execution of the transformer chain.
Return value	Std_ReturnType	The return value is used to indicate communication errors.
Description	Rte_Rips_Read Performs an "explicit" read on a sender-receiver communication data element.	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

⌋(*SRS_Rte_00300*, *SRS_Rte_00301*, *SRS_Rte_00306*, *SRS_BSW_00310*)

D.1.8 Rte_Rips_ReturnResult

[SWS_Rte_89023]{DRAFT} ⌈

Service Name	Rte_Rips_<PlugIn>_ReturnResult_<SwcBswI>_<CGI>(draft)

▽

△

Syntax	`Std_ReturnType Rte_Rips_<PlugIn>_ReturnResult_<SwcBswI>_<CGI> (`
	`  [IN/OUT|OUT] <param_1>,`
	`  [IN/OUT|OUT] <param_n>,`
	`  [Std_TransformerError transformerError]`
	`)`	

Service ID [hex]	0xED	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	<param_1>	The Rte_Rips_ReturnResult API includes zero or more IN/OUT and OUT parameters according SWS_Rte_01111.
Parameters (out)	<param_n>	The Rte_Rips_ReturnResult API includes zero or more IN/OUT and OUT parameters according SWS_Rte_01111.
	transformerError	The OUT parameter transformerError contains the transformer error which occurred during execution of the transformer chain.
Return value	Std_ReturnType	The return value is used to indicate communication errors
Description	Rte_Rips_ReturnResult get the server results of a performed a transformer or cross cluster invocation for clients.	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

⌋(*SRS_Rte_00312*, *SRS_Rte_00306*, *SRS_BSW_00310*)

D.1.9 Rte_Rips_Start

[SWS_Rte_89017] ⌈

Service Name	Rte_Rips_<PlugIn>_Rte_Start

Syntax	`void Rte_Rips_<PlugIn>_Rte_Start (`
	`  void`
	`)`

Service ID [hex]	0xF1
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Rte_Rips_Rte_Start initializes those RTE Implementation Plug-In parts which are relevant for the RTE related operation.
Available via	Rte_Rips_<PlugIn>.h

⌋()

D.1.10 Rte_Rips_Stop

[SWS_Rte_89018] ⌈

Service Name	Rte_Rips_<PlugIn>_Rte_Stop

Syntax	`void Rte_Rips_<PlugIn>_Rte_Stop (`
	`  void`
	`)`

Service ID [hex]	0xF2
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Rte_Rips_Rte_Stop deinitializes those RTE Implementation Plug-In parts which are relevant for the RTE related operation.
Available via	Rte_Rips_<PlugIn>.h

⌋()

D.1.11 Rte_Rips_SchM_Deinit

[SWS_Rte_89019] ⌈

Service Name	Rte_Rips_SchM_Deinit

Syntax	`void Rte_Rips_SchM_Deinit (`
	`  void`
	`)`

Service ID [hex]	0xF3
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Rte_Rips_SchM_Deinit deinitializes those RTE Implementation Plug-In parts which are relevant for the SchM related operations.
Available via	Rte_Rips_<PlugIn>.h

⌋()

D.1.12 Rte_Rips_SchM_Init

[SWS_Rte_89016] ⌈

Service Name	Rte_Rips_<PlugIn>_SchM_Init
Syntax	`void Rte_Rips_<PlugIn>_SchM_Init ( void )`
Service ID [hex]	0xF0
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Rte_Rips_SchM_Init initializes those RTE Implementation Plug-In parts which are relevant for the SchM related operations.
Available via	Rte_Rips_<PlugIn>.h

⌋()

D.1.13 Rte_Rips_SwitchNotificationStatusType

[SWS_Rte_91113]{DRAFT} ⌈

Name	Rte_Rips_SwitchNotificationStatusType (draft)		
Kind	Type		
Derived from	uint8		
Range	RTE_SWITCH_NOTIFICATION_SKIP	0x00	mode switch notification cannot be dequeued
	RTE_SWITCH_NOTIFICATION_ENQUEUED_FIRST	0x01	mode switch notification is enqueued into an empty mode queue
	RTE_SWITCH_NOTIFICATION_ENQUEUED_NOT_FIRST	0x02	mode switch notification is enqueued into a non empty mode queue
	RTE_SWITCH_NOTIFICATION_ENQUEUE_FAILED	0x03	enqueue operation into a non empty mode queue failed
	RTE_SWITCH_NOTIFICATION_DEQUEUED_LAST	0x04	last mode switch notification was enqueued from mode queue
	RTE_SWITCH_NOTIFICATION_DEQUEUED_NOT_LAST	0x05	mode switch notification was enqueued from mode queue, further mode switch notifications are in the queue
Description	Status of the en- and dequeue operation on a mode queue **Tags:**atp.Status=draft		
Available via	Rte_Type.h		

⌋(*SRS_Rte_00321*, *SRS_Rte_00306*, *SRS_BSW_00310*)

D.1.14 Rte_Rips_Switch

[SWS_Rte_91114]{DRAFT} ⌈

Service Name	Rte_Rips_<PlugIn>_Switch_<BswSwcI>_<MMI> (draft)	
Syntax	`void Rte_Rips_<PlugIn>_Switch_<BswSwcI>_<MMI> ( Rte_Rips_SwitchNotificationStatusType switchNotificationStatus, uint32 previousmode, uint32 nextmode )`	
Service ID [hex]	0xB0	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	switchNotificationStatus	Status of the enqueue operation
	previousmode	The value of the ModeDeclaration of the mode being left
	nextmode	The value of the ModeDeclaration of the mode being entered
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Rte_Rips_StartModeSwitch notifies the RTE Implementation Plug-In about an enqueue operation in a mode queue. **Tags:**atp.Status=draft	
Available via	Rte_Rips_<PlugIn>.h	

⌋(*SRS_Rte_00321*, *SRS_Rte_00306*, *SRS_BSW_00310*)

D.1.15 Rte_Rips_DequeueModeSwitch

[SWS_Rte_91115]{DRAFT} ⌈

Service Name	Rte_Rips_<PlugIn>_DequeueModeSwitch_<MMI>_<OsTask> (draft)	
Syntax	`Rte_Rips_SwitchNotificationStatusType Rte_Rips_<PlugIn> _DequeueModeSwitch_<MMI>_<OsTask> ( void )`	
Service ID [hex]	0xB1	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	Rte_Rips_Switch NotificationStatusType	The return value is used indicate the status of the dequeue operation in a mode queue

▽

△

Description	Rte_Rips_DequeueModeSwitch dequeues a mode switch notification from the mode queue when it is called after the last on-entry ExecutableEntity terminated. **Tags:**atp.Status=draft
Available via	Rte_Buffers.h

⌋(*SRS_Rte_00321*, *SRS_Rte_00306*, *SRS_BSW_00310*)

D.1.16 Rte_Rips_Trigger

[SWS_Rte_91117]{DRAFT} ⌈

Service Name	Rte_Rips_<PlugIn>_Trigger_<BswSwcI>_<MMI> (draft)

Syntax	`void Rte_Rips_<PlugIn>_Trigger_<BswSwcI>_<MMI> (`
	`  void`
	`)`

Service ID [hex]	0xB2
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Rte_Rips_Trigger notifies the RTE Implementation Plug-In about a raised trigger. **Tags:**atp.Status=draft
Available via	Rte_Rips_<PlugIn>.h

⌋(*SRS_Rte_00317*, *SRS_Rte_00306*, *SRS_BSW_00310*)

D.1.17 Rte_Rips_Write

[SWS_Rte_89021]{DRAFT} ⌈

Service Name	Rte_Rips_<PlugIn>_Write_[<SwcBswI>][Partition][_<ExE>]_<CGI>(draft)

Syntax	`Std_ReturnType Rte_Rips_<PlugIn>_Write_[<SwcBswI>][Partition][_<ExE>]_`
	`  <CGI> (`
	`  IN <data>,`
	`  [Std_TransformerError transformerError]`
	`)`

Service ID [hex]	0xEB
Sync/Async	Synchronous
Reentrancy	Reentrant

▽

△

Parameters (in)	<data>	The IN parameter <data> pass the received data.
Parameters (inout)	None	
Parameters (out)	transformerError	The OUT parameter transformerError contains the transformer error which occurred during execution of the transformer chain.
Return value	Std_ReturnType	The return value is used to indicate communication errors.
Description	Rte_Rips_Write Performs an "explicit" write on a sender-receiver communication data element.	
Available via	Rte_Rips_<PlugIn>_<SwcBswI>.h	

⌋(*SRS_Rte_00300*, *SRS_Rte_00301*, *SRS_Rte_00306*, *SRS_BSW_00310*)

D.2 OS

See document [12] and [16] as reference for `Os`.

- `TASK`
- `ActivateTask`
- `ChainTask`
- `TerminateTask`
- `GetResource`
- `ReleaseResource`
- `SuspendOSInterrupts`
- `ResumeOSInterrupts`
- `GetSpinlock`
- `ReleaseSpinlock`
- `GetApplicationID`
- `StartScheduleTable`
- `StopScheduleTable`
- `Schedule`
- `DisableAllInterrupts`
- `EnableAllInterrupts`
- `SuspendAllInterrupts`
- `ResumeAllInterrupts`
- `GetResource`
- `ReleaseResource`

- SuspendOSInterrupts
- ResumeOSInterrupts
- ReleaseResource
- GetSpinlock
- ReleaseSpinlock
- GetApplicationID

D.3 NvM

D.3.1 NvM_CancelJobs

[SWS_NvM_00535] ⌈

Service Name	NvM_CancelJobs	
Syntax	`Std_ReturnType NvM_CancelJobs ( NvM_BlockIdType BlockId )`	
Service ID [hex]	0x10	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The job was successfully removed from queue. E_NOT_OK: The job could not be found in the queue.
Description	Service to cancel all jobs pending for a NV block.	
Available via	NvM.h	

⌋(*SRS_Mem_08560*)

D.3.2 NvM_EraseNvBlock

[SWS_NvM_00457] ⌈

Service Name	NvM_EraseNvBlock
Syntax	`Std_ReturnType NvM_EraseNvBlock ( NvM_BlockIdType BlockId )`

▽

△

Service ID [hex]	0x09	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: request has been accepted E_NOT_OK: request has not been accepted
Description	Service to erase a NV block.	
Available via	NvM.h	

⌋(*SRS_Mem_08544*)

D.3.3 NvM_GetDataIndex

[SWS_NvM_00449] ⌈

Service Name	NvM_GetDataIndex	

Syntax	`Std_ReturnType NvM_GetDataIndex (`
	`  NvM_BlockIdType BlockId,`
	`  uint8* DataIndexPtr`
	`)`	

Service ID [hex]	0x02	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	DataIndexPtr	Pointer to where to store the current dataset index (0..255)
Return value	Std_ReturnType	E_OK: The index position has been retrieved successfully. E_NOT_OK: An error occurred.
Description	Service for getting the currently set DataIndex of a dataset NVRAM block	
Available via	NvM.h	

⌋()

D.3.4 NvM_GetErrorStatus

[SWS_NvM_00451] ⌈

Service Name	NvM_GetErrorStatus	
Syntax	`Std_ReturnType NvM_GetErrorStatus ( NvM_BlockIdType BlockId, NvM_RequestResultType* RequestResultPtr )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	RequestResultPtr	Pointer to where to store the request result. See NvM_Request ResultType .
Return value	Std_ReturnType	E_OK: The block dependent error/status information was read successfully. E_NOT_OK: An error occured.
Description	Service to read the block dependent error/status information.	
Available via	NvM.h	

⌋(*SRS_Mem_00020*)

D.3.5 NvM_InvalidateNvBlock

[SWS_NvM_00459] ⌈

Service Name	NvM_InvalidateNvBlock	
Syntax	`Std_ReturnType NvM_InvalidateNvBlock ( NvM_BlockIdType BlockId )`	
Service ID [hex]	0x0b	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: request has been accepted E_NOT_OK: request has not been accepted
Description	Service to invalidate a NV block.	
Available via	NvM.h	

⌋(*SRS_Mem_08011*)

D.3.6 NvM_ReadBlock

[SWS_NvM_00454] ⌈

Service Name	NvM_ReadBlock	
Syntax	`Std_ReturnType NvM_ReadBlock ( NvM_BlockIdType BlockId, void* NvM_DstPtr )`	
Service ID [hex]	0x06	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	NvM_DstPtr	Pointer to the RAM data block.
Return value	Std_ReturnType	E_OK: request has been accepted E_NOT_OK: request has not been accepted
Description	Service to copy the data of the NV block to its corresponding RAM block.	
Available via	NvM.h	

⌋(*SRS_LIBS_08533*, *SRS_Mem_00016*)

D.3.7 NvM_ReadPRAMBlock

[SWS_NvM_00764] ⌈

Service Name	NvM_ReadPRAMBlock	
Syntax	`Std_ReturnType NvM_ReadPRAMBlock ( NvM_BlockIdType BlockId )`	
Service ID [hex]	0x16	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: request has been accepted E_NOT_OK: request has not been accepted
Description	Service to copy the data of the NV block to its corresponding permanent RAM block.	
Available via	NvM.h	

⌋(*SRS_LIBS_08533*, *SRS_Mem_00016*)

D.3.8 NvM_RestoreBlockDefaults

[SWS_NvM_00456] ⌈

Service Name	NvM_RestoreBlockDefaults	
Syntax	`Std_ReturnType NvM_RestoreBlockDefaults ( NvM_BlockIdType BlockId, void* NvM_DestPtr )`	
Service ID [hex]	0x08	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	NvM_DestPtr	Pointer to the RAM data block.
Return value	Std_ReturnType	E_OK: request has been accepted E_NOT_OK: request has not been accepted
Description	Service to restore the default data to its corresponding RAM block.	
Available via	NvM.h	

⌋(*SRS_Mem_00018*)

D.3.9 NvM_RestorePRAMBlockDefaults

[SWS_NvM_00813] ⌈

Service Name	NvM_RestorePRAMBlockDefaults	
Syntax	`Std_ReturnType NvM_RestorePRAMBlockDefaults ( NvM_BlockIdType BlockId )`	
Service ID [hex]	0x18	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: request has been accepted E_NOT_OK: request has not been accepted
Description	Service to restore the default data to its corresponding permanent RAM block.	
Available via	NvM.h	

⌋(*SRS_Mem_00018*)

D.3.10 NvM_SetBlockLockStatus

[SWS_NvM_00548] ⌈

Service Name	NvM_SetBlockLockStatus	
Syntax	`void NvM_SetBlockLockStatus ( NvM_BlockIdType BlockId, boolean BlockLocked )`	
Service ID [hex]	0x13	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
	BlockLocked	TRUE: Mark the RAM.block as locked FALSE: Mark the RAM.block as unlocked
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service for setting the lock status of a permanent RAM block or of the explicit synchronization of a NVRAM block.	
Available via	NvM.h	

⌋(*SRS_Mem_08546*)

D.3.11 NvM_SetBlockProtection

[SWS_NvM_00450] ⌈

Service Name	NvM_SetBlockProtection	
Syntax	`Std_ReturnType NvM_SetBlockProtection ( NvM_BlockIdType BlockId, boolean ProtectionEnabled )`	
Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
	ProtectionEnabled	TRUE: Write protection shall be enabled FALSE: Write protection shall be disabled
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The block was enabled/disabled as requested E_NOT_OK: An error occured.

▽

△

Description	Service for setting/resetting the write protection for a NV block.
Available via	NvM.h

⌋(*SRS_Mem_00127*)

D.3.12 NvM_SetDataIndex

[SWS_NvM_00448] ⌈

Service Name	NvM_SetDataIndex	
Syntax	`Std_ReturnType NvM_SetDataIndex ( NvM_BlockIdType BlockId, uint8 DataIndex )`	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
	DataIndex	Index position (association) of a NV/ROM block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The index position was set successfully. E_NOT_OK: An error occurred.
Description	Service for setting the DataIndex of a dataset NVRAM block.	
Available via	NvM.h	

⌋(*SRS_Mem_08007*)

D.3.13 NvM_SetRamBlockStatus

[SWS_NvM_00453] ⌈

Service Name	NvM_SetRamBlockStatus
Syntax	`Std_ReturnType NvM_SetRamBlockStatus ( NvM_BlockIdType BlockId, boolean BlockChanged )`
Service ID [hex]	0x05
Sync/Async	Synchronous
Reentrancy	Reentrant

▽

△

Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
	BlockChanged	TRUE: Validate the permanent RAM block or the explicit synchronization and mark block as changed. FALSE: Invalidate the permanent RAM block or the explicit synchronization and mark block as unchanged.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The status of the permanent RAM block or the explicit synchronization was changed as requested. E_NOT_OK: An error occurred.
Description	Service for setting the RAM block status of a permanent RAM block or the status of the explicit synchronization of a NVRAM block.	
Available via	NvM.h	

⌋(*SRS_Mem_08545*)

D.3.14 NvM_WriteBlock

[SWS_NvM_00455] ⌈

Service Name	NvM_WriteBlock	

Syntax	`Std_ReturnType NvM_WriteBlock (`
	`  NvM_BlockIdType BlockId,`
	`  const void* NvM_SrcPtr`
	`)`	

Service ID [hex]	0x07	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
	NvM_SrcPtr	Pointer to the RAM data block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: request has been accepted E_NOT_OK: request has not been accepted
Description	Service to copy the data of the RAM block to its corresponding NV block.	
Available via	NvM.h	

⌋(*SRS_Mem_00017*)

D.3.15 NvM_WritePRAMBlock

[SWS_NvM_00793] ⌈

Service Name	NvM_WritePRAMBlock	
Syntax	`Std_ReturnType NvM_WritePRAMBlock ( NvM_BlockIdType BlockId )`	
Service ID [hex]	0x17	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	BlockId	The block identifier uniquely identifies one NVRAM block descriptor. A NVRAM block descriptor contains all needed information about a single NVRAM block.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: request has been accepted E_NOT_OK: request has not been accepted
Description	Service to copy the data of the permanent RAM block to its corresponding NV block.	
Available via	NvM.h	

⌋(*SRS_Mem_00017*)

D.3.16 NvM_SingleBlockCallbackFunction

[SWS_NvM_00467] ⌈

Service Name	NvM_SingleBlockCallbackFunction	
Syntax	`Std_ReturnType NvM_SingleBlockCallbackFunction ( NvM_BlockRequestType BlockRequest, NvM_RequestResultType JobResult )`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	BlockRequest	The request type (read, write, ... etc.) of the previous processed block job
	JobResult	The request result of the previous processed block job.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: callback function has been processed successfully any other: callback function has been processed unsuccessfully
Description	Per block callback routine to notify the upper layer that an asynchronous single block request has been finished.	
Available via	NvM_Externals.h	

⌋(*SRS_BSW_00457*, *SRS_BSW_00360*, *SRS_BSW_00333*)

D.3.17 NvM_InitBlockCallbackFunction

[SWS_NvM_00469] ⌈

Service Name	NvM_InitBlockCallbackFunction	
Syntax	`Std_ReturnType NvM_InitBlockCallbackFunction ( NvM_InitBlockRequestType InitBlockRequest )`	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	InitBlockRequest	The request type (read, restore, ... etc.) of the currently processed block
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: callback function has been processed successfully any other: callback function has been processed unsuccessfully
Description	Per block callback routine which shall be called by the NvM module when default data needs to be restored in RAM, and a ROM block is not configured.	
Available via	NvM_Externals.h	

⌋(*SRS_BSW_00457*, *SRS_BSW_00360*, *SRS_BSW_00333*)

D.3.18 NvM_ReadRamBlockFromNvm

[SWS_NvM_00540] ⌈

Service Name	NvM_ReadRamBlockFromNvm	
Syntax	`Std_ReturnType NvM_ReadRamBlockFromNvm ( const void* NvMBuffer )`	
Service ID [hex]		
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	NvMBuffer	the address of the buffer where the data can be read from
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: callback function has been processed successfully any other: callback function has been processed unsuccessfully
Description	Block specific callback routine which shall be called in order to let the application copy data from NvM module's mirror to RAM block.	
Available via	NvM_Externals.h	

⌋(*SRS_LIBS_08533*, *SRS_BSW_00457*)

D.3.19 NvM_WriteRamBlockToNvm

[SWS_NvM_00539] ⌈

Service Name	NvM_WriteRamBlockToNvm	
Syntax	`Std_ReturnType NvM_WriteRamBlockToNvm ( void* NvMBuffer )`	
Service ID [hex]		
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	NvMBuffer	the address of the buffer where the data shall be written to
Return value	Std_ReturnType	E_OK: callback function has been processed successfully any other: callback function has been processed unsuccessfully
Description	Block specific callback routine which shall be called in order to let the application copy data from RAM block to NvM module's mirror.	
Available via	NvM_Externals.h	

⌋(*SRS_BSW_00457*)

E Referenced Service Interfaces

E.1 Os

E.2 NvM

E.2.1 NvM_BlockIdType

[SWS_NvM_00471] ⌈

Name	NvM_BlockIdType		
Kind	Type		
Derived from	uint16		
Range	0..2^(16- NvMDatasetSelection Bits)-1	–	–
Description	Identification of a NVRAM block via a unique block identifier. Reserved NVRAM block IDs: 0 -> to derive multi block request results via NvM_GetErrorStatus 1 -> redundant NVRAM block which holds the configuration ID		
Variation	–		
Available via	Rte_NvM_Type.h		

⌋()

E.2.2 NvM_BlockRequestType

[SWS_NvM_91002] ⌈

Name	NvM_BlockRequestType		
Kind	Type		
Derived from	uint8		
Range	NVM_READ_BLOCK	0x00	NvM_ReadBlock/ NvM_Read PRAMBlock was performed on the block
	NVM_WRITE_BLOCK	0x01	NvM_WriteBlock/ NvM_Write PRAMBlock was performed on the block
	NVM_RESTORE_BLOCK_ DEFAULTS	0x02	NvM_RestoreBlockDefaults/ Nv M_RestorePRAMBlockDefaults was performed on the block
	NVM_ERASE_NV_BLOCK	0x03	NvM_EraseNvBlock was performed on the block
	NVM_INVALIDATE_NV_BLOCK	0x04	NvM_InvalidateNvBlock was performed on the block
	NVM_READ_ALL_BLOCK	0x05	NvM_ReadAll has finished processing this block
Description	Identifies the type of request performed on a block when signaled via the callback function		
Variation	–		
Available via	Rte_NvM_Type.h		

⌋()

E.2.3 NvM_InitBlockRequestType

[SWS_NvM_00471] ⌈

Name	NvM_BlockIdType		
Kind	Type		
Derived from	uint16		
Range	0..2^(16- NvMDatasetSelection Bits)-1	–	–
Description	Identification of a NVRAM block via a unique block identifier. Reserved NVRAM block IDs: 0 -> to derive multi block request results via NvM_GetErrorStatus 1 -> redundant NVRAM block which holds the configuration ID		
Variation	–		
Available via	Rte_NvM_Type.h		

⌋()

E.2.4 NvM_RequestResultType

[SWS_NvM_00470] ⌈

Name	NvM_RequestResultType		
Kind	Type		
Derived from	uint8		
Range	NVM_REQ_OK	0x00	The last asynchronous request has been finished successfully. This shall be the default value after reset. This status shall have the value 0.
	NVM_REQ_NOT_OK	0x01	The last asynchronous read/write/ control request has been finished unsuccessfully.
	NVM_REQ_PENDING	0x02	An asynchronous read/write/ control request is currently pending.
	NVM_REQ_INTEGRITY_FAILED	0x03	The result of the last asynchronous request NvM_ReadBlock or NvM_ReadAll is a data integrity failure. Note: In case of NvM_ReadBlock the content of the RAM block has changed but has become invalid. The application is responsible to renew and validate the RAM block content.
	NVM_REQ_BLOCK_SKIPPED	0x04	The referenced block was skipped during execution of NvM_ReadAll or NvM_WriteAll, e.g. Dataset NVRAM blocks (NvM_ReadAll) or NVRAM blocks without a permanently configured RAM block.
	NVM_REQ_NV_INVALIDATED	0x05	The referenced NV block is invalidated.
	NVM_REQ_CANCELED	0x06	The multi block request NvM_ WriteAll was canceled by calling NvM_CancelWriteAll. Or Any single block job request (NvM_ ReadBlock, NvM_WriteBlock, Nv M_EraseNvBlock, NvM_Invalidate NvBlock and NvM_RestoreBlock Defaults) was canceled by calling NvM_CancelJobs.
	NVM_REQ_RESTORED_ DEFAULTS	0x08	The referenced NV block had the default values copied to the RAM image.
Description	This is an asynchronous request result returned by the API service NvM_GetErrorStatus. The availability of an asynchronous request result can be additionally signaled via a callback function.		
Variation	–		
Available via	Rte_NvM_Type.h		

⌋()

E.2.5 NvMService

[SWS_NvM_00734] ⌈

Name	NvMService		
Comment	–		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	EraseBlock
Comment	Service to erase a NV block.

Variation	`FOR`
	`  configClass  : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass3 = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`WHERE`
	`  isConfigClass3;`

Possible Errors	E_OK E_NOT_OK

Operation	GetDataIndex	
Comment	Service for getting the currently set DataIndex of a dataset NVRAM block	

Variation	`FOR`
	`  configClass  : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass2 = configClass.value() == "NVM_API_CONFIG_CLASS_2";`
	`WHERE`
	`  isConfigClass2;`	

Parameters	DataIndex	
	Type	uint8
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	GetErrorStatus	
Comment	Service to read the block dependent error/status information.	
Variation	–	
Parameters	RequestResult	
	Type	NvM_RequestResultType
	Direction	OUT
	Comment	–
	Variation	–

▽

△

Possible Errors	E_OK E_NOT_OK

Operation	InvalidateNvBlock
Comment	Service to invalidate a NV block.

Variation	`FOR`
	`  configClass  : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass3 = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`WHERE`
	`  isConfigClass3;`

Possible Errors	E_OK E_NOT_OK

Operation	ReadBlock	
Comment	Service to copy the data of the NV block to its corresponding RAM block.	

Variation	`FOR`
	`  configClass  : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass2 = configClass.value() == "NVM_API_CONFIG_CLASS_2";`
	`  isConfigClass3 = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`WHERE`
	`  isConfigClass2 OR isConfigClass3;`	

Parameters	DstPtr	
	Type	VoidPtr
	Direction	IN
	Comment	The parameter "DstPtr" shall be typed by an ImplementationDataType of category DATA_REFERENCE with the pointer target void to pass an address (pointer) to the RAM Block.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	ReadPRAMBlock
Comment	–

Variation	`FOR`
	`  configClass  : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass2 = configClass.value() == "NVM_API_CONFIG_CLASS_2";`
	`  isConfigClass3 = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`WHERE`
	`  isConfigClass2 OR isConfigClass3;`

Possible Errors	E_OK E_NOT_OK

Operation	RestoreBlockDefaults	
Comment	Service to restore the default data to its corresponding RAM block.	

Variation	`FOR`
	`  configClass  : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass2 = configClass.value() == "NVM_API_CONFIG_CLASS_2";`
	`  isConfigClass3 = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`WHERE`
	`  isConfigClass2 OR isConfigClass3;`	

Parameters	DstPtr	
	Type	VoidPtr
	Direction	IN
	Comment	The parameter "DstPtr" shall be typed by an ImplementationDataType of category DATA_REFERENCE with the pointer target void to pass an address (pointer) to the RAM Block.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	RestorePRAMBlockDefaults
Comment	–

Variation	`FOR`
	`  configClass  : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass2 = configClass.value() == "NVM_API_CONFIG_CLASS_2";`
	`  isConfigClass3 = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`WHERE`
	`  isConfigClass2 OR isConfigClass3;`

Possible Errors	E_OK E_NOT_OK

Operation	SetDataIndex	
Comment	Service for setting the DataIndex of a dataset NVRAM block.	

Variation	`FOR`
	`  configClass  : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass2  = configClass.value() == "NVM_API_CONFIG_CLASS_2";`
	`  isConfigClass3  = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`  blockMgmTypes  = ECV.subEltList("NvM/NvMBlockDescriptor/`
	`  NvMBlockManagementType");`
	`  isMgd(mgmtType) = mgmtType.value() == "NVM_BLOCK_DATASET";`
	`  datasetMgdCount = blockMgmTypes.filter(isMgd).count();`
	`WHERE`
	`  (isConfigClass2 OR isConfigClass3) AND (datasetMgdCount GT 0);`	

Parameters	DataIndex	
	Type	uint8
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	SetRamBlockStatus	
Comment	Service for setting the RAM block status of an NVRAM block.	

Variation	`LET`
	`  nvmBlockUseSetRamBlockStatus = ECV.subEltList("NvM/NvMBlockDescriptor/NvMBlockUseSetRamBlockStatus");`
	`  useSetRamBlockStatus(useApi) = useApi.value() == true;`
	`  useSetRamBlockStatusCount = nvmBlockUseSetRamBlockStatus.filter(useSetRamBlockStatus).count();`
	`WHERE`
	`  (useSetRamBlockStatusCount GT 0);`	

Parameters	BlockChanged	
	Type	boolean
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	WriteBlock	
Comment	Service to copy the data of the RAM block to its corresponding NV block.	

Variation	`FOR`
	`  configClass : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass2 = configClass.value() == "NVM_API_CONFIG_CLASS_2";`
	`  isConfigClass3 = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`WHERE`
	`  isConfigClass2 OR isConfigClass3;`	

Parameters	SrcPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	The parameter "SrcPtr" shall be typed by an ImplementationDataType of category DATA_REFERENCE with the pointer target void to pass an address (pointer) to the RAM Block.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	WritePRAMBlock	
Comment	–	

Variation	`FOR`
	`  configClass : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass2 = configClass.value() == "NVM_API_CONFIG_CLASS_2";`
	`  isConfigClass3 = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`WHERE`
	`  isConfigClass2 OR isConfigClass3;`	

Possible Errors	E_OK E_NOT_OK	

](*)

E.2.6 NvMAdmin

[SWS_NvM_00737] ⌈

Name	NvMAdmin		
Comment	–		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	SetBlockProtection	
Comment	Service for setting/resetting the write protection for a NV block.	

Variation	`FOR`
	`  configClass  : ECV.subEltList("NvM/NvMCommon/NvMApiConfigClass");`
	`LET`
	`  isConfigClass3 = configClass.value() == "NVM_API_CONFIG_CLASS_3";`
	`WHERE`
	`  isConfigClass3;`	

Parameters	ProtectionEnabled	
	Type	boolean
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

⌋()

E.2.7 NvMNotifyJobFinished

[SWS_NvM_00735] ⌈

Name	NvMNotifyJobFinished		
Comment	Callback that is called when a job has finished		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful

Operation	JobFinished
Comment	Callback that gets called if a job has finished
Variation	–
Parameters	BlockRequest

▽

△

	Type	NvM_BlockRequestType
	Direction	IN
	Comment	–
	Variation	–
	JobResult	
	Type	NvM_RequestResultType
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK	

⌋()

E.2.8 NvMNotifyInitBlock

[SWS_NvM_00736] ⌈

Name	NvMNotifyInitBlock		
Comment	Callback that is called by the NvM module when default data needs to be restored to the RAM image		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	RAM block content was updated
	1	RTE_E_RAM_ UNCHANGED	RAM block content was not changed

Operation	InitBlock	
Comment	This callback is called if the initialization of a block has completed.	
Variation	–	
Parameters	InitBlockRequest	
	Type	NvM_InitBlockRequestType
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	–	

⌋()

E.2.9 NvMMirror

[SWS_NvM_00738] ⌈

Name	NvMMirror		
Comment	–		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed

Operation	ReadRamBlockFromNvM	
Comment	Block specific callback routine which shall be called in order to let the application copy data from NvM module's mirror to RAM block.	
Variation	–	
Parameters	SrcPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	The parameter "SrcPtr" shall be typed by an ImplementationDataType of category DATA_REFERENCE with the pointer target void to pass an address (pointer) to the RAM Block.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	WriteRamBlockToNvM	
Comment	Block specific callback routine which shall be called in order to let the application copy data from RAM block to NvM module's mirror.	
Variation	–	
Parameters	DstPtr	
	Type	VoidPtr
	Direction	IN
	Comment	The parameter "DstPtr" shall be typed by an ImplementationDataType of category DATA_REFERENCE with the pointer target void to pass an address (pointer) to the RAM Block.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

⌋()

E.2.10 PS_{Block}

[SWS_NvM_00847] ⌈

Name	PS_{Block}		
Kind	ProvidedPort	***Interface***	NvMService
Description	–		

▽

△

Port Defined Argument Value(s)	***Type***	NvM_BlockIdType
	Value	`FOR nvBlockDescriptor : ECV.subEltList("NvM/NvMBlockDescriptor"); LET Block = nvBlockDescriptor.shortname(); BlockId = nvBlockDescriptor.subElt("NvMNvramBlockIdentifier").value();`
Variation		`FOR nvBlockDescriptor : ECV.subEltList("NvM/NvMBlockDescriptor"); LET Block = nvBlockDescriptor.shortname(); UsePort = nvBlockDescriptor.subElt("NvMBlockUsePort").value() == true; WHERE UsePort;`

⌋()

E.2.11 PAdmin_{Block}

[SWS_NvM_00843] ⌈

Name	PAdmin_{Block}		
Kind	ProvidedPort	***Interface***	NvMAdmin
Description	–		
Port Defined Argument Value(s)	***Type***	NvM_BlockIdType	
	Value	`FOR nvBlockDescriptor : ECV.subEltList("NvM/NvMBlockDescriptor"); LET Block = nvBlockDescriptor.shortname(); BlockId = nvBlockDescriptor.subElt("NvMNvramBlockIdentifier").value();`	
Variation		`FOR nvBlockDescriptor : ECV.subEltList("NvM/NvMBlockDescriptor"); LET Block = nvBlockDescriptor.shortname(); UsePort = nvBlockDescriptor.subElt("NvMBlockUsePort").value() == true; WHERE UsePort;`	

⌋()

E.2.12 PNJF_{Block}

[SWS_NvM_00846] ⌈

Name	PNJF_{Block}		
Kind	RequiredPort	Interface	NvMNotifyJobFinished
Description	–		

Variation	FOR
	  nvBlockDescriptor : ECV.subEltList("NvM/NvMBlockDescriptor");
	LET
	  Block = nvBlockDescriptor.shortname();
	  UsePort = nvBlockDescriptor.subElt("NvMBlockUsePort").
	  value() == true;
	  SingleBlockCallbackDef = nvBlockDescriptor.subElt("
	  NvMSingleBlockCallback").isDefined();
	  SingleBlockCallbackFncDef = nvBlockDescriptor.subElt("
	  NvMSingleBlockCallback/NvMSingleBlockCallbackFnc").isDefined();
	WHERE
	  UsePort AND SingleBlockCallbackDef AND NOT SingleBlockCallbackFncDef;		

⌋()

E.2.13 PNIB_{Block}

[SWS_NvM_00845] ⌈

Name	PNIB_{Block}		
Kind	RequiredPort	Interface	NvMNotifyInitBlock
Description	–		

Variation	FOR
	  nvBlockDescriptor : ECV.subEltList("NvM/NvMBlockDescriptor");
	LET
	  Block = nvBlockDescriptor.shortname();
	  UsePort = nvBlockDescriptor.subElt("NvMBlockUsePort").
	  value() == true;
	  InitBlockCallbackDef = nvBlockDescriptor.subElt("
	  NvMInitBlockCallback").isDefined();
	  InitBlockCallbackFncDef = nvBlockDescriptor.subElt("
	  NvMInitBlockCallback/NvMInitBlockCallbackFnc").isDefined();
	WHERE
	  UsePort AND InitBlockCallbackDef AND NOT InitBlockCallbackFncDef;		

⌋()

E.2.14 PM_{Block}

[SWS_NvM_00844] ⌈

Name	PM_{Block}		
Kind	RequiredPort	***Interface***	NvMMirror
Description	–		

Variation	**FOR**
	  nvBlockDescriptor : ECV.*subEltList*("NvM/NvMBlockDescriptor");
	LET
	  Block = nvBlockDescriptor.*shortname*();
	  UsePort = nvBlockDescriptor.*subElt*("NvMBlockUsePort").*value*()
	  == true;
	  UsePortSyncMech = nvBlockDescriptor.*subElt*("NvMBlockUseSyncMechanism")
	  .*value*() == true;
	WHERE
	  UsePort **AND** UsePortSyncMech;		

⌋()