OCORA

Open CCS On-board Reference Architecture

Generic Safe Computing Platform
High-level Requirements

Gamma Release

Document ID: OCORA-40-013-Gamma
Version: 1.00
Date: 04.12.2020
Status: Final

Revision history

Version	Change Description	Name (Initials)	Date of change
0.01	Draft version	TM	2020-11-12
0.02	Version with proposal for grouping / ordering of requirements and some other updates	PM	2020-11-12
0.99	Draft version for review	TM	2020-11-16
1.00	Final	TM	2020-12-04

Background of this document

The content of this document is the result of the joint work among the initiatives OCORA and Reference CCS Architecture (RCA). At the time of the publication of the OCORA Gamma release, the approval by the RCA initiative is pending.

Table of contents

Table of figures

References

The following references are used in this document:

[1] RCA initiative, see https://www.eulynx.eu/index.php/news

[2] OCORA, see https://github.com/OCORA-Public/Publication

[3] OCORA-10-001-Gamma – Release Notes

[4] OCORA-40-004-Gamma – Generic Computing Platform Whitepaper

[5] OCORA-90-002-Gamma – Glossary

1 Management Summary

The railway sector is currently undergoing the largest technology leap in its history, with many railways in Europe and across the globe aiming to introduce large degrees of automation in rail operation. Beyond the rollout of the European Train Control System (ETCS), most railways are for instance aiming at introducing Automated Train Operation (ATO), in some cases up to fully driverless train operation (grade of automation 4, GoA4), and an automated dispatching of rail operation, typically referred to as a Traffic Management System (TMS).

In this context, the railway initiatives Reference Control Command and Signalling Architecture (RCA) [1] and Open Control Command and Signalling Onboard Reference Architecture (OCORA) [3] are driving a functional architecture for the trackside and onboard functions for future rail operation.

In this context, RCA and OCORA are jointly working toward a **generic safe computing platform approach** for onboard and trackside CCS applications (and possibly other railway applications), in particular aiming to decouple applications from the underlying computing platform, considering their very distinct life cycles, and to achieve platform independence. For further details please refer to the white paper "*An Approach for a Generic Safe Computing Platform for Railway Applications*") [4] published as part of the OCORA beat release.

This document provides a first set of high-level requirements applicable to the safe computing platform and its generic abstraction (API) to the platform independent applications running on the platform.

2 Introduction

2.1 Document context and purpose

This document is published as part of the OCORA Gamma release, together with the documents listed in the release notes [3]. It is the first release of this document and it is still in a preliminary state.

Subsequent releases of this document and topic specific documentation will be developed in a modular and iterative approach, evolving within the progress of the OCORA initiative.

It shall to be noted that even though this document is published under the OCORA initiative, all requirements herein have been developed in a joint working group with the RCA initiative.

2.2 Why should I read this document?

This document addresses experts in the railway safety application platform domain and any other persons interested in platform requirements for on-board CCS solutions. The reader will be able to provide feedback to the authors and can, therefore, engage in shaping the OCORA computing platform requirements.

Prior to reading this document, it is recommended to study the Release Notes [3] and in particular the Approach to a Generic Safe Computing Platform White Paper [4]. The reader should also be aware of the Glossary [5].

2.3 Current situation

From a customer perspective, todays deployed CCS on-board systems are proprietary, monolithic vendor-specific solutions, creating undesired vendor lock-ins resulting in very high cost of ownership. High-priced changes and extensions stall advancements and impede new game-changing technologies.

Safety functions implemented by CCS on-board systems demand adherence to railway specific standards during development, operation and maintenance of the entire systems. The stringent homologation processes imposed by CENELEC (standards such as EN 50126, EN 50128, EN 50129) is exorbitantly expensive and time consuming when applied to proprietary, monolithic products.

OCORA aims to attack the problem by breaking down the CCS on-board system into different layers and components with defined, open interfaces that can be developed, tested and certified independently.

2.4 System under consideration

The system under consideration is the generic safe computing platform, with a key characteristic being the generic abstraction layer: the platform independence API.

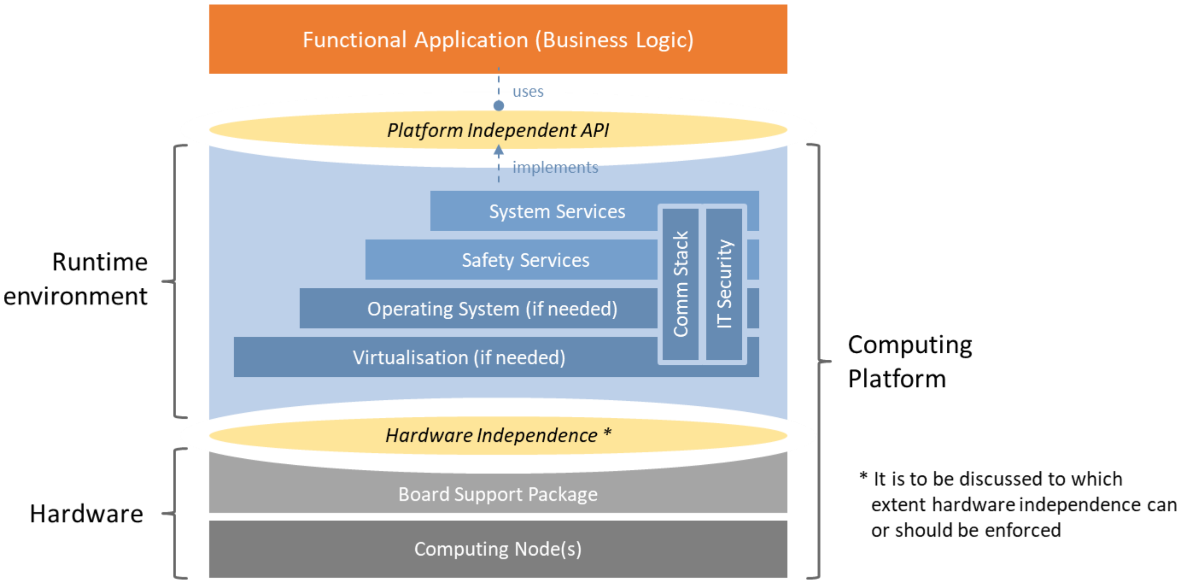

Figure 1 General computing platform principle and terminology

A software abstraction used by functional applications promotes a solution-agnostic and future-proof evolution of the Computing Platform whilst functional applications implementing the business logic of CCS on-board functions remain portable.

A hardware abstraction considers the different life-cycle profiles of software and hardware. A CCS on-board system comprises of different hardware modules: on one hand the computing nodes that run the CCS on-board functional applications and on the other hand all peripheral devices and external systems.

Applications programmed against the PI API are at minimum source code portable, or possibly even binary code portable, between different platform implementations. All safety-related functions not inherent in the application logic are implemented as part of the platform.

Examples of Computing Platform approaches are shown in Figure 2 - actual implementation details are the platform vendors' responsibility.

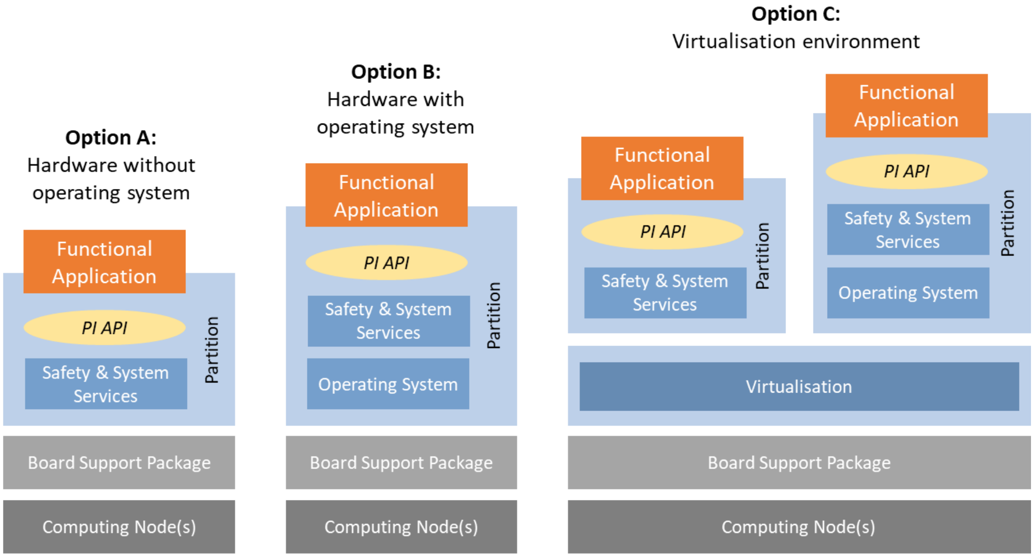

Figure 2. Possible platform options where applications are programmed against an API.

2.5 Definitions

Terms	Definition
Functional Actor	The *Functional Actor* is a fully deterministic functional module of the over-all-system. It has its own task and is the smallest unit that can be restored to a specific point in time between the computation of two incoming messages. For redundancy reasons. There may be multiple replica of a functional actor.
Functional Application	The *Functional Application* is a functional module of the over-all-system. It has its own task (business logic). It consists of one or several Functional Actors.
Voting Unit	A *Voting Unit* is a module implementing the logic to reduce the "same" messages of all Functional Actor replica to the finally valid message for the over-all-system. The *Voting Unit* is a part of the Computing Platform and hence its implementation varies between different platform providers.
Checkpoint	The *Checkpoint* is the stored internal state of a *Functional Actor*. It is stored when the *Functional Actor* has finished computing an incoming message and not yet started to compute the next incoming message. It consists of the values of all variables necessary to start from this snapshot and produce the same stream of outgoing messages again.
Replica-deterministic Time	The replica-deterministic time is guaranteed to be identical for all Functional Actor replica. It has a lower resolution than the standard system time and typically remains unchanged during a Functional Actors scheduled execution cycle.

3 Generic Safe Computing Platform Requirements

ID	Title	Description	Rational Statement / Reason for requirement	Comment	Necessity	Type
General Design						
GD-01	***Computing Platform lifetime***	*The Computing Platform (in the sense of the concept and API design) has a lifetime of at least 40 [tbd] years starting the day of acceptance of the first instance of the platform. During the entire life span, the Computing Platform complying with the same "Form Fit Function Interface Specification" (FFFIS) is available for ordering. (During these 40 [tbd] years the supplier(s) may advance the platform whilst complying to the FFFIS).*	Typically rail equipment has a rather long lifetime. Hence the Computing Platform has to be supported and maintained for decades. As it consists of hard- and software, it is essential that new hardware is being supported as it becomes available on the market - throughout the entire lifetime of the Computing Platform.		Essential	Non-Functional
GD-02	***Maintenance period***	*A productive instance of the Computing Platform has a usage period of at least 20 [tbd] years (176'000 [tbd] hours). Maintenance must be provided for the complete period.*	Typically rail equipment has a rather long lifetime. Hence the Computing Platform has to be supported and maintained for decades.		Essential	Non-Functional
GD-03	***Maximum supported SIL level***	*The maximum supported SIL level of the Safe Compute Platform (in terms of the concept and corresponding API design and its applications) is SIL4.*	Applications running on top of the runtime environment of the platform may comply to non-SIL or SIL0-SIL4. The same may apply to services of the platform.	Not every embodiment of the platform must be SIL4.	Essential	Non-Functional
GD-04	***Mixed SIL support***	*The Computing Platform allows running mixed SIL functional applications side by side.*	Hardware with a multicore processor architecture is commonly available today. It allows running functional applications side-by-side sharing the existing resources. Such hardware allows minimizing the physical space consumption in the train engine. Running mixed SIL applications on the same platform maximizes resource exploitation.	Not every embodiment of the platform must be SIL4.	Conditional	Non-Functional
GD-05	***Unified platform independent application programming interface***	*The Computing Platform implements the Unified Platform Independent API.*	CCS functional application portability is key regarding life-cycle management and certification effort. Functional Applications exclusively using the unified platform independent API shall be easily portable between (in the best case binary compatible operable on) different versions of Computing Platform implementations.		Essential	Technical

ID	Title	Description	Rational Statement / Reason for requirement	Comment	Necessity	Type
GD-06	***Encapsulated, transparent fault tolerance mechanism***	*The Computing Platform supports the implementation of applications with a safety integrity level 4 according to the related railways standards. All safety-related functions not inherent in the application logic are implemented as part of the platform. The Computing Platform transparently encapsulates the safety and fault tolerance mechanism.*	As platform vendors may use their specific approaches to handling safety and fault tolerance, it must be fully encapsulated in the platform e.g. applications must not include any platform specific code related to safety or fault tolerance. The application interacts with the platform only via the unified platform independent API. Vendors may offer different (new) approaches to safety and fault tolerance as they become available on the market - solution agnostic and future-proof.		Essential	Technical
GD-07	***Separation of platform hardware and platform software***	*The Computing Platform enforces a clear separation between the platform hardware and platform software services.*	A clear separation between hard- and software simplifies platform life-cycle management. Typically, the hardware has a much shorter lifetime that the software running on top of it.		Essential for trackside, conditional for onboard	Non-Functional
GD-08	***Multi-vendor hardware support***	*At all times during the maintenance and support period, the Runtime Environment supports hardware of at least two different hardware manufactures.*	To overcome undesired vendor lock-ins, ideally the platform hardware is based completely on COTS modules. If this is not possible, the platform vendor still has the obligation to support hardware of different manufactures.	This is not relevant in public cloud environments	Essential for trackside, conditional for onboard	Technical
GD-09	***Direct hardware sourcing from manufacturer***	*Hardware modules may be ordered directly from the defined hardware manufacturers.*	Direct buy of hardware modules shall help to improve cost efficiency and avoid the vendor lock-in. This applies to either case, e.g. if COTS hardware modules are supported or if the platform manufacturer uses a set of certified hardware modules of at least two different manufactures.	This is not relevant in public cloud environments	Essential for trackside, conditional for onboard	Non-Functional
Certification and Compliance						
CC-01	***Safety Certification responsibility***	*The Computing Platform vendor bears the full platform certification responsibility.*	To comply with the modular safety concept, the platform certification is key. The platform shall be fully certified to run up to SIL4 functional applications if the applications comply to the unified safety application conditions.		Essential	Non-Functional
CC-02	***Unified Safety Application Conditions***	*The Computing Platform defines a unified set of application safety conditions (SRACs) which all safety critical CCS functional applications must comply with in order to be certifiable according to CENELEC safety standards.*	In order to be able to port functional applications from one platform implementation to another, it is key that all platform implementations delegate the exact same set of safety application conditions to the functional applications. Otherwise applications would have to be modified to comply with different conditions on different platform implementations.		Essential	Non-Functional
CC-03	***Meet security standards***	*The platform meets the Security standards, guidelines, policies defined and established by respective railway organisations.*			Essential	Non-Functional

ID	Title	Description	Rational Statement / Reason for requirement	Comment	Necessity	Type
Execution Environments and Real-Time Support						
ER-01	***Strict spatial and temporal isolation of execution environments (partitions)***	*The Computing Platform provides execution environments (partitions) with an isolated memory address space and limited execution time, which are composed of one or several processes.*	To run different CCS applications on the same Computing Platform, it is paramount that they cannot influence each other - e.g. they must be fully independent. Independence is necessary from a spatial perspective - e.g. all functional applications must have their own assigned resources - as well as from a time perspective - e.g. functional applications must have guaranteed CPU time irrespective of what other functional applications are doing.		Essential	Technical
ER-02	***Fault isolation between partitions***	*The Computing Platform provides fault isolation between different partitions to ensure the independence of partitions.*	A system that is designed to fail safe, requires a dedicated failure detection mechanism that exist only for the purpose of fault isolation. This prevents propagation of the failure and guarantees the system can enter the defined failure state.		Essential	Technical
ER-03	***Concurrent execution of partitions***	*The computing platform can execute multiple execution environments (partitions) concurrently.*	Follows from **GSCP-003** and **GSCP-006**	How concurrent execution of partitions is realized exactly (e.g., assignment of different partitions to different CPUs or time-multiplexing on a common CPU) is left to implementation	Essential	Technical
ER-04	***Configurable partition scheduling intervals***	*The Computing Platform executes each partition at defined scheduling intervals, which shall be defined in the configuration.*	Determinism is paramount in a safety critical environment. Therefore, each partition must have a defined execution period (scheduling: time interval).		Essential	Functional
ER-05	***Configurable guaranteed execution time***	*The Computing Platform shall execute each partition for a guaranteed execution time, which shall be defined in the configuration.*	Determinism is paramount in a safety critical environment. Therefore, each partition must have a guaranteed execution time (scheduling: number of ticks).		Essential	Functional
ER-06	***Hard real-time support***	*The Computing Platform provides hard real-time support.*	Real-time computing is key for designing and/or developing predictable safe CCS functional applications. Hard real-time systems are used when it is imperative that an event be reacted to within a strict deadline.		Essential	Technical
ER-07	***Controllable partition states***	*The Computing Platform allows partitions to be active or inactive.*	In order to be able to apply partition updates on a deployed, productive system, it is key to be able to have active and inactive partitions - only inactive partitions can be updated.		Essential	Functional

ID	Title	Description	Rational Statement / Reason for requirement	Comment	Necessity	Type
ER-08	***Partition execution***	*The Computing Platform executes active partitions only.*	Inactive partitions are no longer scheduled. This is key in case of erroneous behaviour of a partition in order to enter and maintain a degraded/safe state.		Essential	Technical
ER-09	***Partitions may deactivate themselves***	*The Computing Platform provides partitions with the ability to deactivate themselves.*	A partition might see the need to put itself into a safe state due to self-monitoring.		Essential	Functional
ER-10	***Dynamic mapping of partitions to different hardware resources***	*The platform enables a dynamic mapping of partitions during operation to hardware resources, for instance in response to hardware failures or triggered by an operator (e.g., for maintenance reasons).*	To mitigate hardware failures.	Only trackside	Essential for trackside	Functional
Communication, I/O and Storage						
CIS-01	***Inter-partition communication***	*The Computing Platform provides an interface for inter-partition communication.*	Applications running in different partitions shall be able to exchange data with each other.		Essential	Functional
CIS-02	***Access to local analogue inputs***	*The Computing Platform provides the ability to partitions to access to local analogue inputs.*	In case there are local analogue inputs directly connected to the hardware of the Computing Platform, these must be accessible to partitions running on the same hardware.		Essential for onboard conditional for trackside	Functional
CIS-03	***Access to local digital inputs***	*The Computing Platform provides the ability to partitions to access local digital inputs.*	In case there are local digital inputs directly connected to the hardware of the Computing Platform, these must be accessible to partitions running on the same hardware.		Essential for onboard conditional for trackside	Functional
CIS-04	***Access UVCC bus attached data inputs***	*The Computing Platform provides the ability to partitions to access data inputs attached to the Universal Vital Control and Command (UVCC) Bus.*	By default, peripheral devices are attached to the Universal Vital Control & Command bus. The platform must allow the allocation of data values published by peripheral devices as input data to partitions.		Essential for onboard conditional for trackside	Functional
CIS-05	***Control local analogue outputs***	*The Computing Platform provides the ability to partitions to control local analogue outputs.*	In case there are local analogue outputs directly connected to the hardware of the Computing Platform, these must be controllable from partitions running on the same hardware.		Essential for onboard conditional for trackside	Functional
CIS-06	***Control local digital outputs***	*The Computing Platform provides the ability to partitions to control local digital outputs.*	In case there are local digital outputs directly connected to the hardware of the Computing Platform, these must be controllable from partitions running on the same hardware.		Essential for onboard conditional for trackside	Functional
CIS-07	***Control UVCC bus attached data outputs***	*The Computing Platform provides the ability to partitions to control data outputs attached to the*	By default, peripheral devices are attached to the Universal Vital Control & Command bus. The		Essential for onboard	Functional

ID	Title	Description	Rational Statement / Reason for requirement	Comment	Necessity	Type
		Universal Vital Control and Command (UVCC) Bus.	platform must allow the allocation of output data values of partitions as input data to peripheral devices.		conditional for trackside	
CIS-08	***Access to persistent storage***	*The Computing Platform provides the ability to partitions to access data stored in persistent memory.*	To store and retrieve configuration data.		Essential	Functional
CIS-09	***Persistent storage access control***	*The Computing Platform allows to control (configure) persistent storage access as either read-only or read-write.*	To avoid accidental data loss.		Essential	Functional
Timing and Synchronisation						
TS-01	***External time synchronisation***	*The Computing Platform allows time synchronisation with an external time server.*	Time synchronisation aims to coordinate otherwise independent clocks. Even when initially set accurately, real clocks will differ after some amount of time due to clock drift, caused by clocks counting time at slightly different rates.		Essential	Functional
TS-02	***Standard time synchronisation protocol***	*The Computing Platform supports time synchronisation using standard time synchronisation protocols.*	Standards form the fundamental building blocks for product development by establishing consistent protocols that can be universally understood and adopted. This helps fuel compatibility and interoperability and simplifies product development, and speeds time-to-market.		Essential	Functional
TS-03	***Obtain current time***	*The Computing Platform provides a mechanism to partitions for obtaining the current time.*	The computing platform provides the time of the synchronized real-time clock of the executing hardware. Important: this time is not replica-deterministic!		Essential	Functional
TS-04	***Obtain replica-deterministic time***	*The Computing Platform provides a mechanism for processes to obtain a replica-deterministic time.*	The replica-deterministic time is the time to be used within functional application replicas. The platform guarantees that all replicas obtain the same time. This time may have a limited resolution.		Essential for majority voting	Functional
Monitoring and Diagnostics						
MD-01	***Monitoring and diagnostics interface***	*The Computing Platform includes a monitoring and diagnostics interface accessible locally and remote connection.*	In order to analyse the system behaviour and performance during development, test and operation, a monitoring interface is vital.		Essential	Functional

ID	Title	Description	Rational Statement / Reason for requirement	Comment	Necessity	Type
MD-02	***Record internal execution errors***	*The Computing Platform stores all internal execution errors persistently.*	To support debugging and fault analysis.		Essential	Functional
MD-03	***Monitoring partitions***	*The Computing Platform supports monitoring of the execution of partitions, for instance by capturing KPIs related memory usage, processor load, etc.*	To support debugging and fault analysis as well as to monitor proper operation of deployed system.		Essential	Functional
MD-04	***Simulation and testing on COTS hardware***	*The Runtime Environment can be executed on a COTS hardware to facilitate simulation and testing.*	To facilitate development and test of the system.		Essential	Technical
Configuration and Update						
CU-01	***Static and/or dynamic platform configuration***	*The Computing Platform supports static (e.g. compile/link time) and dynamic configuration (runtime, during operational phases). The computing platform ensures that dynamic reconfiguration of one partition does not affect other partitions.*	The platform configuration may happen during the build phase of the system and installed during deployment. However, it shall also be possible to use dynamic configuration where this does not affect the safety and performance.		Essential	Functional
CU-02	***Local platform update***	*The Computing Platform provides mechanisms to locally update the run-time environment.*	The ability of updating the platform software is essential. In case remote (e.g., over the air) updates fail for any reason, it must be possible to perform local updates with physical access to the computing platform. Updates shall be uploaded via industry standard interfaces. In case of limited bandwidth and depending on the size, platform updates may have to be deployed locally.		Essential	Functional
CU-03	***Remote (e.g., over-the-air) platform update***	*The Computing Platform provides mechanisms to remotely update the run-time environment.*	The ability of updating the platform software is essential. To minimize maintenance cost, the normal update deployment mechanism shall be remotely (e.g., over-the-air) with no physical presence of any maintenance personnel on site (e.g., on the train).		Essential	Functional
CU-04	***Local platform configuration update***	*The Computing Platform provides mechanisms to locally update the computing platform configuration.*	The ability of updating the platform configuration is essential. In case remote (e.g., over the air) updates fail for any reason, it must be possible to perform local updates with physical access to the computing platform. Updates shall be uploaded via industry standard interfaces.		Essential	Functional

ID	Title	Description	Rational Statement / Reason for requirement	Comment	Necessity	Type
CU-05	***Remote (over-the-air) platform configuration update***	*The Computing Platform provides mechanisms to remotely update the computing platform configuration.*	The ability of updating the platform configuration is essential. To minimize maintenance cost, the normal update deployment mechanism shall be remotely (e.g., over-the-air) with no physical presence of any maintenance personnel on site (e.g., on the train).		Essential	Functional
CU-06	***Local partition update***	*The Computing Platform provides mechanisms to locally update partitions.*	The ability of updating the platform partitions is essential. In case remote (e.g., over the air) updates fail for any reason, it must be possible to perform local updates with physical access to the computing platform. Updates shall be uploaded via industry standard interfaces. In case of limited bandwidth and depending on the size, partition updates may have to be deployed locally.		Essential	Functional
CU-07	***Remote (over-the-air) operational data update***	*The Computing Platform provides mechanisms to remotely update operational data during system operation.*	To be able to leverage new game changing technology it might be essential to periodically update operational data while the system is in operation. Such updates might for example include GIS map data on a vehicle.		Essential	Functional

4 Platform Independent API Requirements

ID	Title	Description	Rational Statement / Reason for requirement	Comment	Necessity	Type
Functional Actor Presence and Integrity						
FAI-01	***State and presence information of functional actors***	*The interface provides a mechanism to obtain presence and state information of functional actors.*	Functional Actors must know if other functional actors that they depend on have stopped working or moved into a different functional state (e.g. degraded mode).		Essential	Functional
FAI-02	***State integrity check of functional actors***	*The computing platform detects and reacts if a replica has a wrong state (cyclic verification/comparison of Checkpoints between all FA replica).*	Long running replicas that use in memory data for their processing, need cyclic verification if their internal data has not changed and is still correct.		Essential for majority voting	Functional
Messaging						
MSG-01	***Transparent high-level messaging mechanism***	*The interface offers a transparent communication mechanism to exchange messages between functional actors.*	The communication between functional actors shall be message based and transparent in such a way, that the functional actor does not need to know where its counterpart is located/deployed: it could be locally on the same computing node or remote on another machine.		Essential	Functional
MSG-02	***Maximum message delivery latency***	*The messaging mechanism guarantees a maximum message delivery latency of 10 ms among FAs located on the same physical platform.*	Safety critical applications need to be able to rely on a maximum message delivery time. The actual time a message delivery takes may vary, but the system must be able to react in case messages are not delivered within a known maximum delivery time.		Essential	Functional
MSG-03	***FIFO atomic message broadcast to functional actor replicas***	*The messaging mechanism ensures that the same sequence of messages is delivered to all Functional Actor replicas.*	FIFO: When multiple messages from the same source are received at a functional actor, these messages are delivered in the same order as these messages were sent. Atomic ("Total Order"): All replicas of a functional actor receive the same sequence of messages, also if the messages are from different sources		Essential for majority voting	Functional
MSG-04	***Messages distribution to Functional Actors according to SIL***	*A message provided by the platform to a functional actor is correct according to the defined SIL level*	One possible example of ensuring the correctness is voting. A voting logic combines the messages from the replicas into a single message, that is protected with information redundancy (e.g. parity, CRC, MAC) and forwards it to the FA.		Essential	Functional

ID	Title	Description	Rational Statement / Reason for requirement	Comment	Necessity	Type
MSG-05	***Message check over multiple FA replica***	*The messaging mechanism detects, and the computing platform reacts if one replica provides once or several times a different message to the voting logic compared to equivalent messages of the other replicas.*	Functional actors reporting a proper internal state but producing messages different to their replica, need to be managed e.g. reported, stopped, restarted, etc.		Essential for majority voting	Functional
MSG-06	***Handling of lost messages***	*The messaging mechanism detects and reacts if messages are being lost.*	In order to take appropriate action, it is important that the system can detect that it has lost messages.		Essential	Functional
Logging and Tracing						
LT-01	***Logging and tracing support***	*The interface provides functions for logging and tracing.*	Logging and tracing are critical when analysing system behaviour and faults. Having a unified logging and tracing concept dramatically simplifies the analysis.		Essential	Functional
LT-02	***Log and trace levels***	*Different log/trace categories and levels are supported. It is possible to configure the log/trace level on a functional application level and on fine granularity.*	Depending on the required information it is important to be able to enable logging only for certain components (applications) and not the entire system.		Essential	Functional
LT-03	***Disable log and trace***	*It is possible to completely disable logging/tracing. If disabled, there is no impact on platform performance at all.*	As all logging has some effect on the temporal behaviour of an application, it is important that logging can be completely disabled in such a way that it has zero impact on the application performance and safety certification.		Essential	Functional
LT-04	***Log and trace performance***	*When enabled the logging and tracing have minimal impact on platform performance.*	As all logging has some effect on the temporal behaviour of an application, it is important that logging is implemented in a way that it minimizes the temporal impact of the observed application and platform.		Essential	Functional