Document Title	Requirements on Runtime Environment
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	083
Document Classification	Auxiliary

Document Version	2.2.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
26.10.2011	2.2.0	AUTOSAR Administration	• RTE00155: Changed description • RTE00154: Changed description • RTE00234: Added requirement • RTE00235: Added requirement
13.10.2010	2.1.0	AUTOSAR Administration	• RTE00210: changed rational • RTE00020: Added access to OS service interface
30.11.2009	2.0.0	AUTOSAR Administration	Added support for concepts: • AUTOSAR Scheduler harmonization • RTE API enhancement • Triggered Event • Enhance Measurement and Calibration • Avoidance of duplicated Type Definitions • Integrity and Scaling at ports • Implicit Communication Enhancement • A2L Generation Support • Support of large data types • Fixed Data Exchange • Variant Handling • Time Determinism • DLT Concept • Memory related Concepts • Build System Enhancement • Multi Core Architectures • Memory Partitioning • Error Handling • VMM AMM Concept • Legal disclaimer revised
15.01.2009	1.2.0	AUTOSAR Administration	• Changed RTE00005 • Removed RTE00044
23.06.2008	1.1.3	AUTOSAR Administration	Legal disclaimer revised

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
31.10.2007	1.1.2	AUTOSAR Administration	• Document meta information extended • Small layout adaptations made
24.01.2007	1.1.1	AUTOSAR Administration	• “Advice for users” revised • “Revision Information” added
05.12.2006	1.1.0	AUTOSAR Administration	• Added RTE00153, RTE00154, RTE00155, RTE00156, RTE00157, RTE00158, RTE00159 Changed RTE00151 • Added RTE00160 • Added RTE00161 • Legal disclaimer revised
12.07.2006	1.0.1	AUTOSAR Administration	• Changed RTE00133, RTE00013, RTE00077, RTE00075, date format changed to dd-mm-yyyy. • Added RTE00152, [14] • Removed RTE00136 because it contradicts with RTE00152
25.04.2006	1.0.0	AUTOSAR Administration	Initial release

Disclaimer

Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Scope of document

The goal of AUTOSAR and of this document is to define the requirements and behavior of the AUTOSAR Run-time environment.

It is not within the remit of AUTOSAR to consider how the RTE is implemented but however all requirements and behavioral specifications are reviewed internally to ensure that at least one feasible implementations is possible.

2 How to read this document

Each requirement has its unique identifier starting with the prefix “BSW” (for “Basic Software”). For any review annotations, remarks or questions please refer to this unique ID rather than chapter or page numbers!

2.1 Conventions used

In requirements, the following specific semantics shall be used (based on the Internet Engineering Task Force IETF).

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as:

- SHALL: This word means that the definition is an absolute requirement of the specification.
- SHALL NOT: This phrase means that the definition is an absolute prohibition of the specification.
- MUST: This word means that the definition is an absolute requirement of the specification due to legal issues.
- MUST NOT: This phrase means that the definition is an absolute prohibition of the specification due to legal constraints.
- SHOULD: This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY: This word, or the adjective „OPTIONAL“, means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation, which does not include a particular option, MUST be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, MUST be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides.)

The priority “Low” indicates that this requirement may not be implemented in version 1.0.

2.2 Requirements structure

Each module specific chapter contains a short functional description of the Basic Software Module. Requirements of the same kind within each chapter are grouped under the following headlines (where applicable):

Functional Requirements:
- Configuration (which elements of the module need to be configurable)
- Initialisation
- Normal Operation
- Shutdown Operation
- Fault Operation
- ...

Non-Functional Requirements:
- Timing Requirements
- Resource Usage
- Usability
- Output for other WPs (e.g. Description Templates, Tooling,...)
- ...

3 Functional Overview

The Run-Time Environment (RTE) is at the heart of the AUTOSAR ECU architecture. The RTE is the realization (for a particular ECU) of the interfaces of the AUTOSAR Virtual Function Bus (VFB) and thus provides the infrastructure services for communication between Application Software Components as well as facilitating access to basic software components including the OS.

Application Software Components contain system software that is CPU and location independent. This means that, subject to constraints imposed by the system designer, an Application Software Component can be mapped to any available ECU during system configuration. The RTE is responsible for ensuring that components can communicate and that the system continues to function as expected wherever the components are mapped.

The RTE encompasses both the *variable elements* of the system infrastructure that arise from the different mappings of components to ECUs as well as *standardized* RTE services. The RTE is generated and/or configured for each ECU to ensure that the RTE is optimal for the ECU.

4 Requirements Specification

4.1 Functional Requirements

4.1.1 Interaction with AUTOSAR OS

The requirements in this section all concern how the RTE interacts with the AUTOSAR OS. The AUTOSAR ECU architecture defines all interactions to occur over a *standardized interface.*

4.1.1.1 [RTE00020] Access to OS

ID:	RTE00020
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Access to OS
Type:	Changed (20.10.2008)
Importance:	high
Description:	The RTE shall abstract the features of Os from AUTOSAR Software Components. Only the access to the service interface of the RTE shall be available for AUTOSAR Software Components directly.
Rationale:	The Application Software Components are intended to be Os independent and therefore should not access any particular Os function directly, except for the service interface. For example, the RTE uses task-based functionality (tasks, resources, events, …) to provide Runnable Entity functionality to the application. The existence of OS tasks is not made visible to the application.
Use Case:	The Os offers a standardized interface. This interface is accessed by Software Components only via the RTE API and hence access is controlled by the RTE. The Os offers a service interface. This is directly accessible by the Software Components.
Dependencies:	RTE00025
Conflicts:	--
Supporting Material:	Specification of the Virtual Functional Bus [6] The AUTOSAR ECU architecture defines a standardized interface for the OS and an AUTOSAR interface for Application Software Components and therefore there can be no direct interaction.
Contributes to:	--

4.1.1.2 [RTE00099] Decoupling of interrupts

ID:	RTE00099
Initiator:	WP RTE
Date:	03.11.2004
Short Description:	Decoupling of interrupts
Type:	new
Importance:	high
Description:	The RTE shall not permit interrupt context to be propagated to Application Software Components. To ensure low latency times and determinism, the interrupt context may have to be propagated to the RTE.
Rationale:	If Application Software Components were able to execute within an interrupt context they would be able to block the system schedule for unacceptably long periods of time.

Use Case:	The RTE 'intercepts' interrupts and enables a Runnable Entity to handle the notification. The Runnable Entity executes in the context of a task.
Dependencies:	--
Conflicts:	--
Supporting Material:	Specification of the Virtual Functional Bus [6] In this requirement, blocking meant to indicate that the RTE shall not suspend (Running-->Waiting) the thread of control executing the callback. It is not meant to indicate that the thread cannot be pre-empted. i.e. blocking is "suspended" but not "pre-empted"
Contributes to:	--

4.1.1.3 [RTE00036] Assignment to OS Applications

ID:	RTE00036
Initiator:	WP RTE
Date:	04.10.2004
Short Description:	Assignment to OS Applications
Type:	Changed (20.10.2008)
Importance:	high
Description:	When partitioning is in use, the RTE generator shall reject configurations where the Runnable Entities of one Software Component instance are not assigned to tasks within the same OS-Application.
Rationale:	All objects (e.g. resources, alarms) which belong to one OS-Application have access to each other – the OS will kill tasks that attempt direct access without being mapped to the same OS application.
Use Case:	Efficient access is required – if mapped to different OS applications then the RTE would be required to implement the protection mode switches which would have a significant impact on efficiency.
Dependencies:	RTE00018
Conflicts:	--
Supporting Material:	Where memory protection is used the tasks mapped for a component instance form a single OS Application – this permits intra-component interactions to occur with minimum overhead.
Contributes to:	--

4.1.1.4 [RTE00049] Construction of task bodies

ID:	RTE00049
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Construction of task bodies
Type:	changed (19.03.2009)
Importance:	high
Description:	The RTE generator shall construct task bodies to execute Runnable Entities and Basic Software Schedulable Entities in a form suitable for the AUTOSAR OS – this will typically be as a function exported with C linkage. The SW-Component description declares the Runnable Entities present in a component. The Basic Software Module description declares the Basic Software Schedulable Entities present in a BSW Module.
Rationale:	The mapping of Runnable Entities and Basic Software Schedulable Entities to tasks forms part of the input to the generator. Automatic mapping is too complex a task (and insufficient data is present in the input) to be considered as part of AUTOSAR at this stage.
Use Case:	Runnable Entities and Basic Software Schedulable Entities in a sequence mapped to the same task. It is possible to provide

	• tasks with only Runnable Entities • tasks with support for interlaced execution of Runnable Entites and Basic Software Schedulable Entities • tasks with only Basic Software Schedulable Entities.
Dependencies:	RTE00219
Conflicts:	--
Supporting Material:	[BRF00020] - Integration of existing BSW Scheduling into the RTE [15]
Contributes to:	--

4.1.1.5 [RTE00193] Support for Runnable Entity execution chaining

ID:	RTE00193
Initiator:	WP Safety
Date:	17.06.2008
Short Description:	Support for Runnable Entity execution chaining
Type:	New
Importance:	High
Description:	The RTE Generator shall allow Runnable Entity monitoring by mapping of the monitored Runnable Entities to one or several OS Tasks and – in case of the usage of several OS Tasks – chain the execution of these OS Tasks.
Rationale:	In order to monitor the execution time of Runnable Entities per individual (or several) Runnable Entity, the mechanism of task chaining shall be used so the monitoring can be implemented using OS Task monitoring. The RTE shall be able to activate (chain) the execution of the chained tasks in order to get the desired Runnable Entity execution order.
Use Case:	Runnable Entities which are configured to be executed sequentially in one OS Task shall be able to be spilt to several OS Tasks in order to apply OS execution time monitoring on a fine grained level (individual Runnable Entity or several Runnable Entities). To get the same behavior the chain task mechanism shall be applied.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00121] – Monitoring of task execution [15]
Contributes to:	--

4.1.1.6 [RTE00210] Support for inter OS application communication

ID:	RTE00210
Initiator:	WP Architecture
Date:	15.07.2008
Short Description:	Support for inter OS application communication
Type:	new
Importance:	high
Description:	RTE shall support the communication between Software Component instance that are allocated to different OS applications, where different OS applications may be located on different memory partitions and/or cores.
Rationale:	As Software Component instances from different OS applications may be located on different cores and different memory partitions, the communication between them may require the use of dedicated communication and signaling methods like the use of a Cross OS Application Communication Module.
Use Case:	Multi core support, memory partitioning support
Dependencies:	RTE00011
Conflicts:	--
Supporting Material:	[BRF00221] Controlled Data Exchange between Cores [15] [BRF00217] Event Mechanism shall work across Cores [15] [BRF00214] Resources across Cores [15]

	[BRF00115] SW-Cs grouped in separate user-mode memory partitions [15]
Contributes to:	--

4.1.2 Interaction with AUTOSAR COM

The requirements in this section all concern how the RTE interacts with the AUTOSAR COM. The AUTOSAR ECU architecture defines all interaction to occur over a *standardized interface.*

4.1.2.1 [RTE00068] Signal initial values

ID:	RTE00068
Initiator:	WP RTE
Date:	04.10.2004
Short Description:	Signal initial values
Type:	new
Importance:	high
Description:	The RTE generator shall ensure that signals for which an INIT_VALUE is specified are initialized.
Rationale:	Data can be read before COM has provided a first value and applications should be prevented from reading un-initialized data.
Use Case:	--
Dependencies:	RTE00108 The INVALIDATE attribute can be used in conjunction with an INIT_VALUE to indicate to an Application Software Component that no data has been received since COM or the RTE started. The INVALIDATE attribute shall be initialized too.
Conflicts:	--
Supporting Material:	Specification of the Virtual Functional Bus [6]
Contributes to:	--

4.1.2.2 [RTE00069] Communication timeouts

ID:	RTE00069
Initiator:	WP RTE
Date:	04.10.2004
Short Description:	Communication timeouts
Type:	new
Importance:	high
Description:	The RTE generator shall include run-time checks for monitoring timeouts specified in the ECU Configuration for blocking communication. When synchronous intra-task client server communication is optimized to a direct function call, no timeout can occur though clients can still be written to expect a timeout were the configuration to change. Therefore this requirement does not apply when the synchronous client server call is optimized to a direct function call.
Rationale:	Prevent infinite blocking of receivers.
Use Case:	A Runnable Entity performs a blocking “read” of a data item. A blocking read will wait forever if no data arrives unless a timeout is applied.
Dependencies:	RTE00147
Conflicts:	--
Supporting Material:	Specification of the Virtual Functional Bus [6] - timeouts are required within components to prevent infinite blocking and thus apply both to inter-ECU communication (that uses COM) and intra-ECU communication (that may or may not use COM).

Contributes to:	--

4.1.2.3 [RTE00073] Atomic transport of Data Elements

ID:	RTE00073
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Atomic transport of Data Elements
Type:	Changed (24.10.2008)
Importance:	high
Description:	The RTE shall ensure that the transmission and reception of data elements (regardless whether they are simple or composite), and all arguments of a single RTE operation are treated as atomic units. Where a parameter is passed by reference rather than by value the RTE is forced to rely on the component not modifying the target of the reference while the parameter is in use by the RTE.
Rationale:	--
Use Case:	Elements of a record cannot be handled separately by the Application Software Component but, instead, the whole record should be treated as a single atomic unit.
Dependencies:	RTE00032 (data consistency) This should not be read as requiring COM to treat the transmission as atomic – it (or a lower layer in the COM stack) shall remain free to split across multiple (network) frames as long as the split is not visible to the RTE.
Conflicts:	--
Supporting Material:	Specification of the Virtual Functional Bus [6] Software Component Template [7]
Contributes to:	--

4.1.2.4 [RTE00082] Standardized communication protocol

ID:	RTE00082
Initiator:	WP RTE
Date:	05.10.2004
Short Description:	Standardized communication protocol
Type:	Changed (24.10.2008)
Importance:	high
Description:	The RTE shall define and implement the protocol (e.g. message sequences) for inter-ECU client-server communication. For communication mechanisms that are not directly provided by the AUTOSAR COM layer (e.g. client-server communication) a standardized protocol that is implemented by every AUTOSAR RTE has to be defined.
Rationale:	This ensures that RTEs of different vendors are interoperable.
Use Case:	If C/S is mapped to paired COM message channels then both RTEs shall implement the same mapping to be compatible.
Dependencies:	RTE00062
Conflicts:	--
Supporting Material:	RTE00091 – common protocol for COM transmissions.
Contributes to:	--

4.1.2.5 [RTE00091] Inter-ECU Marshalling

ID:	RTE00091
Initiator:	WP RTE
Date:	20.10.2004
Short Description:	Inter-ECU Marshalling
Type:	new
Importance:	high
Description:	The RTE shall use a common format for transmitting/receiving data elements or parameters of operations between ECUs. On transmission the (target specific) signal data shall be converted to the common format and the reverse operation performed on reception. It shall be possible to exchange record types between components written in different programming languages.
Rationale:	The RTE is responsible for ensuring that data elements or parameters of operations (e.g. records, parameter lists, …) can be sent between ECUs. Since each ECU may define signals differently in memory a straight transmission cannot be performed and, instead, the sender shall convert the data elements or parameters to a common format before transmission and the reverse transformation shall be performed by the receiving RTE. A common communication protocol enables RTEs from different vendors to interoperate.
Use Case:	--
Dependencies:	RTE00082 – defines common message sequence for client-server.
Conflicts:	--
Supporting Material:	Software Component Template [7] The term “Marshalling” is used synonymously to “Serialization”.
Contributes to:	--

4.1.2.6 [RTE00181] Conversion between internal and network data types

ID:	RTE00181
Initiator:	WP RTE
Date:	01.06.2008
Short Description:	Conversion between internal and network data types
Type:	New
Importance:	Low
Description:	The RTE shall generate conversion routines – when configured – for different data types for data elements used in sender-receiver communication within one ECU (high resolution) and data elements used between ECUs (low resolution) to save serial data link bandwidth and non-productive development work.
Rationale:	Save bandwidth in inter-ECU communication.
Use Case:	Within one ECU the SWCs want to use data types with high resolutions in computations to get a small epsilon (computational deviation). When the same data is transported to another ECU it is converted into the network representation on the sender side (with loss of precision). On the receiver side the data is converted back to the original type.
Dependencies:	--
Conflicts:	The SWC does not see explicitly that the data type resolution becomes lower when the data is sent between two ECUs instead of within one ECU.
Supporting Material:	[BRF00097] - Conversion between internal and Network Data Types for InterECU Communication [15]
Contributes to:	--

4.1.3 Interaction with Application Software Components

Includes Application Software Components, sensor components and actuator components.

4.1.3.1 [RTE00011] Support for multiple Application Software Component instances

ID:	RTE00011
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Support for multiple Application Software Component instances.
Type:	new
Importance:	high
Description:	The RTE shall support multiple instances of the same Application/Sensor/Actuator Software Component type mapped to the same ECU.
Rationale:	Repetition of the same Application Software Component type on an ECU to promote component reuse.
Use Case:	• Having one Application Software Component type for the window lifter which is instantiated four times in a vehicle. • Homogenous SW redundancy is a basic/simple solution for increasing fault tolerance.
Dependencies:	Name space – rules for “instantiating” an application / sensor / actuator have to be defined. RTE00210
Conflicts:	--
Supporting Material:	Software Component Template [7]
Contributes to:	--

4.1.3.2 [RTE00012] Multiple instantiated AUTOSAR software components delivered as binary code shall share code

ID:	RTE00012
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Multiple instantiated AUTOSAR software components delivered as binary code shall share code.
Type:	new
Importance:	high
Description:	The RTE generator shall implement multiple instantiations (delivered as binary code) of the AUTOSAR software component type (on the same ECU) through sharing the same Application Software Component code for all instances. Source code deliverables can be instantiated either by code sharing or code duplication. This depends on the implementation of the RTE.
Rationale:	1) VFB metamodel. 2) Requirement to minimise code space. 3) Cannot modify binary-code software components and therefore all instances must use the same object-code.
Use Case:	--
Dependencies:	RTE00133
Conflicts:	--
Supporting Material:	Software Component Template [7]

	Code that is to be shared between instances should be re-entrant because of the pre-emptive environment it which it will run. Re-entrant code is indicated by the "`supportsMultipleInstantiation`" flag in the software component description.
Contributes to:	--

4.1.3.3 [RTE00013] Per-instance memory

ID:	RTE00013
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Per-instance memory
Type:	New
Importance:	High
Description:	The RTE shall provide per-instance memory to Application Software Components, where each Application Software Component instance has its own copy of memory, not shared with other instances of the same Application Software Component type.
Rationale:	Require variables with lifetime greater than that of a Runnable Entity but remain protected from different instances of the same Software Component type. RTE shall not provide memory shared between instances of an Application Software Component type. Also required for multiple instance support.
Use Case:	--
Dependencies:	RTE00075 – API for accessing per-instance memory
Conflicts:	--
Supporting Material:	Software Component Template [7]
Contributes to:	--

4.1.3.4 [RTE00077] Instantiation of per-instance memory

ID:	RTE00077
Initiator:	WP RTE
Date:	05.10.2004
Short Description:	Instantiation of per-instance memory
Type:	new
Importance:	high
Description:	The RTE generator shall instantiate each per-instance memory section of a software component according to the attributes given in its software component description. The instantiation of per-instance memory shall be either derived from input information or instantiated automatically by the RTE generator (where such information is not available). In the latter case the address of the per-instance memory is assigned by the RTE generator and therefore is not subject to control by the system integrator.
Rationale:	Required by the Software Component Template
Use Case:	RAM mirror from NVRAMManager.
Dependencies:	--
Conflicts:	--
Supporting Material:	The per-instance memory presented as input to the RTE generator shall be uniquely identifiable. Software Component Template [7]
Contributes to:	--

4.1.3.5 [RTE00017] Rejection of inconsistent component implementations

ID:	RTE00017
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Rejection of inconsistent component implementations
Type:	new
Importance:	high
Description:	The RTE generator shall ensure that the compiler can detect (and reject) access to undefined RTE API calls. The RTE generator is required to reject “invalid“ configurations, part of this is rejecting invalid APIs (i.e. calls to an unknown port at compile time).
Rationale:	--
Use Case:	The component description defines the names of the ports that a component “requires” and “provides” and the associated interfaces. The RTE generator can then define only the valid API calls. For example, consider a component that has a Port ‘p1’ with a data items ‘a’ and ‘b’. In this case, the RTE generator will only create an API for the send of data items ‘a’ and ‘b’. Thus an attempt by the component to send any invalid data item, e.g. ‘c’ on the interface shall be detected by the compiler and a warning issued.
Dependencies:	--
Conflicts:	--
Supporting Material:	Software Component Template [7]
Contributes to:	--

4.1.3.6 [RTE00134] Runnable Entity categories supported by the RTE

ID:	RTE00134
Initiator:	WP RTE
Date:	15.12.2004
Short Description:	Runnable Entity categories supported by the RTE
Type:	changed (19.03.2009)
Importance:	high
Description:	The RTE shall support the Runnable Entity categories 1a, 1b and 2 Runnable Entity category: • 1a) The Runnable Entity is only allowed to use implicit reading (DataReadAccess) and writing (DataWriteAcess). A category 1a Runnable Entity cannot block and cannot use explicit read/write. • 1b) The Runnable Entity can use explicit reading and writing (DataReadAccess). A category 1b Runnable Entity cannot block. Implicit read/write is also allowed. • 2) The Runnable Entity may use explicit reading/writing including blocking behavior.
Rationale:	Support several kinds of runnable entity implementation kinds.
Use Case:	It is easier to reason about time behavior for category 1a Runnable Entities that do not invoke RTE API calls that (may) not execute in constant time.
Dependencies:	RTE00128, RTE00129 – Implicit reception and transmission. RTE00098 – Explicit transmission.
Conflicts:	--
Supporting Material:	SW-Component Template [7]
Contributes to:	--

4.1.3.7 [RTE00072] Activation of Runnable Entities

ID:	RTE00072
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Activation of Runnable Entities
Type:	new
Importance:	high
Description:	The RTE shall start/resume a Runnable Entity according to the RTEEvents to which it is linked.
Rationale:	Activations of Runnable Entities due to arrival of data from other components, invocation of operations of one port or time based execution of Runnable Entities is based on the RTEEvent model [7].
Use Case:	Cyclic, time based activation of Runnable Entities; activation of a Runnable Entity due to the arrival of data using the sender-receiver communication pattern.
Dependencies:	[RTE00160] [RTE00161]
Conflicts:	--
Supporting Material:	Software Component Template [7]
Contributes to:	--

4.1.3.8 [RTE00160] Debounced start of Runnable Entities

ID:	RTE00160
Initiator:	WP RTE
Date:	02.11.2006
Short Description:	Debounced start of Runnable Entities
Type:	new
Importance:	high
Description:	The RTE shall allow the configuration of a debounce start time of Runnable Entities to avoid the same Runnable Entity being executed shortly after each other.
Rationale:	In case several RTE Events occur within a short time interval there shall only be a limited amount of executions of the Runnable Entity. It shall be possible to define a minimum time which in which all activations are noticed, but the Runnable Entity will start only after that period has passed.
Use Case:	Runnable Entities being activated with along timing period and additionally activated on several DataReceivedEvents. If the Runnable Entity has just been executed the RTE shall wait for the defined period until the Runnable Entity is executed again.
Dependencies:	[RTE00072]
Conflicts:	--
Supporting Material:	--
Contributes to:	--

4.1.3.9 [RTE00161] Activation offset of Runnable Entities

ID:	RTE00161
Initiator:	WP RTE
Date:	03.11.2006
Short Description:	Activation offset of Runnable Entities
Type:	new
Importance:	high
Description:	The RTE shall allow the definition of an activation offset of Runnable Entites.
Rationale:	In order to allows optimizations in the scheduling (smooth cpu load, mapping

	of Runnable Entities with different periods in the same task to avoid data sharing, etc.), the RTE has to handle the activation offset information from a task shared reference point for time trigger Runnable Entites.
Use Case:	--
Dependencies:	[RTE00072]
Conflicts:	--
Supporting Material:	--
Contributes to:	--

4.1.3.10 [RTE00031] Multiple Runnable Entities

ID:	RTE00031
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Multiple Runnable Entities
Type:	new
Importance:	high
Description:	The RTE shall support multiple Runnable Entities in one Software Component type.
Rationale:	Runnable Entities are used for servers, receivers, feedback, … etc and therefore each component can have many Runnable Entities.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	VFB Metamodel
Contributes to:	--

4.1.3.11 [RTE00032] Data consistency mechanisms

ID:	RTE00032
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Data consistency mechanisms
Type:	new
Importance:	high
Description:	The RTE shall support one or more mechanism for ensuring data consistency *within* an Application Software Component instance. No direct access to data 'outside' the component instance is possible within AUTOSAR. The scope of the mechanism (e.g. exclusive area) shall be all Runnable Entities (that statically specify the same exclusive area – RTE_IN004) in the software component **instance**. If consistency between component instances is required then an additional software component can be created to provide appropriate access semantics to the encapsulated data. A side effect of a data consistency mechanism may be to prevent other Runnable Entities in different component instances from executing, for example, the RTE may lock out all interrupts for a short period of time. However this is not deemed to be an illegal (non-AUTOSAR) communication channel since the set of affected Runnable Entities is not defined and therefore cannot be relied upon by a component author.
Rationale:	Multiple Runnable Entities can be active within an Application Software Component and therefore a mechanism shall exist to prevent concurrency conflicts. An Application Software Component cannot access the OS directly and therefore the RTE shall provide the mechanism.
Use Case:	• Exclusive areas are an example of a mechanism suitable, e.g.:

	`void RTERunnable_a(RTEInstance self)`
	`{ …`
	`  RTEEnter_<region>(self);`
	`  /* read-modify-write data */`
	`  RTEExit_<region>(self);`
	`  ... }`
	
	• An Application Software component, which server function is concurrently called by several clients: in this case multiple executions of runnable entities (associated to each call, supporting "canBeInvokedConcurrently") usually need to access shared data.

Dependencies:	--
Conflicts:	--
Supporting Material:	VFB Requirements (VFB_C60, Sched70) To permit an exclusive area to affect all instances of a software component type would be incorrect since component instances are independent and would also open a non-AUTOSAR communication channel between the components. Note: The APIs using in this requirement that are mentioned are only examples of how the APIs may look like – the API presented in the SWS is subject to change.
Contributes to:	--

4.1.3.12 [RTE00046] Support for "Executable Entity runs inside" Exclusive Areas

ID:	RTE00046
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Support for "Executable Entity runs inside" Exclusive Areas
Type:	changed (19.03.2009)
Importance:	high
Description:	The RTE shall support exclusive areas where a Runnable Entity or a Basic Software Schedulable Entity is declared as "running inside" the Exclusive Area. All Runnable Entities in a SW-Component or Basic Software Schedulable Entities that specify the same "runs inside" Exclusive Area shall be scheduled non preemptively with respect to other Runnable Entities respectively Basic Software Schedulable Entities in the set.
Rationale:	"Runs inside" Exclusive Areas satisfies the requirement from the SW-Component template and Basic Software Module Description template that certain Exclusive Areas can be defined that are automatically entered whenever a Executable Entity is invoked by the RTE / Basic Software Scheduler.
Use Case:	--
Dependencies:	RTE00032
Conflicts:	--
Supporting Material:	SW-Component Template [7] BSWMD Template [5]
Contributes to:	--

4.1.3.13 [RTE00142] InterRunnableVariables

ID:	RTE00142
Initiator:	WP RTE

Date:	13.06.2005
Short Description:	Support for InterRunnableVariables
Type:	new
Importance:	high
Description:	The RTE shall support InterRunnableVariables. A Software Component shall be able to declare one or more InterRunnableVariables used for data consistency purposes. An InterRunnableVariable is use when several Runnable Entities of the same Software Component instance access the same data item. InterRunnableVariables are used to store data item copies to avoid concurrent Runnable Entity accesses to the one original data item.
Rationale:	InterRunnableVariables satisfy the requirement from the software component template that certain InterRunnableVariables can be defined that can be accessed by Runnable Entities of same software component instance to implement the "variable copies" strategy.
Use Case:	Data item write access by Runnable Entity in 10ms task. Several data item read accesses by Runnable Entity in 50ms task. 50ms task can be preempted by 10ms task. Data inconsistency can be prevented if Runnable Entity in 50ms task gets a copy of the original data in an InterRunnableVariable to work on during activation.
Dependencies:	RTE00032
Conflicts:	--
Supporting Material:	SW-Component Template [7]
Contributes to:	--

4.1.3.14 [RTE00033] Serialized execution of Server Runnable Entities

ID:	RTE00033
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Serialized execution of Server Runnable Entities
Type:	changed (24.03.2009)
Importance:	high
Description:	The RTE shall support serialized and non-serialized execution of Server Runnable Entities. The RTE shall complete the processing of one serialized service request before it accepts and dispatches the next request for that server. The serialization is applied on the service level, so one server can handle multiple service calls concurrently (this implies that the service's Runnable Entities are mapped to different tasks and there is no shared data between them). If serialization is supported by a server and how big the actual queue is shall be configurable.
Rationale:	A serialized server only accepts and processes requests atomically and thus avoids potential conflicting concurrent access. Invocation of the server's Runnable Entity shall be encapsulated within an exclusive area when simultaneous intra-ECU and inter-ECU execution is possible.
Use Case:	The NVRAM Manager is capable of handling multiple requests. If for each stored data item there is a separate port generated during configuration the generic serialization mechanism works fine.
Dependencies:	RTE00032 (Per-instance scope of exclusive areas). RTE00110 (Support for buffering).
Conflicts:	--

Supporting Material:	The execution of a server is independent of how it is invoked, in particular, whether the call is synchronous or asynchronous is a property of the client and not the server. This requirement explicitly enforces strict serialization of Server Runnable Entities. An RTE generator can optimize a client/server call to a direct function call only if serialization is maintained. This can be done through the insertion of resource locks or other mechanisms.
Contributes to:	--

4.1.3.15 [RTE00133] Concurrent invocation of Runnable Entities

ID:	RTE00133
Initiator:	WP RTE
Date:	10.12.2004
Short Description:	Concurrent invocation of Runnable Entities
Type:	new
Importance:	high
Description:	RTE has to allow and support the concurrent invocation of a Runnable Entity (means several activations of same Runnable Entity at same time) for those Runnable Entities whose attribute “canBeInvokedConcurrently” is set to TRUE. The RTE generator shall reject input configurations requiring several concurrent activations of a Runnable Entity when the attribute “canBeInvokedConcurrently” of the Runnable Entity is set to FALSE. Note that this is independent of the Runnable Entities ability to be multiple instantiated or not.
Rationale:	Requirement from SwCT Runnable Entities description
Use Case:	Direct client-server calls implementation with Runnable Entity implemented as a server. E.g. needed for Basic-SW services.
Dependencies:	RTE00012
Conflicts:	--
Supporting Material:	Software Component Template [7]
Contributes to:	--

4.1.3.16 [RTE00143] Mode Switches

ID:	RTE00143
Initiator:	WP RTE
Date:	28.09.2005
Short Description:	Mode Switches
Type:	new
Importance:	high
Description:	The RTE shall implement the functionality of ModeSwitchEvent and ModeDisablingDependency.
Rationale:	ModeDisablingDependency is the only mean by which AUTOSAR allows to define sets of Runnable Entities that run only in certain modes. ModeSwitchEvent allows to trigger Runnable Entities on the transitions between modes.
Use Case:	Use cases are, e.g.: Initialization and finalization phases, different communication modes (telling, whether the SW-C can expect the communication partners of the ports to be available).
Dependencies:	RTE00144
Conflicts:	--
Supporting Material:	Software Component Template [7]

	Requirements on Mode Management [16]
Contributes to:	--

4.1.3.17 [RTE00176] Sharing of NVRAM data

ID:	RTE00176
Initiator:	WP RTE
Date:	22.05.2008
Short Description:	Sharing of NVRAM data
Type:	New
Importance:	Medium
Description:	Several Application Software Components shall be able to access the same data – through ports – defined in NvBlockComponentType.
Rationale:	This permits to lower the amount of NVRAM needed on an ECU and to avoid duplication of data and the maintenance of this duplicated data in the engineering processes.
Use Case:	Reuse of common parameters. Reduce NVRAM usage.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00022] - Modification of NVRAM Memory Access Concept [15]
Contributes to:	--

4.1.3.18 [RTE00180] DataSemantics range check during runtime

ID:	RTE00180
Initiator:	WP RTE
Date:	06.06.2008
Short Description:	DataSemantics range check during runtime
Type:	new
Importance:	medium
Description:	Provide functionality in the RTE to enable checking the ranges of data element values for both S/R and C/S communication. If specified at SW-Component level a violation shall be reported to the SW-Component and a correction strategy shall be implemented. If configured during ECU Configuration a violation shall be reported to the DET.
Rationale:	For integration of SW-Components from different providers the check of the actual contract is needed in order to detect violations. For debugging it is useful to check whether SWCs sending data do provide the data within the specified ranges. In case there is a violation a Development Error may be raised.
Use Case:	Even when a SW-Component specifies in the PortInterface that a certain DataElementPrototype value shall be within 0-100 it is technically possible to send a value of 110. The RTE shall be able to check for such range violations. Range check on sender/client side: No propagation of illegal values through the RTE (local and remote). Reaction in case of error: “Out Of Bounds” error code in the status value. Range check on receiver/server side: Protection against reception of out of bounds values on the receiver side (assuming the sender has not already checked). Reaction in case of error: “Out Of Bounds” error code in the status value.
Dependencies:	--
Conflicts:	--

Supporting Material:	[BRF00074] - DataSemantics Ranges Check during Runtime [15]
Contributes to:	--

4.1.3.19 [RTE00182] Self Scaling Signals at Port Interfaces

ID:	RTE00182
Initiator:	WP RTE
Date:	06.06.2008
Short Description:	Self Scaling Signals at Port Interfaces
Type:	new
Importance:	medium
Description:	When explicitly specified, the RTE shall support the connection of ports whose interfaces have incompatible data types or incompatible data semantics according to the compatibility rules of the SWC-T. The RTE shall allow specifying the scaling of signals for ports on the sender/server side and the receiver/client side to allow automatic re-scaling in the RTE. A deterministic generation of the re-scaling shall be supported by the RTE generator independent of its implementation. Dedicated connector needs to be modeled on VFB level, and based on that the RTE generator has to create the adapter. The RTE generator is not allowed to do it completely on its own.
Rationale:	Avoid writing SWC glue code to interface two different SWCs conversion code provided by the integrator / sub-system designer (e.g. using a COMPU-METHOD). Hence, no recalculation of signal resolution in the affected SWCs is necessary. In order to allow deterministic generation of the re-scaling code additional input information might be need to specify the details of the desired re-scaling.
Use Case:	In diagnostics the resolution of signals has to be provided as specified in the ISO document for OBDII (e.g. engine speed). If the RTE could provide these signals in the correct resolution the SWC is not required to do this re-calculation. For the re-scaling of LINEAR to LINEAR data an optional/alternative call-out function from the RTE might be utilized in order to support deterministic conversion formula specification.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00101] - Self Scaling Signals at Port Interfaces [15]
Contributes to:	--

4.1.4 Interaction with Basic Software Components

4.1.4.1 [RTE00152] Support for port-defined argument values

ID:	RTE00152
Initiator:	WP RTE
Date:	2006.04.27
Short Description:	Support for port-defined argument values
Type:	new
Importance:	high
Description:	The mechanism of “port-defined argument values”, as defined in the AUTOSAR Services document [14], has to be supported.
Rationale:	To allow the interaction of Application Software Components with the infrastructural basic software.
Use Case:	Access of a SW-C to the NVRAM Manager.
Dependencies:	--
Conflicts:	--
Supporting Material:	AUTOSAR Services [14]
Contributes to:	--

4.1.4.2 [RTE00022] Interaction with call-backs

ID:	RTE00022
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Interaction with call-backs
Type:	new
Importance:	high
Description:	The RTE shall not suspend execution while executing a call-back.
Rationale:	Blocking COM (e.g. in a call-back) could prevent reception of data and therefore lead to data loss.
Use Case:	--
Dependencies:	RTE00099 – decoupling of interrupts
Conflicts:	--
Supporting Material:	If the RTE cannot process (e.g. pass the information to a Runnable Entity) the call-back immediately then the information must be queued and processed at a later point. A call-back is not the same as activation of a Runnable Entity.
Contributes to:	--

4.1.4.3 [RTE00062] Local access to basic software components

ID:	RTE00062
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Local access to basic software components
Type:	Changed (20.10.2008)
Importance:	high
Description:	The RTE shall permit application and basic software components to *directly* (via RTE) access the AUTOSAR interfaces of basic software components located on the same ECU. The RTE generator shall prevent *direct* access to the AUTOSAR interfaces of remote basic software components. One exception is the BswM module which is allowed to be accessed on other ECUs as well. Indirect access to the AUTOSAR interfaces of basic software components located on a remote ECU shall be possible via the inclusion of an Application Software Component on the remote ECU to 'export' an appropriate AUTOSAR interface to the basic software component.
Rationale:	This requirement is imposed for two reasons: • Efficiency – remote access to a basic software component permits only the lowest level of optimization. For example, the RTE generated would be unable to take advantage of intra-task access to optimize communication to either a direct function call (client-server) or queue write (sender-receiver). • Control – an ECU integrator can know, a priori, that scheduling will not be affected by components on remote ECUs accessing the basic software and blocking access by local components.
Use Case:	• On a given ECU, sensor/actuators components are not allowed to communicate with remote ECU abstraction. This means that sensor/actuator SWCs SHALL be mapped to the same ECU to which the sensor/actuator devices are mapped. • Exception: A mode request to the local BswM is also distributed by the RTE to other ECUs which are configured to need the mode request as well.
Dependencies:	RTE00018 – rejection invalid configurations
Conflicts:	The BswM module is an exception since the mode requests may be distributed to BswM modules on other ECUs.
Supporting Material:	See VFB chapter 4.4.2.2. The location of a service can be implemented by a proxy implemented as an AUTOSAR software component.
Contributes to:	--

4.1.4.4 [RTE00169] Map code and memory allocated by the RTE to memory sections

ID:	RTE00169
Initiator:	WP RTE
Date:	05.06.2008
Short Description:	Map code and memory allocated by the RTE to memory sections.
Type:	New
Importance:	High
Description:	The RTE shall map its generated code and allocated memory to RTE memory sections.
Rationale:	Enable memory mapping control of the whole ECU, not only BSWM and SWC.
Use Case:	Efficiency of the interactions between SWC Runnable Entities and generated code. Memory Mapping of calibration parameters. Memory partitioning and memory mapping of PIM and other variables in the same partition as the users.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00057] - Memory Mapping Concept [15] [BRF00077] - Enable Memory Mapping and Compiler Abstraction to be usable for SWCs [15]
Contributes to:	--

4.1.4.5 [RTE00170] Provide used memory sections description

ID:	RTE00170
Initiator:	WP RTE
Date:	05.06.2008
Short Description:	Provide used memory sections description
Type:	New
Importance:	High
Description:	The RTE generator shall produce an XML file with a description of memory sections used by the means of the BSW Module template.
Rationale:	Enable memory mapping control of the whole ECU, not only BSWM and SWC.
Use Case:	Efficiency of the interactions between SWC Runnable Entities and generated code. Memory Mapping of calibration parameters. Memory partitioning and memory mapping of PIM and other variables in the same partition as the users.
Dependencies:	RTE00169 – Map code and memory allocated by the RTE to memory sections
Conflicts:	--
Supporting Material:	Basic Software Module Description template [5] [BRF00057] - Memory Mapping Concept [15] [BRF00077] - Enable Memory Mapping and Compiler Abstraction to be usable for SWCs [15]
Contributes to:	--

4.1.4.6 [RTE00177] Support of NvBlockComponentType

ID:	RTE00177
Initiator:	WP RTE
Date:	22.05.2008
Short Description:	Support of NvBlockComponentType
Type:	new
Importance:	medium

Description:	The RTE shall support NvBlockComponentType by providing a NvRAM and NvROM block to the NVRAM Manager and access to the data stored in the block.
Rationale:	Allow data to be grouped together in the same NVRAM block to lower the amount of needed NVRAM blocks.
Use Case:	Storage of several small flags in a large NvRAM block.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00022] - Modification of NVRAM Memory Access Concept [15]
Contributes to:	--

4.1.4.7 [RTE00228] Fan-out NvBlock callback function

ID:	RTE00228
Initiator:	WP RTE
Date:	20.10.2008
Short Description:	Fan-out NvBlock callback function
Type:	new
Importance:	high
Description:	The RTE shall support the fan-out (take one incoming callback function and distribute it into several callback function calls) of the NvBlock callback function from the NVRAM Manager to multiple Software Component instances which use the corresponding NvBlock.
Rationale:	It is possible to define several users for one NvBlock,but the NVRAM Manager is not able to handle several callback functions. Therefore the callback function has to be fan-out by the RTE.
Use Case:	Two Software Component Instances accessing the same NvBlock.
Dependencies:	RTE00177
Conflicts:	--
Supporting Material:	[BRF00022] - Modification of NVRAM Memory Access Concept [15]
Contributes to:	--

4.1.4.8 [RTE00233] Generation of the Basic Software Module Description

ID:	RTE00233
Initiator:	WP RTE
Date:	20.06.2009
Short Description:	Generation of the Basic Software Module Description
Type:	new
Importance:	high
Description:	The RTE Generator shall provide a Basic Software Module Description of the actual generated RTE.
Rationale:	The Basic Software Module Description provides information how the described module interacts with other modules and needs to be integrated.
Use Case:	Support the integration of AUTOSAR software.
Dependencies:	RTE00169, RTE00170
Conflicts:	--
Supporting Material:	Specification of Basic Software Module Description [5]
Contributes to:	--

4.1.5 Generation of the BSW Scheduler

4.1.5.1 [RTE00211] Cyclic time based scheduling of BSW Schedulable Entities

ID:	RTE00211
Initiator:	WP RTE
Date:	14.07.2008
Short Description:	Cyclic time based scheduling of BSW Schedulable Entities
Type:	new
Importance:	high
Description:	The RTE shall support to the cyclic time based scheduling of BSW Schedulable Entities.
Rationale:	Many BSW Modules rely on the cyclic time based call of their Schedulable Entities in order to fulfill their functionality.
Use Case:	Call of the function “Com_MainFunctionTx” to achieve periodic sending of IPdus.
Dependencies:	RTE00072
Conflicts:	--
Supporting Material:	[BRF00020] - Integration of existing BSW Scheduling into the RTE [15]
Contributes to:	--

4.1.5.2 [RTE00212] Activation Offset of BSW Schedulable Entities

ID:	RTE00212
Initiator:	WP RTE
Date:	14.07.2008
Short Description:	Activation Offset of BSW Schedulable Entities
Type:	new
Importance:	high
Description:	The RTE shall allow the definition of an Activation Offset of BSW Schedulable Entities.
Rationale:	In order to allow optimizations in the scheduling the RTE has to handle the activation offset information from a task shared reference point for time trigger BSW Schedulable Entities.
Use Case:	Mapping of BSW Schedulable Entities with different periods in the same task
Dependencies:	RTE00211 , RTE00161
Conflicts:	--
Supporting Material:	[BRF00020] - Integration of existing BSW Scheduling into the RTE [15]
Contributes to:	--

4.1.5.3 [RTE00213] Mode Switches for BSW Modules

ID:	RTE00213
Initiator:	WP RTE
Date:	14.07.2008
Short Description:	Mode Switches for BSW Modules
Type:	new
Importance:	high
Description:	The RTE shall support Mode Switches for BSW modules. BSW Schedulable Entities are scheduled dependent on modes or are activated by entering and exiting a mode.
Rationale:	Conditional scheduling of BSW Schedulable Entities dependent on different operating modes of the ECU.
Use Case:	• Initialization and finalization phases • Different communication modes
Dependencies:	RTE00144
Conflicts:	--

Supporting Material:	[BRF00020] - Integration of existing BSW Scheduling into the RTE [15]
Contributes to:	--

4.1.5.4 [RTE00214] Common Mode handling for Basic SW and Application SW

ID:	RTE00214
Initiator:	WP RTE
Date:	14.07.2008
Short Description:	Common Mode handling for Basic SW and Application SW
Type:	new
Importance:	high
Description:	The RTE shall support the coordinated switching of a Mode affecting BSW Modules and Application Software Components.
Rationale:	Synchronized behavior during a mode transition controlling AUTOSAR BSW Modules and Application Software Components.
Use Case:	ECU initialization and finalization phase.
Dependencies:	RTE00144 , RTE00143 , RTE00213
Conflicts:	--
Supporting Material:	[BRF00020] - Integration of existing BSW Scheduling into the RTE [15]
Contributes to:	--

4.1.5.5 [RTE00215] API for Mode switch notification to the SchM

ID:	RTE00215
Initiator:	WP RTE
Date:	28.09.2005
Short Description:	API for Mode switch notification to the SchM
Type:	New
Importance:	High
Description:	The SchM shall provide APIs for communication of active modes from the BSW mode manager to the SchM.
Rationale:	The SchM needs to disable the execution of certain BSW Schedulable Entities depending on modes, therefore it needs to know the current modes.
Use Case:	See RTE00213
Dependencies:	RTE00213 , RTE00214 This might be implemented via a Port-Based API or a direct C-API.
Conflicts:	--
Supporting Material:	[BRF00020] - Integration of existing BSW Scheduling into the RTE [15] [BRF00260] - Support of at Runtime dynamically schedulable BSW Modules [15]
Contributes to:	--

4.1.5.6 [RTE00216] Triggering of BSW Schedulable Entities by occurrence of External Trigger

ID:	RTE00216
Initiator:	WP RTE
Date:	21.05.2008
Short Description:	Triggering of BSW Schedulable Entities by occurrence of External Trigger
Type:	new
Importance:	high
Description:	The RTE shall support the triggering of BSW Schedulable Entities by the occurrence of External Triggers. Particular BSW Schedulable Entities in BSW dependent from External Triggers shall be executed after occurrence of the event in a defined and deterministic order specified by the integrator. The occurrence of the External Trigger is either reported via API to the RTE

	or by means of the OS (e.g. expiration of an OS Alarm). Restriction: This is only applicable for intra-ECU usage.
Rationale:	Sporadic and non timing based periodic activation of BSW Schedulable Entities in different BSW Modules.
Use Case:	Angle periodic triggering of the ignition for a combustion engine.
Dependencies:	RTE00218
Conflicts:	--
Supporting Material:	[BRF00031] Triggered Event [15] [BRF00020] Integration of existing BSW Scheduling into the RTE [15]
Contributes to:	--

4.1.5.7 [RTE00230] Triggering of BSW Schedulable Entities by occurrence of Internal Trigger

ID:	RTE00230
Initiator:	WP RTE
Date:	19.03.2009
Short Description:	Triggering of BSW Schedulable Entities by occurrence of Internal Trigger
Type:	new
Importance:	high
Description:	The Basic Software Scheduler shall support the triggering of BSW Schedulable Entities by the occurrence of Internal Trigger. Particular BSW Schedulable Entities in BSW dependent from Internal Events shall be executed after occurrence of the event in a defined and deterministic order specified by the integrator. The occurrence of the Internal Trigger is either reported via API to the RTE or by means of the OS (e.g. expiration of an OS Alarm). Restriction: This is only applicable for intra-ECU usage.
Rationale:	Decoupling from Interrupt Context inside a Basic Software Module.
Use Case:	An interrupt which shall not exceed a certain WCET activates a BSW Schedulable Entity to process more time consuming algorithms.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00031] Triggered Event [15] [BRF00020] Integration of existing BSW Scheduling into the RTE [15]
Contributes to:	--

4.1.5.8 [RTE00217] Synchronized activation of Runnable Entities and BSW Schedulable Entities

ID:	RTE00217
Initiator:	WP RTE
Date:	14.07.2008
Short Description:	Synchronized activation of Runnable Entities and BSW Schedulable Entities
Type:	New
Importance:	High
Description:	The RTE shall support the triggering of both, Runnable Entities and BSW Schedulable Entities by the same Triggered Events.
Rationale:	Synchronous activation of routines in AUTOSAR BSW modules and Application Software Components.
Use Case:	Angle periodic triggering of the routines in Application Software Components and Complex Device Drivers for a combustion engine.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00020] - Integration of existing BSW Scheduling into the RTE [15]
Contributes to:	--

4.1.5.9 [RTE00218] API for Triggering BSW modules by Triggered Events

ID:	RTE00218
Initiator:	WP RTE
Date:	14.07.2008
Short Description:	API for Triggering BSW modules by Triggered Events
Type:	New
Importance:	High
Description:	The RTE shall provide an API usable for BSW modules to notify the RTE about the occurrence of Triggered Events.
Rationale:	The sources for Triggered Events may be captured in the BSW and need to be forwarded to the application SWCs.
Use Case:	See RTE00216
Dependencies:	RTE00216 , RTE00217
Conflicts:	--
Supporting Material:	[BRF00020] - Integration of existing BSW Scheduling into the RTE [15]
Contributes to:	--

4.1.5.10 [RTE00219] Support for interlaced execution sequences of Runnable Entities and BSW Schedulable Entities

ID:	RTE00219
Initiator:	WP RTE
Date:	14.07.2008
Short Description:	Support for interlaced execution sequences of Runnable Entities and BSW Schedulable Entities
Type:	new
Importance:	high
Description:	The RTE shall support the interlaced execution sequences of Runnable Entities and BSW Schedulable Entities within the same Os Task. The whole execution sequence for all Runnable Entities and BSW Schedulable Entities which are mapped to the same OS Task can be arbitrarily defined.
Rationale:	Usage of OS Tasks for scheduling of Application Software Components and BSW Modules.
Use Case:	Reduce response time of a closed loop control by configuration of a signal flow orientated calculation sequence (plant determination, controller, actuator). Reduce number of tasks.
Dependencies:	--
Conflicts:	Such interlaced configuration might not be supported in case of partitioning and multi-core usage.
Supporting Material:	[BRF00020] - Integration of existing BSW Scheduling into the RTE [15]
Contributes to:	--

4.1.5.11 [RTE00220] ECU life cycle dependent scheduling

ID:	RTE00220
Initiator:	WP RTE
Date:	14.07.2008
Short Description:	ECU life cycle dependent scheduling
Type:	new
Importance:	high
Description:	The RTE shall support the exclusive Scheduling of BSW modules dependent from the ECU life cycle. Before the RTE is fully initialized (call of Rte_Start) or after the RTE is finalized (call of Rte_Stop) only BSW Schedulable Entities shall be scheduled.

Rationale:	Support different life-cycles of BSW Modules and Application Software Components.
Use Case:	Exclusive Scheduling of BSW Modules during start-up and shut-down phase of the ECU.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00020] - Integration of existing BSW Scheduling into the RTE [15]
Contributes to:	--

4.1.5.12 [RTE00221] Support for "BSW integration" builds

ID:	RTE00221
Initiator:	WP RTE
Date:	14.07.2008
Short Description:	Support for "BSW integration" builds
Type:	new
Importance:	high
Description:	The RTE generator shall provide means to generate the API for only for BSW Modules (with the related BSW Scheduling code), excluding the API for Application Software Components. When the input information contains Application Software Component parts these shall be ignored in this mode. The complete input information must be valid, including the ignored parts.
Rationale:	Support integration of BSW Modules without Application Software Components.
Use Case:	Pre-integration and test of BSW Module packages.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00020] - Integration of existing BSW Scheduling into the RTE [15]
Contributes to:	--

4.1.5.13 [RTE00222] Support shared exclusive areas in BSW Service Modules and the corresponding Service Component

ID:	RTE00222
Initiator:	WP RTE
Date:	16.07.2008
Short Description:	Support shared exclusive areas in BSW Service Modules and the corresponding Service Component
Type:	new
Importance:	high
Description:	The RTE shall provide APIs to enter or exit exclusive areas for both, BSW Service Modules and the corresponding Service Component.
Rationale:	Coordinated access to shared memory between BSW Schedulable Entities and BSW Runnable Entities of the SAME module (i.e. the AUTOSAR Service which has both, a BSW aspect and a SW-C aspect).
Use Case:	Coordinate the access to the NvM job buffer from the NvM module Schedulable Entities and the NvM server Runnable Entities called by the Application SW-Components.
Dependencies:	RTE00046
Conflicts:	--
Supporting Material:	[BRF00020] - Integration of existing BSW Scheduling into the RTE [15]
Contributes to:	--

4.1.5.14 [RTE00229] Support for Variant Handling of BSW Modules

Initiator:	WP RTE
Date:	30.10.2008
Short Description:	Support for Variant Handling of BSW Modules
Type:	new
Importance:	high
Description:	The RTE Generator shall support the generation of the BSW scheduling where “PreCompileTime” and “PostBuild” variability is left open and shall be resolved after the generation.
Rationale:	Some variability may stay in the input information after the RTE Generation phase.
Use Case:	A simple NvM only needs one Schedulable Entity while the full-featured NvM requires several Schedulable Entities. Both variants can be described in one input information and the generated RTE shall contain means to switch between both variants after the generation.
Dependencies:	RTE00201
Conflicts:	--
Supporting Material:	[BRF00020] - Integration of existing BSW Scheduling into the RTE [15] [BRF00029] - Variant Handling on VFB Level [15]

4.1.6 Support for Measurement and Calibration

4.1.6.1 [RTE00153] Support for Measurement

ID:	RTE00153
Initiator:	WP RTE
Date:	26.07.2006
Short Description:	Support for Measurement
Type:	new
Importance:	high
Description:	The RTE generator shall create code allowing read out of ECU internal communication data and variable contents. Responsibility of RTE is to supply RAM locations where the measurement data can be read by other SW (e.g. Basic SW, external measurement tools). This read out might be asynchronous to all RTE actions. The RTE is not responsible to deliver the measurement values to ECU external instances.
Rationale:	Measurement is needed to get knowledge about ECU internal behavior when ECU is running.
Use Case:	Monitor SWC internal signals (e.g.InterRunnableVariables), VFB communication or mode states.
Dependencies:	--
Conflicts:	--
Supporting Material:	Software Component Template [7] chapter “Measurement & Calibration”
Contributes to:	--

4.1.6.2 [RTE00154] Support for Calibration

ID:	RTE00154
Initiator:	WP RTE
Date:	26.07.2006
Short Description:	Support for Calibration
Type:	new
Importance:	high
Description:	RTE and BSW Scheduler shall support calibration process of ECUs. RTE and BSW Scheduler are not responsible to make parameter or interpolation curve/map modifications by itself but must support calibration

	data emulation. RTE and BSW Scheduler are neither responsible to handle exchange of calibration parameter values nor for communication with ECU external instances.
Rationale:	Calibration is the process of adjusting an ECU SW to fulfill its tasks to control physical processes resp. to fit to special project needs or environments.
Use Case:	Adapt ECU SW to motor specific properties. Environment specific adaptation of ECU SW.
Dependencies:	RTE00153 – Support of Measurement: calibration needs means of measurement to work properly.
Conflicts:	--
Supporting Material:	Software Component Template [7] chapter “Measurement & Calibration” ASAP Standard (www.asam.net)
Contributes to:	--

4.1.6.3 [RTE00156] Support for different calibration data emulation methods

ID:	RTE00156
Initiator:	WP RTE
Date:	07.08.2006
Short Description:	Support for different calibration data emulation methods
Type:	new
Importance:	high
Description:	The RTE generator shall support these data emulation methods for calibration purposes: directAccess Calibration data is stored in ROM and accessed directly. This method can be used with appropriate calibration hardware. Single pointered method Calibration data accesses are done via one indirection over a pointer table in RAM Double pointered method Calibration data accesses are done via a base pointer in RAM and over a pointer table in ROM/FLASH InitRAM parameter method RTE accesses calibration parameters located in RAM directly (without any indirection) and copies the values from ROM/FLASH during startup Methods 2-4 need SW support from RTE.
Rationale:	Projects in different domains have different requirements and different RAM availabilities.
Use Case:	DirectAccess method: No overhead. Appropriate HW support present or after rebuild for production Single pointered method: more available RAM present than with InitRAM method, only 1 indirection, no time for initial copy Double pointered method: less RAM needs than single pointered method when calibration is off, activate several modified parameters simultaneously InitRAM parameter method: Only few parameters to calibrate
Dependencies:	RTE00154 – Support of Calibration
Conflicts:	--
Supporting Material:	Software Component Template [7] chapter “Measurement & Calibration”
Contributes to:	--

4.1.6.4 [RTE00157] Support for calibration parameters in NVRAM

ID:	RTE00157
Initiator:	WP RTE

Date:	26.07.2006
Short Description:	Support for calibration parameters in NVRAM
Type:	new
Importance:	high
Description:	RTE shall support allocation of calibration parameters in NVRAM
Rationale:	Allocation in NVRAM allows independent parameter manipulation by other instances without re-flashing the ECU
Use Case:	Modify a NVRAM calibration parameter via a diagnostic service, e.g. modify window lifter speed or enable a SW option
Dependencies:	RTE00154 – Support of Calibration
Conflicts:	--
Supporting Material:	Software Component Template [7] chapter “Measurement & Calibration”
Contributes to:	--

4.1.6.5 [RTE00158] Support separation of calibration parameters

ID:	RTE00158
Initiator:	WP RTE
Date:	26.07.2006
Short Description:	Support separation of calibration parameters
Type:	new
Importance:	high
Description:	RTE shall support separation of calibration parameters
Rationale:	Separation required e.g. due to security reasons
Use Case:	Separate calibration parameters for monitoring purposes from the other calibration parameters to get independency from parameters for normal functional operation in case of partly corrupted memory.
Dependencies:	RTE00154 – Support of Calibration
Conflicts:	--
Supporting Material:	Software Component Template [7] chapter “Measurement & Calibration”
Contributes to:	--

4.1.6.6 [RTE00159] Sharing of calibration parameters

ID:	RTE00159
Initiator:	WP RTE
Date:	09.08.2006
Short Description:	Sharing of calibration parameters
Type:	new
Importance:	high
Description:	Several software components (and also several instances of software components) shall be able to share same calibration parameters defined in CalprmComponentTypes.
Rationale:	Avoids potential inconsistencies between several calibration parameters for same item on 1 ECU, reduces ECU resource consumption
Use Case:	Common use of calibration parameters like maximum vehicle speed, left/right steering wheel, temperature sensor interpolation curve, ..
Dependencies:	RTE00154 – Support of Calibration
Conflicts:	--
Supporting Material:	Software Component Template [7] chapter “Measurement & Calibration”
Contributes to:	--

4.1.6.7 [RTE00189] A2L Generation Support

ID:	RTE00189
Initiator:	WP RTE

Date:	17.06.2008
Short Description:	A2L Generation Support
Type:	New
Importance:	High
Description:	The RTE generator shall support the generation of output information in order to support the later generation of a complete A2L file.
Rationale:	The RTE generator is allocating the variables which shall be measurable. Therefore the information about the allocated variables shall be exported for further usage by subsequent tools.
Use Case:	In order to measure some RTE allocated variables the symbols used in the allocation have to be available for the measurement tools.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00021] - A2L Generation Support [15]
Contributes to:	--

4.1.7 General Requirements

4.1.7.1 [RTE00021] Per-ECU RTE customization

ID:	RTE00021
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Per-ECU RTE customization
Type:	new
Importance:	high
Description:	The RTE shall be customizable (generated and/or configured) for each ECU. The RTE generator should avoid, where possible, the use of generic functions and should, instead, favor functions that are configured/generated to specifically implement the required communication patterns.
Rationale:	Generic functions are considered to be too computationally expensive since the function needs to dynamically determine what actions to perform (e.g. switch on parameters). In contrast, statically configured/generated functions know implicitly what needs to be done and therefore avoid these costs and are therefore considered necessary for the production of optimal systems.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	An ECU with two or more micro-controllers can be configured using either shared memory (and hence a single OS, single basic software set, etc) or with separate memory (multiple OSs, multiple basic software sets, etc.). In the first case there is only a single ECU according to the AUTOSAR ECU architecture and therefore only one RTE. In the second case there are multiple, independent, ECUs and therefore multiple RTEs.
Contributes to:	--

4.1.7.2 [RTE00065] Deterministic generation

ID:	RTE00065
Initiator:	WP RTE
Date:	04.10.2004
Short Description:	Deterministic generation
Type:	Changed (27.10.2008)
Importance:	High
Description:	A given version of a RTE generator from a vendor - for an identical set of

	input files - shall reproduce, every time it is invoked, the same RTE code with the exception of time-related information in code comments, like 'generated at..', which may differ.
Rationale:	The generated RTE code shall be equal in case the same generator (version) and input information is used.
Use Case:	There shall be no difference (other than information in comments) between RTEs generated by the same generator for the same input files.
Dependencies:	--
Conflicts:	--
Supporting Material:	--
Contributes to:	--

4.1.7.3 [RTE00028] "1:n" Sender-receiver communication

ID:	RTE00028
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	"1:n" Sender-receiver communication
Type:	new
Importance:	high
Description:	The RTE shall support "1:n" sender-receiver communication. Sender-receiver communication is message passing and the RTE shall support scenarios with a single-sender-multiple-receivers ("1:n").
Rationale:	VFB Specification requires support for single-sender-multiple-receiver ("1:n")
Use Case:	--
Dependencies:	RTE00131 – "n:1: communication
Conflicts:	--
Supporting Material:	VFB Specification
Contributes to:	--

4.1.7.4 [RTE00131] "n:1" Sender-receiver communication

ID:	RTE00131
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	"n:1" Sender-receiver communication
Type:	New
Importance:	High
Description:	The RTE shall support "n:1" sender-receiver communication. Sender-receiver communication is message passing and the RTE shall support scenarios with multiple-senders-single-receiver ("n:1").
Rationale:	VFB Specification requires support for multiple-senders-one-receiver ("n:1")
Use Case:	--
Dependencies:	RTE00028 – "1:n" communication
Conflicts:	--
Supporting Material:	VFB Specification
Contributes to:	--

4.1.7.5 [RTE00029] "n:1" Client-server communication

ID:	RTE00029
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	"n:1" Client-server communication
Type:	new

Importance:	medium
Description:	The RTE shall support multiple-client-single-server (“n:1”) client-server (function invocation) communication. Individual clients are independent – there is no coordination of requests between clients. Single-client-multiple-server (“1:n”) communication is not required. Such communication raises issues about buffering and selection of results that are application dependent and therefore not considered to be the domain of the RTE.
Rationale:	--
Use Case:	The fog light shall serve as backup for the brake light this can be implemented by one “fog light” - server switching fog light on/off and two clients, the “fog light”-client and the “brake light”-client both switching fog lights on and off for different purposes.
Dependencies:	VFB requires support multiple-clients-one-server (“n:1”) but explicitly does not require to support single-client-multiple-server (“1:n”) communication
Conflicts:	--
Supporting Material:	VFB Specification
Contributes to:	--

4.1.7.6 [RTE00079] Single asynchronous client-server interaction

ID:	RTE00079
Initiator:	WP RTE
Date:	05.10.2004
Short Description:	Single asynchronous client-server interaction
Type:	new
Importance:	high
Description:	The RTE shall support at most one asynchronous call at a time from a single operation in a required port categorized by a client-server interface (i.e. there can only be one outstanding request per "AsynchronousServerCallPoint"). Note that a single client can simultaneously have multiple outstanding requests provided each is to *different* server operations. When a SW-component instance restarts it may receive a stale reply – replies to a request made before the component was restarted. The RTE shall forward stale replies and it is the job of the SW-component instance to detect and reject the reply, for example, through sequence numbers.
Rationale:	Requirement from VFB spec (4.1.4.2 Client-Server Communication). There is no queuing (of parameters and return locations) on the client-side and therefore only one single outstanding request can be supported.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	Software Component Template [7] VFB Specification [6]
Contributes to:	--

4.1.7.7 [RTE00080] Multiple requests of servers

ID:	RTE00080
Initiator:	WP RTE
Date:	05.10.2004
Short Description:	Multiple requests of servers
Type:	new
Importance:	high
Description:	The RTE shall support the queuing of concurrent calls to a server (by

	different clients). A server specified using the “BUFFERING queue(n)” attribute may have queued requests from multiple clients. Requests shall be read from the server’s queue using first-in-first-out semantics. Depending on the RTE implementation the queue may be present in the either in the RTE or in COM.
Rationale:	Requirement from VFB spec (4.1.4.2 Client-Server Communication)
Use Case:	--
Dependencies:	RTE00033
Conflicts:	--
Supporting Material:	Queues are applied at the operation level, i.e. each operation in a client-server interface has a dedicated queue.
Contributes to:	--

4.1.7.8 [RTE00162] “1:n” External Trigger communication

ID:	RTE00162
Initiator:	WP RTE
Date:	21.05.2008
Short Description:	“1:n” External Trigger communication
Type:	new
Importance:	high
Description:	The RTE shall support the communication of External Trigger events from one trigger source to multiple trigger sinks (“1:n”). The Runnable Entity(s) in the trigger sink(s) linked to the event shall be executed after occurrence of the event in a defined and deterministic order defined by the integrator. Restriction: This is only applicable for intra-ECU usage.
Rationale:	Sporadic and non timing based periodic activation of Runnable Entities in different Software Components.
Use Case:	Angle periodic triggering of the Mass Air Flow calculation for a combustion engine.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00031] - Triggered Event [15]
Contributes to:	--

4.1.7.9 [RTE00163] Support for InterRunnableTriggering

ID:	RTE00163
Initiator:	WP RTE
Date:	21.05.2008
Short Description:	Support for InterRunnableTriggering
Type:	new
Importance:	high
Description:	The RTE shall support InterRunnableTriggering. A software component instance shall be able to explicitly trigger the execution of Runnable Entities of the same component instance.
Rationale:	Decoupling of calculation and processing sequences inside a Software Component instance.
Use Case:	A time base triggered Runnable Entity which shall not exceed a certain WCET activates a second Runnable Entity of the same SW-C instance in case of error to process more time consuming exception handling.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00031] - Triggered Event [15]
Contributes to:	--

4.1.7.10 [RTE00235] Support queued triggers

ID:	RTE00235
Initiator:	WP RTE
Date:	20.06.2011
Short Description:	Support queued triggers
Type:	new
Importance:	high
Description:	External and internal trigger event communication shall support queuing the number of triggers issued by a trigger source. When the trigger source is informed of the end of execution of all triggered executable entities, the RTE shall (if any trigger is in the queue) dequeue a trigger by activating again the triggered executable entities.
Rationale:	--
Use Case:	There are use cases existing where it is important to queue the number of activations done by a trigger source when triggers are faster issued in the trigger source as the runnables in the trigger target can be activated.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00031] - Triggered Event [15]
Contributes to:	--

4.1.7.11 [RTE00025] Static communication

ID:	RTE00025
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Static communication
Type:	new
Importance:	high
Description:	The RTE shall support only those communication connections known when the RTE is generated – the source(s) and destination(s) of all communication shall be known statically. Static communication is considered to include Application Software Component access to the publisher-subscriber service – components are statistically configured to access the service and then subscribers are dynamically chosen from a statically configured set.
Rationale:	Dynamic communication is deemed too expensive (both at run-time and in code overhead) and would therefore limit the range of devices for which the RTE is suitable.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	VFB Specification In AUTOSAR (and in COM) only static communication connections are permitted. If dynamic communication will be allowed in future, all specifications have to be reworked.
Contributes to:	--

4.1.7.12 [RTE00144] Mode Switch notification via AUTOSAR interfaces

ID:	RTE00144
Initiator:	WP RTE

Date:	28.09.2005
Short Description:	RTE shall support the notification of mode switches via AUTOSAR interfaces
Type:	New
Importance:	High
Description:	RTE shall use the well defined mechanisms of AUTOSAR interfaces for the communication of active modes from the mode manager to the mode dependent software component.
Rationale:	Use the flexibility and configuration mechanisms defined for AUTOSAR interfaces.
Use Case:	See RTE00143
Dependencies:	RTE00143
Conflicts:	--
Supporting Material:	AUTOSAR. Software Component Template. Version 1.04 – Final, 04 2005 (p. 61, L 7-8)
Contributes to:	--

4.1.7.13 [RTE00018] Rejection of invalid configurations

ID:	RTE00018
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Rejection of invalid configurations
Type:	new
Importance:	high
Description:	The RTE generator shall detect, and reject where appropriate, the invalid deployment and communication configuration of application and basic software components.
Rationale:	The RTE is required to reject “invalid“ configurations, e.g. wait point in category 1a or 1b Runnable Entity, interface incompatibility, …
Use Case:	Multiple instantiation of a component where the “`supportsMultipleInstantiation`” flag is not set. The RTE generator shall reject the mapping of event-triggered and “communication triggered” Runnable Entities to the same *basic* task. (An implementation is possible, if inefficient, for extended tasks).
Dependencies:	RTE00062 – local access to basic software
Conflicts:	--
Supporting Material:	A valid RTE cannot be generated for an invalid configuration. For example, AUTOSAR is required to be interoperable with “legacy ECUs” (Requirement MAIN190, AUTOSAR_MainRequirements_v2.2_r.doc, p. 32). The capabilities of such ECUs may not be precisely compatible with AUTOSAR and therefore some configurations, e.g. client-server communication with the legacy ECU, should be rejected – that’s an invalid configuration.
Contributes to:	--

4.1.7.14 [RTE00055] Use of global namespace

ID:	RTE00055
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	RTE’s use of global namespace
Type:	New
Importance:	High
Description:	The RTE specification shall define standard naming conventions for all the symbols created by the RTE generator that are visible within the global namespace.

	Creating symbol definitions within the global namespace using this naming convention is the exclusive right of the RTE generator. Application and/or basic software components shall not create symbols defined by this naming convention within the global namespace.
Rationale:	Prevents conflicts with symbols created by application and/or basic software components.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	All symbols use the prefix “RTE”.
Contributes to:	--

4.1.7.15 [RTE00164] Ensure a unique naming of generated types visible in the global namespace

ID:	RTE00164
Initiator:	WP RTE
Date:	20.05.2008
Short Description:	Ensure a unique naming of generated types visible in the global namespace.
Type:	New
Importance:	High
Description:	The RTE shall define standard naming conventions for all the type symbols created by the RTE generator that are visible within the global namespace. The naming convention shall be defined in a way, that each type requiring an own implementation within the global namespace is named uniquely.
Rationale:	Prevent type conflicts within the global namespace caused by integration of SWC provided from different suppliers.
Use Case:	--
Dependencies:	RTE00055
Conflicts:	--
Supporting Material:	[BRF00078] - Avoidance of duplicated Type Definitions in RTE Types Header File [15]
Contributes to:	--

4.1.7.16 [RTE00165] Suppress identical "C" type re-definitions

ID:	RTE00165
Initiator:	WP RTE
Date:	20.05.2008
Short Description:	Suppress identical "C" type re-definitions.
Type:	New
Importance:	High
Description:	The RTE generator shall suppress the re-definition of compatible type declarations resulting in identical "C" type definition.
Rationale:	ECU Extract can contain (compatible) type definitions which are resulting in the identical type definitions.
Use Case:	Improvement of code quality by avoidance of type re-definitions. Enabling stricter type checking and suppress compiler errors/warnings.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00078] - Avoidance of duplicated Type Definitions in RTE Types Header File [15]
Contributes to:	--

4.1.7.17 [RTE00166] Use the AUTOSAR Standard Types in the global namespace if the AUTOSAR data type is mapped to an AUTOSAR Standard Type

ID:	RTE00166
Initiator:	WP RTE
Date:	20.05.2008
Short Description:	Use the AUTOSAR Standard Types in the global namespace if the AUTOSAR data type is mapped to an AUTOSAR Standard Type.
Type:	New
Importance:	High
Description:	The RTE shall suppress the re-declaration of AUTOSAR data types mapped to the AUTOSAR Standard Types and shall use directly the AUTOSAR Standard Types instead.
Rationale:	Improving code quality by avoidance of type casts in case of communication with Basic Software and in case of library calls.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00078] - Avoidance of duplicated Type Definitions in RTE Types Header File [15]
Contributes to:	--

4.1.7.18 [RTE00167] Encapsulate a Software Component local name space

ID:	RTE00167
Initiator:	WP RTE
Date:	20.05.2008
Short Description:	Encapsulate a Software Component local name space.
Type:	New
Importance:	High
Description:	The RTE generator shall encapsulate a Software Component local name space for declarations and definitions for APIs and types related to one Software Component.
Rationale:	From a SW-C providers point of view this is the only name space which can be ensured by the SW-C provider. Therefore the RTE Generator has to provide the mapping from the SW-C locally used names within the application header file to the names used within the global namespace.
Use Case:	Support embedding of SW-Components in different ECUs by avoiding type conflicts.
Dependencies:	RTE00055, RTE00087
Conflicts:	--
Supporting Material:	[BRF00078] - Avoidance of duplicated Type Definitions in RTE Types Header File [15]
Contributes to:	--

4.1.7.19 [RTE00126] C support

ID:	RTE00126
Initiator:	WP RTE
Date:	18.11.2004
Short Description:	C language support
Type:	changed (25.03.2009)
Importance:	high
Description:	The RTE generator shall support SW-Components and BSW Modules created using ‘ANSI C’.
Rationale:	Support common used programming language.

Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	Specification of the Virtual Functional Bus [6] ANSI/ISO 9899-1989, “Programming Languages – C”
Contributes to:	--

4.1.7.20 [RTE00138] C++ support

ID:	RTE00138
Initiator:	WP RTE
Date:	01.02.2005
Short Description:	C++ language support
Type:	changed (25.03.2009)
Importance:	high
Description:	The RTE generator shall support SW-Components and BSW Modules created using ISO C++.
Rationale:	Support common used programming language.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	Specification of the Virtual Functional Bus [6] ISO/IEC 14882-1998, “Programming Languages - C++”
Contributes to:	--

4.1.7.21 [RTE00051] RTE API mapping

ID:	RTE00051
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	RTE API mapping
Type:	new
Importance:	high
Description:	The RTE specification shall define a standard naming convention for all RTE API artifacts visible by a component author that are created by the RTE generator. The names of RTE API artifacts shall not include the component instance names.
Rationale:	The requirement for an API mapping enables the signature of generated RTE functions to be hidden from users and permits targeted optimization depending on configuration. The hiding of signatures is desirable for two reasons: The names of generated RTE functions may be long (to ensure name uniqueness) and therefore unwieldy for users to reference directly. The generated function name may include information not known when the component is compiled, such as the instance name.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	At the point the component is written the component instance name is not defined (deployment has not be performed) and therefore the component instance name cannot be included in the API. However, the instance name is required by the RTE generator when actually generating the RTE to ensure name uniqueness and therefore the RTE generator shall implement a **well-defined** API mapping from the RTE API to the generated RTE API functions.
Contributes to:	--

4.1.7.22 [RTE00048] RTE Generator input

ID:	RTE00048
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	RTE Generator input
Type:	new
Importance:	high
Description:	The RTE generator shall accept input consisting of zero or more databases/files.
Rationale:	It is not reasonable to expect input as a single file. The RTE generator shall collect information from multiple input files and check their consistency.
Use Case:	• Provide each Software Component type description in a separate input file. • Provide the System description in a separate input file. • Be prepared to not find any input file and provide proper error reporting.
Dependencies:	--
Conflicts:	--
Supporting Material:	AUTOSAR design flow does not restrict input to one source and therefore RTE generator must be flexible.
Contributes to:	--

4.1.7.23 [RTE00023] RTE Overheads

ID:	RTE00023
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	RTE Overheads
Type:	new
Importance:	high
Description:	The RTE generator shall provide a configuration option specifying the overall design goal of the generated RTE - minimizing memory and/or run-time overhead.
Rationale:	Unfortunately, the different feasible directions of optimizations can contradict each other and only a trade-off close to the overall design goal can be found. But this may sufficient in order to meet the constraints of the ECU or the mapping of that special functionality to the ECU is not possible, anyway. Thus, this requirement requests that the RTE generator should be able to generate RTEs that fit on a wide a range of devices as possible (obviously depending on configuration and component deployment).
Use Case:	The RTE generator shall generate RTEs for the ECUs needs with respect to the given resources (processor speed, memory, etc.).
Dependencies:	--
Conflicts:	--
Supporting Material:	VFB_C20 – Rephrased since requirement is inherently untestable – one can never know when requirements have been minimized (e.g. local minima). However, rejection does not absolve an implementation from a general requirement to be “efficient” with ECU resources… The test is – does the generated RTE fit for the ECUs resources, but not more!
Contributes to:	--

4.1.7.24 [RTE00024] Source-code AUTOSAR software components

ID:	RTE00024
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Source-code AUTOSAR software components
Type:	New
Importance:	High
Description:	The RTE shall support AUTOSAR software components where the source is available (“source-code software components”).
Rationale:	AUTOSAR software components as source-code increase the optimization potential for the generated RTE.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	Software Component Template [7]
Contributes to:	--

4.1.7.25 [RTE00140] Binary-code AUTOSAR software components

ID:	RTE00140
Initiator:	WP RTE
Date:	08.02.2005
Short Description:	Binary-code AUTOSAR software components
Type:	new
Importance:	high
Description:	The RTE shall support AUTOSAR software components where only the object code (“binary-code software components”) is available.
Rationale:	Binary-code AUTOSAR software components are required for IP hiding.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	Software Component Template [7] Support for binary-code AUTOSAR software components requires the same compiler type and compiler version.
Contributes to:	--

4.1.7.26 [RTE00083] Optimization for source-code components

ID:	RTE00083
Initiator:	WP RTE
Date:	08.10.2004
Short Description:	Optimization for source-code components
Type:	new
Importance:	high
Description:	The RTE generator should provide optimized communication when the source-code of an Application Software Component is available. Optimizations envisaged include elimination of the RTE for that communication channel.
Rationale:	VFB_C20
Use Case:	Conversion of intra-task S-R communication to direct variable write. This can only be performed for source-code components since the deployment is not known when a binary-code component is compiled.
Dependencies:	RTE00023 (minimize overheads) has been rephrased from the specification of VFB_C20 to be testable. This requirement is considered testable provided

	the “contemporary” solution is suitable for comparison.
Conflicts:	--
Supporting Material:	VFB_C20 requires that optimizations should enable the RTE to impose zero overhead when compared with “contemporary” implementations. When comparing solutions, one should make sure that the AUTOSAR system has the same features as the "contemporary solution", for example, by using the “`SupportsMultipleInstantiation`" attribute of the "Implementation" class.
Contributes to:	--

4.1.7.27 [RTE00027] VFB to RTE mapping shall be semantic preserving

ID:	RTE00027
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	VFB to RTE mapping shall be semantic preserving
Type:	new
Importance:	high
Description:	The RTE generator shall configure the RTE to implement the specified communication paths while retaining their semantics.
Rationale:	The mapping from VFB model expressed in the XML input to generated RTE is required to be semantic preserving. This requirement applies regardless of whether communication is done by COM, by the RTE directly or if the RTE generator optimizes the generated RTE to bypasses the RTE completely for certain communication paths.
Use Case:	The RTE generator is not permitted to modify the semantics of communication, for example, converting synchronous to asynchronous.
Dependencies:	--
Conflicts:	--
Supporting Material:	VFB_C10
Contributes to:	--

4.1.7.28 [RTE00190] Support for variable-length Data Types

ID:	RTE00190
Initiator:	WP RTE
Date:	17.06.2008
Short Description:	Support for variable-length Data Types
Type:	new
Importance:	medium
Description:	The RTE shall support the transfer of data with variable size (could be bigger than 8 bytes but has a fixed maximum size). The variable length support shall be applicable to • strings • primitive byte arrays
Rationale:	The support of variable length data allows a more efficient utilization of resources in the ECU and on the communication busses. It also supports the implementation of dynamic communication protocols (like SAE J1939). Handling of strings with always a fixed length is not preferable, since a lot of ECU resources will be allocated by this approach. The usage of an always fixed size would result in wasting runtime and bandwidth in case of not used but reserved space. The alternative usage of a couple of interfaces and signals to serve different sizes will complicate the APIs and waste configuration freedom.
Use Case:	• Transferring message and information strings of variable length

	between ECU's, e.g. Central ECU and Instrument Cluster. • Transferring the content of a received SMS to display it. • Transferring (variable) strings to a display. • Primitive byte arrays which are similar to strings but are not zero terminated.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00004] – Support for variable-length Data Types [15] [BRF00290] – Support of Array Type with dynamic Size [15]
Contributes to:	--

4.1.7.29 [RTE00234] Support for Record Type sub-setting

Initiator:	WP-1.2
Date:	23.07.2011
Short Description:	Support for Record Type sub-setting
Type:	new
Importance:	high
Description:	The RTE shall support that ports are connected which are typed by different interfaces and where the elements of the provide port are typed by composite data types which composite elements are mapped to elements of the require port. Hereby the require port might contain only a sub set of the elements contained in the provide port.
Rationale:	Since the handling of data in a consistent manner requires using a record type, it shall be allowed at the receiver of a RecordType to only receive a sub-set of the sent record data elements. Since different receivers do require a different sub-set of the provided data.
Use Case:	4 wheel speed signals and the movement direction signal are provided in one record. If a receiver is only interested in the movement direction information all of the other information from this record does not have to be considered at this specific receiver.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.1.7.30 [RTE00098] Explicit Sending

ID:	RTE00098
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Explicit Sending
Type:	new
Importance:	high
Description:	The RTE shall provide a mechanism for making requests for explicit sending of AUTOSAR signals (i.e. an implementation of DataSendPoint). The *DataSendPoint* of a Runnable Entity references an instance of a data-element in a provided port. Using the *DataSendPoint*, a Runnable Entity can use an explicit RTE API call to write new values of the specified data-element (which may cause an immediate send depending on component and communication configuration).
Rationale:	Implementation of internal component model from VFB Specification.
Use Case:	--
Dependencies:	RTE00134, RTE00128 and RTE00129 – The current SwCT and VFB specifications require that the Runnable Entity is of cat 2 for explicit sending. This situation is being revised so that cat 1b and 2 will be able to access DataSendPoints (e.g. extended to cat 1b).

Conflicts:	--
Supporting Material:	VFB Specification [6] Software Component Template [7]
Contributes to:	--

4.1.7.31 [RTE00129] Implicit Sending

ID:	RTE00129
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Implicit Sending
Type:	New
Importance:	High
Description:	The RTE shall provide a mechanism for the implicit sending of data elements. The mechanism shall grant write-access to a data element of a provided port that may be freely changed until the Runnable Entity returns. The presence of DataWriteAccess means that the Runnable Entity will potentially modify the DataElement in the pPort. The Runnable Entity has free access to the data-element while it is running but the Runnable Entity should ensure that the data-element is in a consistent state when it returns. When using DataWriteAccess the new values of the data-element are made available, by the RTE, when the Runnable Entity returns. Depending on the configuration the RTE may either have nothing to do or it may need to actually initiate sending of the data element.
Rationale:	Implementation of internal component model from VFB Specification.
Use Case:	--
Dependencies:	RTE00134 and RTE00128 – Previous SwCT and VFB specifications required that the Runnable Entity is (at most?) of cat 1b. This situation is being revised so that cat 1a are allowed to access DataReadAccess and DataWriteAccess.
Conflicts:	--
Supporting Material:	VFB Specification [6] Software Component Template [7]
Contributes to:	--

4.1.7.32 [RTE00128] Implicit Reception

ID:	RTE00128
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Implicit Reception
Type:	new
Importance:	high
Description:	The RTE shall provide a mechanism for the implicit reception of data elements. The mechanism shall grant read-access to a data element of a required port that will not be modified by the RTE and may be freely read until the Runnable Entity returns. The presence of DataReadAccess means that the Runnable Entity will require access to the DataElement in the rPort. The Runnable Entity expects that the contents of this data does not change during execution of the Runnable Entity.
Rationale:	--
Use Case:	--
Dependencies:	RTE00134 and RTE00129 – Previous SwCT and VFB specifications required that the Runnable Entity is (at most?) of cat 1b. This situation is

	being revised so that cat 1a are allowed to access DataReadAccess and DataWriteAccess.
Conflicts:	--
Supporting Material:	VFB Specification [6] Software Component Template [7]
Contributes to:	--

4.1.7.33 [RTE00141] Explicit Reception

ID:	RTE00141
Initiator:	WP RTE
Date:	04.03.2005
Short Description:	Explicit Reception
Type:	new
Importance:	high
Description:	The RTE shall provide a mechanism for making requests for explicit reception of AUTOSAR signals (i.e. an implementation of DataReceivePoint). The *DataReceivePoint* of a Runnable Entity references an instance of a data-element in a required port. Using the *DataReceivePoint*, a Runnable Entity can use an explicit RTE API call to receive new values of the specified data-element (e.g. the 'next' value is read out of the local queue).
Rationale:	Implementation of internal component model from VFB Specification.
Use Case:	--
Dependencies:	RTE00134, RTE00128, RTE00098, and RTE00129
Conflicts:	--
Supporting Material:	VFB Specification [6]
Contributes to:	--

4.1.7.34 [RTE00092] Implementation of VFB model “waitpoints”

ID:	RTE00092
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Implementation of VFB model “waitpoints”
Type:	New
Importance:	High
Description:	The RTE API shall support wait points at which Runnable Entities will block until an “RTEEvent” occurs. A category 2 Runnable Entity shall, through the RTE API, be able to suspend its execution (i.e. block) until a well-defined event occurs. This requirement does not mean that “wait points” shall be explicitly specified in the API and could be satisfied by blocking calls that suspend the caller until an event (defined in the VFB meta-model) occurs.
Rationale:	Runnable Entities need to be able to suspend execution (block) until a defined event occurs.
Use Case:	• Waiting for the arrival of data • Waiting for the return of a call
Dependencies:	RTE00027
Conflicts:	--
Supporting Material:	This requirement is a special case of RTE00027. Software Component Template [7]
Contributes to:	--

4.1.7.35 [RTE00145] Compatibility mode

ID:	RTE00145
Initiator:	WP RTE
Date:	16.11.2005
Short Description:	Compatibility mode
Type:	New
Importance:	High
Description:	The RTE Generator shall provide a compatibility operating mode that guarantees compatibility between different RTE implementations both for source code and object code components.
Rationale:	For IP hiding purposes a component may be delivered as object code only. Then it has to be precompiled against a header file created by an RTE implementation that may not be the RTE implementation that is used in the integration environment.
Use Case:	--
Dependencies:	RTE00146
Conflicts:	--
Supporting Material:	--
Contributes to:	--

4.1.7.36 [RTE00146] Vendor mode

ID:	RTE00146
Initiator:	WP RTE
Date:	16.11.2005
Short Description:	Vendor mode
Type:	New
Importance:	High
Description:	The RTE Generator may provide a vendor operating mode allowing vendor-specific optimizations.
Rationale:	The vendor mode does not need to rely on predefined data structures and gives the individual RTE implementations the freedom for further optimizations for an additional reduction of the RTE overhead.
Use Case:	--
Dependencies:	RTE00145
Conflicts:	--
Supporting Material:	--
Contributes to:	--

4.1.7.37 [RTE00148] Support “Specification of Memory Mapping”

ID:	RTE00148
Initiator:	WP RTE
Date:	30.01.2006
Short Description:	Support “Specification of Memory Mapping”
Type:	New
Importance:	High
Description:	The document “Specification of Memory Mapping” shall be supported by RTE implementations.
Rationale:	To allow the integration of several software modules in one ECU.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	Specification of Memory Mapping [11]

Contributes to:	--

4.1.7.38 [RTE00149] Support “Specification of Compiler Abstraction”

ID:	RTE00149
Initiator:	WP RTE
Date:	30.01.2006
Short Description:	Support “Specification of Compiler Abstraction”
Type:	New
Importance:	High
Description:	The document “Specification of Compiler Abstraction” shall be supported by RTE implementations.
Rationale:	--
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	Specification of Compiler Abstraction [13]
Contributes to:	--

4.1.7.39 [RTE00150] Support “Specification of Platform Types”

ID:	RTE00150
Initiator:	WP RTE
Date:	30.01.2006
Short Description:	Support “Specification of Platform Types”
Type:	New
Importance:	High
Description:	The document “Specification of Platform Types” shall be supported by RTE implementations.
Rationale:	--
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	Specification of Platform Types [12]
Contributes to:	--

4.1.7.40 [RTE00151] Support RTE relevant requirements of the “General Requirements on Basic Software Modules”

ID:	RTE00151
Initiator:	WP RTE
Date:	30.01.2006
Short Description:	Support RTE relevant requirements of the “General Requirements on Basic Software Modules”
Type:	New
Importance:	High
Description:	The following requirements of the “General Requirements on Basic Software Modules” shall be supported by RTE implementations: [BSW007] [BSW101] [BSW161] [BSW00300] [BSW00305] [BSW00307] [BSW00308] [BSW00310] [BSW00312] [BSW00326] [BSW00327] [BSW00330] [BSW00336] [BSW00337] [BSW00338] [BSW00342] [BSW00345] [BSW00346] [BSW00347] [BSW00353] [BSW00397] [BSW00399] [BSW00400] [BSW00405] [BSW00407] [BSW00415] [BSW00447]
Rationale:	--
Use Case:	--
Dependencies:	--

Conflicts:	--
Supporting Material:	General Requirements on Basic Software Modules [2], only a subset of the requirements needs to be taken into account for the RTE.
Contributes to:	--

4.1.7.41 [RTE00171] Support for fixed and constant data

ID:	RTE00171
Initiator:	WP RTE
Date:	03.06.2008
Short Description:	Support for fixed and constant data
Type:	new
Importance:	medium
Description:	The RTE shall support the access to fixed and constant data shareable between SWCs.
Rationale:	SWCs shall be able to access fixed data which are commonly defined for several SWCs.
Use Case:	Definition of general constant values to be used by many SWCs (like pi, avogadro).
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00105] - System Parameter [15]
Contributes to:	--

4.1.7.42 [RTE00178] Data consistency of NvBlockComponentType

ID:	RTE00178
Initiator:	WP RTE
Date:	22.05.2008
Short Description:	Data consistency of NvBlockComponentType
Type:	new
Importance:	medium
Description:	The RTE shall protect the data defined in NvBlockComponentType against concurrent write and read access (by SWCs or NVRAM Manager).
Rationale:	The data of a NvBlockComponentType is shared amongst SWCs (and also with the NVM). The data needs to be protected against concurrent write and read access.
Use Case:	--
Dependencies:	RTE00176
Conflicts:	--
Supporting Material:	[BRF00022] - Modification of NVRAM Memory Access Concept [15] Software Component Template [7]
Contributes to:	--

4.1.7.43 [RTE00179] Support of Update Flag for Data Reception

ID:	RTE00179
Initiator:	WP RTE
Date:	09.06.2008
Short Description:	Support of Update Flag for Data Reception
Type:	new
Importance:	low
Description:	In case of Sender Receiver communication with last is best semantics, if the configuration requires, RTE shall support an update flag that indicates whether there has been an update of the data since the last read operation from the Software Component to the data element. The update flag shall be set during reception of the data by the RTE and reset during the read

	operation from the software component.
Rationale:	Allows polling for updates.
Use Case:	This allows a Runnable Entity - that is, e.g., triggered by the FlexRay cycle - to take action depending on the availability of new data. It shall be possible to refrain from re-reading the data element, if the data is not updated.
Dependencies:	RTE00110
Conflicts:	--
Supporting Material:	[BRF00092] Extension of the Receive Queue Behavior [15]
Contributes to:	--

4.1.7.44 [RTE00184] RTE Status “Never Received”

ID:	RTE00184
Initiator:	WP RTE
Date:	06.06.2008
Short Description:	RTE Status “Never Received”
Type:	new
Importance:	medium
Description:	The RTE shall support the RTE-Status “never received”. This is the new initial status of each data element for which it is configured. This initial status will be cleared when the first reception occurs.
Rationale:	This additional status establishes the possibility to check, whether a data element has been changed since system start or partition restart.
Use Case:	Get the information whether involved data have been received at any time since system start or partition restart.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00023] - Additional RTE Status ‘Never Received’ [15] [BRF00104] - Mode Dependent Reset of Initial Values [15]
Contributes to:	--

4.1.7.45 [RTE00191] Support for Variant Handling

ID:	RTE00191
Initiator:	WP Methodology and Templates
Date:	19.05.2009
Short Description:	Support for Variant Handling
Type:	new
Importance:	high
Description:	The RTE shall support the resolution and implementation of the AUTOSAR Variant Handling mechanism.
Rationale:	With the support of variant handling it is also possible to leave some variation until the RTE generation. Then the RTE Generator needs to support the resolution of these variation points.
Use Case:	• Using the same ECU (hardware and software) for several vehicle lines. This leads to different communication matrices, which need to be used depending in which vehicle line the ECU is build in. Also the functionality may be slightly different in each vehicle line. • A project can choose out of different existing implementations of AUTOSAR SWCs respectively SW-compositions with same or compatible interfaces to implement the sum of the system functionality. This addresses pre-built variant handling.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00029] [BRF00155]
Contributes to:	--

4.1.7.46 [RTE00201] Contract Phase with Variant Handling support

ID:	RTE00201
Initiator:	WP Methodology and Templates
Date:	19.06.2008
Short Description:	Contract Phase with Variant Handling support
Type:	new
Importance:	high
Description:	The RTE shall support the generation of the Application Software Component header file where “PreCompileTime” or “PostBuildLoadable” variability is left open and shall be resolved later.
Rationale:	Some variability may be defined as having the binding time “PreCompileTime” or “PostBuildLoadable”. This variability has to be represented in the generated application software component header file.
Use Case:	The existence of a Port is specified to be variable with a binding time “PreCompile”. Then in the application software component header file the corresponding APIs could be wrapped in #IFDEFs so the compiler can actually benefit from the binding during compiling. When the binding time is PostBuildLoadable the application software component header file shall consider the superset of all possible variants.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00029] Variant Handling on VFB Level [15]
Contributes to:	--

4.1.7.47 [RTE00202] Support for array size variants

ID:	RTE00202
Initiator:	WP Methodology and Templates
Date:	19.06.2008
Short Description:	Support for array size variants
Type:	New
Importance:	High
Description:	The RTE generator shall be able generate arrays whose size is depending on a model attribute.
Rationale:	Software supporting a variable – but fixed during runtime - number of entities.
Use Case:	“Length of Arrays, size of Calibration Axis”: depending on the system, the number of cylinders varies, values required per cylinder are combined in arrays to implement scalable algorithms.
Dependencies:	This shall be resolved until “PreCompile” time.
Conflicts:	--
Supporting Material:	[BRF00029] Variant Handling on VFB Level [15]
Contributes to:	--

4.1.7.48 [RTE00204] Support the selection / deselection of SWC prototypes

ID:	RTE00204
Initiator:	WP Methodology and Templates
Date:	19.06.2008
Short Description:	Support the selection / de-selection of SWC prototypes
Type:	New
Importance:	High
Description:	The RTE shall support the configuration - post-build time - a software component to run or not. When the SWC does not run, the RTE needs to behave as if the software

	component does not exist, therefore all RTE events for that SWC's Runnable Entities shall be suppressed.
Rationale:	Handling of optional functionality.
Use Case:	Disabling trailer functionality if not used.
Dependencies:	The behavior of a "deselected" SW-Component prototype shall be defined.
Conflicts:	--
Supporting Material:	[BRF00029] Variant Handling on VFB Level [15]
Contributes to:	--

4.1.7.49 [RTE00206] Support the selection of a signal provider

ID:	RTE00206
Initiator:	WP Methodology and Templates
Date:	19.06.2008
Short Description:	Support the selection of a signal provider
Type:	New
Importance:	High
Description:	RTE shall support selection of a signal provider at "PostBuildLoadable" time.
Rationale:	The provider of a signal could come from one/several internal software component, or via a network signal. The RTE shall support the selection of the source post-build loadable. Support variants where the source of a signal can be either internal or external to the ECU.
Use Case:	A system where in one variant the temperature sensor is connected to the local ECU and is available locally, another variant the temperature value is provided via the network.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00029] Variant Handling on VFB Level [15]
Contributes to:	--

4.1.7.50 [RTE00207] Support N to M communication patterns while un-resolved variations are affecting these communications

ID:	RTE00207
Initiator:	WP Methodology and Templates
Date:	19.06.2008
Short Description:	Support N to M communication patterns while un-resolved variations are affecting these communications.
Type:	New
Importance:	High
Description:	In a system N:M communication shall be allowed when this communication is subject to a variation point. After all affecting variants have been resolved only 1:N or N:1 communication shall be possible. This shall be available for PostBuildLoadable variants.
Rationale:	Some variants will have multiple components publish the same data. The receivers require the data from any publisher. But after variant resolving only one publisher is actually sending the data.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00029] Variant Handling on VFB Level [15]
Contributes to:	--

4.1.7.51 [RTE00231] Support native interface between Rte and Com for Strings and uint8 arrays

ID:	RTE00231
Initiator:	WP RTE
Date:	31.03.2009
Short Description:	Support native interface between Rte and Com for Strings and uint8 arrays
Type:	new
Importance:	high
Description:	Com does support the sending / receiving of an array of uint8 natively. The Rte shall use this native interface if the communicated data type is supported by Com.
Rationale:	Allow an easy access to data natively supported by Com.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--
Contributes to:	--

4.1.7.52 [RTE00232] Synchronization of runnable entities

ID:	RTE00232
Initiator:	WP RTE
Date:	19.05.2009
Short Description:	Synchronization of runnable entities
Type:	new
Importance:	high
Description:	The RTE shall support the synchronization of runnable entities. Synchronization means that the mechanisms that implement the runnable entity activation (OsAlarm, OsTask, OsEvent, etc...) are triggered (expired, activated, set, etc...) at the same point of time. Synchronization shall be possible even if runnable entity are mapped to different OsTasks, different OsPartitions, different cores or even different ECUs.
Rationale:	In some cases, multiple sensors or actuators have to be acquired or driven by different runnable entities at the same point of time.
Use Case:	On a Flexray cluster, some processus are time triggered and shall be synchronized between ECUs.
Dependencies:	Global time service for synchronization over ECU is required.
Conflicts:	--
Supporting Material:	[BRF00120] Synchronization of ECUs local time within a cluster
Contributes to:	--

4.1.8 VFB Tracing

4.1.8.1 [RTE00005] Support for ‘trace’ build

ID:	RTE00005
Initiator:	WP RTE
Date:	04.10.2004
Short Description:	The RTE generator shall support ‘trace’ builds
Type:	Changed 08.07.2008
Importance:	high
Description:	If the RTE provides means for tracing which cost additional RAM and/or ROM and/or RUNTIME in the ECU. It shall be possible to switch these features off statically during RTE generation.
Rationale:	Allow monitoring of VFB communication and runtime behavior.

Use Case:	DLT, Debugging
Dependencies:	RTE00045, RTE00008
Conflicts:	--
Supporting Material:	VFB90, VFB_C10
Contributes to:	--

4.1.8.2 [RTE00045] Standardized VFB tracing interface

ID:	RTE00045
Initiator:	WP RTE
Date:	04.10.2004
Short Description:	Standardized VFB tracing interface
Type:	new
Importance:	medium
Description:	In 'trace' build the RTE implementation shall provide a standardized interface to make data values and events available to VFB tracing tools. When in 'trace' build the RTE generator inserts hook calls at interesting points (e.g. API invocation, interactions with COM, task start, Runnable Entity start, etc).
Rationale:	By defining a standardized VFB tracing interface tool vendors can adapt existing trace tools quickly – this will promote the adoption of AUTOSAR ECUs.
Use Case:	--
Dependencies:	RTE00005
Conflicts:	--
Supporting Material:	VFB Specification (VFB90); VFB Specification V1.03, Sect. 4.4.3, p. 94 An RTE implementation can define 'null' implementations of the hooks – to enable an RTE to build – but then permit them to be re-implemented by tool vendors to target vendor specific interfaces to standard tracing tools. The VFB tracing interface shall be configurable to interface with the AUTOSAR Debugging [18] or Diagnostic Log and Trace [19] functionality.
Contributes to:	--

4.1.8.3 [RTE00008] VFB tracing configuration

ID:	RTE00008
Initiator:	WP RTE
Date:	04.10.2004
Short Description:	VFB tracing configuration
Type:	new
Importance:	medium
Description:	If the RTE is configured for tracing communication across the VFB, it shall be possible to detail the configuration of the RTE for what has to be logged and traced.
Rationale:	Tracing only of interesting signals/activations/system states, to reduce overhead in RAM+RUNTIME.
Use Case:	--
Dependencies:	RTE00005
Conflicts:	--
Supporting Material:	VFB Specification (VFB90)
Contributes to:	--

4.1.8.4 [RTE00192] Support multiple trace clients

ID:	RTE00192
Initiator:	WP Debugging
Date:	17.06.2008
Short Description:	Support multiple trace clients
Type:	new
Importance:	high
Description:	The RTE Generator shall be able to support multiple trace clients for the same trace event.
Rationale:	It shall be possible to configure several trace functions on the same trace event. The individual trace functions shall not need to know about each other.
Use Case:	The Debugger and the Diagnostic Log and Trace (DLT) are interested in the same trace event.
Dependencies:	RTE00008
Conflicts:	--
Supporting Material:	[BRF00224] – Allow monitoring of DEM, RTE and COM to improve diagnostics [15] [BRF00296] – RTE/VFB Trace interface needed for Log&Trace [15]
Contributes to:	--

4.1.8.5 [RTE00003] Tracing of sender-receiver communication

ID:	RTE00003
Initiator:	WP RTE
Date:	04.10.2004
Short Description:	Tracing of sender-receiver communication
Type:	new
Importance:	medium
Description:	The ‘trace’ builds of the RTE generator shall support tracing of sender-receiver signals on the VFB. The RTE should provide means for the tracing of transported signals of sender-receiver communication. It should be possible to trace both intra-ECU and inter-ECU communication.
Rationale:	Log data and supply it for debugging purposes.
Use Case:	--
Dependencies:	RTE00005
Conflicts:	--
Supporting Material:	VFB Specification (VFB90)
Contributes to:	--

4.1.8.6 [RTE00004] Tracing of client-server communication

ID:	RTE00004
Initiator:	WP RTE
Date:	04.10.2004
Short Description:	Tracing of client-server communication
Type:	new
Importance:	medium
Description:	The ‘trace’ builds of the RTE generator shall support the tracing of client-server communication. The RTE should provide means for the tracing of transported signals of client-server communication. It should be possible to trace both intra-ECU and inter-ECU communication.

Rationale:	Log data and supply it for debugging purposes.
Use Case:	--
Dependencies:	RTE00005
Conflicts:	--
Supporting Material:	VFB Specification (VFB90)
Contributes to:	--

4.1.9 Application Software Component Initialization and Finalization

4.1.9.1 [RTE00052] Initialization and finalization of components

ID:	RTE00052
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	Initialization and finalization of components
Type:	new
Importance:	high
Description:	The RTE shall support initialization and finalization of Application Software Components. The term “initialization of a component” refers to the phase of a software components life cycle which will be executed before entering the normal operational mode, normally in order to set up an appropriate environment for executing the application. The term “finalization of a component” refers to the phase of a software components life cycle which will be executed after the normal operational mode, normally in order to reset the operational environment to a determined state.
Rationale:	The general ECU life-cycle is characterized by transitions from inoperational states to run states and back to the inoperational states of the ECUStateManager. When in run states the OS scheduler is responsible for the ECUs schedule and Runnable Entities will be executed with respect to the OS scheduler. In all other states the ECUStateManager is responsible for schedule of the ECU, but Runnable Entities can only be executed in synchronous and sequential way. Since the initialization and finalization phase of components refer to the transition from inoperational state to run states or vice versa, the Runnable Entities given for initialization and finalization can be executed synchronously as well as asynchronously depending on the intention of the system designer. That means the RTE shall provide means to invoke those Runnable Entities in the one or the other way and shall ensure that Runnable Entities of Application Software Components always communicate at least conceptually with modules of the basic software via the RTE.
Use Case:	Reading application specific parameters from non-volatile RAM while initialization application specific software component, writing application specific parameters to the non-volatile RAM while finalization of the Application Software Component.
Dependencies:	SWS ECUStateManager
Conflicts:	--
Supporting Material:	VFB Specification (Sched42)
Contributes to:	--

4.1.9.2 [RTE00070] Invocation order of Runnable Entities

ID:	RTE00070
Initiator:	WP RTE
Date:	01.10.2004

Short Description:	Invocation order of Runnable Entities
Type:	Changed 01.09.2008
Importance:	high
Description:	The RTE generator shall respect the invocation order of Runnable Entities as specified in the input information.
Rationale:	The invocation order of Runnable Entities may be of importance to the functionality of the algorithm or its timing. The invocation order may be provided with the description of the SW-Components or in the configuration of the RTE.
Use Case:	A control loop implemented with several SW-Components. The execution of the individual Runnable Entities of this control loop shall be respected by the generated RTE code.
Dependencies:	--
Conflicts:	--
Supporting Material:	Software Component Template [7]
Contributes to:	--

4.1.10 API

4.1.10.1 [RTE00100] Compiler independent API

ID:	RTE00100
Initiator:	WP RTE
Date:	04.11.2004
Short Description:	Compiler independent API
Type:	new
Importance:	high
Description:	The RTE API (for a particular programming language) shall be compiler and platform independent.
Rationale:	There shall be no need to change an Application Software Component source-code when the component is moved between ECUs and/or the compiler is changed.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	In addition the RTE should, ideally, also be retargetable (i.e. portable) with minimum effort. This is explicitly not stated as an RTE requirement since it is a statement about RTE implementation and not RTE behavior.
Contributes to:	--

4.1.10.2 [RTE00168] Typing of RTE API

ID:	RTE00168
Initiator:	WP RTE
Date:	20.05.2008
Short Description:	Typing of RTE API.
Type:	New
Importance:	High
Description:	For parameters which are typed by an AUTOSAR data type the RTE API shall either use data types related to the whole AUTOSAR data type or data types related to the used Base Type. The kind of API is defined in the SWC-Description and is part of the contract phase input.
Rationale:	Dependent from the SWCs implementation either usage of Base Type related types (comprising only the "C" implementation aspect) or full AUTOSAR data type related types (comprising the semantic of the data type as well) is more advantageous. This depends how many library functions are used in the SW-C's implementation. In general the RTE shall support both

	typing to avoid unnecessary type casts as far as possible and to support strong type checking.
Use Case:	Strong type checking with a static code analyzer.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00078] - Avoidance of duplicated Type Definitions in RTE Types Header File [15]
Contributes to:	--

4.1.10.3 [RTE00059] RTE API passes 'in' primitive data types by value

ID:	RTE00059
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	RTE API shall pass 'in' primitive data types by value
Type:	new
Importance:	high
Description:	An API input parameter that is a primitive data type (with the exception of a string) shall be passed by value.
Rationale:	Pass by value is efficient for small data types.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	In the context of this requirement, primitive data types include integers (both signed and unsigned), floating point and opaque types.
Contributes to:	--

4.1.10.4 [RTE00060] RTE API shall pass 'in' composite data types by reference

ID:	RTE00060
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	RTE API shall pass 'in' composite data types by reference.
Type:	new
Importance:	high
Description:	The RTE API shall pass all input parameters that are composite data types (i.e. a record or an array) or strings by reference.
Rationale:	Pass by reference is efficient for large data types.
Use Case:	--
Dependencies:	--
Conflicts:	This requirement may be in conflict in systems using memory protection mechanism. If the sender of the composite data and the receiver of the composite data are belonging to different memory domains RTE must copy the composite data to a memory area accessible by the receiver in order to avoid memory violation faults caused by the receiver.
Supporting Material:	RTE00036 Assignment to OS Applications
Contributes to:	--

4.1.10.5 [RTE00061] 'in/out' and 'out' parameters

ID:	RTE00061
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	'in/out' and 'out' parameters
Type:	new

Importance:	high
Description:	The RTE API shall pass ‘in/out’ and “out” formal parameters by reference.
Rationale:	Required so that modifications to the actual parameters made by the called function are visible to the caller.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	--
Contributes to:	--

4.1.10.6 [RTE00115] API for data consistency mechanism

ID:	RTE00115
Initiator:	WP RTE
Date:	17.11.2004
Short Description:	API for data consistency mechanism
Type:	new
Importance:	high
Description:	The RTE shall provide an API to access the data consistency mechanism(s).
Rationale:	The data mechanism may rely on AUTOSAR OS mechanisms (e.g. resources) which cannot be accessed directly by Application Software Components.
Use Case:	--
Dependencies:	RTE00032 – data consistency mechanism
Conflicts:	--
Supporting Material:	VFB Requirements (VFB_C60, Sched70)
Contributes to:	--

4.1.10.7 [RTE00075] API for accessing per-instance memory

ID:	RTE00075
Initiator:	WP RTE
Date:	05.10.2004
Short Description:	API for accessing per-instance memory
Type:	new
Importance:	high
Description:	The RTE generator shall generate an API in the application header file through which the Runnable Entities of a component instance can access their per-instance memory for reading and writing.
Rationale:	Required by the software component template
Use Case:	--
Dependencies:	RTE00013 – per-instance memory
Conflicts:	--
Supporting Material:	“per-instance memory” is synonymous to “Individual data of a component instance” described in Software Component Template [7] The RTE does not impose a data consistency mechanism on access to per-instance memory. If a component requires consistency then the `RTEEnter` and `RTEExit` API calls should be used.
Contributes to:	--

4.1.10.8 [RTE00107] Support for INFORMATION_TYPE attribute

ID:	RTE00107
Initiator:	WP RTE
Date:	17.11.2004

Short Description:	Support for INFORMATION_TYPE attribute
Type:	new
Importance:	high
Description:	The RTE generator shall support the INFORMATION_TYPE attribute with values “data” and “event”. The RTE generator shall support different information types for each data item in an AUTOSAR interface. The RTE generator shall raise a configuration-time error if the specification of INFORMATION_TYPE is inconsistent for sender and receiver.
Rationale:	Required by VFB Specification.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	VFB Specification v1.03, p. 40 VFB Specification v1.03, Section 4.1.7.4 VFB Specification v1.03, Table 4-15 VFB Specification v1.04, p. 61 line 14 When “data” is specified, the RTE shall presume the following; Receive mode shall be (explicit read) “data_read_access”. Buffering shall be “last_is_best”. Specification of an initial value is required. Specification of “TIME_FOR_RESYNC” is required. Specification of “LIVELIHOOD” is required. When “event” is specified, the RTE shall presume the following: The “TIME_FOR_RESYNC” is not specified. The “LIVELIHOOD” is not specified. An attempt to redefine the presumptions shall cause a configuration time error. Failure to fulfill the presumptions shall cause a configuration time error.
Contributes to:	--

4.1.10.9 [RTE00108] Support for INIT_VALUE attribute

ID:	RTE00108
Initiator:	WP RTE
Date:	17.11.2004
Short Description:	Support for INIT_VALUE attribute
Type:	new
Importance:	high
Description:	The RTE generator shall support the INIT_VALUE attribute for both intra-ECU and inter-ECU communication (though the latter is expected to requires no direct support if AUTOSAR COM is used). If an initial value is specified for a receiver and not a sender (or vice versa) the RTE generator shall apply the same initial value to both sender and receiver. If an initial value is specified for both sender and receiver the RTE generator shall use the specifications for the receiver.
Rationale:	The input information can contain conflicting values for the INIT_VALUE attribute (sender and receiver).
Use Case:	The INIT_VALUE is different in the sender- and receiver com spec.
Dependencies:	RTE00068 – Signal initial values

Conflicts:	--
Supporting Material:	VFB Specification v1.03, p. 42
Contributes to:	--

4.1.10.10 [RTE00109] Support for RECEIVE_MODE attribute

ID:	RTE00109
Initiator:	WP RTE
Date:	17.11.2004
Short Description:	Support for RECEIVE_MODE attribute
Type:	new
Importance:	high
Description:	The RTE generator shall support the RECEIVE_MODE attribute with the values “data_read_access”, “wake_up_of_wait_point” and “activation_of_runnable_entity”. The RTE generator shall support different receive modes for each data item in an AUTOSAR interface.
Rationale:	Derived from the VFB Specification.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	VFB Specification v1.03, p. 43 When “data_read_access” is specified the RTE generator shall create a non-blocking read API for the data item. The name of the API could include the port name and data item name. When “wake_up_of_wait_point” is specified the RTE generator shall create a blocking read API for the data item. The name of the API shall include the port name and data item name. The API could support a timeout specified at configuration time. When “activation_of_runnable_entity” is specified the RTE generator shall invoke a Runnable Entity when data is received passing the received data as parameters to the Runnable Entity. The name of the Runnable Entity could include the port name and data item name.
Contributes to:	--

4.1.10.11 [RTE00110] Support for BUFFERING attribute

ID:	RTE00110
Initiator:	WP RTE
Date:	17.11.2004
Short Description:	Support for BUFFERING attribute
Type:	new
Importance:	high
Description:	The RTE generator shall support the BUFFERING attribute with the values “last_is_best” (sender/receiver only), “queue” and “no” (client/server only). The RTE generator shall support different buffering specifications for each data item in an AUTOSAR interface. Note the queues may be implemented by either the RTE or by COM.
Rationale:	--
Use Case:	--
Dependencies:	RTE00033 – Serialization of Server Runnable Entities
Conflicts:	--

Supporting Material:	VFB Specification v1.03, p. 43 When “last_is_best” is specified the RTE generator Shall create a non-consuming read API for the data item. The name of the API shall include the port name and data item name. Shall store received data shall be stored in a single-element queue and new data shall overwrite existing data. When “queue” is specified for sender/receiver the RTE generator Shall create a consuming read API for the data item. The name of the API shall include the port name and data item name. Shall store received data in a queue (the length of which is specified by the “queue” attribute value) accessed on a first-in-first-out basis. Shall discard new data if the queue is full. When “no” or “queue” is specified for client/server see RTE00033.
Contributes to:	--

4.1.10.12 [RTE00111] Support for CLIENT_MODE attribute

ID:	RTE00111
Initiator:	WP RTE
Date:	17.11.2004
Short Description:	Support for CLIENT_MODE attribute
Type:	new
Importance:	high
Description:	The RTE generator shall support the CLIENT_MODE attribute with the values “synchronous” and “asynchronous”. The RTE generator shall support different client mode specifications for each operation in an AUTOSAR interface.
Rationale:	--
Use Case:	--
Dependencies:	RTE00049 – construction of task bodies
Conflicts:	--
Supporting Material:	VFB Specification v1.03, p. 51 When “synchronous” is specified the RTE generator Shall create an API that invokes the operation synchronously. The name of the API could include the port name and operation name. Shall support a timeout specified at the configuration time. The RTE generator should ignore any timeout specified for intra-task communication. When “asynchronous” is specified the RTE generator Shall create an API that invokes the operation asynchronously. The name of the API could include the port name and operation name. Reject configurations that specify asynchronous invocation of server where both Runnable Entities are mapped to the same task.
Contributes to:	--

4.1.10.13 [RTE00121] Support for FILTER attribute

ID:	RTE00121
Initiator:	WP RTE
Date:	17.11.2004
Short Description:	Support for FILTER attribute
Type:	new
Importance:	high
Description:	The RTE generator shall support the FILTER attribute. If specified, the

	attribute value shall specify the filter type used. The RTE generator shall support different filter specifications for each data item in an AUTOSAR interface. The RTE generator shall ensure that value-based filtering is available for all receivers whether communication occurs via COM or is handled by the RTE.
Rationale:	Same behavior independent of RTE implementation and component deployment.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	VFB Specification v1.03, p. 44
Contributes to:	--

4.1.10.14 [RTE00147] Support for communication infrastructure time-out notification

ID:	RTE00147
Initiator:	WP RTE
Date:	27.01.2006
Short Description:	Support for communication infrastructure time-out notification
Type:	New
Importance:	High
Description:	The RTE shall support the notification of time-outs on cyclically received signals/signal-groups via COM. The deadline monitoring has to be enabled for these signals and the callback has to be configured in COM. This is only applicable for sender-receiver communication with info-type “data”.
Rationale:	Indicate the missing update of signals received via COM.
Use Case:	When the “vehicle speed” signals is not updated because of communication infrastructure errors it needs to be indicated to the SW-Components interested.
Dependencies:	RTE00069
Conflicts:	--
Supporting Material:	nonBSW Feature list (v0.40) F049 The value is specified as “aliveTimeout” in the Software Component Template [7] (Required ComSpec). The value is specified as “timeout” in combination with “needsOutdatedIndication” in the System Template (SystemSignal). COM already performs the “deadline monitoring” and notifies the RTE.
Contributes to:	--

4.1.10.15 [RTE00078] Support for Data Element Invalidation

ID:	RTE00078
Initiator:	WP RTE
Date:	02.02.2006
Short Description:	Support for Data Element Invalidation
Type:	Changed (27.10.2008)
Importance:	High
Description:	The RTE shall support the invalidation of Data Elements. The RTE shall provide an API to invalidate a Data Element and to query if a Data Element has been invalidated. Also a notification on the reception of an invalid Data Element shall be supported.
Rationale:	The “canInvalidate” communication attribute shall be visible to the software

	components.
Use Case:	Data invalid
Dependencies:	--
Conflicts:	--
Supporting Material:	--
Contributes to:	--

4.1.10.16 [RTE00122] Support for Transmission Acknowledgement

ID:	RTE00122
Initiator:	WP RTE
Date:	17.11.2004
Short Description:	Support for Transmission Acknowledgement
Type:	Changed (27.10.2008)
Importance:	High
Description:	The RTE generator shall support Transmission Acknowledgement for outgoing communication.
Rationale:	Allow the transmitting Application Software Component to wait for the acknowledgement and handle the successful / failed transmission.
Use Case:	--
Dependencies:	RTE00125
Conflicts:	--
Supporting Material:	--
Contributes to:	--

4.1.10.17 [RTE00125] Rejection of “1:n” communication with the Transmission Acknowledgement

ID:	RTE00125
Initiator:	WP RTE
Date:	18.11.2004
Short Description:	Rejection of “1:n” communication with the Transmission Acknowledgement
Type:	Changed (27.10.2008)
Importance:	high
Description:	The RTE generator shall reject configurations of Transmission Acknowledgement where there are multiple receivers.
Rationale:	Too high overhead for communication, which would need to aggregate the results of multiple communication protocols in the COM-Stack.
Use Case:	--
Dependencies:	RTE00018 – rejection invalid configurations
Conflicts:	--
Supporting Material:	VFB Specification v1.03, p. 42
Contributes to:	--

4.1.10.18 [RTE00094] Communication and Resource Errors

ID:	RTE00094
Initiator:	WP RTE
Date:	09.10.2004
Short Description:	Communication and Resource Errors
Type:	new
Importance:	high
Description:	The RTE shall handle errors related to communication or resources. The RTE is required to handle communication errors (e.g. message

	transmission failed) and resource errors (e.g. network not available) and to notify the relevant software component through the RTE API.
Rationale:	--
Use Case:	--
Dependencies:	RTE00084 – Support infrastructural errors
Conflicts:	--
Supporting Material:	Software Component Template [7] VFB v1.03 explicitly delegates the definition of the error handling mechanism to be defined by WP RTE.
Contributes to:	--

4.1.10.19 [RTE00084] Support infrastructural errors

ID:	RTE00084
Initiator:	WP RTE
Date:	08.10.2004
Short Description:	Support infrastructural errors
Type:	new
Importance:	high
Description:	The RTE API shall support the forwarding of infrastructural errors (see [6]) to components. This can occur synchronously with API calls (e.g. read, send) or asynchronously (i.e. activation of Runnable Entity). Infrastructural errors include communication and resource errors.
Rationale:	VFB Specification [6]
Use Case:	--
Dependencies:	RTE00094 – Communication and Resource Errors
Conflicts:	--
Supporting Material:	VFB Specification [6] Software Component Template [7]
Contributes to:	--

4.1.10.20 [RTE00123] Forwarding of application level server errors

ID:	RTE00123
Initiator:	WP RTE
Date:	17.11.2004
Short Description:	The RTE shall forward application level errors from server to client
Type:	new
Importance:	high
Description:	The RTE shall pass the application error ID together with the communication reply from the server to the client. The RTE shall only pass the application error, if no structural error is present.
Rationale:	For client-server communication, the application SW components require a method to transfer application specific errors under the condition that there are no structural errors on the communication path.
Use Case:	A crypto library provides a server. An application error should be returned if the arguments of the call are miss-configured.
Dependencies:	RTE00124 – API for application level server errors
Conflicts:	--
Supporting Material:	--
Contributes to:	--

4.1.10.21 [RTE00124] API for application level server errors

ID:	RTE00124
Initiator:	WP RTE

Date:	17.11.2004
Short Description:	API for application level errors during Client Server communication
Type:	new
Importance:	high
Description:	The RTE shall communicate application level errors on the same path as structural errors of the communication stack. The RTE shall receive error information from the server operation's return value.
Rationale:	This requirement enables the efficient use of return values to pass error IDs. By a common use of the return value for structural and application errors, the application only has to check once for "OK".

Use Case:	`Rte_StatusType`
	`sqrt(Rte_Instance self,`
	`  Double p, Double *result)`
	`{`
	`  if (p < (Double)0.0) {`
	`  /* Set application error`
	`  * (API to be defined) */`
	`  return ERROR_IMAGINARY_NUMBER;`
	`  }`
	`  *result = (Double)sqrt(p);`
	`  return RTE_E_OK;`
	`}`

Dependencies:	RTE00123 – Forwarding of application level server errors
Conflicts:	--
Supporting Material:	--
Contributes to:	--

4.1.10.22 [RTE00089] Independent access to interface elements

ID:	RTE00089
Initiator:	WP RTE
Date:	08.10.2004
Short Description:	Independent access to interface elements
Type:	new
Importance:	high
Description:	The RTE API shall support independent access to data items (sender-receiver interface) or operations (client-server interface).
Rationale:	Required by the VFB Specification
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	VFB Specification Data items (or operations) in an interface form multiple logical channels between the same end-points (ports). Each logical channel is handled independently – data can be sent and received or operations invoked without reference to other logical channels. Since logical channels are independent there is not guarantee of consistency between sends over different channels.
Contributes to:	--

4.1.10.23 [RTE00137] API for mismatched ports

ID:	RTE00137
Initiator:	WP RTE

Date:	01.02.2004
Short Description:	API for mismatched ports
Type:	new
Importance:	high
Description:	The RTE generator shall provide null API calls for data elements or operations for ports where more elements/operations are provided than required. The API for an unconnected provided data element or operation shall discard the input parameters and return “no error”.
Rationale:	--
Use Case:	A provided sender-receiver port defines two data elements ‘a’ and ‘b’ yet is required by a port with only element ‘a’. The API call for ‘b’ shall be generated but shall have no effect.
Dependencies:	--
Conflicts:	--
Supporting Material:	Software Component Template [7]
Contributes to:	--

4.1.10.24 [RTE00139] Support for unconnected ports

ID:	RTE00139
Initiator:	WP RTE
Date:	01.02.2004
Short Description:	Support for unconnected ports
Type:	Changed (26.10.2009)
Importance:	high
Description:	The RTE shall handle ports, whether required or provided, that are not connected. The APIs for an unconnected required sender/receiver port shall return a dedicated status code that the sender is not connected. The result value shall be the init value. The API to call a client-server port for an unconnected required port shall return a dedicated status code that the server is not connected. The API to collect the result from an asynchronous client-server port for an unconnected required port shall return a dedicated status code that the server is not connected. The API for an unconnected provided sender/receiver port shall discard the input parameters and return “no error”.
Rationale:	In a component based system design there is a high chance to end up with unconnected ports for software components. In a variant rich system design unconnected ports can occur.
Use Case:	A not connected port of a software component.
Dependencies:	RTE00200
Conflicts:	--
Supporting Material:	Software Component Template [7]
Contributes to:	--

4.1.10.25 [RTE00200] Support of unconnected R-Ports

ID:	RTE00200
Initiator:	WP RTE
Date:	30.06.2008
Short Description:	Support of unconnected R-Ports

Type:	new
Importance:	high
Description:	The RTE Generator shall support generating an RTE with unconnected R-Ports. The strict checking of unconnected R-Ports shall be supported via configuration as well.
Rationale:	During the development of an ECU there are intermediate building states when not all needed communication partners are available yet, however it shall be possible to generate an RTE anyways.
Use Case:	The RTE Generator issues a warning for each unconnected R-Port during the generation if strict checking is enabled.
Dependencies:	RTE00139
Conflicts:	--
Supporting Material:	--
Contributes to:	--

4.1.10.26 [RTE00155] API to access calibration parameters

ID:	RTE00155
Initiator:	WP RTE
Date:	03.08.2006
Short Description:	API to access calibration parameters
Type:	new
Importance:	high
Description:	The SW-C and basic software module source code shall be independent from the actual calibration method (data emulation with SW or HW support) chosen for the needed calibration parameters. To abstract from the different access methods to calibration parameters the RTE and SchM shall provide an API.
Rationale:	The SW-C source code shall use a dedicated API to access the calibration parameters.
Use Case:	The SW-C code uses the same API call regardless whether the calibration parameter is stored directly in ROM or is stored in a structure to support data emulation with SW support.
Dependencies:	RTE00154 – Support of Calibration
Conflicts:	--
Supporting Material:	Software Component Template [7] chapter “Measurement & Calibration”
Contributes to:	--

4.1.10.27 [RTE00183] RTE Read API returning the dataElement´s value

ID:	RTE00183
Initiator:	WP RTE
Date:	01.06.2008
Short Description:	RTE Read API returning the dataElement´s value
Type:	new
Importance:	low
Description:	For explicit sender-receiver communication, when it is configured, the RTE generator shall provide an API returning the value of a primitive DataElement directly in the C-language return statement of the API.
Rationale:	When the SWC implementation does not need the Std_ReturnType value, this more efficient API returns the result as a return value.
Use Case:	Many calls to RTE_Read() are expected for Application SWCs. Allow the RTE to generate efficient code which does not need special code optimization compilers to benefit from.
Dependencies:	--
Conflicts:	--

Supporting Material:	[BRF00024] - Additional RTE read API using Return Value [15]
Contributes to:	--

4.1.10.28 [RTE00185] RTE API with Rte_IFeedback

ID:	RTE00185
Initiator:	WP RTE
Date:	01.06.2008
Short Description:	RTE API with Rte_IFeedback
Type:	new
Importance:	high
Description:	The RTE shall support the generation of an API to retrieve the transmission acknowledgement in an implicit communication.
Rationale:	Allow to query the transmission state also for implicit communication.
Use Case:	Enable a Runnable Entity to check whether the information provided from the last execution (in an implicit API) has actually been transmitted.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00259] - Extend RTE API with Rte_IFeedback [15]
Contributes to:	--

4.1.10.29 [RTE00203] API to read system constant

ID:	RTE00203
Initiator:	WP Methodology and Templates
Date:	19.06.2008
Short Description:	API to read system constant
Type:	New
Importance:	High
Description:	The RTE shall provide an API to read a system constant value. This API shall be usable for the C-preprocessor code or the C-compiler code. Support for the following kinds if information shall be generated: • Read the actual value of the SystemConstantDef • Read the setting of an attribute (e.g. array size) • Check the existence of a variable element
Rationale:	The software module implementation can use the value of the system constant in its execution code. The software module implementation can query the existence of a variable element. This is applicable for Basic Software as well as for application software components.
Use Case:	The existence of a Port is specified to be variable with a binding time “PreCompile”. The implementation of the SWC can query the value of the system constant used to enable/disable the port, in order to change its behavior.
Dependencies:	This requirement is dedicated to variant handing, a more generic requirement is RTE00171 which might lead to the same implementation.
Conflicts:	--
Supporting Material:	[BRF00029] Variant Handling on VFB Level [15]
Contributes to:	--

4.1.11 C/C++ API

4.1.11.1 [RTE00087] Software Module Header File generation

ID:	RTE00087
Initiator:	WP RTE

Date:	08.10.2004
Short Description:	Software Module Header File generation
Type:	new
Importance:	high
Description:	The RTE Generator shall create exactly one software module header file to be explicitly included in each C/C++ application or basic software component type that defines that component's RTE API. There may be a hierarchy of include files implicitly included.
Rationale:	Required to define API mapping and to perform optimizations and monitoring targeted for specific components. The software component header file is generated and can therefore include component specific information, including task header files for zero-overhead access to OS facilities.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	AUTOSAR design flow. This requirement does not preclude a component including its own header files.
Contributes to:	--

4.1.12 Initialization and Finalization Operation

Requirements for component finalization are considered above (RTE00052).

4.1.12.1 [RTE00116] RTE Initialization and finalization

ID:	RTE00116
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	RTE Initialization and finalization
Type:	Changed (29.09.2008)
Importance:	high
Description:	The RTE generator shall provide mechanisms to initialize and finalize the RTE in two steps: Step 1: • set all initial values of the communication and mode machine instances • start the schedule of BSW modules • treat the communication with application SW-Cs like unconnected remote communication Step2: • start the schedule and communication of all application SW-Cs The RTE startup shall support the startup of SW-C and BSW modules mapped to different OS applications, specifically different cores and memory partitions.
Rationale:	Support the initialization of the BSW Scheduler and the RTE.
Use Case:	Initialize the mode management and the scheduling of BSW before initializing and executing the Application SW-Components.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00020] - Integration of existing BSW Scheduling into the RTE [15] [BRF00261] Enable Integrator to optimize Startup/Shutdown Behavior of BSW Modules [15]
Contributes to:	--

4.1.13 Partition Restarting and Termination

4.1.13.1 [RTE00195] No activation of Runnable Entities in terminated or restarting partitions

ID:	RTE00195
Initiator:	WP Error Handling
Date:	26.06.2008
Short Description:	No activation of Runnable Entities in terminated or restarting partitions
Type:	new
Importance:	high
Description:	When a partition is terminated or restarting, the RTE shall ensure that RTEEvents will not activate Runnable Entities of the terminated or restarting OS-Application (RTE may use OS means to implement this requirement).
Rationale:	Terminated partitions shall not be executed. The RTE shall ensure that they will not resuscitate in any way. During restarting activities partitions are not ready to be executed before their environment has been cleaned up. This should be enforced by RTE.
Use Case:	Error handling strategy in case of a memory or timing fault.

	Avoid usage of potentially inconsistent data.
Dependencies:	RTE00223, RTE00224
Conflicts:	--
Supporting Material:	[BRF00275] – Capability for Application Level SWC Management (stop, start, restart) [15]
Contributes to:	--

4.1.13.2 [RTE00196] Inter-partition communication consistency

ID:	RTE00196
Initiator:	WP Error Handling
Date:	26.06.2008
Short Description:	Inter-partition communication consistency
Type:	new
Importance:	high
Description:	When two SWCs communicate, if one SWC is on a terminated partition (resp. a partition being restarted), it should behave for the initiating SWC as if it was mapped on a shutdown remote ECU (resp. an ECU being restarted). The RTE may provide the feedback with a timeout error immediately.
Rationale:	Servers on terminated partitions will not answer and server calls should result in timeouts for the clients. Servers' feedback should be dropped if the client is terminated or has been restarted since it sent the request. Received data during a restart should be discarded and init values should be proposed to the SWCs.
Use Case:	Avoid transmission of potentially inconsistent data. Avoid inconsistent feedback to clients (sequence number needed).
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00275] – Capability for Application Level SWC Management (stop, start, restart) [15]
Contributes to:	--

4.1.13.3 [RTE00223]Callout for partition termination notification

ID:	RTE00223
Initiator:	WP Error Handling
Date:	24.07.2008
Short Description:	Callout for partition termination notification
Type:	new
Importance:	high
Description:	The RTE generator shall provide an API to indicate to the RTE that a partition will be terminated or restarted.
Rationale:	The RTE should stop communications with the specified partition.
Use Case:	Call from a ProtectionHook / ErrorHandling manager.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00275] – Capability for Application Level SWC Management (stop, start, restart) [15]
Contributes to:	--

4.1.13.4 [RTE00224]Callout for partition restart request

ID:	RTE00224
Initiator:	WP Error Handling
Date:	24.07.2008
Short Description:	Callout for partition restart request

Type:	new
Importance:	high
Description:	The RTE generator shall provide an API to indicate to the RTE that a partition needs to be restarted. This API should restore an initial RTE environment for the partition and re-activate communication with this partition.
Rationale:	In case of restart, the partition is first terminated from the ProtectionHook, and needs to be restarted when the infrastructure is ready. Init values will be expected by the restarted SWCs.
Use Case:	Call from the OSRestartTask, after the BSW have been also notified.
Dependencies:	--
Conflicts:	--
Supporting Material:	[BRF00275] – Capability for Application Level SWC Management (stop, start, restart) [15]
Contributes to:	--

4.1.14 Fault Operation

Errors are directly reported to invoking Software Component (see RTE00094).

4.2 Non-Functional Requirements

4.2.1 General Requirements

4.2.1.1 [RTE00064] AUTOSAR Methodology

ID:	RTE00064
Initiator:	WP RTE
Date:	04.10.2004
Short Description:	AUTOSAR methodology
Type:	new
Importance:	high
Description:	The RTE generator shall operate according to the AUTOSAR methodology.
Rationale:	--
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	AUTOSAR Methodology
Contributes to:	--

4.2.1.2 [RTE00019] RTE is the communication infrastructure

ID:	RTE00019
Initiator:	WP RTE
Date:	01.10.2004
Short Description:	RTE is the communication infrastructure
Type:	new
Importance:	high
Description:	All communication between Application Software Components and between Application Software Components and basic software components shall occur, at least conceptually, via the RTE. Note that communication between modules within the basic software and to shared libraries does NOT occur through the RTE. This is a basic requirement and ensures that the RTE controls all communication involving Application Software Components. This requirement is not intended to prevent the RTE generator providing optimizations that bypass the RTE such as optimizing client-server to direct function call.
Rationale:	AUTOSAR ECU architecture
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	VFB Specification This requirement applies regardless of whether communication is done by COM, by the RTE directly or if the RTE generator optimizes the generated RTE to bypasses the RTE completely for certain communication paths. The phrase “at least conceptually” is used to indicate that on the conceptual (model M2/M1 levels) all communication occurs via the virtual function bus and, since the RTE is the realization of the VFB for an ECU, via the RTE. However this is only conceptual since the actual implementation (model M0 level) may not use the RTE for communication. For example, client-server communication conceptually occurs via the RTE but may be implemented as a direct function call (and hence bypass the RTE).

Contributes to:	--

5 References

5.1 Deliverables of AUTOSAR

[1] Glossary,
AUTOSAR_TR_Glossary.pdf

[2] General Requirements on Basic Software Modules,
AUTOSAR_SRS_BSWGeneral.pdf

[3] Requirements on RTE,
AUTOSAR_SRS_RTE.pdf

[4] Specification of RTE Software,
AUTOSAR_SWS_RTE.pdf

[5] Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate

[6] Virtual Functional Bus,
AUTOSAR_EXP_VFB.pdf

[7] Software Component Template,
AUTOSAR_TPS_SoftwareComponentTemplate.pdf

[8] Methodology,
AUTOSAR_MOD_Methodology.pdf

[9] Specification of ECU Configuration,
AUTOSAR_TPS_ECUConfiguration.pdf

[10] Specification of ECU Configuration Parameters,
AUTOSAR_MOD_ECUConfigurationParameters.pdf

[11] Specification of Memory Mapping,
AUTOSAR_SWS_MemoryMapping.pdf

[12] Specification of Platform Types,
AUTOSAR_SWS_PlatformTypes.pdf

[13] Specification of Compiler Abstraction,
AUTOSAR_SWS_CompilerAbstraction.pdf

[14] AUTOSAR Services,
AUTOSAR_Services.pdf

[15] Feature Specification of the BSW Architecture and the RTE,
AUTOSAR_TR_BSWAndRTEFeatures.pdf

[16] Requirements on Mode Management,
AUTOSAR_SRS_ModeManagement.pdf

[17] Specification of Multi-Core OS Architecture
AUTOSAR_SWS_MultiCoreOS,

[18] Specification of Debugging in AUTOSAR,
AUTOSAR_SWS_Debugging.pdf

[19] Specification of Diagnostic Log and Trace ,
AUTOSAR_SWS_DiagnosticLogAndTrace.pdf