Document Title	Requirements on Software Cluster Connection
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	973

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	Intitial release

Disclaimer

Table of Contents

1 Scope of Document

The goal of AUTOSAR and of this document is to define the requirements on the software clusters connection in the Classic Platform.

2 Conventions to be used

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

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

2.1 Requirements Guidelines

As described in section 4.1 the Software Cluster Connection has several logical subunits which is expressed by the different nouns in the requirements.

- Binary Manifest
- Cross Cluster Communication
- Proxy Module
  - High Proxy Module
  - Low Proxy Module

3 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 3.1: Acronyms and Abbreviations

Terms:	Description:
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.

Terms:	Description:
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	The Software Cluster which contains mainly software components only selected BSW modules (e.g. a Service module, transformers, e.t.c.)
Host Software Cluster	The Software Cluster which contains major part of the BSW and especially the micro controller dependent lower layer BSW Modules, e.g. OS and MCAL.
Substitution Software Cluster	A Software Cluster which 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	A part of the Proxy Module residing in the Applicative Software Cluster.
Low Proxy Module	The part of the Proxy Module residing 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).
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` which handles the communication towards other `Software Cluster`. This includes the Transformer handling if intra ECU transformation is configured.

Table 3.2: Terms

4 Requirements Specification

This chapter describes all requirements driving the work to define the Software Cluster Connection.

4.1 Functional Overview

The overall software of a Classic Platform Architecture can be split into `Software Cluster`s. Each `Software Cluster` is an independent Build Unit and the result of the cluster specific build processes are the Binary Objects. The Software Cluster Connection provides the ability to

- connect the Binary Objects deployed on the same machine
- substitute not locally available BSW modules in an Applicative Software Cluster, which 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

4.2 Functional Requirements

4.2.1 Requirements on Binary Manifest

This section contains the requirements on the Binary Manifest enabling the connection of software clusters on the basis of Binary Objects.

[SRS_SwCluC_00001] Easy target machine interpretation ⌈

Type:	draft
Description:	The Binary Manifest of the Software Cluster Connection shall be interpretable on the target machine with low resource consumption (ROM, RAM, runtime)
Rationale:	–
Dependencies:	–
Use Case:	Connect Binary Objects after reprogramming of a single Software Cluster on the target machine, which is a micro controller with limited resources.
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00014] Standardized persistence in memory ⌈

Type:	draft
Description:	The Binary Manifest of the Software Cluster Connection shall have a standardized persistence in memory.
Rationale:	Support utilization of Software Cluster Connector tools and implementations from different vendors.
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00002] Identification of intended connections by unique IDs ⌈

Type:	draft
Description:	The Binary Manifest of the Software Cluster Connection shall support the identification of to be connected provided and required resources by unique IDs
Rationale:	Binary Manifest shall be very compact (UUIDs are too memory consuming) Short hashes are not sufficiently deterministic to avoid unintended number collisions.
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00003] Bidirectional connections ⌈

Type:	draft
Description:	The Binary Manifest of the Software Cluster Connection shall support bidirectional exchange of handles between a resource provider and resource requester.
Rationale:	–
Dependencies:	–
Use Case:	The interfacing between a user of a BSW Service in one Software Cluster and the BSW Service module in another Software Cluster requires on the one hand the knowledge about interface IDs of the BSW module by the service user and on the other hand the service user has to provide the information about callback functions towards the Software Cluster of the BSW Module.
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00004] Connection multiplicity ⌈

Type:	draft
Description:	The Binary Manifest of the Software Cluster Connection shall support following connection multiplicities: A resource is provided and can be required by at most one SWCL A resource is provided and can be required by n (n=>1) Software Clusters. In this case the resource can serve for multiple requesters.
Rationale:	–
Dependencies:	–

▽

△

Use Case:	The interfacing between a user of a BSW Service in one Software Cluster and the BSW Service module in another Software Cluster requires on the one hand the knowledge about interface IDs of the BSW module by the service user and on the other hand the service user has to provide the information about callback functions towards the Software Cluster of the BSW Module.
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00005] Substitute Resource Providers ⌈

Type:	draft
Description:	The Binary Manifest of the Software Cluster Connection shall support that a Resource is provided by a substitute instead of the original Resource Provider
Rationale:	–
Dependencies:	–
Use Case:	Allows patching of signals without updating the originally providing Software Cluster by just by adding a Software Cluster providing sane value or calculating a replacement value and rerouting those signals (i.e. data elements on ports)
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00006] C API ⌈

Type:	draft
Description:	The Binary Manifest of the Software Cluster Connection shall provide a well-defined C Interface towards other sub-units of the Software Cluster Connection and Complex Drivers in the Software Cluster. The C-Interface shall support following features: • Get handles of resources from the Binary Manifest • Get information weather and which SWCL is connected • Provide an indexed access for a group of handles • Get informative fields of resources from the Binary Manifest
Rationale:	The inner structure of the Binary Manifest is driven by the connection use case and may change for different target machines. Such dependencies shall not be propagated into the other functional sub-units of the Software Cluster Connection. Modular design of the Software Cluster Connection.
Dependencies:	–
Use Case:	Create a project specific Complex Driver which is Software Cluster aware.

▽

△

Supporting Material:	–

⌋()

[SRS_SwCluC_00007] Standardized configuration ⌈

Type:	draft
Description:	The Binary Manifest of the Software Cluster Connection shall provide a standardized configuration towards other sub-units of the Software Cluster Connection and Complex Drivers in the Software Cluster.
Rationale:	Modular design of the Software Cluster Connection. Create a project specific Complex Driver which is Software Cluster aware and requires entries in the Binary Manifest.
Dependencies:	–
Use Case:	Create a project specific Complex Driver which is Software Cluster aware.
Supporting Material:	–

⌋()

[SRS_SwCluC_00008] Retrieve connection status and connected Software Cluster ⌈

Type:	draft
Description:	The Binary Manifest of the Software Cluster Connection shall support to retrieve whether an interface is connected to a software cluster and to which Software Cluster it is connected.
Rationale:	–
Dependencies:	–
Use Case:	RTE shall return unconnected information in case a required port stays unconnected. Disabling scheduling of a specific Software Cluster. RTE shall return unconnected information in case a required port stays unconnected. Disabling scheduling of a specific Software Cluster.
Supporting Material:	–

⌋()

[SRS_SwCluC_00009] Support missing interface partners ⌈

Type:	draft
Description:	The Binary Manifest of the Software Cluster Connection shall support that required and provided resources stay unconnected. In this case default handles shall substitute the missing interface partner.
Rationale:	–
Dependencies:	–
Use Case:	Support stepwise introduction of new interfaces. Support independent development and stepwise integration of Software Clusters. Support Software Clusters with a super-set of interfaces.
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00010] Static safeguard of Software Cluster connections ⌈

Type:	draft
Description:	The Binary Manifest of the Software Cluster Connection shall support to safeguard the connections between Software Clusters. This means the Binary Manifest shall enable the storage of information which can be used to ensure that only required and provided resources of Software Clusters are connected, if the fundamental properties are equal.
Rationale:	–
Dependencies:	–
Use Case:	Ensure that only compatible interfaces are connected.
Supporting Material:	Such information can be hash values or readable values.

⌋*()*

[SRS_SwCluC_00011] Separation of immutable memory and memory modifiable at the connection phase ⌈

Type:	draft
Description:	The Binary Manifest of the Software Cluster Connection shall strictly separate immutable memory which only needs to be read and memory which needs to be modified during the connection phase.
Rationale:	Limit the impact of the Binary Software Cluster connections to a limited and well-defined memory area.
Dependencies:	–
Use Case:	–

▽

△

Supporting Material:	–

⌋()

[SRS_SwCluC_00012] direct linkage ⌈

Type:	draft
Description:	The Binary Manifest of the Software Cluster Connection shall support the direct linkage between the inner tables of the Binary Manifest and the implementation of the resource provider and requesters. This is a bidirectional dependency since the using code of the Binary Manifest expects to get some handles (pointers, ID values) from the Binary Manifest and the initialization of the Binary Manifest tables requires some static initializes from the manifest using code (functions declarations, variable declarations, symbols)
Rationale:	Keep interfacing between Software Clusters lean. Especially for data interfaces a high amount of connections is expected.
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋()

[SRS_SwCluC_00013] Initialization with C-compiler and linker means ⌈

Type:	draft
Description:	The Binary Manifest of the Software Cluster Connection shall support the initialization of its immutable tables by the C-compiler and linker.
Rationale:	Support standard build tools for Binary Manifest implementation.
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋()

4.2.2 Requirements on Cross Software Cluster Communication

This section contains the requirements on the `Cross Software Cluster Communication` enabling the VFB communication between Software Clusters.

[SRS_SwCluC_00100] Cross Software Cluster Communication Plug-Ins ⌈

Type:	draft
Description:	The Cross Software Cluster Communication of the Software Cluster Connection shall implement the APIs and functionality of a `Cross Software Cluster Communication Plug-In`
Rationale:	The cross cluster communication is implemented by another vendor than the RTE
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00101] ’1:n’ Sender-receiver communication ⌈

Type:	draft
Description:	The Cross Software Cluster Communication of the Software Cluster Connection shall support ’1:n’ sender-receiver communication.
Rationale:	VFB Specification requires support for single-sender-multiple-receiver (’1:n’)
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00102] ’n:1’ Sender-receiver communication ⌈

Type:	draft
Description:	The Cross Software Cluster Communication of the Software Cluster Connection shall support ’n:1’ sender-receiver communication.
Rationale:	VFB Specification requires support for multiple-sender-single-receiver (’n:1’)
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00103] ’n:1’ Client-server communication ⌈

Type:	draft
Description:	The Cross Software Cluster Communication of the Software Cluster Connection shall support multiple-client-single-server ('n:1') client-server communication. Individual clients are independent - there is no coordination of requests between clients. In addition, the Cross Software Cluster Communication shall strictly decouple the call contexts between clients and servers.
Rationale:	VFB requires support multiple-clients-one-server ('n:1') but explicitly does not require to support single-client-multiple-server ('1:n') communication. One Software Cluster shall not depend on the implementation and configuration of another Software Cluster.
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00104] '1:n' Mode Switch Communication ⌈

Type:	draft
Description:	The Cross Software Cluster Communication of the Software Cluster Connection shall support '1:n' Mode Switch Communication.
Rationale:	VFB Specification requires support for single mode managers multiple mode users ('1:n')
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00105] '1:n' External Trigger communication ⌈

Type:	draft
Description:	The Cross Software Cluster Communication of the Software Cluster Connection shall support the communication of External Trigger events from one trigger source to multiple trigger sinks ('1:n') in different software clusters.
Rationale:	Sporadic and non timing based periodic activation of Runnable Entities in different Software Components in different Software Clusters.
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00106] '1:n' Parameter Communication ⌈

Type:	draft
Description:	The Cross Software Cluster Communication of the Software Cluster Connection shall support '1:n' Parameter Communication.
Rationale:	VFB Specification requires support for single mode managers multiple mode users ('1:n')
Dependencies:	–
Use Case:	Share calibration parameters between Software Clusters
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00107] Support unspecific preemption scenarios ⌈

Type:	draft
Description:	The Cross Software Cluster Communication of the Software Cluster Connection shall ensure data consistency for any communication between software clusters. Furthermore, the implementation has to work for all preemption scenarios, since the execution order is only known after the connection phase.
Rationale:	The preemption scenario on communication graphs is only known after the connection of software clusters.
Dependencies:	–
Use Case:	–
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00108] Prevent from writing directly to memory of other Software Clusters ⌈

Type:	draft
Description:	The Cross Software Cluster Communication of the Software Cluster Connection shall prevent from writing directly to memory of other SWCLs.
Rationale:	The implementation of a software cluster shall not assume any writability of memory of other software clusters.
Dependencies:	–
Use Case:	–
Supporting Material:	Strict usage of pull patterns or implementation of memory access via Host.

⌋*()*

4.2.3 Requirements on Proxy Modules

This section contains the requirements on the Proxy Modules enabling the substitution of BSW modules in Applicative Software Clusters and the connection to the real BSW module in the Host Software Cluster.

[SRS_SwCluC_00201] Standardized AUTOSAR Interfaces for software components ⌈

Type:	draft
Description:	The High Proxy Module of the Software Cluster Connection shall implement the Standardized AUTOSAR Interfaces of the according AUTOSAR Service substituted by the proxy.
Rationale:	–
Dependencies:	
Use Case:	Provide Standardized AUTOSAR Interfaces for software components
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00202] Standardized Interfaces for local BSW modules ⌈

Type:	draft
Description:	The High Proxy Module of the Software Cluster Connection shall implement the Standardized Interfaces of the BSW module substituted by the proxy which are relevant for modules in higher layers or same layer. Interfaces for initialization and de-initialization are excluded since those can be implemented for the Software Cluster Connection commonly.
Rationale:	–
Dependencies:	
Use Case:	Provide Standardized Interfaces for a subset of local BSW modules in a software cluster
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00203] Id abstraction ⌈

Type:	draft
Description:	The Proxy Module functionality of the Software Cluster Connection shall support that the Ids - identifying a specific channel - passed to the BSW module of Host Software Cluster can change without reconfiguration or rebuild of the Applicative Software Cluster using the specific channel.

▽

△

Rationale:	Ids configured in BSW modules can change arbitrarily from one integration to another since those values needs to be continuous.
Dependencies:	
Use Case:	Independent development and integration of Software Clusters
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00204] Modular Software Cluster Connection ⌈

Type:	draft
Description:	The Proxy Module functionality of the Software Cluster Connection shall support the deactivation of specific Proxy Modules. This means in case of deactivation neither the static code nor any code generation, model generation, or validation for this specific Proxy Module is active.
Rationale:	–
Dependencies:	
Use Case:	Deactivate specific Proxy Modules in order to substitute it by a platform specific CDD
Supporting Material:	–

⌋*()*

[SRS_SwCluC_00205] Safeguarding connections between Software Clusters ⌈

Type:	draft
Description:	The Proxy Module of the Software Cluster Connection shall safeguard the individual connections between High Proxy Module(s) and Low Proxy Module by aggregation of the essential functional properties in the static safeguard of the Binary Manifest.
Rationale:	Ensure that only compatible interfaces are connected.
Dependencies:	
Use Case:	Independent development and integration of Software Clusters
Supporting Material:	[SRS_SwCluC_00010]

⌋*()*

[SRS_SwCluC_00206] NV blocks in Applicative Software Cluster ⌈

Type:	draft
Description:	The Proxy Module of the Software Cluster Connection shall enable the usage of NV blocks in Applicative Software Cluster including the usage of the related API for single block requests
Rationale:	–
Dependencies:	
Use Case:	Non volatile memory usage by Applicative Software Cluster
Supporting Material:	[SRS_SwCluC_00010]

⌋*()*

4.2.4 Requirements on workflow integration

This section contains the requirements on the `Cross Software Cluster Communication` enabling the integration in development workflows.

[SRS_SwCluC_00300] A2L Generation Support ⌈

Type:	draft
Description:	The Software Cluster Connection shall support the generation of output information in order to support the later generation of a complete A2L file.
Rationale:	In case Software Cluster Connection is allocating the variables which shall be measurable or constants which shall be calibratable, the information about the allocated variables/constants shall be exported for further usage by subsequent tools.
Dependencies:	–
Use Case:	Make default values calibratable via measurement tools.
Supporting Material:	–

⌋*()*

4.3 Non-Functional Requirements (Qualities)

5 Requirements Tracing

The following table references the features specified in [3] and links to the fulfillments of these.

Feature	Description	Satisfied by

6 References

[1] System Template
AUTOSAR_TPS_SystemTemplate

[2] Glossary
AUTOSAR_TR_Glossary

[3] Requirements on AUTOSAR Features
AUTOSAR_RS_Features