OCORA

Open CCS On-board Reference Architecture

Generic Safe Computing Platform

High-Level Requirements

Document ID: OCORA-TWS03-020

Version: 2.01

Release: Delta

Date: 30.06.2021

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 Initiative] and Open Control Command and Signalling Onboard Reference Architecture [OCORA Initiative] 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”) [OCORA-TSW03-010] published as part of the OCORA beta 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.

Revision History

Version	Change Description	Initials	Date of change
1.0	Official version for OCORA Gamma Release	TM	07.12.2020
2.01	Official version for OCORA Delta Release	TM	30.06.2021
		-	

Table of Contents

References

Reader's note: please be aware that the document id's in square brackets, e.g. [OCORA-BWS01-010], as per the list of referenced documents below, is used throughout this document to indicate the references to external documents. Wherever a reference to a TSI-CCS SUBSET is used, the SUBSET is referenced directly (e.g. SUBSET-026). OCORA always reference to the latest available official version of the SUBSET, unless indicated differently.

[OCORA-BWS01-010] – Release Notes
[OCORA-BWS01-020] – Glossary
[OCORA-BWS01-030] – Question and Answers
[OCORA-BWS01-040] – Feedback Form
[OCORA-BWS03-010] - Introduction to OCORA
[OCORA-BWS04-010] - Problem Statements
[OCORA-TWS03-010] – Computing Platform – Whitepaper

[RCA Initiative], see https://www.eulynx.eu/index.php/news
[OCORA Initiative], see https://github.com/OCORA-Public/Publication

1 Introduction

1.1 Purpose of the document

The purpose of this document is to provide the collection of all Safe Computing Platform Requirements in a structured manner.

This document is addressed to experts in the CCS domain and to any other person, interested in the OCORA concepts for on-board CCS. The reader is invited to provide feedback to the OCORA collaboration and can, therefore, engage in shaping OCORA. Feedback to this document and to any other OCORA documentation can be given by using the feedback form [OCORA-BWS01-040] .

If you are a railway undertaking, you may find useful information to compile tenders for OCORA compliant CCS building blocks, for tendering complete CCS system, or also for CCS replacements for functional upgrades or for life-cycle reasons.

If you are an organization interested in developing CCS building blocks according to the OCORA standard, information provided in this document can be used as input for your development.

1.2 Applicability of the document

The document is currently considered informative but may become a standard at a later stage for OCORA compliant on-board CCS solutions. Subsequent releases of this document will be developed based on a modular and iterative approach, evolving within the progress of the OCORA collaboration.

1.3 Context of the document

This document is published as part of the OCORA Delta release, together with the documents listed in the release notes [OCORA-BWS01-010] . Before reading this document, it is recommended to read the Release Notes [OCORA-BWS01-010]. If you are interested in the context and the motivation that drives OCORA we recommend to read the Introduction to OCORA [OCORA-BWS03-010] , and the Problem Statements [OCORA-BWS04-010] . The reader should also be aware of the Glossary [OCORA-BWS01-020] and the Question and Answers [OCORA-BWS01-030].

1.4 Requirements Engineering Process

This OCORA requirement document is developed, using the Requirements Management Guideline [OCORA-TWS05-010]. The requirements are engineered in a top-down manner:

- As a starting point all **"Stakeholder Requirements"** towards the OCORA initiative **(A-Level requirements)** are captured and formalised.

- In a second step, the **"Program- and Design Requirements" (B-Level requirements)** are developed. These requirements define tools, processes, methodologies and design rules to be used within the program and to be considered during the system analysis and the system design/architecture work.

- As a next step, the A- and B-Level requirements are further developed in the MBSE analysis to become **"System Requirements" (C-Level requirements).**

- As part of the MBSE architecture work, building blocks are identified taking into account the MBSE analysis (C-Level requirements). All applicable requirements (A-Level, B-Level, and C-Level) are apportioned to the identified building blocks, resulting in **"Building Block Requirements" (D-Level requirements)**, forming the OCORA tender templates, together with the applicable program & design requirements.

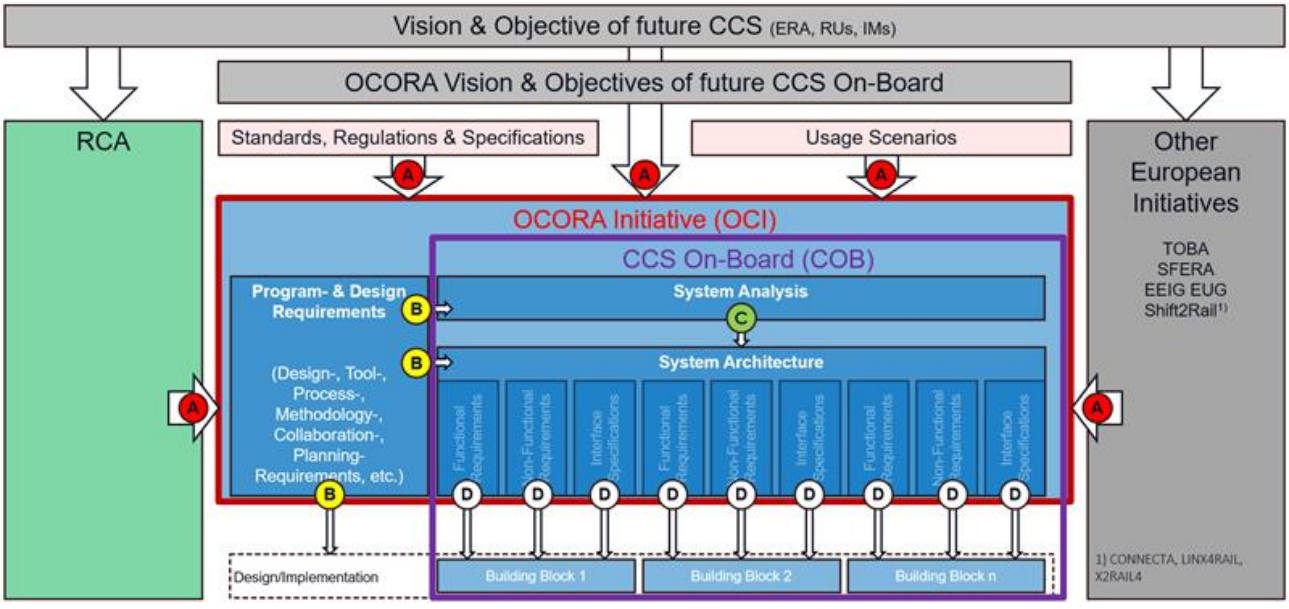

Figure 1 OCORA Requirements Engineering Process

Please note, that the A-Level requirements are applicable to the OCORA Initiative (OCI) while the B- and C-Level requirements are targeted towards the CCS On-Board System (COB) and its architecture. D-Level requirements are applicable to the respective building blocks.

1.5 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.

1.6 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.

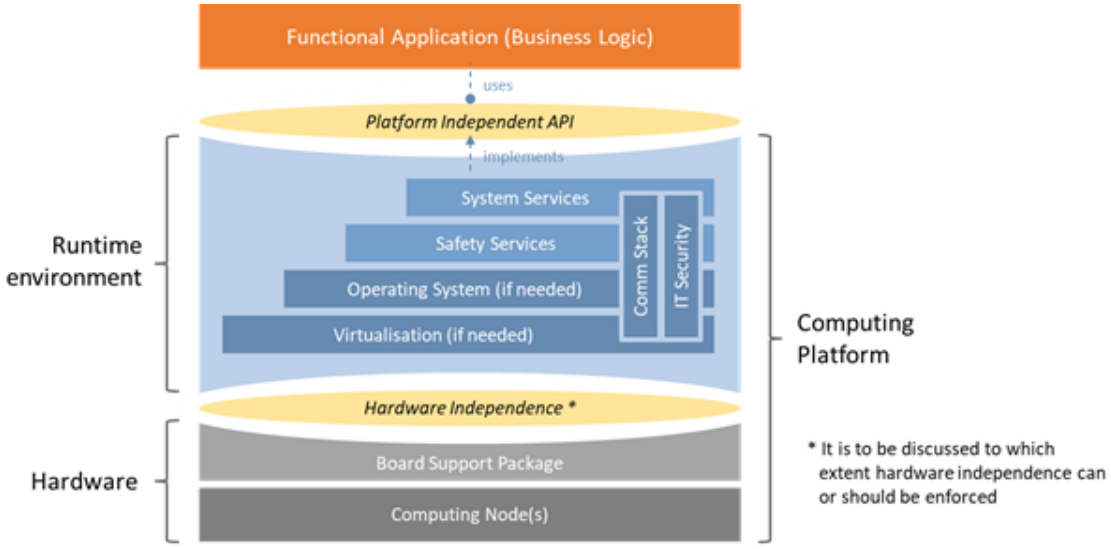

Figure 2 General computing platform principle and terminology

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.

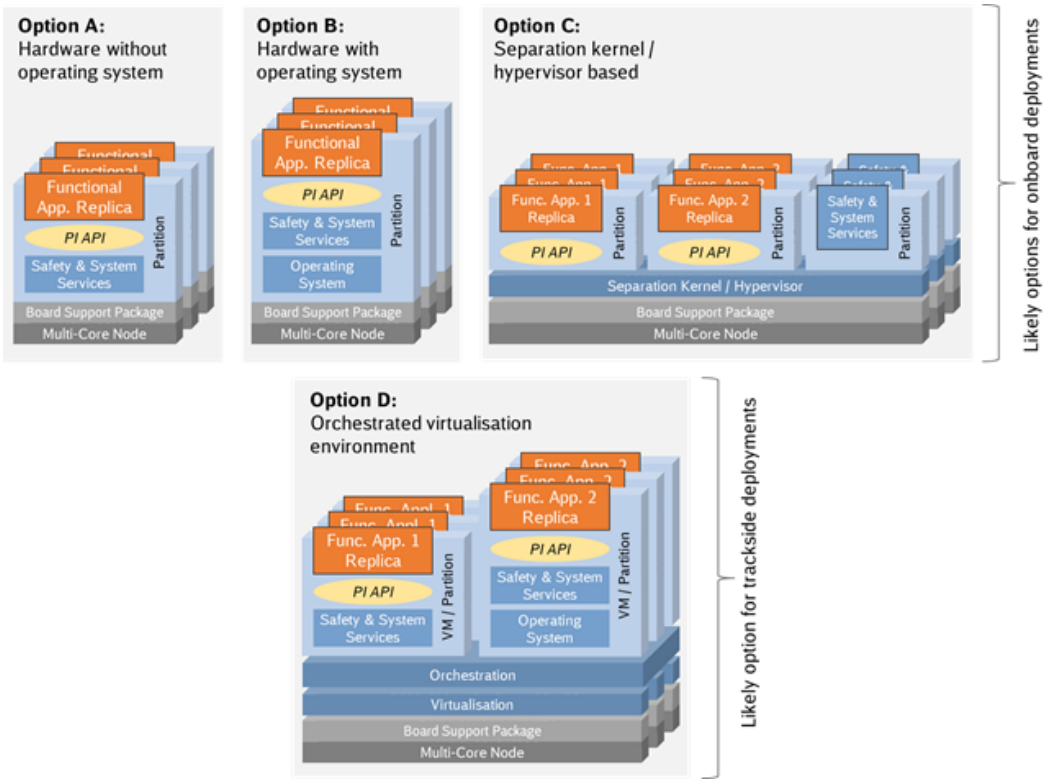

Figure 3 Possible platform options where applications are programmed against an API.

1.7 Definitions

Terms	Definitions
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.

2 Platform Requirements

2.1 General Design

OCORA-145, [GD-01] - Computing Platform Lifetime

The Computing Platform (in the sense of the concept and API design) has a lifetime of at least 40 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 years the supplier(s) may advance the platform whilst complying to the FFFIS).

Status	✓ Approved
Req. Class	Requirement
Rational	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.
Remark	*Delta Release:* removed "[tbd]"

OCORA-146, [GD-02] - Maintenance period

A productive instance of the Computing Platform has a usage period of at least 20 years (176'000 hours). Maintenance must be provided for the complete period

Status	✓ Approved
Req. Class	Requirement
Rational	Typically rail equipment has a rather long lifetime. Hence the Computing Platform has to be supported and maintained for decades.
Remark	*Delta Release:* removed "[tbd]"

OCORA-147, [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.

Status	✓ Approved
Req. Class	Requirement
Rational	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.
Remark	*Note:* Not every embodiment of the platform must be SIL4.

OCORA-148, [GD-04] - Mixed SIL support

The Computing Platform allows running mixed SIL functional applications side by side.

Status	✔ Approved
Req. Class	Optional Requirement
Rational	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.
Remark	*Note:* Not every embodiment of the platform must be SIL4.

OCORA-475, [new] - Dedicated Platform Instances for Functional Applications of different SIL

The Computing Platform allows running mixed SIL functional applications using dedicated platform instances for functional applications of different SIL.

Status	✔ Approved
Req. Class	Requirement
Rational	Hardware with a multicore processor architecture is commonly available today. It allows running functional applications side-by-side sharing the existing resources. Running applications of different SIL on dedicated platform instances maximizes flexibility regarding safety and security.
Remark	*Delta Release:* this requirement has been newly added

OCORA-149, [GD-05] - Unified platform independent application programming interface

The Computing Platform implements the Unified Platform Independent API.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-150, [GD-06] - Encapsulated, transparent fault tolerance mechanism

The Computing Platform supports the hosting 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

Status	✓ Approved
Req. Class	Requirement
Rational	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.
Remark	*Delta Release:* changed "implementation of applications" to "hosting of applications"

OCORA-151, [GD-07] - Separation of platform hardware and platform software

The Computing Platform enforces a clear separation between the platform hardware and the runtime environment.

Status	✓ Approved
Req. Class	Optional Requirement
Rational	A clear separation between hard- and software simplifies platform life-cycle management. Typically, the hardware has a much shorter lifetime than the software running on top of it.
Remark	*Requirement Class:* Mandatory Requirement for trackside *Delta Release:* changed "software services" to "runtime environment"

OCORA-152, [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.

Status	✔ Approved
Req. Class	Optional Requirement
Rational	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.
Remark	*Requirement Class:* Mandatory Requirement for trackside *Note:* This is not relevant in public cloud environments

OCORA-153, [GD-09] - Direct hardware sourcing from manufacturer

Hardware modules may be ordered directly from the defined hardware manufacturers.

Status	✔ Approved
Req. Class	Optional Requirement
Rational	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.
Remark	*Requirement Class:* Mandatory Requirement for trackside *Note:* This is not relevant in public cloud environments

2.2 Certification and Compliance

OCORA-154, [CC-01] - Safety Certification responsibility

The Computing Platform vendor is responsible to ensure (by provision of tooling, documentation, generic product certification, etc.) that a solution using the platform is certifiable according to CENELEC safety standards if functional applications comply to the unified safety application conditions.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	*Delta Release:* completely re-phrased the requirement to clarify responsibility.

OCORA-155, [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.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-156, [CC-03] - Meet security standards

The platform meets the Security standards, guidelines, policies defined and established by respective railway organisations.

Status	✔ Approved
Req. Class	Requirement
Rational	Using a standards based approach, ensures that adequate controls, processes and procedures are in place to ensure the protection of the platform confidentiality, integrity and availability.
Remark	*Delta Release:* This requirement will be expanded later based on the progress of the OCORA Cyber Security Workstream.

2.3 Execution Environments and Real-Time Support

OCORA-157, [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.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-158, [ER-02] - Fault isolation between partitions

The Computing Platform provides fault isolation between different partitions to ensure the independence of partitions.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-159, [ER-03] - Concurrent execution of partitions

The computing platform can execute multiple execution environments (partitions) concurrently.

Status	✔ Approved
Req. Class	Requirement
Rational	To run different CCS applications on the same Computing Platform.
Remark	*Note:* 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

OCORA-160, [ER-04] - Configurable partition scheduling intervals

The Computing Platform executes each partition at defined scheduling intervals, which shall be defined in the configuration.

Status	✔ Approved
Req. Class	Requirement
Rational	Determinism is paramount in a safety critical environment. Therefore, each partition must have a defined execution period (scheduling: time interval).
Remark	

OCORA-161, [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.

Status	✔ Approved
Req. Class	Requirement
Rational	Determinism is paramount in a safety critical environment. Therefore, each partition must have a guaranteed execution time (scheduling: number of ticks).
Remark	

OCORA-162, [ER-06] - Hard real-time support

The Computing Platform provides hard real-time support.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-163, [ER-07] - Controllable partition states

The Computing Platform allows partitions to be active or inactive

Status	✓ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-164, [ER-08] - Partition execution

The Computing Platform executes active partitions only.

Status	✓ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-165, [ER-09] - Partitions may deactivate themselves

The Computing Platform provides partitions with the ability to deactivate themselves.

Status	✓ Approved
Req. Class	Requirement
Rational	A partition might see the need to put itself into a safe state due to self-monitoring.
Remark	

OCORA-338, [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).

Status	✔ Approved
Req. Class	Optional Requirement
Rational	To mitigate hardware failures.
Remark	*Requirement Class:* Mandatory Requirement for trackside

2.4 Communication, I/O and Storage

OCORA-339, [CIS-01] - Inter-partition communication

The Computing Platform provides an interface for inter-partition communication.

Status	✔ Approved
Req. Class	Requirement
Rational	Applications running in different partitions shall be able to exchange data with each other.
Remark	

OCORA-340, [CIS-02] - Access to local analogue inputs

The Computing Platform provides the ability to partitions to access to local analogue inputs.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	*Requirement Class:* Optional Requirement for trackside

OCORA-341, [CIS-03] - Access to local digital inputs

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	*Requirement Class:* Optional Requirement for trackside

OCORA-342, [CIS-04] - Access CCN attached data inputs

The Computing Platform provides the ability to partitions to access data inputs attached to the CCS Communication Network (CCN).

Status	✔ Approved
Req. Class	Requirement
Rational	By default, peripheral devices are attached to the CCS Communication Network. The platform must allow the allocation of data values published by peripheral devices as input data to partitions.
Remark	*Requirement Class:* Optional Requirement for trackside

OCORA-343, [CIS-05] - Control local analogue outputs

The Computing Platform provides the ability to partitions to control local analogue outputs.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	*Requirement Class:* Optional Requirement for trackside

OCORA-361, [CIS-06] - Control local digital outputs

The Computing Platform provides the ability to partitions to control local digital outputs.

Status	✓ Approved
Req. Class	Requirement
Rational	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.
Remark	*Requirement Class:* Optional Requirement for trackside

OCORA-344, [CIS-07] - Control CCN attached data outputs

The Computing Platform provides the ability to partitions to control data outputs attached to the CCS Communication Network (CCN).

Status	✓ Approved
Req. Class	Requirement
Rational	By default, peripheral devices are attached to the CCS Communication Network. The platform must allow the allocation of output data values of partitions as input data to peripheral devices.
Remark	*Requirement Class:* Optional Requirement for trackside

OCORA-345, [CIS-08] - Access to persistent storage

The Computing Platform provides the ability to partitions to access data stored in persistent memory.

Status	✓ Approved
Req. Class	Requirement
Rational	To store and retrieve configuration data.
Remark	

OCORA-346, [CIS-09] - Persistent storage access control

The Computing Platform allows to control (configure) persistent storage access as either read-only or read-write.

Status	✔ Approved
Req. Class	Requirement
Rational	To avoid accidental data loss.
Remark	

2.5 Timing and Synchronisation

OCORA-347, [TS-01] - External time synchronisation

The Computing Platform allows time synchronisation with an external time server.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-348, [TS-02] - Standard time synchronisation protocol

The Computing Platform supports time synchronisation using standard time synchronisation protocols.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-349, [TS-03] - Obtain current time

The Computing Platform provides a mechanism to partitions for obtaining the current time.

Status	✔ Approved
Req. Class	Requirement
Rational	The computing platform provides the time of the synchronized real-time clock of the executing hardware. Important: this time is not replica-deterministic!
Remark	

2.6 Monitoring and Diagnostics

OCORA-351, [MD-01] - Monitoring and diagnostics interface

The Computing Platform includes a monitoring and diagnostics interface accessible locally and remote connection.

Status	✔ Approved
Req. Class	Requirement
Rational	In order to analyse the system behaviour and performance during development, test and operation, a monitoring interface is vital.
Remark	

OCORA-352, [MD-02] - Record internal execution errors

The Computing Platform stores all internal execution errors persistently.

Status	✔ Approved
Req. Class	Requirement
Rational	To support debugging and fault analysis.
Remark	

OCORA-353, [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.

Status	✔ Approved
Req. Class	Requirement
Rational	To support debugging and fault analysis as well as to monitor proper operation of deployed system.
Remark	

OCORA-354, [MD-04] - Simulation and testing on COTS hardware

The Runtime Environment can be executed on a COTS hardware to facilitate simulation and testing.

Status	✔ Approved
Req. Class	Requirement
Rational	To facilitate development and test of the system.
Remark	

2.7 Configuration and Update

OCORA-355, [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.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-356, [CU-02] - Local platform update

The Computing Platform provides mechanisms to locally update the run-time environment.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-357, [CU-03] - Remote (e.g., over-the-air) platform update

The Computing Platform provides safe and secure mechanisms to remotely update the run-time environment.

Status	✔ Approved
Req. Class	Requirement
Rational	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).
Remark	*Delta Release:* specified that update mechanisms need to be "safe & secure"

OCORA-358, [CU-04] - Local platform configuration update

The Computing Platform provides mechanisms to locally update the computing platform configuration.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-359, [CU-05] - Remote (over-the-air) platform configuration update

The Computing Platform provides safe and secure mechanisms to remotely update the computing platform configuration.

Status	✓ Approved
Req. Class	Requirement
Rational	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).
Remark	*Delta Release:* specified that update mechanisms need to be "safe & secure"

OCORA-488, [CU-06] - Local partition update

The Computing Platform provides mechanisms to locally update partitions.

Status	✓ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-571, [new] - Remote (over-the-air) partition update

The Computing Platform provides mechanisms to remotely update partitions.

Status	✓ Approved
Req. Class	Requirement
Rational	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.
Remark	*Delta Release:* this requirement has been newly added

OCORA-360, [CU-07] - Remote (over-the-air) operational data update

The Computing Platform provides safe and secure mechanisms to remotely update operational data during system operation.

Status	✓ Approved
Req. Class	Requirement
Rational	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.
Remark	*Delta Release:* specified that update mechanisms need to be "safe & secure"

3 Platform Independent API Requirements

3.1 Functional Actor Presence and Integrity

OCORA-364, [FAI-01] - State and presence information of functional actors

The interface provides a mechanism to obtain presence and state information of functional actors.

Status	✓ Approved
Req. Class	Requirement
Rational	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).
Remark	

OCORA-365, [FAI-02] - State integrity check of functional actors

The computing platform detects if a replica has a wrong state (e.g. cyclic verification/comparison of Checkpoints between all FA replica).

Status	✓ Approved
Req. Class	Requirement
Rational	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.
Remark	*Delta Release:* removed "reacts" from the requirement. A dedicated requirement regarding the platform reactions has been added.

3.2 Timing

OCORA-350, [TS-04] - Obtain replica-deterministic time

The Computing Platform provides a mechanism for functional actors to obtain a replica-deterministic time.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	*Requirement Class:* Mandatory Requirement for majority voting *Delta Release:* this requirement has been moved from GSCP section OCORA-229 - Timing and Synchronisation to the new API section OCORA-483 - Timing

3.3 Messaging

OCORA-366, [MSG-01] - Transparent high-level messaging mechanism

The interface offers a transparent communication mechanism to exchange messages between functional actors.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-367, [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.

Status	✔ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-368, [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.

Status	✓ Approved
Req. Class	Requirement
Rational	*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
Remark	

OCORA-369, [MSG-04] - Message 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

Status	✓ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-370, [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.

Status	✓ Approved
Req. Class	Requirement
Rational	Functional actors reporting a proper internal state but producing messages different to their replica, need to be managed e.g. reported, stopped, restarted, etc.
Remark	

OCORA-371, [MSG-06] - Handling of lost messages

The messaging mechanism detects and reacts if messages are being lost.

Status	✔ Approved
Req. Class	Requirement
Rational	In order to take appropriate action, it is important that the system can detect that it has lost messages.
Remark	

3.4 Logging and Tracing

OCORA-372, [LT-01] - Logging and tracing support

The interface provides functions for logging and tracing.

Status	✔ Approved
Req. Class	Requirement
Rational	Logging and tracing are critical when analysing system behaviour and faults. Having a unified logging and tracing concept dramatically simplifies the analysis.
Remark	

OCORA-373, [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.

Status	✔ Approved
Req. Class	Requirement
Rational	Depending on the required information it is important to be able to enable logging only for certain components (applications) and not the entire system.
Remark	

OCORA-374, [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.

Status	✓ Approved
Req. Class	Requirement
Rational	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.
Remark	

OCORA-469, [LT-04] - Log and trace performance

When enabled the logging and tracing have minimal impact on platform performance.

Status	✓ Approved
Req. Class	Requirement
Rational	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.
Remark	