Document Title	Requirements on RTE Software
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Identification No	083
Document Classification	Auxiliary

Document Version	1.2.0
Document Status	Final
Part of Release	3.1
Revision	0002

Document Change History

Date	Version	Changed by	Change Description
15.01.2009	1.2.0	AUTOSAR Administration	• Changed RTE00005 • Removed RTE00044
23.06.2008	1.1.3	AUTOSAR Administration	Legal disclaimer revised
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

Page left intentionally blank

Disclaimer

Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents 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 Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, 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 this 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 Requirements Specification

3.1 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 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.

The AUTOSAR VFB Specification defines two communication models within the RTE core services; sender-receiver (signal passing) and client-server (function invocation). Each communication model can be applied to one of three distribution patterns; intra-task, inter-task and inter-ECU. Intra-task communication occurs between runnable entities that are mapped to the same OS task whereas inter-task communication occurs between runnable entities mapped to different tasks and can therefore involve a context switch. In contrast, inter-ECU communication occurs between runnable entities in components mapped to different ECUs and so is inherently concurrent and potentially unreliable.

3.2 Functional Requirements

3.2.1 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.*

3.2.1.1 [RTE00020] Access to OS

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	Access to OS
Type:	new
Importance:	high
Description:	The RTE shall only use the OS in order to provide its own functionality to the AUTOSAR application components.

	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.
Rationale:	The application software components are intended to be OS independent and therefore should not access any particular OS directly.
Use Case:	The OS offers a standardized interface. This interface is accessed by application software components only via the RTE API and hence access is controlled by the RTE.
Dependencies:	RTE00025
Conflicts:	--
Supporting Material:	See VFB Specification chapter 4.2.5.2.2 The AUTOSAR ECU architecture defines a standardised interface for the OS and an AUTOSAR interface for application software components and therefore there can be no direct interaction.

3.2.1.2 [RTE00099] Decoupling of interrupts

Initiator:	WP4.2.1.1
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:	VFB Specification, Section 4.2.5.2.2 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"

3.2.1.3 [RTE00037] The RTE shall be able to invoke functions across protection boundaries

Initiator:	WP4.2.1.1
Date:	04.10.2004
Short Description:	The RTE shall be able to invoke functions across protection boundaries
Type:	new
Importance:	high
Description:	The RTE shall be able to invoke functions across protection boundaries – for example, a function in a different OS application – and therefore shall be a trusted OS application.

Rationale:	Only trusted OS applications will be able to call the OS API – the components are not trusted and therefore the RTE shall 'switch' protection boundaries before invoking the function. The RTE is the only element in the system that can make the switch for application software components.
Use Case:	When operating in the context of a component the RTE shall switch "protection mode" before invoking functions outside of the current protection boundary.
Dependencies:	--
Conflicts:	--
Supporting Material:	AUTOSAR OS Specification v1.2, Sections 7.7.1 (Protection errors) and 10.5.1 (Protection hook).

3.2.1.4 [RTE00036] Assignment to OS Applications

Initiator:	WP4.2.1.1
Date:	04.10.2004
Short Description:	Assignment to OS Applications
Type:	new
Importance:	high
Description:	When memory protection is in use, the RTE generator shall reject configurations where the runnables of an instance of an AUTOSAR software-component 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 – rejection invalid configurations
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.

3.2.1.5 [RTE00049] Construction of task bodies

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	Construction of task bodies
Type:	new
Importance:	high
Description:	The RTE generator shall construct task bodies to execute runnable entities in a form suitable for the AUTOSAR OS – this will typically be as a function exported with C linkage. The component description declares the runnable entities present in a component.
Rationale:	The mapping of runnable 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 in a sequence mapped to the same task.
Dependencies:	--

Conflicts:	--
Supporting Material:	--

3.2.2 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.*

3.2.2.1 [RTE00068] Signal initial values

Initiator:	WP4.2.1.1
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 regardless of whether they are transported by COM or by the RTE.
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 component that no data has been received since COM or the RTE started. The INVALIDATE attribute shall be initialized too.
Conflicts:	--
Supporting Material:	VFB Specification

3.2.2.2 [RTE00069] Communication timeouts

Initiator:	WP4.2.1.1
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:	VFB Specification - 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).

3.2.2.3 [RTE00073] Data items are atomic

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	Data items are atomic
Type:	new
Importance:	high
Description:	The RTE API shall assure that the transmission and reception of single data elements (and ALL arguments of a single 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:	VFB_C60 SwCT 1.0.0.15 ch-6.1.1 p-66

3.2.2.4 [RTE00082] Standardized communication protocol

Initiator:	WP4.2.1.1
Date:	05.10.2004
Short Description:	Standardized communication protocol
Type:	new
Importance:	high
Description:	The RTE shall implement the defined 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:	--
Conflicts:	--
Supporting Material:	RTE00091 – common protocol for COM transmissions.

3.2.2.5 [RTE00091] Inter-ECU Marshalling

Initiator:	WP4.2.1.1
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:	SwCT 1.0.0.15 ch-6.3.1.1 p-70 Titus XDR – external Data Representation. “Marshalling” is defined as the conversion of a record to transportable representation (e.g. splitting a complex data item into primitive data types) suitable for passing to COM for transmission. The conversion process will be defined in the SWS.

3.2.3 Interaction with Application Components

Includes application software components, sensor components and actuator components.

3.2.3.1 [RTE00011] Support for multiple application software component instances

Initiator:	WP4.2.1.1
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:	--
Dependencies:	Name space – rules for “instantiating” an application / sensor / actuator have to be defined.
Conflicts:	--
Supporting Material:	SwCT defines the “supportsMultipleInstantiation” attribute for a component – this requires that all runnable entities in the component are re-entrant.

	SwCT 1.0.0.18 ch-8.2 p-149

3.2.3.2 [RTE00012] Multiply instantiated AUTOSAR software components delivered as binary code shall share code

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	Multiply 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:	--
Conflicts:	--
Supporting Material:	SwCT, Section 8.8.2 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.

3.2.3.3 [RTE00013] Per-instance memory

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	The RTE shall support per-instance memory
Type:	new
Importance:	high
Description:	"per-instance memory" is accessible by all runnable entities of an application software component instance but is not shared by all instances of the component.
Rationale:	Require variables with lifetime greater than that of a runnable entity but remain protected from different instances of the same component. Also required for multiple instance support.
Use Case:	--
Dependencies:	RTE00075 – API for accessing per-instance memory
Conflicts:	--
Supporting Material:	SwCT v2.0.0-RC4 Section 5.9

3.2.3.4 [RTE00077] Instantiation of per-instance memory

Initiator:	WP4.2.1.1
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:	SwCT 5.9 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.

3.2.3.5 [RTE00017] Rejection of inconsistent component implementations

Initiator:	WP4.2.1.1
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:	SwCT v1.0.0.15 Section 4.2

3.2.3.6 [RTE00134] Runnable entity categories supported by the RTE

Initiator:	WP4.2.1.1
Date:	15.12.2004

Short Description:	Runnable entity categories supported by the RTE
Type:	new
Importance:	high
Description:	The RTE shall support the runnable entity categories 1a, 1b and 2. Runnable 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 (data_read_access). A category 1b runnable entity cannot block. Implicit read/write is also allowed. 2) The runnable entity may use explicit reading/writing including blocking behaviour. It shall be possible to map category 1a and 1b runnable entities to either AUTOSAR OS Basic or Extended tasks. Category 2 runnable entities shall be mapped to AUTOSAR OS Extended tasks.
Rationale:	Support VFB and SWCT specified concepts.
Use Case:	It is easier to reason about time behaviour 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:	The material in this section will be expanded in the SWS. The following tables are a brief summary of the planned SWS contents. The different category of runnable entities support the following receive modes from the VFB Specification (v1.04, p. 43): Receive Mode 1a 1b 2 DataReadAccess – implicit read. Y Y N data_read_access - explicit read (non-blocking) using the RTE API. N Y Y wake_up_of_wait_point – explicit read (blocking) using the RTE API. N N Y The VFB Specification defines a third “receive mode”; activation_of_runnable_entity, that is useable by all categories of runnable entity. This mode is handled similarly to DataReadAccess (i.e. implicit read) with the addition that a specified runnable is activated when the data is received. The different category of runnable entities support the following write mechanisms: Write Mode 1a 1b 2 Implicit (DataWriteAccess) Y Y N Explicit (never blocking) N Y Y The SUCCESS feedback methods specified in the VFB Specification (v1.04, p. 42) are supported as follows: Feedback method 1a 1b 2 data_read_access N Y Y wake_up_of_wait_point N N Y activation_of_runnable_entity Y Y Y For category 1a and 1b runnable entities, data could be passed to/from runnable entities via the parameters of the runnable. For category 2 data is only passed via the RTE API parameters.

	SwCT and VFB. This requirement is anticipating feedback done to the relevant WPs that should be discussed in the future – in particular the enumeration of runnable entity types is subject to change.

3.2.3.7 [RTE00072] Activation of runnable entities

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	Activation of runnable entities
Type:	new
Importance:	high
Description:	The RTE shall start/resume a runnable 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 (s. VFB-Spec, 1.03, ch-4.5.4.3 p-108, Figure 28)
Use Case:	Cyclic, time based activation of runnable entities; activation of a runnable due to the arrival of data using the sender-receiver communication pattern.
Dependencies:	[RTE00160] [RTE00161]
Conflicts:	--
Supporting Material:	SwCT 1.0.0.15 ch-7.3.2 p-102, ch-7.8 p-117 VFB-Spec, 1.03, ch-4.5.4.3 p-110, p. 112 SwCT Requirement CONTENT080 – defines period for time-triggered runnable entities. SwCT Requirement CONTENT085 VFB requirement Sched35 and Sched37

3.2.3.8 [RTE00160] Debounced start of runnable entities

Initiator:	WP4.2.1.1
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 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:	--

3.2.3.9 [RTE00161] Activation offset of runnable entities

Initiator:	WP4.2.1.1
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 runnables 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 runnables.
Use Case:	--
Dependencies:	[RTE00072]
Conflicts:	--
Supporting Material:	--

3.2.3.10 [RTE00031] Multiple runnable entities

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	Multiple runnable entities
Type:	new
Importance:	high
Description:	The RTE shall support multiple runnable entities in AUTOSAR software components.
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

3.2.3.11 [RTE00032] Data consistency mechanisms

Initiator:	WP4.2.1.1
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

	runnables 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 runnables 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);
	...
	}

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.

3.2.3.12 [RTE00046] Support for “runnable runs inside” Exclusive areas

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	Support for “runnable runs inside” Exclusive areas
Type:	new
Importance:	high
Description:	The RTE shall support exclusive areas where a runnable is declared as “running inside” the exclusive area. All runnable entities in a component that specify the same “runs inside” exclusive area shall be scheduled non pre-emptively with respect to other runnable entities in the set.
Rationale:	“Runs inside” exclusive areas satisfy the requirement from the software component template that certain exclusive areas can be defined that are automatically entered whenever a runnable entity is invoked by the RTE.
Use Case:	--
Dependencies:	RTE00032 – scope of a exclusive area is the component instance.
Conflicts:	--
Supporting Material:	SwCT v1.0.1.18, Section 7.9.2

3.2.3.13 [RTE00142] InterRunnableVariables

Initiator:	WP4.2.1.1
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 useful when several runnable entities of the component access same data item. InterRunnableVariables are used to store data item copies to avoid concurrent runnable 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 in 10ms task. Several data item read accesses by runnable in 50ms task. 50ms task can be preempted by 10ms task. Data inconsistency can be prevented if runnable in 50ms task gets a copy of the original data in an InterRunnableVariable to work on during activation.
Dependencies:	RTE00032
Conflicts:	--
Supporting Material:	SwCT v1.02 Section 7.9.3

3.2.3.14 [RTE00033] Serialization of server runnables

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	Serialization of server runnables
Type:	new
Importance:	high
Description:	The RTE shall support serialized and non-serialized server runnables. The RTE shall complete the processing of one serialized service request before it accepts and dispatches the next request for that server. The serialisation is applied on the service level, so one server can handle multiple service calls concurrently (this implies that the service's runnables are mapped to different tasks and there is no shared data between them). 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 a exclusive area when simultaneous intra-ECU and inter-ECU execution is possible. If serialization is supported by a server and how big the actual queue is shall be configurable.
Rationale:	Requirement from VFB spec (4.1.4.2 Client-Server Communication)
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 serialisation mechanism works fine.

Dependencies:	RTE00032 (Per-instance scope of exclusive areas). RTE00110 (Support for buffering).
Conflicts:	At the moment there is no support in SWCT for concurrent multiple service execution. But in VFB it is specified through the buffering attribute.
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 runnables. 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. SwCT Version 1.02 chapter 7.2.2.2.

3.2.3.15 [RTE00133] Concurrent invocation of runnable entities

Initiator:	WP4.2.1.1
Date:	10.12.2004
Short Description:	Support of runnable entity attribute “canBeInvokedConcurrently”
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 (SwCT 2.0.0-RC4 ch-5.2.2)
Use Case:	Direct client-server calls implementation with runnable entity implemented as a server. E.g. needed for Basic-SW services.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

3.2.3.16 [RTE00143] Mode Switches

Initiator:	WP4.2.1.1
Date:	28.09.2005
Short Description:	Mode Switches shall be supported
Type:	new
Importance:	high
Description:	The RTE shall implement the functionality of ModeSwitchEvents and ModeDisabelingDependencies.
Rationale:	ModeDisabelingDependencies are the only means by which AUTOSAR allows to define sets of runnables that run only in certain modes. ModeSwitchEvents allow to trigger runnables 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) Modes could be used as abstraction of schedule tables
Dependencies:	RTE00144
Conflicts:	--
Supporting Material:	AUTOSAR. Software Component Template. Version 1.04 – Final, 04 2005 (C 4.8; pp. 61ff)

3.2.4 Interaction with Basic Software Components

3.2.4.1 [RTE00152] Support for port-defined argument values

Initiator:	WP4.2.1.1
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]

3.2.4.2 [RTE00022] Interaction with call-backs

Initiator:	WP4.2.1.1
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.

3.2.4.3 [RTE00062] Local access to basic software components

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	Local access to basic software components
Type:	new
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. Indirect access to the AUTOSAR interfaces of basic software components located on a remote ECU shall be possible via the inclusion of an application 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.
Dependencies:	RTE00018 – rejection invalid configurations
Conflicts:	--
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.

3.2.5 Support for Measurement and Calibration

3.2.5.1 [RTE00153] Support of Measurement

Initiator:	WP4.2.1.1
Date:	26.07.2006
Short Description:	Support of 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 me asynchronously 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:	SWC-T [7] chapter “Measurement & Calibration”

3.2.5.2 [RTE00154] Support of Calibration

Initiator:	WP4.2.1.1
Date:	26.07.2006
Short Description:	Support of Calibration
Type:	new
Importance:	high
Description:	RTE shall support calibration process of ECUs. The RTE is not responsible to make parameter or interpolation curve/map modifications by itself but must support calibration data emulation. The RTE is 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:	SWC-T [7] chapter “Measurement & Calibration” ASAP Standard (www.asam.net)

3.2.5.3 [RTE00156] Support different calibration data emulation methods

Initiator:	WP4.2.1.1
Date:	07.08.2006
Short Description:	Support different calibration data emulation methods
Type:	new
Importance:	high
Description:	The RTE generator shall support these data emulation methods for calibration purposes: 1. directAccess Calibration data is stored in ROM and accessed directly. This method can be used with appropriate calibration hardware. 2. Single pointered method Calibration data accesses are done via one indirection over a pointer table in RAM 3. Double pointered method Calibration data accesses are done via a base pointer in RAM and over a pointer table in ROM/FLASH 4. 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:	SWC-T [7] chapter “Measurement & Calibration”

3.2.5.4 [RTE00157] Support calibration parameters in NVRAM

Initiator:	WP4.2.1.1
Date:	26.07.2006
Short Description:	Support 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:	SWC-T [7] chapter “Measurement & Calibration”

3.2.5.5 [RTE00158] Support separation of calibration parameters

Initiator:	WP4.2.1.1
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:	SWC-T [7] chapter “Measurement & Calibration”

3.2.5.6 [RTE00159] Sharing of calibration parameters

Initiator:	WP4.2.1.1
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:	SWC-T [7] chapter “Measurement & Calibration”

3.2.6 General Requirements

3.2.6.1 [RTE00021] Per-ECU RTE customization

Initiator:	WP4.2.1.1
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.

3.2.6.2 [RTE00065] Deterministic generation

Initiator:	WP4.2.1.1
Date:	04.10.2004
Short Description:	The RTE generator shall be deterministic
Type:	new
Importance:	high
Description:	The RTE generator shall be able to reproduce an RTE with identical behavior from the same input files.
Rationale:	--
Use Case:	There shall be no difference (other than information in comments) between RTEs generated by the same generator for the same input files – if this were not true then working with the RTE becomes difficult!
Dependencies:	--

Conflicts:	--
Supporting Material:	--

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

Initiator:	WP4.2.1.1
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

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

Initiator:	WP4.2.1.1
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

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

Initiator:	WP4.2.1.1
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:	<why is this necessary>
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

3.2.6.6 [RTE00079] Single asynchronous client-server interaction

Initiator:	WP4.2.1.1
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 a single outstanding request can be supported.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	SWC meta-model VFB Specification v1.03, Section 4.1.4.2

3.2.6.7 [RTE00080] Multiple requests of servers

Initiator:	WP4.2.1.1
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.

3.2.6.8 [RTE00025] Static communication

Initiator:	WP4.2.1.1
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 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.

3.2.6.9 [RTE00144] Mode switch notification via AUTOSAR interfaces

Initiator:	WP4.2.1.1
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)

3.2.6.10 [RTE00018] Rejection of invalid configurations

Initiator:	WP4.2.1.1
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, 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.

3.2.6.11 [RTE00055] Use of global namespace

Initiator:	WP4.2.1.1
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”.

3.2.6.12 [RTE00126] C support

Initiator:	WP4.2.1.1
Date:	18.11.2004
Short Description:	C language support
Type:	new
Importance:	high
Description:	The RTE generator shall support SW-components created using ‘ANSI C’.

Rationale:	--
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	VFB Specification v1.03, p. 36 Glossary Section 3.13 (API) ANSI/ISO 9899-1989, “Programming Languages – C”

3.2.6.13 [RTE00138] C++ support

Initiator:	WP4.2.1.1
Date:	01.02.2005
Short Description:	C++ language support
Type:	new
Importance:	high
Description:	The RTE generator shall support SW-components created ISO C++.
Rationale:	--
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	VFB Specification v1.03, p. 36 Glossary Section 3.13 (API) ISO/IEC 14882-1998, “Programming Languages - C++”

3.2.6.14 [RTE00051] RTE API mapping

Initiator:	WP4.2.1.1
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.

3.2.6.15 [RTE00048] RTE Generator input

Initiator:	WP4.2.1.1
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… therefore the RTE generator shall collect information from multiple input files and check their consistency.
Use Case:	--
Dependencies:	--
Conflicts:	--
Supporting Material:	AUTOSAR design flow does not restrict input to one source and therefore RTE generator must be flexible.

3.2.6.16 [RTE00023] RTE Overheads

Initiator:	WP4.2.1.1
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!

3.2.6.17 [RTE00024] Source-code AUTOSAR software components

Initiator:	WP4.2.1.1

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:	SwCT v1.0.1.18, Section 2.1.4

3.2.6.18 [RTE00140] Binary-code AUTOSAR software components

Initiator:	WP4.2.1.1
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:	SwCT v1.0.1.18, Section 2.1.4 Support for binary-code AUTOSAR software components requires the same compiler type and compiler version.

3.2.6.19 [RTE00083] Optimization for source-code components

Initiator:	WP4.2.1.1
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.

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

Initiator:	WP4.2.1.1
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

3.2.6.21 [RTE00053] AUTOSAR data types

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	AUTOSAR data types
Type:	new
Importance:	high
Description:	The RTE API shall support AUTOSAR data types defined in the component description. AUTOSAR data types (both pre-defined and user-defined) are defined in the software component template using a notation that includes the data type's name and the permitted value range. The RTE API shall support the following pre-defined AUTOSAR primitive data types: Boolean, Float, Float_with_NaN, Double, Double_with_NaN, UInt4, UInt8, UInt16, UInt32, SInt4, SInt8, SInt16, SInt32, Char8, Char16. The RTE API shall also support *complex* data types including records and arrays. Note that a record can contain other records and an array can consist of an array of records.
Rationale:	SwCT v1.0.0.15 Section 6.3.2
Use Case:	A component that needs to transfer data through an AUTOSAR interface can call a RTE API function provided by the RTE passing this data as a parameter.
Dependencies:	--
Conflicts:	--

Supporting Material:	SwCT v1.0.0.15 Section 6.3.3 Units and DataTypes proposal, InterWP, WP 10.x

3.2.6.22 [RTE00056] Pre-defined primitive data types cannot be redefined

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	Pre-defined primitive data types cannot be redefined
Type:	new
Importance:	high
Description:	The RTE generator shall prevent the redefinition within a component description of the supported pre-defined primitive data types.
Rationale:	Prevents a component having its own view as to what a “Uint8” (etc) should be.
Use Case:	--
Dependencies:	RTE00053
Conflicts:	--
Supporting Material:	--

3.2.6.23 [RTE00098] Explicit Transmission

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	Explicit Transmission
Type:	new
Importance:	high
Description:	The RTE shall provide a mechanism for making requests for explicit transmission 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 can use an explicit RTE API call to write new values of the specified data-element (which may cause an immediate transmission 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 is of cat 2 for explicit transmission. 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 v1.04, Section 4.5.4 SwCT 1.0.0.15 ch-7.5.2 p-112

3.2.6.24 [RTE00129] Implicit Transmission

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	Implicit Transmission
Type:	New
Importance:	High

Description:	The RTE shall provide a mechanism for the implicit transmission 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 will potentially modify the DataElement in the pPort. The runnable has free access to the data-element while it is running but the runnable 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 returns. Depending on the configuration the RTE may either have nothing to do or it may need to actually initiate transmission of the data element.
Rationale:	Implementation of internal component model from VFB Specification.
Use Case:	--
Dependencies:	RTE00134, RTE00128 and RTE00129 – Previous SwCT and VFB specifications required that the runnable 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 v1.04, Section 4.5.4.1 SwCT 1.0.0.15 ch-7.5.1 p-111

3.2.6.25 [RTE00128] Implicit Reception

Initiator:	WP4.2.1.1
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 will require access to the DataElement in the rPort. The runnable expects that the contents of this data does not change during execution of the runnable entity.
Rationale:	--
Use Case:	--
Dependencies:	RTE00134, RTE00128 and RTE00129 – Previous SwCT and VFB specifications required that the runnable 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 v1.04, Section 4.5.4.1 SwCT 1.0.0.15 ch-7.5.1 p-111

3.2.6.26 [RTE00141] Explicit Reception

Initiator:	WP4.2.1.1
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 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 v1.04, Section 4.5.4

3.2.6.27 [RTE00092] Implementation of VFB model “waitpoints”

Initiator:	WP4.2.1.1
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 should, 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:	--
Dependencies:	RTE00027
Conflicts:	--
Supporting Material:	This requirement is a special case of RTE00027. SwCT 1.0.0.15 ch-7.3.2 p-102

3.2.6.28 [RTE00145] Compatibility mode

Initiator:	WP4.2.1.1
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:	--

3.2.6.29 [RTE00146] Vendor mode

Initiator:	WP4.2.1.1
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:	--

3.2.6.30 [RTE00148] Support “Specification of Memory Mapping”

Initiator:	WP4.2.1.1
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]

3.2.6.31 [RTE00149] Support “Specification of Compiler Abstraction”

Initiator:	WP4.2.1.1
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]

3.2.6.32 [RTE00150] Support “Specification of Platform Types”

Initiator:	WP4.2.1.1
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]

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

Initiator:	WP4.2.1.1
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: [BSW00300] [BSW00304] [BSW00305] [BSW00307] [BSW00308] [BSW00310] [BSW00312] [BSW00326] [BSW00327] [BSW00330] [BSW007]
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.

3.2.7 VFB Tracing

3.2.7.1 [RTE00005] Support for ‘trace’ build

Initiator:	WP4.2.1.1
Date:	04.10.2004
Short Description:	The RTE generator shall support ‘trace’ builds
Type:	Changed 29.08.2008
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:	--
Dependencies:	RTE00045, RTE00008

Conflicts:	--
Supporting Material:	VFB90, VFB_C10

3.2.7.2 [RTE00045] Standardized VFB tracing interface

Initiator:	WP4.2.1.1
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 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 hook functions and their semantics are defined in the RTE specification.

3.2.7.3 [RTE00008] VFB tracing configuration

Initiator:	WP4.2.1.1
Date:	04.10.2004
Short Description:	VFB tracing configuration
Type:	new
Importance:	medium
Description:	If the RTE provides means for tracing communication across the VFB, it shall be possible to configure 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); VFB Specification V1.03, Sect. 4.4.3, p. 94

3.2.7.4 [RTE00003] Tracing of sender-receiver communication

Initiator:	WP4.2.1.1
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)

3.2.7.5 [RTE00004] Tracing of client-server communication

Initiator:	WP4.2.1.1
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)

3.2.8 Application Component Initialization and Finalization

3.2.8.1 [RTE00052] Initialization and finalization of components

Initiator:	WP4.2.1.1
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 runnables will be executed with respect to the OS scheduler. In all other states the ECUStateManager is responsible for schedule of the ECU, but runnables 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 runnables 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 runnables in the one or the other way and shall ensure that runnables 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)

3.2.8.2 [RTE00070] Invocation order of runnables

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	Invocation order of runnable entities
Type:	new
Importance:	high
Description:	The RTE generator shall respect the invocation order of runnable entities as described through a behavior description.
Rationale:	The invocation order of runnable entities (seen as pieces of code) may be of importance, e.g., while the initialization and finalization phase. This requirement shall be applicable for runnable entities that are mapped to tasks. The invocation order of different application software components may be expressed by the invocation order of their runnables, when these runnable entities can be mapped to the same tasks.
Use Case:	Let a runnable for initialization finish before an other runnable is allowed to start.
Dependencies:	--
Conflicts:	--
Supporting Material:	SwCT 1.0.0.15 ch. 4.8.6 p. 59 and ch-7.9 p-117

3.2.9 API

3.2.9.1 [RTE00100] Compiler independent API

Initiator:	WP4.2.1.1
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.

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

Initiator:	WP4.2.1.1
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.

3.2.9.3 [RTE00060] RTE API shall pass 'in' complex data types by reference

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	RTE API shall pass 'in' complex data types by reference.
Type:	new
Importance:	high
Description:	The RTE API shall pass all input parameters that are complex 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 complex data and the receiver of the complex data are belonging to different memory domains RTE must copy the complex data to a memory area accessible by the receiver in order to avoid memory violation faults caused by the receiver.
Supporting Material:	RTE00001 [rejected] Implementation of the OS Protection Hook RTE00036 Assignment to OS Applications RTE00037 The RTE shall be able to invoke functions across protection boundaries

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

Initiator:	WP4.2.1.1
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:	--

3.2.9.5 [RTE00115] API for data consistency mechanism

Initiator:	WP4.2.1.1
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)

3.2.9.6 [RTE00075] API for accessing per-instance memory

Initiator:	WP4.2.1.1
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 SwCT (v2.0.0-RC4) section 5.9. 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.

3.2.9.7 [RTE00107] Support for INFORMATION_TYPE attribute

Initiator:	WP4.2.1.1
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:	RTE00112 [rejected] – TIME_FOR_RESYNC
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.

3.2.9.8 [RTE00108] Support for INIT_VALUE attribute

Initiator:	WP4.2.1.1
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:	--
Use Case:	--
Dependencies:	RTE00068 – Signal initial values
Conflicts:	--

Supporting Material:	VFB Specification v1.03, p. 42

3.2.9.9 [RTE00109] Support for RECEIVE_MODE attribute

Initiator:	WP4.2.1.1
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.

3.2.9.10 [RTE00110] Support for BUFFERING attribute

Initiator:	WP4.2.1.1
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 runnables
Conflicts:	--
Supporting Material:	VFB Specification v1.03, p. 43

	When “last_is_best” is specified the RTE generator 1. 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. 2. 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 1. Shall create a consuming read API for the data item. The name of the API shall include the port name and data item name. 2. 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. 3. Shall discard new data if the queue is full. When “no” or “queue” is specified for client/server see RTE00033.

3.2.9.11 [RTE00111] Support for CLIENT_MODE attribute

Initiator:	WP4.2.1.1
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 1. Shall create an API that invokes the operation synchronously. The name of the API could include the port name and operation name. 2. 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 1. Shall create an API that invokes the operation asynchronously. The name of the API could include the port name and operation name. 2. Reject configurations that specify asynchronous invocation of server where both runnable entities are mapped to the same task.

3.2.9.12 [RTE00121] Support for FILTER attribute

Initiator:	WP4.2.1.1
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

3.2.9.13 [RTE00122] Support for SUCCESS attribute

Initiator:	WP4.2.1.1
Date:	17.11.2004
Short Description:	Support for SUCCESS attribute
Type:	New
Importance:	Low
Description:	The RTE generator shall support the SUCCESS attribute. If specified, the attribute values shall indicate whether an acknowledgement of reception or transmission is required. The RTE generator shall support different SUCCESS specifications for each data item in an AUTOSAR interface. The RTE generator shall reject attempts to configure both a transmission and reception acknowledgement for the same data item.
Rationale:	--
Use Case:	--
Dependencies:	RTE00125 – Interaction of “1:n” communication with the SUCCESS attribute
Conflicts:	--
Supporting Material:	VFB Specification v1.04, p. 45 This requirement has a low priority because the whole reliability concept is not well defined up to now. Therefore this requirement for the success attribute may change.

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

Initiator:	WP4.2.1.1
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 SWCT (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.

3.2.9.15 [RTE00078] Support for INVALIDATE attribute

Initiator:	WP4.2.1.1
Date:	02.02.2006
Short Description:	Support for INVALIDATE attribute
Type:	New
Importance:	High
Description:	The RTE shall support the INVALIDATE communication attribute for signals. The RTE shall provide an API to invalidate a signal and to query if a signal has been invalidated. Also a notification on the reception of an invalid signal shall be supported.
Rationale:	The INVALIDATE communication attribute shall be visible to the software components.
Use Case:	Data invalid
Dependencies:	RTE00107 – INFORMATION_TYPE attribute
Conflicts:	--
Supporting Material:	nonBSW Feature list (v0.40) F048 Table 4.5 of the VFB specification V1.03

3.2.9.16 [RTE00125] Interaction of “1:n” communication with the SUCCESS attribute

Initiator:	WP4.2.1.1
Date:	18.11.2004
Short Description:	Interaction of “1:n” communication with the SUCCESS attribute
Type:	new
Importance:	high
Description:	The RTE generator shall reject attempts to specify transmission acknowledgement where there are multiple receivers.
Rationale:	Too high overhead both for communication and for application component (which would have to handle and aggregate results from all receivers).
Use Case:	--
Dependencies:	RTE00018 – rejection invalid configurations
Conflicts:	--
Supporting Material:	VFB Specification v1.03, p. 42

3.2.9.17 [RTE00094] Communication and Resource Errors

Initiator:	WP4.2.1.1
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:	SwCT 1.0.0.15 p.90 lines 5-12 VFB v1.03 explicitly delegates the definition of the error handling mechanism to be defined by WP4.2.1.1.

3.2.9.18 [RTE00084] Support infrastructural errors

Initiator:	WP4.2.1.1
Date:	08.10.2004
Short Description:	Support infrastructural errors
Type:	new
Importance:	high
Description:	The RTE API shall support the forwarding of infrastructural errors 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 v1.03, Section 4.6 (esp. figure 4-31) SwCT 1.0.0.15 p.90 lines 5-12
Use Case:	--
Dependencies:	RTE00094 – Communication and Resource Errors
Conflicts:	--
Supporting Material:	Error events are returned using the RTE API return code.

3.2.9.19 [RTE00123] Forwarding of application level server errors

Initiator:	WP4.2.1.1
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 math library provides a server that determines the square root of an argument. An application error should be returned if the argument is negative.
Dependencies:	RTE00124 – API for application level server errors
Conflicts:	--
Supporting Material:	See presentation

	https://svn.autosar.org/repos/04PapersAndResults/04WPs/26WP4.2.1.1/04FromOtherWPs/ErrorHandling/AR_EH_ErrorsOnVFB-Concept.ppt from error handling team

3.2.9.20 [RTE00124] API for application level server errors

Initiator:	WP4.2.1.1
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:	See presentation https://svn.autosar.org/repos/04PapersAndResults/04WPs/26WP4.2.1.1/04FromOtherWPs/ErrorHandling/AR_EH_ErrorsOnVFB-Concept.ppt from error handling team

3.2.9.21 [RTE00089] Independent access to interface elements

Initiator:	WP4.2.1.1
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.

3.2.9.22 [RTE00130] API to determine executing runnable entity

Initiator:	WP4.2.1.1
Date:	01.12.2004
Short Description:	API to determine executing runnable entity
Type:	new
Importance:	high
Description:	The RTE shall provide an API to determine the currently executing runnable entity.
Rationale:	--
Use Case:	The OS protection hook, created by the system integrator, invokes the API to determine what action to take.
Dependencies:	--
Conflicts:	--
Supporting Material:	The RTE provided API, when given an OS application ID, returns the runnable entity currently executing in that application. The function would then be used (by the system integrator?) to create a custom protection hook that determines the action to take after invoking the RTE supplied function.

3.2.9.23 [RTE00137] API for mismatched ports

Initiator:	WP4.2.1.1
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:	SWcT-C V1.0.1.18, p.106

3.2.9.24 [RTE00139] API for unconnected ports

Initiator:	WP4.2.1.1
Date:	01.02.2004
Short Description:	API for unconnected ports
Type:	new
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 status code as if a sender was connected, but did not transmit anything within the timeout period. The API to collect the result from an asynchronous client-server port for an unconnected required port shall return the status code as if a server was connected but did not transmit a reply within the timeout period. The API for an unconnected provided sender/receiver port shall discard the input parameters and return “no error”.
Rationale:	--
Use Case:	A not connected port of a software component.
Dependencies:	--
Conflicts:	--
Supporting Material:	SWcT-C V1.0.1.18, p.106

3.2.9.25 [RTE00155] API to access calibration parameters

Initiator:	WP4.2.1.1
Date:	03.08.2006
Short Description:	API to access calibration parameters
Type:	new
Importance:	high
Description:	The SW-C 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 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:	SWC-T [7] chapter “Measurement & Calibration”

3.2.10 C/C++ API

3.2.10.1 [RTE00087] Application Header File

Initiator:	WP4.2.1.1
Date:	08.10.2004
Short Description:	Application Header File
Type:	new
Importance:	high
Description:	The RTE Generator shall create exactly one application 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 application 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. The name of the header file shall be `RTE_<component>.h` where `<component>` is the software component type name. This requirement does not preclude a component including its own header files.

3.2.11 Initialization and Finalization Operation

Requirements for component finalization are considered above (RTE00052).

3.2.11.1 [RTE00116] RTE Initialization and finalization

Initiator:	WP4.2.1.1
Date:	01.10.2004
Short Description:	RTE Initialization and finalization
Type:	new
Importance:	high
Description:	The RTE generator shall provide mechanisms to initialize and finalize the RTE. The initialization “function” is used to initialize the RTE and would typically be called once by an ECU soon after coming out of reset. The finalization “function” is used to shutdown the RTE. The mechanism details (e.g. function contents) are highly dependent on the RTE implementation. In particular, an RTE which consisted of macros and libraries may need no function at all in which case the requirement is satisfied by a null macro.
Rationale:	The RTE shall be properly setup before the application components.
Use Case:	--
Dependencies:	ModeManager
Conflicts:	At which time is this function called? If it is called during the Startup-hook, no AUTOSAR OS system calls are allowed. It should be stated, which calls are allowed in RTE Start.
Supporting Material:	--

3.2.12 Fault Operation

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

3.3 Non-Functional Requirements (Qualities)

3.3.1 General Requirements

3.3.1.1 [RTE00064] AUTOSAR Methodology

Initiator:	WP4.2.1.1
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

3.3.1.2 [RTE00019] RTE is the communication infrastructure

Initiator:	WP4.2.1.1
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 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).

4 References

4.1 Deliverables of AUTOSAR

[1] Glossary,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_Glossary.pdf

[2] General Requirements on Basic Software Modules,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_General.pdf

[3] Requirements on RTE,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_RTE.pdf

[4] Specification of RTE Software,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_RTE.pdf

[5] Specification of Basic Software Module Description,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_BSW_ModuleDescription.pdf

[6] Specification of the Virtual Functional Bus,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_VirtualFunctionBus.pdf

[7] Software Component Template,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SoftwareComponentTemplate.pdf

[8] Methodology,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_Methodology.pdf

[9] Specification of ECU Configuration,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_ECU_Configuration.pdf

[10] Specification of ECU Configuration Parameters,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_ECU_ConfigationParameters.pdf

[11] Specification of Memory Mapping,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_MemoryMapping.pdf

[12] Specification of Platform Types,
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_PlatformTypes.pdf

[13] Specification of Compiler Abstraction
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_CompilerAbstraction.pdf[1]

[1]

[14] AUTOSAR Services
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_Services.pdf