OCORA

Open CCS On-board Reference Architecture

High-Level Requirements

Generic Safe Computing Platform

Document ID: OCORA-TWS03-020

Version: 4.1

Date: 05.12.2022

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 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
3.0	Official version for OCORA Release R1	TM	26.11.2021
3.1	Decoupled document evolution from OCORA release cycle	TM	01.06.2022
4.0	• Updated Requirements based on API workshop results • Computing Platform Requirements have been moved to a different Polarion Project (Modular Safe Computing Platform) and therefore have new IDs	TM	16.11.2022
4.1	Official version for OCORA Release R3	TM	05.12.2022

References

Reader's note: please be aware that the document ids in square brackets, e.g. [OCORA-BWS01-010], as per the list of referenced documents below, are 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] – Generic Safe Computing Platform for Railway Applications – Whitepaper
[OCORA-TWS03-030] - Specification of the PI API between Application and Platform

[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, functional upgrades or for life-cycle reasons.

If you are an organisation 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. This document can be considered as the final version published by OCORA, as the work on modular IT platforms for rail operation will now be taken forward in the Europe's Rail System and Innovation pillars.

1.3 Context of the document

This document is published as part of an OCORA 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.

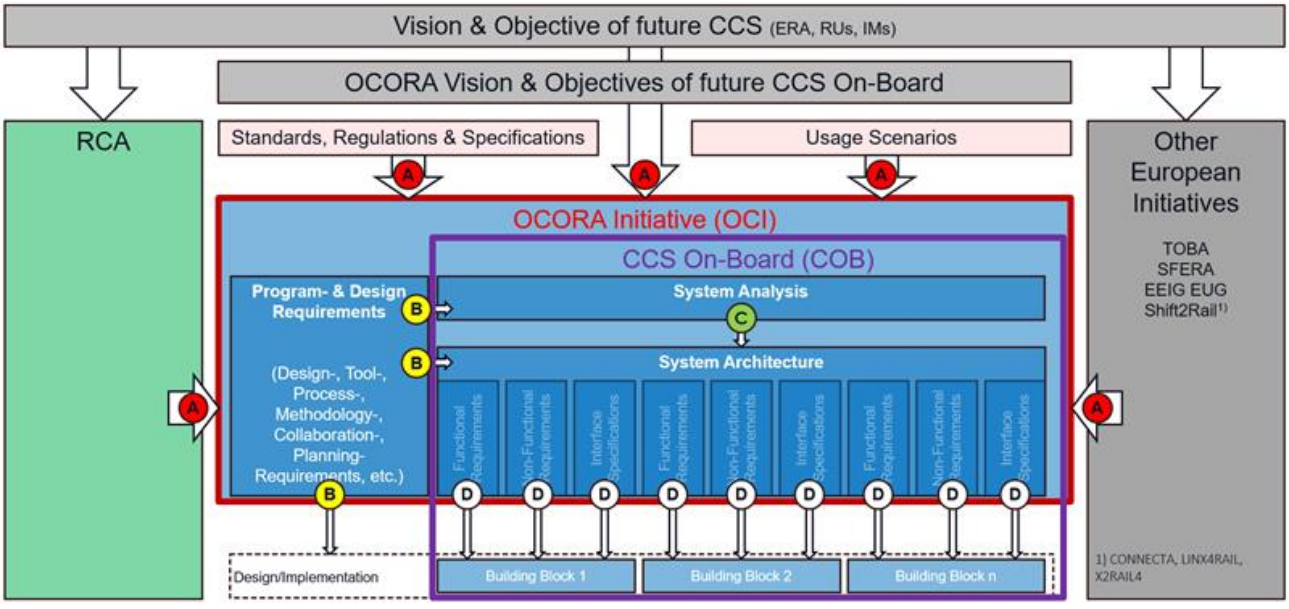

Figure 1 OCORA Requirements Engineering Process

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.

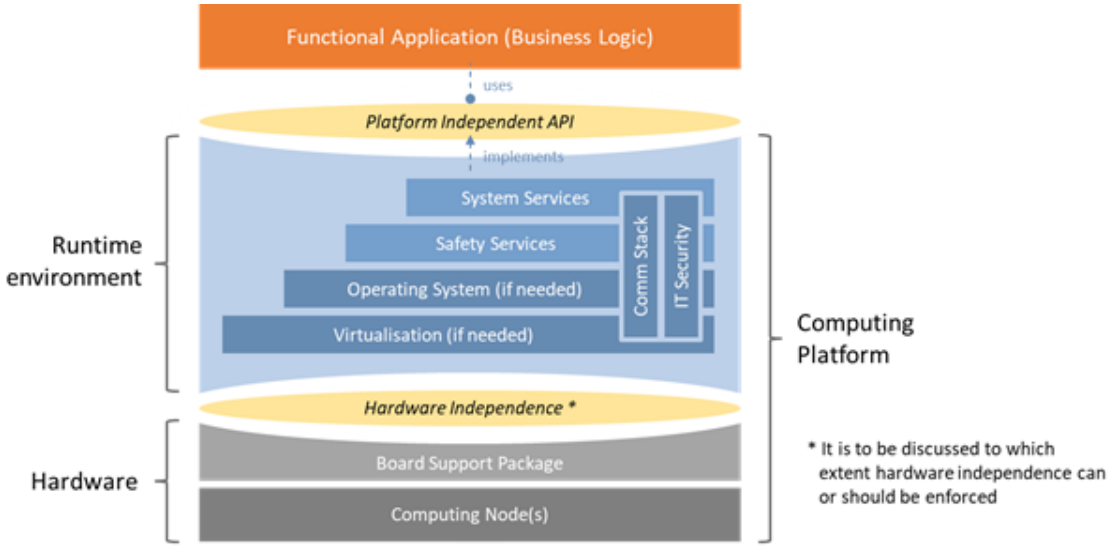

Figure 2 General Computing Platform principle and terminology

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 3 - actual implementation details are the platform vendors' responsibility.

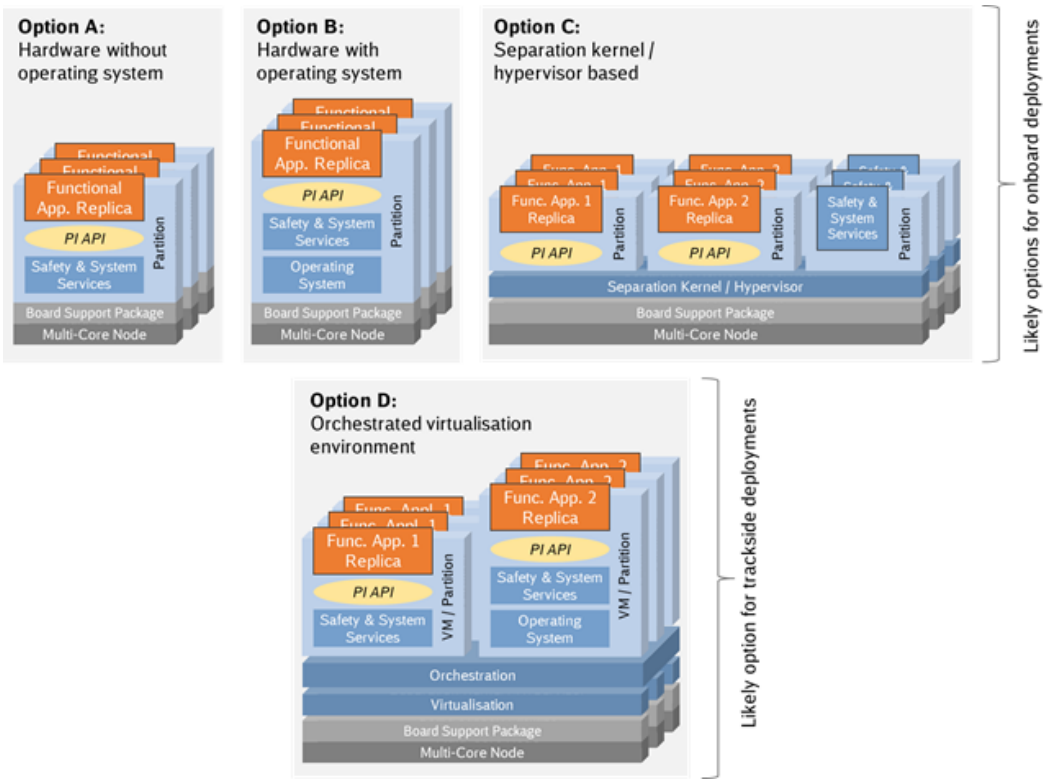

Figure 3 Possible platform deployment options where applications are programmed against an API.

Further details on the objectives and key design paradigms related to the Safe Computing Platform can be found under [OCORA-TWS03-010] . Furthermore, a first specification for the possible PI API among applications and platform has been developed among railways and suppliers and is available under [OCORA-TWS03-030] . As mentioned in Section 1.2, these works are now expected to be taken further forward in the Europe's Rail System and Innovation Pillars.

1.7 Definitions

Term	Definition
Functional Actor	A fully deterministic functional module that provides a specific application functionality. All functionality within a Functional Actor has a common functional safety requirement. A Functional Actor is the entity to which the Platform applies composite fail safety to meet the functional safety requirement. It is assumed that the split of a Functional Application into multiple Functional Actors is left to the discretion of the application vendor.
Functional Application	A comprehensive set of application functionality, assumed to be provided as one product by a single vendor. A Functional Application could for instance correspond to a subsystem as defined in the RCA architecture. Application-level communication among Functional Applications is expected to follow standardized interfaces, for instance defined by RCA or OCORA. Functions within one Functional Application may have different functional safety requirements.
Functional Actor Replica	An instance of a Functional Actor that is run on a single Computing Element, possibly jointly with Replicas of other or the same Functional Actors, if mechanisms are provided by the Platform that ensure sufficient independence (e.g., in CPU and memory usage) between Functional Actor Replicas to fulfill the CENELEC norms EN 5012x. It can be restored to a specific state to be in sync again with other Replicas after an error. The Platform applies voting to the outputs of all Replicas of a single Functional Actor. Toward a single Replica, it is not visible that other Replicas of the same Functional Actor are running.
Computing Element	A single compute element, which may comprise multiple cores, used by a (Computing) Platform. It is assumed that different Functional Actors (and different Functional Actor Replica of the same or different Functional Actors) may concurrently run on the same Computing Element if mechanisms are provided that ensure sufficient independence (e.g., in CPU and memory usage) between the Functional Actors and Functional Actor Replica to fulfill the CENELEC norms EN 5012x.
Voting Unit	A Voting Unit is a module implementing the logic to reduce the “same” messages of all Functional Actor Replicas 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.
Replica-synchronized Time	The replica-synchronized 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.
(Computing) Platform	Self-contained deployment of a Safe Computing Platform as defined in the white paper [OCORA-TWS03-010] , comprising multiple Computing Elements and RTE Instances running on these. Note that a Platform may be geographically distributed (though the distribution is transparent to application and hence not relevant to this specification).

Term	Definition
PI API	The Platform Independent Application Programming Interface refers to a standardised abstraction between the Runtime Environment and the Functional Actors. It is the key element in supporting portability and platform independence of Functional Actors. The PI API consists of two subsets: the SIL4 API and the Basic Integrity API.
PI SIL4 API	A constrained subset of the PI API that is certified to be used for implementing safe (up to SIL4), replicated Functional Actors.
PI Basic Integrity API	The subset of the PI API that is only certified to be used for implementing Functional Actors that require Basic Integrity.
Other API (e.g. Management API)	Other API refers to potentially standardized abstractions between the Runtime Environment and external systems.
Flow	A Flow is a messaging relation between Functional Actors. Flows may be joined or disjoined, registered or subscribed to by Functional Actors (possibly within constraints defined via configuration). Once a Flow is established, Functional Actors may use it to exchange messages. A Flow may have various properties relating to the usage of voting, the usage of specific Safe Communication Protocols, quality of service, etc.

2 Platform Requirements

2.1 General Design

MSCP-26 - Computing Platform Lifetime

The Computing Platform (in the sense of the concept and API design) shall have a lifetime of at least 30 years starting the day of acceptance of the first deployment of the platform. It shall be guaranteed that during the lifetime of 30 years Computing Platform realizations are available for ordering that comply with the same "Form Fit Function Interface Specification" (FFFIS). (During these 30 years the supplier(s) may advance the platform whilst complying to the FFFIS).

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	All APIs
Rationale	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	

MSCP-22 - Maintenance period

A productive instance of the Computing Platform shall have a usage period of at least 20 years (176'000 hours). Maintenance shall be provided for the complete period (with respect to replacement to faulty parts, security patches, etc.).

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	All APIs
Rationale	Typically rail equipment has a rather long lifetime. Hence the Computing Platform has to be supported and maintained for decades.
Remark	

MSCP-21 - Maximum supported SIL level

The maximum supported SIL level of the Safe Computing Platform (in terms of the concept and corresponding API design and its applications) shall be SIL4.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Applications running on top of the runtime environment of the platform may require basic integrity up to SIL4. The same may apply to services of the platform.
Remark	*Note:* Not every embodiment of the platform must be SIL4.

MSCP-20 - Mixed SIL support

The Computing Platform shall allow running mixed SIL Functional Actor Replicas side by side.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	none
Rationale	Hardware with a multicore processor architecture is commonly available today. It allows running Functional Actor Replica side-by-side sharing the existing resources. Such hardware allows minimizing the physical space consumption in the train engine. Running mixed SIL Functional Actor Replicas on the same platform maximizes resource exploitation.
Remark	*Note:* Not every embodiment of the platform must be SIL4.

MSCP-17 - Platform independent application programming interface

The Computing Platform shall implement the Platform Independent API.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI API (SIL4 & Basic Integrity)
Rationale	CCS functional application portability is key regarding life-cycle management and certification effort. Functional Applications exclusively using the platform independent API shall be easily portable between (in the best case binary compatible operable on) different versions of Computing Platform implementations.
Remark	

MSCP-23 - Encapsulated, transparent fault tolerance mechanism

All safety-related functions not inherent in the application logic shall be implemented as part of the platform. The Computing Platform shall transparently encapsulate the safety and fault tolerance mechanism.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	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	

MSCP-18 - Separation of platform hardware and platform software

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

Status	✔ Approved
Classification on-board	Optional Requirement
Classification track-side	Requirement
Involved API	none
Rationale	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	

MSCP-19 - Multi-vendor hardware support

At all times during the maintenance and support period, the Runtime Environment shall support hardware of at least two different hardware manufactures.

Status	✔ Approved
Classification on-board	Optional Requirement
Classification track-side	Requirement
Involved API	none
Rationale	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	

MSCP-24 - Direct hardware sourcing from manufacturer

It shall be possible to order Hardware modules directly from the defined hardware manufacturers.

Status	✓ Approved
Classification on-board	Optional Requirement
Classification track-side	Requirement
Involved API	none
Rationale	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	

MSCP-57 - Simulation and testing on COTS hardware

It shall be possible to execute the Runtime Environment on a COTS hardware to facilitate simulation and testing.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	All APIs
Rationale	To facilitate development and test of the system.
Remark	

2.2 Certification and Compliance

MSCP-29 - Computing Platform vendor independent (re-)homologation

The Computing Platform vendor shall be responsible to ensure (by provision of tooling, documentation, generic product certification, etc.) that a solution using the platform is certifiable according to CENELEC without the explicit involvement of the Computing Platform vendor.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	All APIs
Rationale	A full decoupling of the life-cycles of the Computing Platform and the Functional Applications requires that a deployment of Platform and Applications can be homologated without the explicit involvement of the Computing Platform vendor.
Remark	

MSCP-89 - Application vendor independent (re-)homologation

The Application vendor shall be responsible to ensure, by providing all required artefacts, that a Functional Application can be integrated on a Computing Platform and homologated without the explicit involvement of the Application vendor.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	All APIs
Rationale	A full decoupling of the life-cycles of the Computing Platform and the Functional Applications requires that a deployment of Platform and Applications can be homologated without the explicit involvement of the application(s) vendor(s).
Remark	

MSCP-28 - Unified Safety Related Application Conditions

The Computing Platform shall define a unified set of safety related application conditions (SRACs) (at least to the extent that they relate directly to the Functional Application) which all safety critical CCS Functional Applications must comply with in order to be certifiable according to CENELEC safety standards.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	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 related application conditions to the Functional Applications. Otherwise Applications would have to be modified to comply with different conditions on different platform implementations.
Remark	

MSCP-30 - Meet security standards

The Computing Platform is able to meet the relevant security system requirements of IEC 62443-3-3:2013 with Security Level 3 (SL3).

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	All APIs
Rationale	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	The requirement is based on the initial security level target (SL-T) definition and will be refined and apportioned to component requirements to the Safe Computing Platform later in the subsequent phases of the development process. For the time being, the reference to the railway notes of TS 50701:2022 has been removed until to ongoing controversies have been resolved.

MSCP-109 - Authentication and authorisation of Functional Actors

The Computing Platform shall ensure the authentication and authorisation of Functional Actors.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	The Platform has to ensure that only authenticated Functional Actors can use the platform. Further Functional Actors need to be able to trust, that the entities they are receiving messages from or transmitting messages to are the entities they claim to be.
Remark	

2.3 Functional Actor Replica execution

MSCP-36 - Independence of Functional Actor Replicas

The Computing Platform shall ensure sufficient independence (e.g., in CPU and memory usage) between Functional Actor Replicas to fulfill the CENELEC norm EN 50129:2018 or later.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	none
Rationale	This is required for CENELEC compliance EN 50129:2018 or later.
Remark	

MSCP-38 - Controllable Functional Actor states

The Computing Platform shall offer a management interface to set a Functional Actor to be active or inactive where inactive means that it is not executed and can be moved/replaced/updated.

Status	✓ Approved
Classification on-board	Optional Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	In order to be able to apply Functional Actor updates on a deployed, productive system, it is key to be able to have active and inactive Functional Actors - only inactive Functional Actors can be updated.
Remark	*Note:* it is assumed that messages sent to temporarily inactive Functional Actors are queued.

MSCP-33 - A Functional Actor may deactivate itself

The Computing Platform shall provide a Functional Actor the ability to deactivate itself.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	A Functional Actor might see the need to deactivate itself due to self-monitoring.
Remark	

MSCP-91 - Management Information on Functional Actor state changes

The Computing Platform shall provide a management interface on which information is provided when a Functional Actor changes state.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	If a Functional Actor deactivates itself this likely requires some external action.
Remark	

MSCP-37 - Automatic dynamic mapping of Functional Actor Replicas to different Computing Elements

The Computing Platform shall be able to dynamically map Functional Actor Replicas during operation to other Computing Elements in response to hardware failures.

Status	✓ Approved
Classification on-board	Optional Requirement
Classification track-side	Optional Requirement
Involved API	none
Rationale	To mitigate hardware failures.
Remark	

MSCP-94 - Operator triggered activation of Computing Element

The Computing Platform shall provide a management interface to allow a Computing Element to be activated meaning that it is from then on available to be used.

Status	✓ Approved
Classification on-board	Optional Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	To be able to add additional Computing Elements to a running Computing Platform.
Remark	

MSCP-92 - Operator triggered deactivation of Computing Element

The Computing Platform shall provide a management interface to deactivate Computing Element(s) in order to relocate all Functional Actor Replicas (active & inactive) to another Computing Element(s).

Status	✓ Approved
Classification on-board	Optional Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	This feature is required to replace or upgrade hardware during operation.
Remark	*Note: If insufficient other Computing Elements are available, the Platform shall reject the deactivation.*

MSCP-93 - Transparent deployment of Functional Actor Replicas to Computing Elements

For a Functional Actor it shall be transparent on which Computing Elements its Replicas or those of other Functional Actors are deployed to.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	none
Rationale	For any function implemented within the Functional Actor, it is not needed to know where it is deployed.
Remark	

MSCP-41 - Concurrent execution of Functional Actor Replicas

The Computing Platform shall be able to run multiple Functional Actor Replicas concurrently (at the same time).

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	none
Rationale	Hardware with a multicore processor architecture is commonly available today. It allows running Functional Actor Replicas side-by-side sharing resources.
Remark	

MSCP-39 - Configurable Functional Actor Replicas scheduling intervals

The Computing Platform shall be able to assign deterministic execution behaviour to Functional Actor Replicas in regular scheduling intervals, based on configuration.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Determinism is paramount in a safety critical environment. Therefore, each Functional Actor Replica must have a defined execution period (scheduling: time interval).
Remark	

MSCP-97 - One-shot timer based scheduling of Functional Actor Replicas

The Computing Platform shall be able to assign deterministic execution behaviour to Functional Actor Replicas triggered by an one-shot-timer event.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Beside a regular scheduling interval, additional execution of a Functional Actor Replica might be needed, therefore one-shot-timer triggered execution shall be provided by the platform.
Remark	

MSCP-96 - Event triggered scheduling of Functional Actor Replicas

The Computing Platform shall be able to schedule Functional Actor Replicas for execution triggered by an event such as the receiving of a message by the Functional Actor.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Functional Actors may need to react to explicit events prior to their next regular scheduling.
Remark	

MSCP-40 - Configurable guaranteed execution budget

The Computing Platform shall be able to execute each Functional Actor Replica for a guaranteed execution budget, which shall be defined in the configuration.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Determinism is paramount in a safety critical environment. Therefore, each Functional Actor Replica must have a guaranteed execution e.g., to be scheduled for configured number of time ticks.
Remark	Configuration for all different scheduling paradigms need to be provided, e.g., execution time guarantees might differ between interval and event triggered scheduling and as well between onboard and trackside applications needs.

MSCP-35 - Strict deadlines and deterministic scheduling (used to be Hard Real Time Support)

The Computing Platform shall be able to provide strict deadlines and maximum tolerable jitter for deterministic scheduling of Functional Actor Replicas.

Status	✔ Approved
Classification on-board	Optional Requirement
Classification track-side	Optional Requirement
Involved API	none
Rationale	Real-time computing is key for designing and/or developing predictable, safe CCS functional applications.
Remark	

MSCP-108 - Detect and handle errors according to EN 50129:2018

The Platform shall detect and handle errors according to EN 50129:2018 (with both the definition of “error” and the handling of these according to EN 50129:2018)

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	This is required to fulfil the norm EN 50129:2018.
Remark	

MSCP-118 - Monitor Functional Actor Replica processing time

The Platform shall monitor whether a Functional Actor is able to conclude processing within a defined time period.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	none
Rationale	It is important that Functional Actor Replicas can conclude on processing, e.g., incoming messages within a certain CPU resource. The platform has to monitor this to be able to potentially react.
Remark	

MSCP-119 - Inform Functional Actor if exceeding processing time

The Platform shall inform a Functional Actor if one or multiple of its Functional Actor Replicas have (once or multiple times) not been able to conclude processing in the allocated processing time, and take action if configured so.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Only the Platform knows about the exceeding and informs the Functional Actor that it might have taken appropriate actions.
Remark	

MSCP-106 - Ensure individual Functional Actor Replicas process the same messages

When the Platform invokes the multiple Replicas of the same Functional Actor, it shall ensure that individual Replicas process the same messages.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	This is required, as otherwise it could not be guaranteed that the Replicas yield the exact same output.
Remark	

2.4 Communication, I/O and Storage

MSCP-44 - Communication between Functional Actors

The Computing Platform shall provide standardised mechanisms for communication between Functional Actors.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Applications and their respective Functional Actor shall be able to exchange data with each other using a standardised message paradigm, specified in the PI API.
Remark	

MSCP-49 - Access to local I/O

The Computing Platform shall provide the ability to Functional Actors to access local inputs and outputs.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Optional Requirement
Involved API	PI Basic Integrity API
Rationale	In case there are local I/Os directly connected to the hardware of the Computing Platform, these must be made accessible to Functional Actors.
Remark	Some onboard Computing Platforms need to supply SIL4 outputs (e.g., Emergency Brake) and it is the responsibility of the Computing Platform implementation to realise such functional safe I/Os.

MSCP-51 - Access to Communication Networks

The Computing Platform shall provide the ability to Functional Actors to communicate via communication networks.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI Basic Integrity API
Rationale	Fanctional Actors deployed on different Computing Elements need to communicate with each other.
Remark	It is assumed that the Computing Platform needs to provide network access via commonly used network protocols as e.g., TCP/IP, etc.

MSCP-48 - Access to persistent storage

The Computing Platform shall provide the ability to Functional Actors to access data stored in persistent memory.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	To store and retrieve configuration data.
Remark	

MSCP-43 - Persistent storage access control

The Computing Platform shall apply access controls for persistent storage.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	• to avoid accidental data loss • to apply access rights • to apply controls, as e.g., read-only or read-write such as defined in a configuration
Remark	

2.5 Time Synchronisation

MSCP-55 - External time synchronisation

The Computing Platform shall allow time synchronisation with an external time server.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	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	

MSCP-53 - Standard time synchronisation protocol

The Computing Platform shall support time synchronisation using standard time synchronisation protocols.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	The usage of standardised protocols ensure compatibility, interoperability, simplify product development and speed up time-to-market.
Remark	

2.6 Monitoring, Diagnostics, Logging and Tracing

MSCP-60 - Monitoring and diagnostics interface

The Computing Platform shall include a monitoring and diagnostics interface accessible locally and via remote connection.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	In order to analyse the system behaviour and performance during development, test and operation, a diagnostics interface is needed between Computing Platform deployments and the MDCM (Monitoring, Diagnostics, Configuration, and Maintenance).
Remark	One possible use case for diagnostic communication via the interface could be the Update and Configuration Management, interacting with the MDCM.

MSCP-58 - Record internal execution errors

The Computing Platform shall store all internal execution errors persistently.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	To support fault analysis.
Remark	

MSCP-59 - Monitoring Functional Actor Replicas

The Computing Platform shall support monitoring of the execution of Functional Actor Replicas, for instance by capturing KPIs related memory usage, processor load, etc.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	To support fault analysis as well as to monitor proper operation of deployed system.
Remark	

MSCP-84 - Logging and tracing support

The Computing Platform shall be able to provide logging and tracing information to an external entity.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	Logging and tracing are critical when analysing system behaviour and faults. Having a unified logging and tracing concept dramatically simplifies the analysis.
Remark	

MSCP-126 - Configuration of logging and tracing support

The Computing Platform shall allow an external entity to configure the extent of logging and tracing.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	Logging and tracing are critical when analysing system behaviour and faults. Having a unified logging and tracing concept dramatically simplifies the analysis.
Remark	

MSCP-85 - Logging and tracing impact on Computing Platform performance

The Computing Platform shall ensure that logging and tracing have a negligible impact on platform performance.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	none
Rationale	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	

2.7 Configuration and Update

MSCP-62 - Static configuration

The Computing Platform shall support static configuration that doesn't get modified during operational phases.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Functional Actors running on the platform need configuration to specify their needs of resources towards the platform.
Remark	This configuration must be standardized to enable platform independence in case it belongs to the PI API.

MSCP-99 - Dynamic configuration

The Computing Platform shall support dynamic configuration of Functional Actors (runtime, during operational phases). The Computing Platform ensures that dynamic reconfiguration of one Functional Actor does not affect other Functional Actors.

Status	✔ Approved
Classification on-board	Optional Requirement
Classification track-side	Requirement
Involved API	PI Basic Integrity API
Rationale	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	

MSCP-67 - Local platform update

The Computing Platform shall provide safe and secure mechanisms to locally update the run-time environment.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	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.
Remark	

MSCP-66 - Remote (e.g., over-the-air) platform update

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

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	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 need for physical presence of any maintenance personnel on site (e.g., on the train).
Remark	Remote updates must not affect the proper operation of Computing Platforms and might need explicit planned scheduling to be applied.

MSCP-68 - Local platform configuration update

The Computing Platform shall provide safe and secure mechanisms to locally update the Computing Platform configuration.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	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	

MSCP-63 - Remote (over-the-air) platform configuration update

The Computing Platform shall provide safe and secure mechanisms to remotely update the Computing Platform configuration.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	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 need for physical presence of any maintenance personnel on site (e.g., on the train).
Remark	Remote configuration updates must not affect the proper operation of Computing Platforms and might need explicit planned scheduling to be applied.

MSCP-69 - Local Functional Actor update

The Computing Platform shall provide safe and secure mechanisms to locally update Functional Actors.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	The ability of updating the Functional Actors 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	Software updates of the Functional Actors are performed by the platform, but software updates of entities controlled by the Functional Actor are in the responsibility of the Functional Actor itself.

MSCP-64 - Remote (over-the-air) Functional Actor update

The Computing Platform shall provide safe and secure mechanisms to remotely update Functional Actors.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	Other API (e.g. Management API)
Rationale	The ability of updating Functional Actors is essential. To minimize maintenance cost, the normal update deployment mechanism shall be remotely (e.g., over-the-air) with no need for physical presence of any maintenance personnel on site (e.g., on the train).
Remark	Software updates of the Functional Actors are performed by the platform, but software updates of entities controlled by the Functional Actor are in the responsibility of the Functional Actor itself.

3 Platform Independent API Requirements

MSCP-112 - Leverage existing API specifications

The PI API design shall maximally leverage API specifications that are already available (to the extent that this is possible)

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	All APIs
Rationale	• in order to facilitate portability of existing Applications to the new API • in order to ensure that platform realizations can maximally leverage existing implementations (open source etc.)
Remark	

MSCP-113 - Restricted number of API functions

The number of API functions shall be as few as possible.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	All APIs
Rationale	• for the ease of portability of applications among platform realizations • for the ease of certification and acceptance
Remark	

MSCP-121 - Evolvability of PI API specification

The PI API shall be evolvable over time, making sure to provide backward-compatibility.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	All APIs
Rationale	It is expected that the PI API can evolve over time. Future evolved versions are expected to be decently backward-compatible to make sure that existing applications can be integrated on Runtime Environments, implementing a future versions of the API.
Remark	

MSCP-114 - Maximize common functions for onboard and trackside

The largest part of the API specification shall be the same for onboard and trackside.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	All APIs
Rationale	• for the sake of portability, reusability of Functional Actors • reduce overhead for learning and training for developers • simplification of specification and effort to conclude on it
Remark	

3.1 Functional Actor Presence

MSCP-73 - State and presence information of Functional Actors

The interface shall provide a mechanism to obtain presence and state information of Functional Actors.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	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	

3.2 Timing

MSCP-54 - Obtain replica-sychronized time

The Computing Platform shall provide a mechanism to Functional Actors Replicas for obtaining the current replica-synchronized time.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Functional Actors need consistent, replica-synchronised time information which is exactly the same for all replicas and can be used to create output that is voted on
Remark	This is important if the time stamp has an impact on any (voted) output of the Functional Actor.

MSCP-98 - Obtain un-synchronized time

The Computing Platform shall provide a mechanism to Functional Actor Replicas for obtaining the current un-synchronised time.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Functional Actor Replicas need un-synchronised time information e.g., replica for specific logging.
Remark	“Unsynchronised time” corresponds to the time at the point when a Functional Actor Replica requests this (and for which different Replicas of the same Functional Actor may obtain a different result)

MSCP-102 - Complement messages with timestamps

The Platform shall complement messages with timestamps.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Timestamps are important, so that receiving Functional Actors can check whether and how strongly received messages are outdated and possibly take appropriate action (i.e., either discard such messages or take other action).
Remark	

MSCP-105 - Inform Functional Actor Replica about exceeding defined processing time period

The Platform shall inform the Functional Actor if it has not been able to conclude processing within a defined time period (once or multiple times, as configured) and (if configured) shut down the Functional Actor.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	If a Functional Actor is not able to conclude processing within a defined time period (once or multiple times), it either has to scale down its load (e.g., by shifting tasks to other FAs, where possible), or the Platform has to shut it down.
Remark	

3.3 Messaging

MSCP-76 - Standardised messaging mechanism between Functional Actors

The Computing Platform shall provide an API to access a standardized communication mechanism, called Flow, to exchange messages between Functional Actors.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	A standardized communication mechanism is needed in order to abstract safe and secure communication and to enable a transparent encapsulated safety and fault tolerance mechanism, realized by the platform.
Remark	

MSCP-116 - Location Transparency

It shall be transparent to a Functional Actor whether it is communicating to a local entity (i.e., residing on the same local Platform deployment) or a remote entity (i.e., residing on a remote Platform deployment, and possibly involving Safe Communication Protocols and/or gateway functions in between).

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	• Enables the ability to (re-)deploy Functional Actors to computing elements • simplifies the application communication, since Functional Actors simply use flows without the need to know how addressing and routing gets established in the platform
Remark	

MSCP-115 - Replication Transparency

It shall be transparent to Functional Actors whether they themselves, and the Functional Actors they are exchanging messages with, are replicated or not.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Since a Functional Actor does not know if itself, or another Functional Actor it is communicating to, is replicated, flows provided by the platform must handle the exchange of messages.
Remark	

MSCP-123 - Avoidance of side channel communication

Functional Actors shall only communicate among each other via Flows.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	• Communication between Functional Actors shall be standardized and safe and secure • to enable transparent replication • for the ease of portability of applications among platform realizations • for the ease of certification and acceptance
Remark	

MSCP-101 - Supervise maximum message delivery times

The Computing Platform shall supervise that messages are not delayed beyond the maximum message delivery times defined for the related Flows.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Functional Actors depend on reliable, reasonably deterministic communication with other Functional Actors. It is hence required that the Platform considers defined maximum message deliverable times in, e.g., the scheduling of Functional Actor Replicas.
Remark	

MSCP-79 - Maximum message delivery latency

The messaging mechanism of the platform shall provide a maximum message delivery latency of TBD among Functional Actors residing on the same physical platform.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	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	

MSCP-81 - Message delivery to Functional Actor Replicas

Flows shall ensure that messages posted by a Functional Actor are delivered to all recipient Functional Actor Replicas in the exact same order as they have been published.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	A Functional Actor can rely on the correct message distribution order without the need to implement order checking logic, which contributes to consistency among its Functional Actor Replicas.
Remark	

MSCP-78 - Message voting over the output of multiple Functional Actor Replicas

Where composite fail safety is used and Functional Actors run in replicas, the Computing Platform shall detect if one Functional Actor Replica provides once or several times a different message to the voting logic compared to equivalent messages of the other replicas of the same Functional Actor. If configured so, the Computing Platform takes action.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	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	

MSCP-80 - Quality of Service

The Computing Platform shall comply with the requirements and follow the recommendations of EN 50159.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	In terms of safe communication, EN 50159 provides guidelines which shall be followed to ensure that we have a common base for communication requirements.
Remark	Issues are identified by the platform (e.g., through the usage of message sequence numbers or some other platform-specific mechanism).

MSCP-117 - Time stamping of messages

The Computing Platform shall supplement messages with their time of creation.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	This is needed, so that receivers are able to determine how old messages are, and whether they should still be processed or discarded, etc.
Remark	This might require the notion of synchronized platform clocks (also among distributed platforms) at least to the extent/granularity (e.g., on the order of tens of ms) that is required to detect outdated messages. When exactly the time stamping happens needs to be discussed.

3.4 Monitoring, Diagnostics, Logging and Tracing

MSCP-111 - Provision of diagnostics information

The Computing Platform shall provide a bi-directional interface to exchange diagnostics information with Functional Actors.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Functional Actors can use the API, e.g., to receive diagnostics information or to report diagnostic information to the platform..
Remark	Diagnostic information could for instance comprise the health status of the platform or the Functional Actor.

MSCP-124 - Provision of logging and tracing API

The Computing Platform shall provide an interface towards Functional Actors for logging and tracing.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Applications have the need to provide log and trace data to the platform.
Remark	

MSCP-86 - Logging and tracing levels

The Computing Platform shall provide different logging and tracing categories and levels.

Status	✔ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	All APIs
Rationale	Depending on the required information, it is important to be able to enable logging and tracing only for certain Functional Actors and not for the entire system.
Remark	

MSCP-87 - Configure logging and tracing categories

The Computing Platform shall allow the configuration of logging and tracing categories and levels per Functional Actor and for the whole platform.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	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 negligible impact on the application performance and no impact on safety certification.
Remark	The Computing Platform shall be able to disable logging and tracing completely. To clarify if this is possible while the system is running or only if the system is stopped.

MSCP-127 - No impact of logging and tracing configuration on safety certification

The Computing Platform shall ensure that the safety certification is not violated by a change in the logging and tracing configuration.

Status	✓ Approved
Classification on-board	Requirement
Classification track-side	Requirement
Involved API	PI SIL4 API
Rationale	Logging and tracing functionality is required on an operational system i.e., a homologated system.
Remark	Most likely there will be certain safety related application conditions around the possible enabled logging and tracing categories/levels in a productive system.