D2.3

Report on 'TCMS framework concept' design, security concepts and assessment

Project number:	730830
Project acronym:	Safe4RAIL
Project title:	Safe4RAIL: SAFE architecture for Robust distributed Application Integration in rolling stock
Start date of the project:	1st of October, 2016
Duration:	24 months
Programme:	H2020-S2RJU-OC-2016-01-2

Deliverable type:	Report
Deliverable reference number:	ICT-730830 / D2.3 / 1.2
Work package	WP2
Due date:	March 2018 – M18
Actual submission date:	30th of March, 2018

Responsible organisation:	IKL
Editor:	Iñigo Odriozola
Dissemination level:	Public
Revision:	1.2

Abstract:	This report describes the TCMS Framework concept design with safety and security considerations and the assessment.
Keywords:	TCMS Functional Distribution Framework, Architecture, Safety Concept, Security Concept

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730830.

Editor

Iñigo Odriozola (IKL)

Contributors (ordered according to beneficiary numbers)

Iñigo Odriozola, Ekain Azketa, Asier Larrucea (IKL)

Bernd Löhr, Iris Bosse (NEW)

Rosa Iglesias, Aitor Uribarren, José Luis Flores (IKL)

Stefano La Rovere (NIER)

Bernhard Nölte, Alexander Piechullek-Königer (TÜV)

Executive Summary

The main task of WP2 of Safe4RAIL is to provide the "Functional Distribution" architecture concept for a mixed criticality embedded platform, offering an execution environment for multiple Train Control and Monitoring System (TCMS) application functions with a virtual bus inside the end-system.

In this context, task 2.4 and the resulting present deliverable targets the design of a high level 'functional distribution architecture' framework concept that could be potentially instantiated on different available frameworks and COTS solutions. This reference framework concept should consider functional distribution among devices distributed along the vehicle, service negotiation/plug and play (e.g. for functional open coupling), mixed-criticality (integration of functions with different criticality), railway certification standards, railway domain product life-cycle, hardware abstraction, abstract communication services, security and railway domain-specific requirements.

This framework is completed by a systematic assessment of deviations from the nominal behaviour of the implemented functions and of threats to data communication, producing a list of hazards, mitigated by a set of safety and security requirements to be met in the implementation of the proposed architecture. The associated 'safety concept' and 'security concept' have been assessed with respect to railway functional safety and security by a certification authority.

This deliverable gathers the result of a design process in form of a reference architecture for the Functional Distribution Framework. This architecture is described in terms of its conceptual view, the set of components that is made of its structural view, and the phases to be undergone in order to make it work. The steps followed to realize a Safety Concept and a Security Concept of this architecture are detailed, before completing the deliverable with their assessment. This assessment focuses on the requirements and expectations on the FDF design to support safety and security aspects and concludes with a set of appraisals.

Contents

Executive Summary **3**
Contents **4**
List of Figures **8**
List of Tables **10**
Chapter 1 Introduction **1**
1.1 About this document 1
1.2 Functional Distribution Framework Concept 1
1.2.1 Motivation 1
1.2.2 High-level requirements 2
1.2.2.1 Technical Requirements *3*
1.2.2.2 Non-technical characteristics *5*
1.2.3 Integrated Modular Platform Concept 6
Chapter 2 Design concept **7**
2.1 Introduction 7
2.2 Conceptual view 7
2.2.1 Elements 7
2.2.1.1 Variable *7*
2.2.1.2 Message *7*
2.2.1.3 SharedMemory *7*
2.2.1.4 Function *7*
2.2.1.5 Process *8*
2.2.1.6 Partition *8*
2.2.1.7 Schedule *8*
2.2.2 Mapping of elements 8
2.3 Structural view – software components 9
2.3.1 Hardware Access Services 10
2.3.1.1 IODriverManager *10*
2.3.1.2 NICDriverManager *10*
2.3.1.3 WDDriverManager *11*
2.3.1.4 ECUDriverManager *11*
2.3.2 Operating System Services 11
2.3.2.1 FileManager *11*
2.3.2.2 MemoryManager *11*
2.3.2.3 ConcurrencyManager *11*
2.3.2.4 TimeManager *11*
2.3.2.5 SocketManager *12*

2.3.2.6 LibraryManager 12
2.3.2.7 ExecutionManager 12
2.3.3 Functional Distribution Services 12
2.3.3.1 VariableManager 12
2.3.3.2 MessageManager 13
2.3.3.3 ConfigurationManager 13
2.3.3.4 NetworkManager 14
2.3.3.5 MonitoringManager 14
2.3.3.6 IOManager 14
2.3.3.7 SynchronizationManager 16
2.3.3.8 FunctionManager 16
2.3.3.9 FrameworkManager 16
2.3.3.10 HealthManager 17
2.3.3.11 LogManager 18
2.3.3.12 TopologyManager 19
2.3.3.13 Redundancy Manager 19
2.3.3.14 DeploymentManager 19
2.3.3.15 CryptoManager 20
2.3.3.16 UserAccountManager 21
2.3.3.17 SecurityMonitoringManager 21
2.3.4 FDF Detailed structural view 24
2.4 Behavioural view 26
2.4.1 Configuration phase 26
2.4.2 Initialization phase 27
2.4.2.1 Driver initialisation 28
2.4.2.2 Data initialisation 30
2.4.2.3 Function initialisation 33
2.4.3 Execution phase 42
2.4.3.1 Data monitoring 44
2.4.3.2 Data distribution 44
2.4.3.3 Global synchronisation 47
2.4.3.4 Watchdog refreshing 48
2.4.3.5 Input reading 48
2.4.3.6 Output writing 49
2.4.3.7 Redundancy management 51
2.4.3.8 Data logging 52
2.4.3.9 Data user function execution 53
2.4.3.10 Data topology discovery 54
2.4.3.11 Deadline checking 54
2.4.3.12 Disable execution 55
2.4.3.13 Load checking 55
2.4.3.14 Output checking 55
2.4.3.15 Temperature checking 56

2.4.3.16 Reset platform 56
2.4.3.17 Executable and configuration deployment 57
Chapter 3 Safety concept 58
3.1 FDF Functional model 58
3.2 FDF PHA Methodology 60
3.3 FDF PHA Results 61
3.3.1 System Hazards 62
3.3.2 Countermeasures 65
3.3.3 Application conditions 72
3.3.4 Recommendations 73
3.3.5 CONNECTA functional requirements mapping 74
Chapter 4 Security concept 76
4.1 Introduction 76
4.2 Motivation 76
4.3 Scope 77
4.4 Objective 78
4.5 Risk analysis – Security objectives 79
4.5.1 FDF brief description 79
4.5.2 Security dimensions or attributes 80
4.5.3 System assets 81
4.5.4 Use case: Bogie Monitoring System 81
4.5.4.1 General Description 82
4.5.4.2 Operational Description 82
4.5.4.3 Assets Used 83
4.5.4.4 Possible Threats/Attacks 83
4.5.5 Security objectives 86
4.6 Security requirements 87
4.7 Risk assessment 89
4.7.1 Security Level Target 90
4.7.2 Determination of the severity of the risk 91
4.8 Security countermeasures 92
4.8.1 Countermeasure 1: Trusted Platform Module (TPM) 92
4.8.2 Countermeasure 2: Password policy 93
4.8.3 Countermeasure 3: User profiles and application profiles policies 94
4.8.4 Countermeasure 4: Role-based access control (RBAC) 95
4.8.5 Countermeasure 5: Encryption 95
4.8.6 Countermeasure 6: Session bindings 96

4.8.7 Countermeasure 7: Network limited bandwidth 96
4.8.8 Countermeasure 8: Asset inventory 97
4.8.9 Countermeasure 9: Software-based memory protection unit....97
4.9 Functional Security Assessment Requirements.... 97
4.10 Conclusions and next steps.... 100
Chapter 5 Assessment of the safety and security concepts 101
5.1 Requirements given to the FDF design.... 101
5.1.1 General....101
5.1.2 Safety 102
5.1.3 Security....102
5.2 Assessment of the safety concept 103
5.2.1 Requirements....103
5.2.2 Approach and findings 103
5.2.3 Appraisal....104
5.3 Assessment of the security concept.... 104
5.3.1 Requirements....104
5.3.2 Approach and findings 104
5.3.3 Appraisal....105
Chapter 6 Integration of the Framework in the IMP 106
Chapter 7 Summary and conclusion.... 107
Chapter 8 List of Abbreviations 109
Chapter 9 Bibliography 112
ANNEX A: FDF Process Hazard Analysis.... 114
ANNEX B: Functional Security Assessment Requirements table.... 121

List of Figures

Figure 1. FDF in the context of a train 2
Figure 2. Integrated Modular Platform overview 6
Figure 3. Example of a Logical to physical mapping and accesses. The red arrow represents RO access whereas the black one RW. 9
Figure 4. FDF software components 10
Figure 5. FDF Dataflow perspective 25
Figure 6. Behaviour of the FDF in 3 phases. 26
Figure 7. Configuration phase. 27
Figure 8. Initialization phase. 28
Figure 9. Driver initialisation. 28
Figure 10. Initialization of IO drivers. 29
Figure 11. Initialization of NIC driver. 29
Figure 12. Initialization of watchdog drivers. 30
Figure 13. Initialization of ECU driver. 30
Figure 14. Data initialisation phase. 30
Figure 15. Initialization of Messages 31
Figure 16. Initialization of Variables. 32
Figure 17. Initialization of Topology objects. 33
Figure 18. Function initialisation phase. 34
Figure 19. Initialization of Synchronization functions 35
Figure 20. Initialization of Health functions. 35
Figure 21. Initialization of IOFunctions. 36
Figure 22. Initialization of MessageFunctions. 37
Figure 23. Initialization of NetworkFunctions. 38
Figure 24. Initialization of RedundancyFunctions 38
Figure 25. Initialization of LogFunctions 39
Figure 26. Initialization of MonitoringFunctions. 40
Figure 27. Initialization of TopologyFunctions. 40
Figure 28. Initialization of Deployment functions. 41
Figure 29. Initialization of UserFunctions (EXE) 41
Figure 30. Initialization of UserFunctions (DLL). 42
Figure 31. Cyclic execution phase. 43
Figure 32. Data monitoring use case. 44
Figure 33. Message composing use case 45
Figure 34. Message parsing use case. 45

Figure 35. Get Message use case. 46
Figure 36. Receive Message use case. 46
Figure 37. Send Message use case. 47
Figure 38. Set Message use case. 47
Figure 39. Global time synchronisation use case. 48
Figure 40. Refresh Watchdog use case. 48
Figure 41. Read Analog Input use case. 49
Figure 42. Read Digital Input use case. 49
Figure 43. Write Digital Output use case. 50
Figure 44. Write Analog Output use case. 51
Figure 45. Redundancy Manager use case. 52
Figure 46. Data logging use case. 53
Figure 47. Execution of user application use case. 53
Figure 48. Data topology discovery use case. 54
Figure 49. Deadline checking use case. 54
Figure 50. Disable execution use case. 55
Figure 51. Load checking use case. 55
Figure 52. Output checking use case. 56
Figure 53. Temperature checking use case. 56
Figure 54. Reset platform use case. 57
Figure 55. Executable and configuration deployment use case. 57
Figure 56. Logical View of the FDF with internal and external communication 77
Figure 57. Steps in the security concept 78
Figure 58. Elements for potential risk analysis. Source: Magerit [10]. 79
Figure 59. Primary and secondary system assets 81
Figure 60. Logical and Physical view of the bogie monitoring systems (BMSA). 82
Figure 61. Severity of risk. 91
Figure 62. Excerpt of Functional Security Assessment requirements for ISASecure certification for two functional requirements 'Access Control' and 'Use Control' 99
Figure 63. Integrated modular platform overview. 106

List of Tables

Table 1: FDF's fundamental Functions 59
Table 2: FDF's fundamental Services 59
Table 3: FDF's Functions and Services 59
Table 4: FDF PHA, Guidewords and deviations 60
Table 5: FDF PHA form, Functional failure mode and Failure effects 61
Table 6: FDF PHA form, Measures specification 61
Table 7: FDF PHA, List of System Hazards and relevant FDF Functions and deviations 64
Table 8: FDF PHA - countermeasures, Communication function 65
Table 9: FDF PHA - countermeasures, Configuration management 66
Table 10: FDF PHA - countermeasures, Framework management function 68
Table 11: FDF PHA - countermeasures, Functions management 68
Table 12: FDF PHA - countermeasures, Input/Output function 69
Table 13: FDF PHA - countermeasures, Message function 70
Table 14: FDF PHA - countermeasures, Holding Brake countermeasures 70
Table 15: FDF PHA - countermeasures, Time management function 71
Table 16: FDF PHA - countermeasures, Fault management function 72
Table 17: FDF PHA, Application conditions 73
Table 18: FDF PHA, Recommendations 74
Table 19: CONNECTA requirements – FDF Software Components mapping 75
Table 20: Security concept - Assets used 83
Table 21: Security Requirements of FDF 89
Table 22: Security objective coverage. 89
Table 23: List of Abbreviations 111
Table 24: FDF Process Hazard Analysis 120
Table 25: Security Requirements 129

Chapter 1 Introduction

1.1 About this document

This document focuses on describing the proposed Design concept for the TCMS Functional Distribution Framework (FDF), considering both safety and security concepts. It also aims to provide the proposed Safety concept and Security concept for the TCMS framework as well as summarise the conclusions obtained from the reviewing and assessment activities of the 'TCMS framework concept'.

This deliverable is organized in this way: Chapter 1 explains the motivation for creating the Functional Distribution Framework and explains the characteristics of the same. Chapter 2 describes the proposed 'Design concept' for the 'TCMS framework', considering both the safety and security concepts, by explaining the conceptual, structural and behavioural views. Then, the Safety Process Hazard Analysis and the resulting Safety Concept are described in chapter 3. After this, chapter 4 details the tasks that have been carried out in the process of creating the Security Concept and shows its results. In chapter 5, we can find the summary of the conclusions obtained from the reviewing and assessment activities of the 'TCMS framework concept'. Finally, chapter 6 explains how the FDF is integrated into the Integrated Modular Platform (IMP) before showing the summary of the activities in this task in chapter 7.

1.2 Functional Distribution Framework Concept

Functional Distribution Framework (FDF), the application framework concept for modular integration of TCMS applications, aims to host distributed safety-critical and non-critical application side-by-side on the same hardware platform in distributed next-generation TCMS systems. This solution will have to provide solutions to fulfil functional safety-critical and non-critical requirements and non-functional requirements (including security) that support functional distribution, interoperability, reconfiguration, deterministic inter-partition communication, hardware and communication abstraction and virtual coupling of services. The Functional Distribution Framework for the next generation TCMS needs to fulfil a set of requirements, in order to overcome today's TCMS limitations and provide further functionalities and enhancements.

1.2.1 *Motivation*

The main goal is then to provide the "Functional Distribution" architecture concept for a mixed criticality embedded platform, offering an execution environment for distributed TCMS safe and secure applications up to SIL4. This execution environment must ensure a strict temporal and spatial partitioning, location transparency and abstraction from the underlying network protocols and hardware. Figure 1 illustrates the FDF layer in the context of two train cars of a train. Two Ethernet Consist Networks (ECN) can be seen, one for each train car and a set of Electronic Control Units (ECU) distributed among the ECNs. Some applications of different SIL run in the ECUs and the Functional Distribution Framework abstracts these from the everything underneath, i.e., the FDF abstracts the applications from the Input/Output management and the network which will be the so-called Drive-by-Data technology, which aims to provide an Ethernet-based deterministic network also in the context of this project. Therefore, in the end, the FDF is an abstraction layer from the I/O Management and communication and this implies location too since the application will not know anything about the origin or destination of the data they consume or provide. Besides, the FDF also abstracts the applications from the

synchronisation of the different FDF nodes within a consist, since it is responsible for getting the global clock from the network and updating the system clock of each concrete ECU.

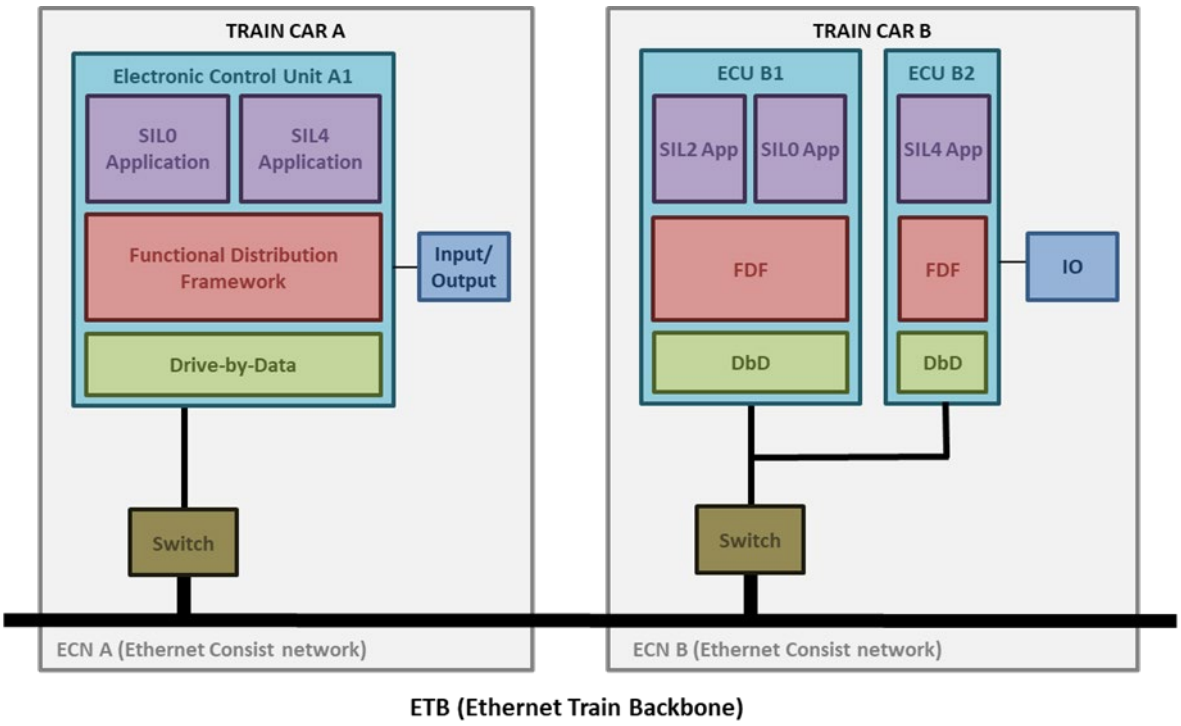

Figure 1. FDF in the context of a train

In order to achieve the previously mentioned goals, the Functional Distribution Framework must provide a set of services:

- Initialization.
- Global clock synchronisation.
- Scheduled execution of applications (of different SIL).
- Safe local data distribution.
- Safe and secure remote data distribution.
- Transparent IO reading and writing.
- Health-monitoring.
- Remote monitoring.
- Logging.
- Deployment, which means providing the ability to update an application without altering the rest.

1.2.2 *High-level requirements*

If we go more into detail, the Functional Distribution Framework for the next generation TCMS needs to fulfil a set of requirements to overcome today's TCMS limitations and provide further functionalities and enhancements. These are described in the following lines.

1.2.2.1 Technical Requirements

Configuration and management services

- Configuration services: The framework needs to offer services to read, parse and load data from a configuration file, which will contain all required information in order to set up the system. At the same time, it must check the content for coherency and integrity. The framework also needs to allow the online system reconfiguration during the train inauguration process.
- Partition management: A memory manager needs to guarantee the isolation of memory spaces. Besides, a cyclic executive scheduler must give and take away access to the processor when corresponds.
- Process management: In order to perform optimal process management, the framework must offer services to create and manage timers for sequential execution and semaphores for sequential and concurrent execution. Each process must wait for a "start semaphore" to be signalled and, when execution is finished, it must signal a so-called "finish semaphore". The identifiers of both semaphores are specified in the configuration file.
- Function management: For correct function management, the framework must offer services to create and manage threads and timers. The main configuration parameters of a thread would be its priority and the function or functions that are executed. It must be able to execute functions sequentially, i.e. by the use of only one thread for all, or concurrently in a multithread way.
- Time management: Having a global system time is essential to execute distributed applications, especially when they are time-triggered. In order to have a correct global system time, the Framework offers a service to get the global time that arrives from the network.
- Memory management: The Framework offers services to create, configure and manage shared memories.
- Communication management: The FDF shall allow sending and receiving messages to and from other FDF nodes of the network in a transparent way and without knowledge of the underlying networking technology.
- Data exchange management: The Framework offers services to create exchange variables for data sharing between different processes. Variables are data structures defined by a unique identifier, a data type, an updating semantic (e.g. sample, buffer) and some quality of service parameters (e.g. deadline, validity, freshness, persistence). Analogously, the framework offers services to create Messages, which are data structures used to communicate with remote applications in other nodes.
- Incremental certification and re-certification of applications: Among the management services, the FDF should facilitate the certification of concrete applications, ensuring that the rest of the applications are unaffected.

Time services: Since the next generation TCMS is supposed to be a functional distribution architecture framework which can host different applications, from this point of view, the time inside this system should be unique and independent of partition execution within an integrated module. All the integrated modules should use the unique time and all time values or capacities reference only to this unique time, instead of relative to any partition execution. Apart from unique time mechanism, TCMS must also provide other time management services such as creating timers.

Input/output services: Another aspect of the Framework is that it must offer a service to access an I/O device, which can be configured as input or output, analogue or digital and also

map this value to an exchange variable where the value is written to and read from. The framework will guarantee that at the beginning of each basic cycle (loop) the current value of every used input is stored in the associated exchange variable. Similarly, the framework will guarantee that at the end of each basic cycle the current value of every used output is set according to the containment of the associated exchange variable.

Real-time support: Scheduling of partitions should be feasible through the standard application programming interface (API) which is provided by the framework. Partitions could be scheduled on a cyclic basis, which enforces the operating system (OS) to maintain a major time frame for all the partitions. Major time frame will periodically repeat throughout the integrated module's runtime operation. The target framework is supported to provide hard real-time. Mechanisms need to be designed to ensure the hard real-time so that the framework can fulfil SIL4 functions requirements. Scheduling of the threads within the same partition should be designed to meet the requirement that some threads should not be pre-empted, in order to implicitly ensure the real-time support of the architecture. At the same time, the processor will always be granted to the highest priority of all the threads.

Fault isolation: This goal framework must be designed to have fault containment. The applicable way is that this execution environment ensures strict space partitioning so that it is not possible for a partition to access the memory space of another partition. Robust partitioning for TCMS should comprise the protection of each partition's addressing space, through specific memory protection mechanisms (e.g. mechanisms implemented in a hardware memory management unit (MMU)). At the same time, functional protection should be implemented to manage the privilege levels and restrictions to the execution of privileged instructions.

Health monitoring and error-handling: Health monitoring is another of the facets of the FDF. It must provide with the recognition of system status concerning errors and failures that might occur or have occurred and as such help to identify faults in the system and mitigate their consequences, i.e. maintaining safe behaviour. This can be the result of, e.g. timeouts for process data and/or collecting and analysing status information of components and devices. Health monitoring will take into account different error sources, log them and determine recovery actions configured by the system designer.

Safety services: Safety is a significant virtue for the platform, as it must be able to host next-generation TCMS applications up to SIL4. In order to achieve such a SIL level, safety measures must be taken throughout the whole framework. The CRC check shall be used to protect the configuration at system setup and the SDT layer must be used to guarantee that a message has not been corrupted in the way to the destination. Moreover, the variable stores must be mirrored and the IO readings carried out redundantly.

Security services: The target FDF shall grant data integrity, authenticity and confidentiality. In order to do so, software or hardware security mechanisms shall be used throughout the framework. As an example, the framework must provide cryptographic mechanisms in the communication component to decrypt incoming messages and encrypt outgoing ones or make use of strict access control for the use of the monitoring functionality.

Requirements for underlying hardware: TCMS should meet some specific requirements for the processors. The processing capacity should be sufficient to meet the worst-case timing requirements and it must be granted that the processor has access to required I/O and memory resources and also to time resources to implement the time services.

It must also be assured also that the processor provides atomic operations for implementing processing control constructs and a mechanism to transfer control to the OS if the partition attempts to perform an invalid operation. If we are to achieve SIL3/4, lock-step architecture is needed for the processors. MMU hardware or a BSP software layer providing Built-in Tests (BIT) could be some of the solutions.

In order to realise the FDF for the next generation TCMS, another aspect to be taken into account is that interrupts need to be strictly forbidden to disturb the time partitioning. Besides, for the goal framework, we need not only the definition of the use of multiple threads within a partition scheduled to execute concurrently on different processor cores, but also the definition of scheduling behaviours associated with multiple partitions which need to be scheduled to execute concurrently on different processor cores.

1.2.2.2 Non-technical characteristics

A need for System Architecture Engineering Method: A system architecture engineering method, which is a systematic, documented, intended way how system architecture engineering is realized, needs to be established for the Safe4Rail project. The reason is that systematic approach is needed to engineer good quality system architecture and a consistent set of its representations (views, models, visions, quality cases, analysis reports, simulations). The system architecture is critical since it supports achievement of critical architecturally significant requirements; it enables engineering of system quality characteristics and attributes and also drives all logically downstream activities. Finally, it greatly affects cost, schedule and risk.

Moreover, quality characteristics such as performance, safety, security, availability and interoperability can be considered main architectural drivers for the system that is the subject of the Safe4RAIL project. Thus, the evaluation of the architecture should be based on the architectural quality cases which should be developed for the particular quality characteristics and their attributes. The architectural quality case consists of Architectural Claims, Architectural Arguments that justify belief in those claims and Architectural Evidence which support the arguments.

Safety and the relevant standards: The set of standards containing the EN 50126 series [2], EN 50129 [4] and EN 50128 [3], comprise the railway sector equivalent of the EN 61508 series, a general standard for functional safety in electronic safety-related systems, as far as Railway Communication, Signalling and Processing Systems are concerned. To cover the safety-related communication in such kind of systems that set of standards was completed by EN 50159. Even though the new versions of EN 50129 and EN 50126 have been published, the original versions are active, so the current pre-norms should be the working versions in the Safe4Rail project and must be considered when defining the FDF too.

Security and the relevant standards: As far as the IT security in the Railway domain is concerned no such a set of standards as that addressing functional safety in railway applications has yet come into existence. However, the work on it has already started. Currently, the NWIP (New Work Item Proposal) of the security standard called Railway Applications - Communication, signalling and processing systems – IT security requirements for electronic systems for signalling is under preparation in SC9XA of CENELEC. This standard, if finished, would have likely provided most answers to the security issues related to the new generation of TCMS. Even though we will have to do without it in the Safe4Rail project the approach to the security of electronic railway systems for signalling indicated in that NWIP will surely provide good guidance. It happens that this standard will be based on IEC 62443 series, which deals with the cybersecurity in industrial systems - the studies have shown that there is a considerable degree of overlap in both domains as far as the IT security regulations and rules are concerned. Therefore this standard should be considered foremost, along with others such as "IEC 15408 – Common Criteria" and "DIN VDE V 0831-104" [20] and "VDE V 0831-102" [19], both draft standards elaborated by the German DKE standardisation committee.

1.2.3 *Integrated Modular Platform Concept*

The goal of the Integrated Modular Platform (IMP) is the facilitation of system integration, interfacing and information transfer from one application partition to another application partition in the networked system. It focuses on all system integration capabilities required to define an integrated modular platform which can host different TCMS, door control, braking, safety or other non-critical functions in one system. The integrated modular platform hosts application functions and provides specific services to critical and non-critical applications, to establish robust software abstraction and provide all resources and timely information (sensors, global variables) access to applications.

The IMP does not depend on applications. Modular applications hosted on an Integrated Modular Platform can be tested in isolation and integrated into the system, without unintended interactions and interdependencies. The IMP represents the lower part of the integrated system, see Figure 2.

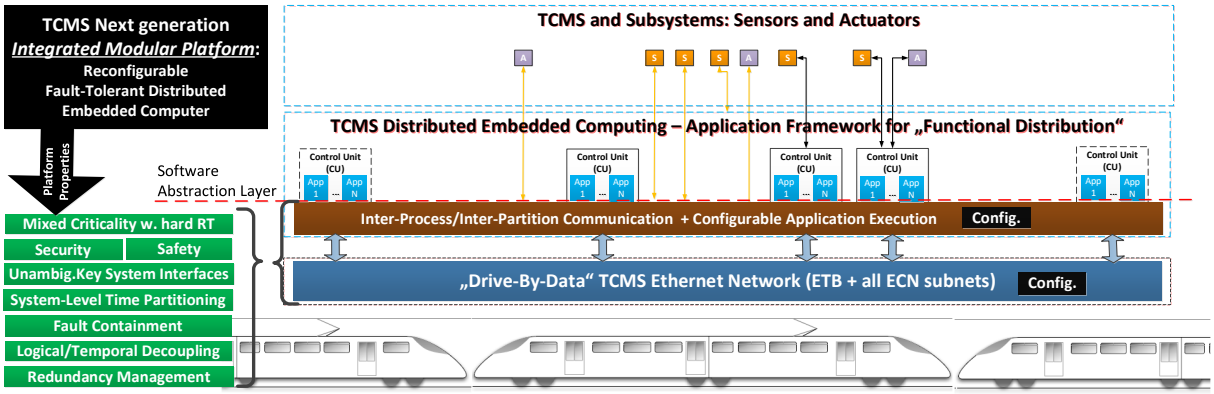

Figure 2. Integrated Modular Platform overview

In the figure, the inter-process and inter-partition communication and configurable application execution are part of the so-called Functional Distribution Framework (highlighted in brown colour), whereas TCMS Ethernet network represents the inter-node communication system or the so-called Drive-By-Data framework. Both parts are described in detail in the next two sections.

The IMP approach implies a paradigm shift from the current federated architecture of loosely connected applications to a much more integrated view, where resources are fundamentally shared between multiple applications. The IMP is a subsystem, whose only function is to host different applications. The configuration of this subsystem, since it is underlying all other applications, is of fundamental importance and must ensure the safe and reliable operation of all applications that make use of it.

The usage of the IMP in a safe and reliable context is ensured through its safe-by-design components as indicated in the next chapters, combined with a unified methodology to its configuration. The configuration of integrated modular platform components adapts the integrated modular platform to a specific use case and topology or architecture.

Chapter 2 Design concept

2.1 Introduction

This chapter explains the design concept of the FDF. First, the conceptual view shows the fundamental elements in which the FDF is based. The structural view shows the architecture of the proposed design by clearly describing the functionality and characteristics of each of the components and how they make use of the previously mentioned fundamental elements. Finally, the behaviour of the FDF is explained in terms of sequence diagrams of concrete use cases in the behavioural view subchapter.

2.2 Conceptual view

This chapter describes all the physical and logical elements that interact within the Functional Distribution Framework. Concrete devices and peripherals such as the Network Interface Card, the Input-Output Interface Card, the Watchdog, storage or CPU also interact with the FDF but are not described in this chapter.

2.2.1 *Elements*

2.2.1.1 Variable

- Data structure to share information between parts of the applications. It can also be a complex nested structure.

2.2.1.2 Message

- Data structure to share Variables between parts of the applications residing in different computing nodes.

2.2.1.3 SharedMemory

- Memory space that can be simultaneously accessed by several Processes.
- It is statically created at compilation time and filled at runtime.
- Depending on the content type there are VariableMemories and MessageMemories.

2.2.1.4 Function

- Schedulable software unit that implements some logic.
- It typically reads the required input data from Variables, Messages or Hardware devices and writes the generated output data to Variables, Messages or Hardware devices.
- Types:
    - ApplicationFunction:
        - Implements the logic of the application
        - It is provided and instantiated by the user.
    - ServiceFunction:

- Implements logic of a FDF service.
- It is provided and instantiated by the FDF.

2.2.1.5 Process

- Executable unit managed by the operating system with isolated memory address space protected against usage from other processes.
- A process consists of one or more threads.
- Temporal separation to other processes is not guaranteed.
- Executes some Functions according to the scheduling plan specified in the Configuration.
- Shared memory can be read by functions of many processes but written by functions of only one process.
- It is configured and executed at runtime.

2.2.1.6 Partition

- Logical unit of isolation with exclusive access to predetermined memory space and to the processor in predetermined time slots.
- A partition is composed of one or several processes.
- Processes can run sequentially (one after the other) or concurrently (with specific priority levels assigned to the processes).
- It is statically created at compilation time.

2.2.1.7 Schedule

- The plan, sequence or time allocation of an execution.
- Types:
  - Partition Schedule:
    - Scheduling plan of the Partitions.
    - A Partition can belong to different Partition Schedules.
    - There can be many Partition Schedules, but only one is loaded at a time.
  - Process Schedule:
    - Scheduling plan of the Processes.
    - A Process can belong to different Process Schedules.
  - Function Schedule:
    - Scheduling plan of the Functions.
    - A Function can belong to different Function Schedules.

2.2.2 *Mapping of elements*

This chapter explains how every FDF element interacts with each other. In Figure 3 we can see the mapping between the different logic and physical elements. Variables are stored in

Variables Shared Memories and Messages Shared Memories contain Messages. A partition can contain one or more processes of the same SIL, while a process can have one or more functions running in it. There can be as much as needed partitions. As can be seen in the picture, a process will have Read-Write (RW) access only to its own Variables Shared Memory, which will be of the same SIL, and Read-Only (RO) access to the rest. Besides, all processes can have RW access to the messages shared memory, regardless of the SIL. The reason for this is that the messages shared memory will always be SIL0 because typically the network will use a non-safe protocol implementation. Every process can have RW access to the rest of physical elements, such as, for instance, the Watchdog, NIC or IOC. The concrete access right of the different processes to the elements will be given per configuration.

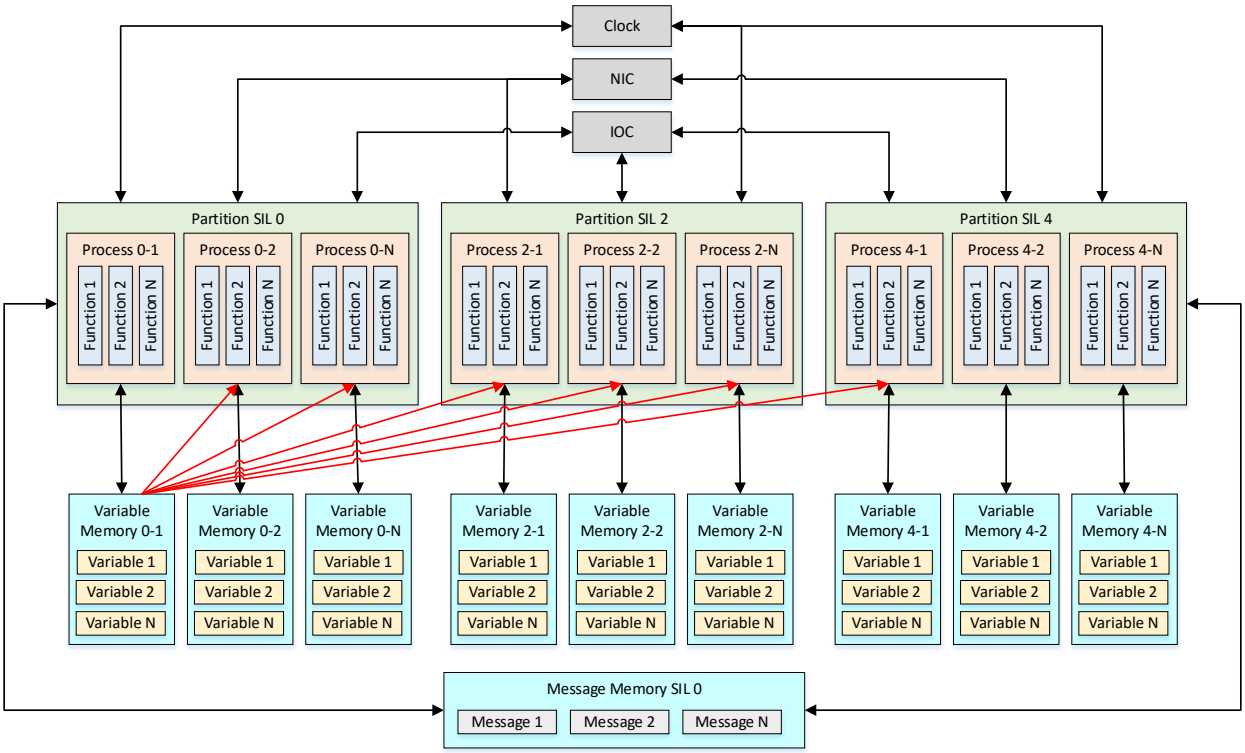

Figure 3. Example of a Logical to physical mapping and accesses. The red arrow represents RO access whereas the black one RW.

2.3 Structural view – software components

The exposed physical and logical elements of the FDF are implemented and managed by some software components, which are defined in terms of the functionality they offer and the interfaces they provide and require. A software component may have different versions depending on the SIL they can be used on, typically by using more rigorous development methodologies and by adding more safety-related functionality to the lower SIL versions. On the other hand, the provided interfaces are required and other components provide the required interfaces. Figure 4 shows the components of the Framework. As can be seen, they are clustered in the group's Hardware Access Services, Operating System Services and Functional Distribution Services, which are explained in the following chapters.

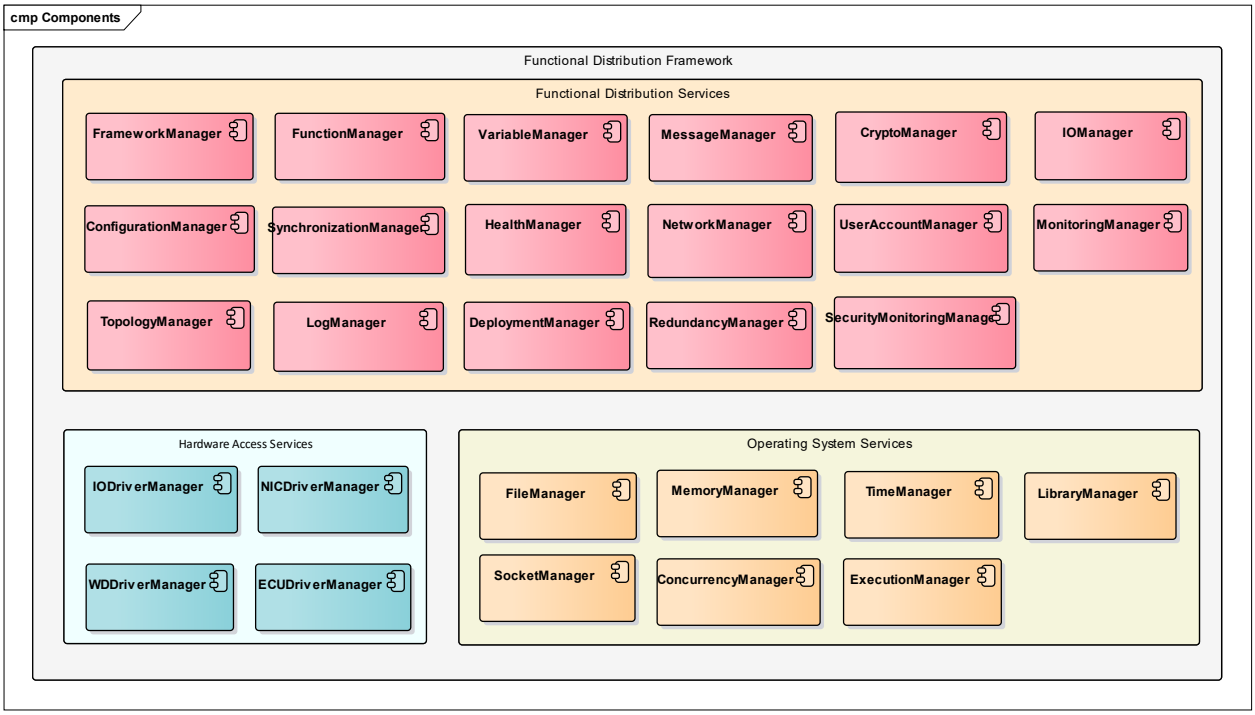

Figure 4. FDF software components.

2.3.1 *Hardware Access Services*

These components provide access to the underlying hardware. They may provide either complete implementations of the hardware access services or wrapper functions to the hardware access services provided by the underlying Drivers. The components provide an independent access layer to the hardware and are thus partly hardware-dependent. They have the same interface but different implementations for different IO and NIC hardware.

2.3.1.1 IODriverManager

- **Functionality:** Provide services to access Input and Output Cards.
- **Provided interfaces:**
  - **IAIDriver:** Read analog input values and errors.
  - **IDIDriver:** Read digital input values and errors.
  - **IAODriver:** Read and write analog output values and errors.
  - **IDODriver:** Read and write digital output values and errors.

2.3.1.2 NICDriverManager

- **Functionality:** Provide services to access Network Interface Cards.
- **Provided interfaces:**
  - **IBEDriver:** Send and receive best-effort Messages.
  - **IRTDriver:**
    - Send and receive real-time Messages.
    - Obtain the global time.

2.3.1.3 WDDriverManager

- **Functionality:** Provide services to access Watchdogs.
- **Provided interfaces:**
  - **IWDDriver:** Configure and refresh the Watchdog.

2.3.1.4 ECUDriverManager

- **Functionality:** Provide services to access the ECU data.
- **Provided interfaces:**
  - **IECUDriver:** Get load and temperature of the ECU.

2.3.2 *Operating System Services*

These components provide an abstraction layer of services related to the Operating System. Due to their dependence on the underlying OS services, they may have different implementation for different Platforms/Operating Systems. However, the interface to access the services shall remain OS-independent. They may provide either complete implementations of their services or wrapper functions to the services provided by the underlying Operating System.

2.3.2.1 FileManager

- **Functionality:** Provide services to manage files.
- **Provided interfaces:**
  - **IFile:** Create, delete, read and write files.

2.3.2.2 MemoryManager

- **Functionality:** Provide services to manage Shared Memories.
- **Provided interfaces:**
  - **ISharedMemory:** Open, close and read Shared Memories.

2.3.2.3 ConcurrencyManager

- **Functionality:** Provide services to synchronise the concurrent access to shared resources and concurrent executions.
- **Provided interfaces:**
  - **ISemaphore:** Open, wait and post semaphores.
  - **IMutex:** Open, lock and unlock mutexes.

2.3.2.4 TimeManager

- **Functionality:** Provide services to manage the system clock and timers.
- **Provided interfaces:**

- **IClock:** Get and set the time of the system clock.
- **ITimer:** Create, delete and configure timers.

2.3.2.5 SocketManager

- **Functionality:** Provide services to set communication channels through sockets.
- **Provided interfaces:**
  - **ISocket:** Create sockets, establish output connections and listen to inputs connections, send and receive data.
- **Required interfaces:** *IBEDriver* and *IRTDriver*.

2.3.2.6 LibraryManager

- **Functionality:** Provide services to handle dynamic libraries.
- **Provided interfaces:**
  - **ILibraryManager:** Opens, closes and handles errors of dynamic libraries.

2.3.2.7 ExecutionManager

- **Functionality:**
  - Guarantee isolation between Partitions.
  - Execute Partitions according to the Partition Schedule.
  - Execute Processes according to the Process Schedules.
- **Provided interfaces:**
  - **IScheduling:** Change from one Partition Schedule to another.

2.3.3 *Functional Distribution Services*

The Functional Distribution Services provide the actual middleware services that are used by applications. The code of these components is portable across different Platforms/Operating Systems because the Hardware Access Service and Operating System Service layers provide specified interfaces. Applications that make use of the FDF services are functionally portable across platforms. **The three last software components, i.e. CryptoManager, UserAccountManager and SecurityMonitoringmanager, are product of the Security Concept**, as they provide countermeasures against security threats.

2.3.3.1 VariableManager

- **Functionality:** Read, write, force and unforce Variable values; read and write Variable qualities; read Variable update time.
- **Provided interfaces:**
  - **IVariable:** It encapsulates a Variable and provides functions to read, write, force and unforce its value and read and write its quality. A Variable is stored in a Shared Memory and if it is safe is also stored mirrored in a second Shared Memory. When a safe Variable is read the mirrored instance is unmirrored and

compared field by field with the main instance and if something is different, the quality is set to bad.

- **Required interfaces:** *IMutex* and *IClock*.

2.3.3.2 MessageManager

- **Functionality:** Provide services to compose, to parse and to access the content of Messages.
- **Provided interfaces:**
  - **IMessage:** It encapsulates a Message and provides functions to read and write the value and the SDT layer. When the value of the Message is set, an internal timestamp is updated.
  - **IComposeFunction:** It reads the state of the system and if it is safe it reads the default values of the specified Variables, else it reads the values and qualities. After that composes a Message with them, with the quality of good for the default values. If the Message contains safe Variables, it computes the SDT layer and attaches it to the Message.
  - **IParseFunction:** It reads a Message, parses from it the values and qualities, and writes the values to the specified Variables. If the Message contains safe Variables, checks the SDT layer of the Message and if it is wrong sets the qualities of all the Variables to bad, else sets the qualities of all the Variables to the parsed ones.
- **Required interfaces:** *IVariable*, *IReceiveFunction*, *ISendFunction*, *IMutex* and *IClock*.

2.3.3.3 ConfigurationManager

- **Functionality:** Provide services to manage the Configuration of all the services provided by FDF This manager checks and loads the configuration and saves it into shared memory, then the other managers read the configuration.
- **Provided interfaces:**
  - **IConfiguration:** It encapsulates the Configuration and offers a function to initialise it. If it is a safe process, it opens the Shared Memory with the specified name, maps its content to a Configuration object and checks its CRC. If it is not a safe process, it opens, reads and closes the Configuration file with the specified name and creates the Configuration object. If it is configured as a safety configuration loader, it opens the Shared Memory with the specified name and maps the Configuration object to it. Then, checks the coherency of the Configuration. Finally, it returns the corresponding error code depending on the following situations:
    - Configuration with the specified name does not exist.
    - The File *fopen* function returns an error.
    - The File *fread* function returns an error.
    - The File *fclose* function returns an error.
    - The SharedMemory *shm_open* function returns an error.
    - The SharedMemory *mmap* function returns an error.
    - The CRC is not correct.

- The coherency of the Configuration is not correct.
- Everything is ok.

- **Required interfaces:** *IFile* and *ISharedMemory*.

2.3.3.4 NetworkManager

- **Functionality:** Provide services to send and receive Messages to and from remote nodes.
- **Provided interfaces:**
  - **ISendFunction:** It reads the specified Message and sends it to the specified remote nodes using a communication protocol through a socket. Finally, it writes the corresponding error code in the result Variable depending on the following situations:
    - The Socket *sendto* function returns an error.
    - Everything is ok.
  - **IReceiveFunction:** It reads the specified datagram of a communication protocol through a socket. If the receiving function returns an ok, it sets the received datagram as the value of the specified Message. Finally, it writes the corresponding error code in the result Variable depending on the following situations:
    - The Socket *recvfrom* function returns an error.
    - The Socket *recvfrom* function returns an ok but no message.
    - Everything is ok.
- **Required interfaces:** *IMessage* and *ISocket*.

2.3.3.5 MonitoringManager

- **Functionality:** Provide services to monitor Variables remotely.
- **Provided interfaces:**
  - **IMonitoringFunction:** Provide remote access to Variables. It executes a server that implements the monitoring protocol. When a request arrives, it replies with the information of the monitored Variables: type, default value, value, quality, forced and timestamp. It is executed in a process different from the ones that produce the monitored Variables and the access to the Shared Memories of those Variables is configured and assured by the Framework as read-only. Finally, it writes the corresponding error code in the result Variable depending on the following situations:
    - The Socket *accept* function returns an error.
    - The Socket recv function returns an error.
    - The Socket send function returns an error.
- **Required interfaces:** *IVariable* and *ISocket*.

2.3.3.6 IOManager

- **Functionality:** Provide services to move data from Inputs to Variables and from Variables to Outputs.
- **Provided interfaces:**
  - **IAIFunction:** It reads an analog input from the *AIDriver* and writes the value to the input Variable value. If the return of the read function is ok sets the quality of the Variable to good and else to bad. In any case, it writes the corresponding error code in the result Variable depending on the following situations:
    - The *AIDriver* read function returns an error.
    - Everything is ok.
  - **IDIFunction:** It reads a digital input from the *DIDriver* and writes the value to the input Variable value. If the return of the read function is ok sets the quality of the Variable to good and else to bad. In any case, it writes the corresponding error code in the result Variable depending on the following situations:
    - The DIDriver read function returns an error.
    - Everything is ok.
  - **IDOFunction:** It reads the quality of the state variable and if it is good reads its value. If the quality is bad or the quality is good and the value equal to safe, it reads the default value of the output Variable, else reads the quality. It the quality is bad, it reads the default value, else reads the current value. Then it writes the read value to the digital output through the *DODriver*. After that, it reads the digital output and compares it to the output Variable. Finally, it writes the corresponding error code in the result Variable depending on the following situations:
    - The quality of the output Variable is bad.
    - The *DODriver* write returns an error.
    - The *DODriver* read returns an error
    - The written and read values are different.
    - Everything is ok.
  - **IAOFunction:** It reads the quality of the state Variable and if it is good reads its value. If the quality is bad or the quality is good and the value equal to safe, it reads the default value of the output Variable, else reads the quality. It the quality is bad, it reads the default value, else reads the current value. Then writes the read value to the analog output through the *AODriver*. After that, it reads the analog output and compares it to the output Variable. Finally, it writes the corresponding error code in the result Variable depending on the following situations:
    - The quality of the output Variable is bad.
    - The *AODriver* write returns an error.
    - The *AODriver* read returns an error
    - The written and read values are different.
    - Everything is ok.
- **Required interfaces:** *IAIDriver*, *IDIDriver*, *IAODriver*, *IDODriver* and *IVariable*.

2.3.3.7 SynchronizationManager

- **Functionality:** Provide services to synchronise the local clock with the global time.
- **Provided interfaces:**
  - **ISynchronizationFunction:** It gets the global time from the *RTDriver* and the local time from the Clock. If both functions return an ok and the difference between them is greater than a specified value, it sets the local clock with the global time. Then it writes the corresponding code in the result Variable depending on the following situations**:**
    - The Clock *gettime* function returns an error.
    - The *RTDriver gettime* function returns an error.
    - The *RTDriver gettime* function returns a time not greater than in the previous cycle.
    - The Clock *settime* function returns an error.
    - Everything is ok.
- **Required interfaces:** *IRTDriver* and *IClock*.

2.3.3.8 FunctionManager

- **Functionality:** Provide services to execute the registered Application and Service Functions.
- **Provided interfaces:**
  - **IFunctionManager:** It provides functions to register Functions to be executed and to execute all the registered Functions. In the execution phase, it reads the quality of the execution flag Variable of the Function. If the quality is good, it reads the value of the Variable, else the default value. If it is true, gets the current time, executes the Function, and gets the current time again to calculate the execution time of the Function, which is written in the execution time Variable of the Function. If any of the current time functions call returns an error, it sets the quality of the execution time Variable to bad. Finally, it writes the corresponding code in the result Variable depending on the following situations**:**
    - Any *gettime* function returns an error.
    - Everything is ok.
- **Required interfaces:** *IFunction*, *IClock* and *IVariable*.

2.3.3.9 FrameworkManager

- **Functionality:** Provide service to instantiate/open all the resources specified in the Configuration and expose the API.
- **Provided interfaces:**
  - **IFrameworkManager:** It provides functions to:
    - Configure: Receives a name and calls the initialise function of *IConfiguration*.

- Initialize: According to the configuration, initialises the specified drivers (WD, IO, NIC and ECU) and stores (Variables, Messages, Topology). Then it initialises and registers in *FunctionManager* the configured functions (Synchronization, Health, IO, Message, Network, Redundancy, Monitoring, Log, Topology, Deployment, UserDLL and *UserEXE*).
- Execute: It enters in a loop and waits in the partition semaphore. When partition starts it signals that semaphore to awake all the processes of the partition. The process reads the value of its execution flag and if it is true gets the current time and calls the execute function of the *FunctionManager*. When this function returns, gets the current time, computes the execution time of the process and writes it in the execution time Variable of the process.
- Register: It registers the specified *Function* in the *FunctionManager*.
- Get variable: It returns the reference of the specified Variable.
- Get topology: It returns the reference of the Topology.
- Get log: It returns the reference of the specified Log.

- **Required interfaces:** All except for *SocketManager* and *ExecutionManager*.

2.3.3.10 HealthManager

- **Functionality:**
  - Provide check services:
    - Deadlines of Functions/Processes.
  - Provide react services:
    - Change Schedule Functions
    - Disable Execution Functions.
    - Terminate Execution Processes.
    - Reset the ECU.
    - WDT function
- **Provided interfaces:**
  - **ITemperatureFunction:** It reads the temperature of the specified device (CPU, Board or Rack) through the ECUDriver. If the function returns an error it sets the value of the quality of the temperature error Variable to bad; else it sets to good. Then checks if the temperature is within the specified range and sets the result in the value of the error Variable. Finally, it writes the corresponding error code in the result Variable depending on the following situations:
    - The ECUDriver get_temperature function returns an error.
    - Everything is ok.

- **ILoadFunction:** It reads the load of the specified device (CPU, Board or Rack) through the ECUDriver. If the function returns an error it sets the value of the quality of the load error Variable to bad; else it sets to good. Then checks if the load is not greater the specified maximum and sets the result in the value of the error Variable. Finally, it writes the corresponding error code in the result Variable depending on the following situations:
  - The ECUDriver get_load function returns an error.
  - Everything is ok.
- **IOutputFunction:** It reads the timestamps of the specified Variables, checks if they have been updated in the current cycle and sets the value of the output error Variable to true if yes and to false if not.
- **IDeadlineFunction:** It reads the quality of the execution time Variable. If it is bad sets the value of the deadline error Variable to true, else reads the value of the execution time Variable and if it is not greater than the specified deadline sets the value of the deadline error Variable to false, else to true.
- **IDisableExecutionFunction:** It reads the quality of the triggering Variable and if it is bad gets the default value else the current value. If the value is true sets the value of the execution flag Variable to false, else to true.
- **ITerminateProcessFunction:** It reads the quality of the triggering Variable and if it is bad gets the default value, else gets the current value. If the value of the triggering Variable and if it is true sets the value of the execution flag Variable to false, else to true.
- **IResetPlatformFunction:** It reads the quality of the triggering Variables and they are bad gets the default values else the current values. If all values are false refreshes the Watchdog.
- **IWDFunction:** It refreshes the Watchdog through the WDDriver.

- **Required interfaces:** *IWDDriver* and *IVariable*.

2.3.3.11 LogManager

- **Functionality:** Provide services to write Variable values in files.
- **Provided interfaces:**
  - **ILog:** It encapsulates the Log object. Provides functions to record Log entries and to write the Log object in a file. When the maximum size of the Log object is reached, it overwrites previously recorded Log entries, starting from the oldest.
  - **ILogFunction:** If a given Variable is true, it creates a log entry with the specified message and the values of the specified Variables and writes in the Log. Then, if the write file flag is true calls the function of the Log object to write it in the file. Finally, it writes the corresponding code in the result Variable depending on the following situations:
    - The Log add function returns and error.

- The Log write function returns and error.
- Everything is ok.

- **Required interfaces:** IVariable and IFile.

2.3.3.12 TopologyManager

- **Functionality:** Provide services to get and update the train topology information.
- **Provided interfaces:**
  - **ITopology:** It encapsulates the train topology information and provides functions to read and write the value and the quality of the Topology. When the value of the Topology is set, an internal timestamp is updated.
  - **ITopologyFunction:** It listens to periodic messages indicating whether the Topology has been updated or not. If the message is not received within the expected period, it sets the quality of the Topology to bad. When receives a message indicating an update it sends a request for the new Topology information to the server. If the reply from the server is not received with the expected deadline, it sets the quality of the Topology to bad, else updates the Topology information and sets its quality to good. Finally, it writes the corresponding code in the result Variable depending on the following situations**:**
    - The periodic update message is not received within the deadline.
    - The reply message is not received within the deadline.
    - The Socket recvfrom function returns an error.
    - The Socket send function returns an error.
    - The Socket recv function returns an error.
    - Everything is ok.
- **Required interfaces:** *ISocket*, *IMutex* and *IClock*.

2.3.3.13 Redundancy Manager

- **Functionality:** Provide services to manage redundant Functions (enable/disable redundant instances).
- **Provided interfaces:**
  - **IRedundantFunction:** It reads the current time and the last update time of the keepalive Variable sent by the master Function. If the current time function returns and ok and the current time is greater than the update time plus the specified deadline, it enables the execution of the redundant Functions by setting to true the values of their execution flag Variables. If the current time function returns an error, it sets the quality of the execution flag Variable to bad, else to good. Finally, it writes the corresponding code in the result Variable depending on the following situations**:**
    - The Clock gettime function returns an error.
    - Everything is ok.
- **Required interfaces:** *IVariable*.

2.3.3.14 DeploymentManager

- **Functionality:** Provide service to update configuration files and executables remotely.
- **Provided interfaces:**
  - **IDeploymentFunction:** It executes a server that implements a secure file transfer protocol such as FTPS or SFTP. When a request arrives, it checks if the client is authorised and if so, allows it to get and put files (executables, libraries and configurations) from and in the system. Finally, it writes the corresponding error code in the result Variable depending on the following situations:
    - The Socket accept function returns an error.
    - The Socket recv function returns an error.
    - The Socket send function returns an error.
    - Everything is ok.

2.3.3.15 CryptoManager

- **Functionality:**
  - Providing cryptographic services to software components:
    - Encryption
    - Decryption
    - Hashing
    - Signature
  - Providing interface with the Trusted Platform Module (see Section 4.8).
  - It is used by most of the software components.
- **Provided interfaces:**
  - **IPublicKeyGenerationFunction:** This function generates the corresponding public key.
  - **IPublicKeyVerificationFunction:** This function performs an embedded public key validation.
  - **IKeyManagementFunction:** This function deals with storage, use and deletion.
  - **ISignatureGenerationFunction:** This function generates the corresponding signature for a hashed message.
  - **ISignatureVerificationFunction:** This function verifies the corresponding signature for a hashed message.
  - **IHashGenerationFunction:** This function computes the hashing for a block of data, based on MAC key, key length, data and data_length.
  - **IHashVerificationFunction:** This function verifies hash.
  - **IEncryptFunction:** This function encrypts data based on cipher algorithm, user key, key length, and so on to generate the cipher text from a plaintext.

- **IDecryptFunction:** This function decrypts data based on cipher algorithm, user key length, and so on to generate the plaintext from a cipher text.
- **IBase64EncodeFunction:** This function encodes data in base 64. This will be a way to protect data from human-readable format, and when.
- **IBase64DecodeFunction:** This function decodes data in base 64.

- **Required interfaces:** *IVariable.*

2.3.3.16 UserAccountManager

- **Functionality:**
  - Provides services to manage user accounts.
- **Provided interfaces:**
  - **IUserManagementFunction:**
    - **Creation:** This function generates a unique identifier for the user account being created, and adds all parameters required for user profile, such as, first name, surname, email, user manager, role, current password, previous password etc. A certain number of old passwords will be associated with a user account to verify passwords are changed properly.
    - **Deletion:** it deletes a user account, if user is authorized.
    - **Update:** it modifies user parameters, if user is authorized. For example, password shall be able to be changed.
  - **IPasswordCheckFunction:** This function checks if password satisfies password policy, for example, alpha-numeric characters, long, and so on, during the creation of the password.
  - **IPrivilegesSettingFunction:** This function based on user role assigns certain privileges or permissions, applying least privileges philosophy.
  - **IKeyManagementFunction:** This function deals with assigning a key to the created user, modifying or deleting it.
- **Required interfaces:** *IVariable, IFile* and a subset from those in *CryptoManager component.*

2.3.3.17 SecurityMonitoringManager

- **Functionality:**
  - Authentication: Providing user, application and device authentication.
  - Authorisation based user account.
  - Session management: close session, monitoring timeouts etc.

- Monitoring FDF behaviour: at the network level, service availability, application behaviour, modification of data.
- Monitoring user and application aspects: session open, application partitioning etc.
- Monitoring transmitted information.
- Generation of reports and notification: login user history, audits, and notification of the use by alerts, emails etc. These reports will be fully configurable for any FDF.
- Prevents execution of untrusted code.
- Modification in configuration files.
- Notification of an attack.

- **Provided interfaces:**
  - **IApplicationIdentificationFunction**: This function will assigned a unique ID to an application on the FDF. This ID will be used for tracing application behaviour, that is for monitoring correct operation of access to CPU and network, data modification and for notifying to a higher system or administrator when abnormal behaviour is discovered. This information can be stored in the TPM.
  - **IApplicationProfileFunction**: This function will create an application profile based on configuration files to trace application permissions.
  - **IAuthenticationVerificationFunction:** This function authenticates user based on user and password, and on the USB or smartcard containing credentials, device for example by serial number and applications based on unique identifier.
  - **ISessionManagementFunction:** This function is in charge of creating a session, locking a session if timeout and closing it.
  - **ILoginManagementFunction:** This function checks logins.
  - **IAuditEventsConfigurationFunction:** This function enables the configuration of audit events like login, timestamps, audit trail, information for non-repudation, modification, deletion, user, location, etc.
  - **IAuditReportingConfigurationFunction:** This function enables the configuration of the audit events defined for reporting. All reports will provide timestamps based on system time, and additional information considered relevant, user location etc. This information shall be encrypted and store in a secure way but means of the CrytoManager.
  - **IUserLoginReportingFunction:** This function enables to list all user accounts and login history.
  - **IGetReportFunction:** this function enables to get a report, only authorized users shall get this information. This information will be protected by encryption, digital signature, digital message reports and timestamps.

- **INetworkMonitoringFunction:** This function shall check all related network issues: insertion of packages, data flooding, loss of communication, replay of messages, messages to provoke a DoS attack.
- **IMonitoringunction:** This function will monitor deletion or insertion of configuration data or detection of insertion of malicious code, very critical.
- **IUseNotificationunction:** This function will inform administrator about user access and actions performed.
- **IAttackNotificationFunction:** This function shall be used in case of determination of a possible attack: access to CPU, modification of configuration files during execution or not. This communication can be done by means of email, text messages or any other means.
- **IIncidentSupportConfigurationFunction:** This function will enable the configuration of automated incident notification services to whom corresponds (user or system).
- **IIncidentNotificationFunction:** This function will notify to an authorized user or system about an incident. This can be made by e-mails, text messages, or any other means configured before.
- **IPasswordExpirationNotificationFunction:** this function shall notify user to modify password after a period of time defined by an administrator.
- **IPasswordStrenghEnforcementFunction:** This function shall guarantee that criteria defined for strength: minimum length, use of upper/lower cases, non-alpha characters etc. In FSA-AC-2.18 it is set a minimum of 6 characters for passwords.
- **IAdministratorAccessVerificationFunction:** This function will notify administrator for getting approval of a user access. This is needed to fulfil FSA-AC-1.2 Dual Approval Access. The result will be encrypted.
- **IMulticastTransmissionVerificationFunction:** This function will verify the source and integrity of the transmissions.
- **IMulticastTransmissionHandlingFunction:** This function will register authorized applications to subscribe to multicast transmission, and authorized applications enabled to send multicast transmissions.
- **IErrorHandlingFunction:** This function will handle error conditions without providing information that could be exploited by adversaries.
- **IBlacklistingCreationFunction:** This function will be used for creating blacklists to protect FDF against executable code. Administrator can use either black lists or white lists.
- **IWhitelistingCreationFunction:** This function will be used for creating whitelists to protect FDF against executable code.

- **ICommunicationVerificationFunction:** This function will check a loss in the communication for inputs/outputs or any other transmitted message to be applied upon loss of communications.
- **IBackupCreationFunction:** This function will create a backup for recovering the system either as a result of an attack or for any other reason like a failure. This backup will be at the user level and system level. Only authorized entities will be able to create it and it will be saved in the TPM.
- **IFDFRecoveryFunction:** This function will restore the system by means of a secure backup after a disruption or failure in the system.

- **Required interfaces:** Several interfaces of VariableManager, FileManager, MonitoringManager, ExecutionManager, FunctionManager, NetworkManager, IOManager, CryptoManager, TimeManager and HealthManager.

2.3.4 *FDF Detailed structural view*

The following picture shows the full usage of the FDF from the dataflow perspective. Above the Framework API, there are shown the applications which could belong to different SIL Levels. They have access to the variable stores and also the message store. From there the Message Manager passes the composed messages on to the Communication Manager or vice versa receives composed messages. The Communication Manager is responsible for the protocol handling. The different paths for deterministic and non-deterministic data are depicted.

The Network Manager offers a common socket type layer giving either access to the TSN ports (deterministic/time-sensitive data) or the socket interface of, e.g. the TRDP protocol for non-deterministic data. The non-deterministic data can either be sent to UDP (PD/MD) or TCP/IP for MD. The Network Manager is also responsible for de-encryption as well as authentication using, for example, IPsec or MACsec. The Configuration Manager passes the configuration on to the respective managers whereas an IO Manager offers transparent access to Local I/O. The File Manager provides a File API. In case of reception of SDT (Safe Data Transmission) secured messages, the Message Manager evaluates them or calculates and adds for sending the correct SDT fields.

Besides, the blocks inside the Communication Manager show the different handling of the typical communication patterns such as cyclic PD (Process Data) push or MD (Message Data) pull. In parallel, there is the need for a Hard Real-Time Data Handler for the TSN traffic making use of the Time Manager and its PTP protocol.

Finally, the Topology Manager provides the TTDB access and services as well as the DNR (Domain Name Resolver) and TTI (Train Topology Information) services.

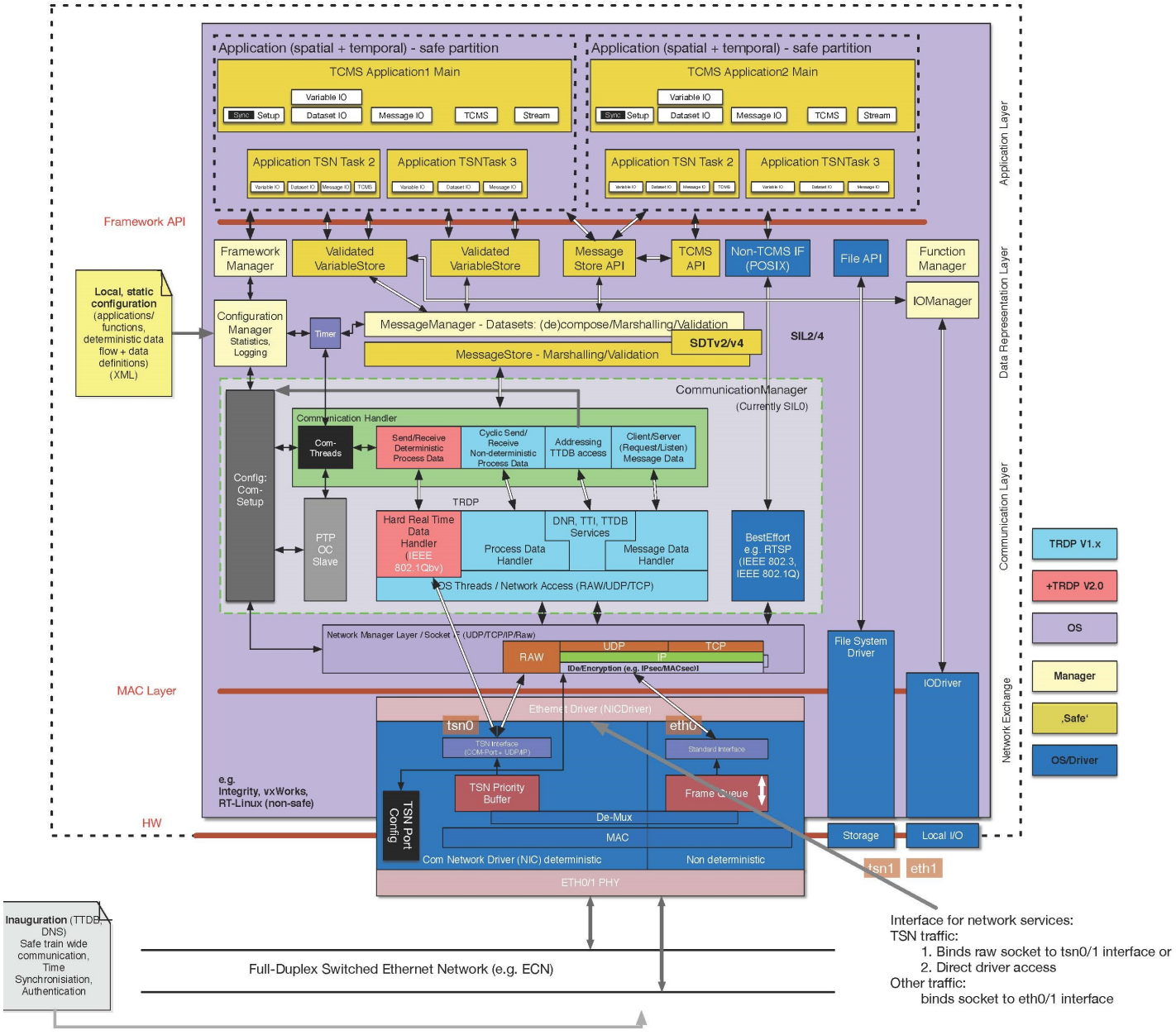

Figura 5. FDF Dataflow perspective

2.4 Behavioural view

In order to understand how the framework works, it is necessary to explain its behaviour by explaining the three different phases it runs. First of all, during the setup phase, the configuration is loaded, the shared memories are linked and the logic elements are mapped. This setup phase can be divided into two phases: Configuration phase and initialisation phase. Once these preliminary steps are completed, the regular execution phase starts, which means the Functional Distribution Framework (FDF) begins to execute cyclically.

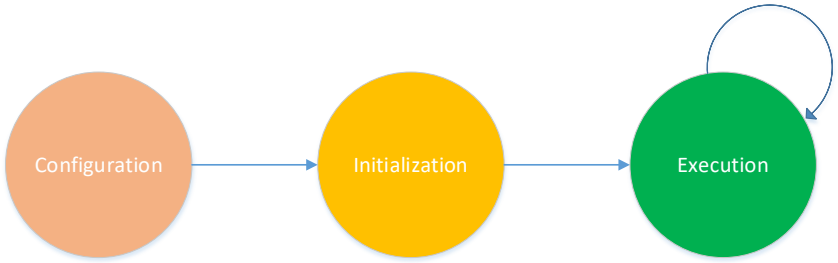

Figure 6. Behaviour of the FDF in 3 phases.

2.4.1 *Configuration phase*

The configuration phase covers the loading of the platform configuration (see Figure 7). The user application initiates the trigger to start the configuration phase. It commands the *FrameworkManager* to configure, and this Manager does so through the *ConfigurationManager*. A SIL0 process reads the configuration file and then stores it in shared memory. Then safe processes can read the configuration from this shared memory before moving to the initialisation phase.

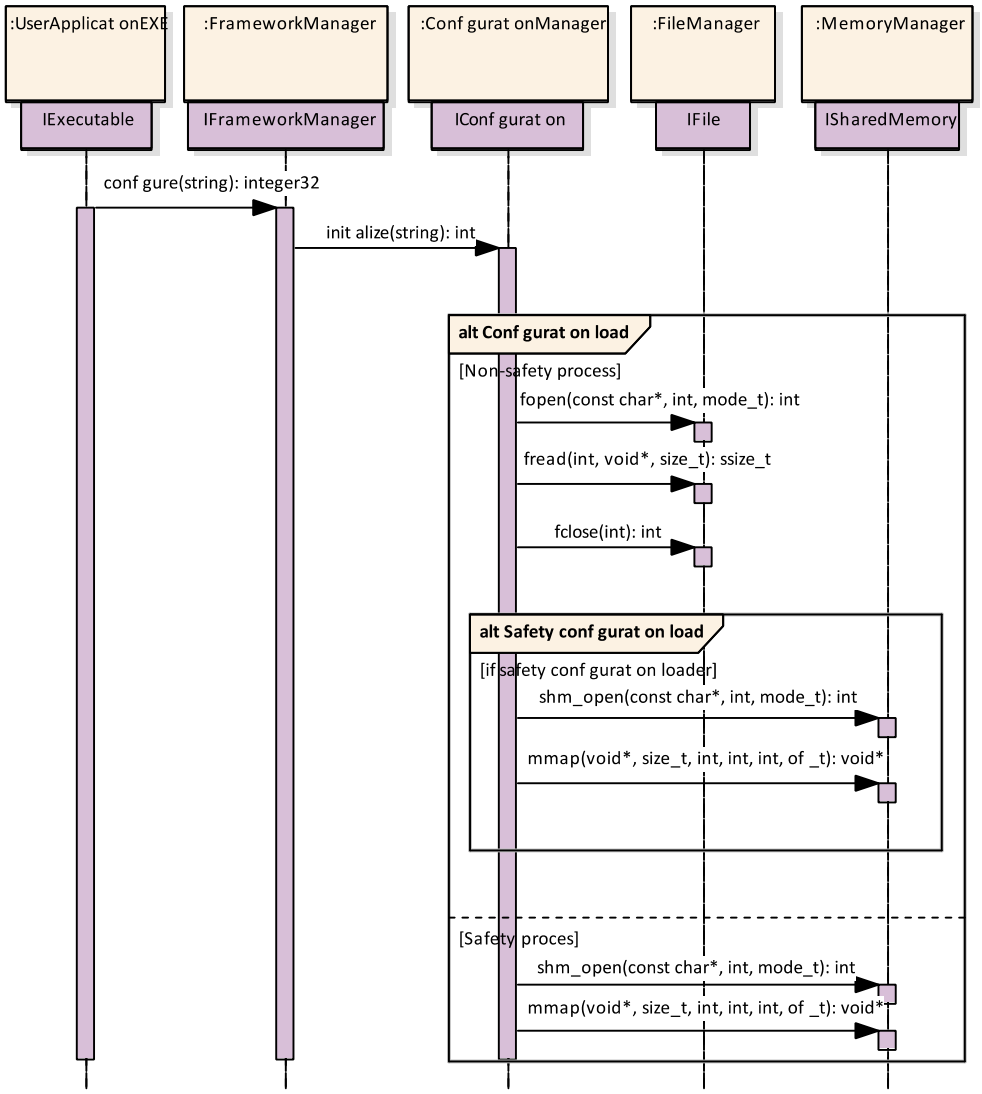

Figure 7. Configuration phase.

2.4.2 *Initialization phase*

Once the configuration has been loaded correctly, the initialisation phase begins (see Figure 8). This phase is divided into three sub-phases which deal with the driver initialisation, data initialisation and function initialisation.

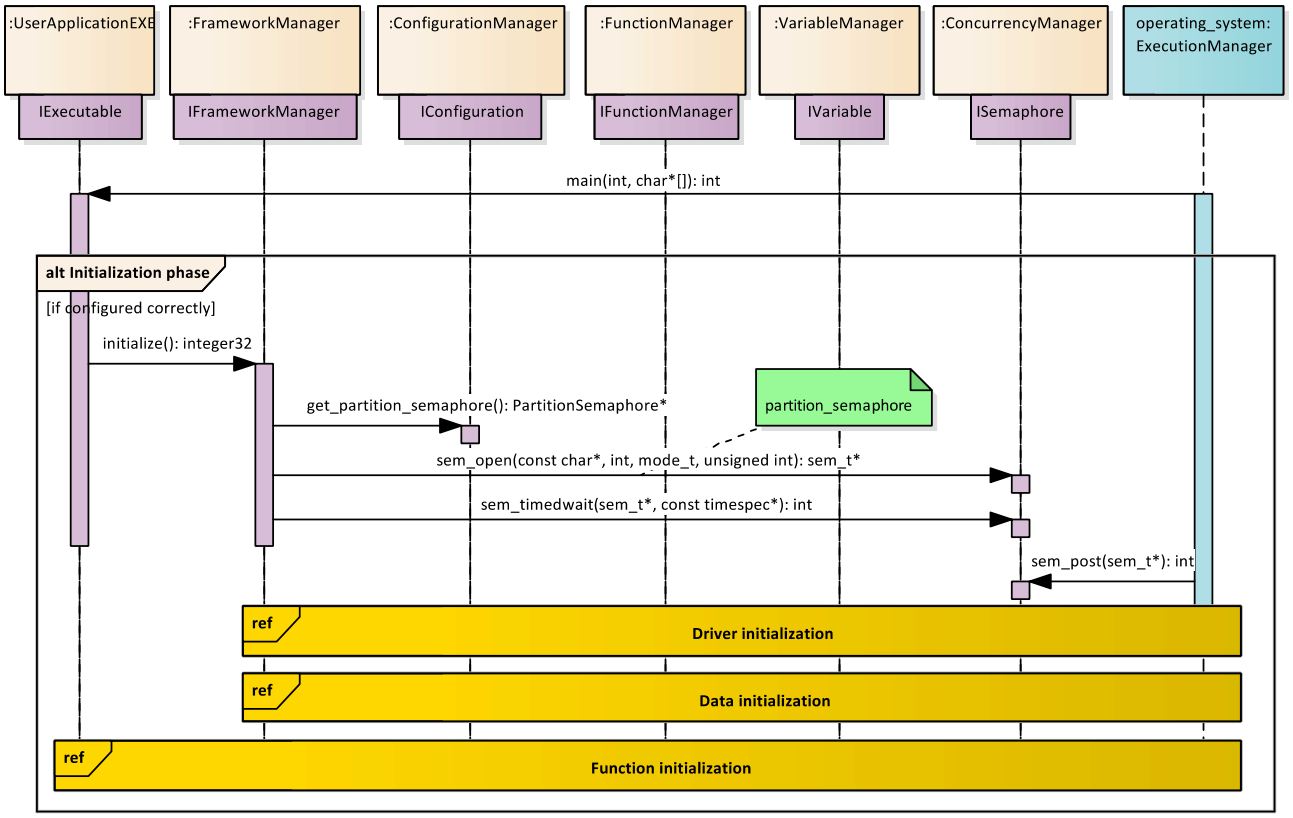

Figure 8. Initialization phase.

2.4.2.1 Driver initialisation

This section explains how the initialisation of the different driver's is made (see Figure 9). This step includes the initialisation of IO and NIC drivers and that for the watchdog and ECU too.

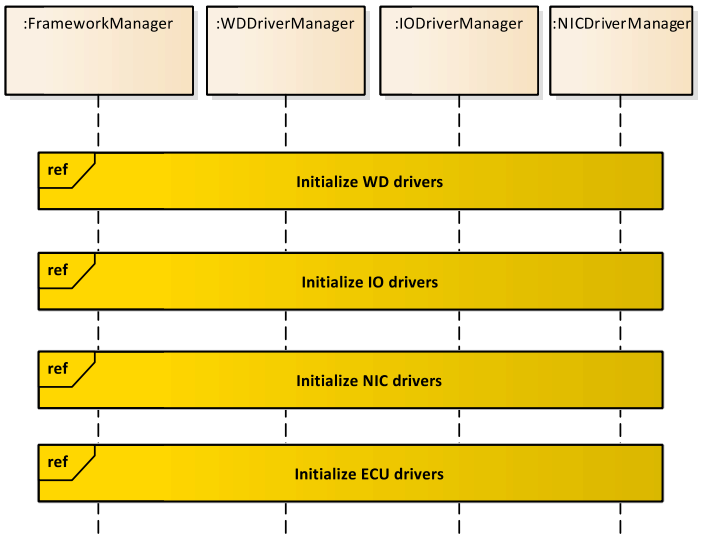

Figure 9. Driver initialisation.

Firstly the IO drivers are initialised. Digital IO and analog input drivers can be found in this set of drivers. As can be seen in Figure 10, one by one, the *FrameworkManager* retrieves the necessary information from the *ConfigurationManager* and then commands the corresponding driver, encapsulated under the *IODriverManager* component, to initialise with this piece of configuration by the use of its concrete interface.

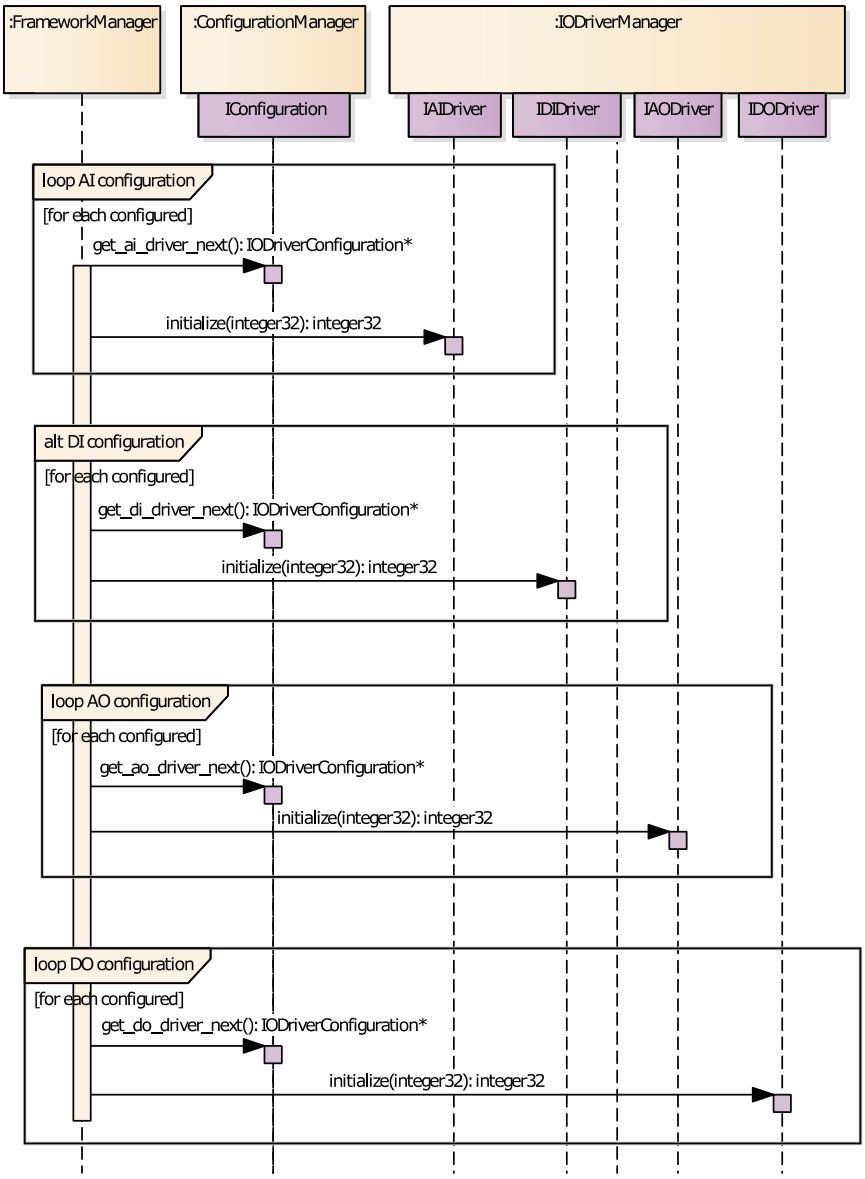

Figure 10. Initialization of IO drivers.

The same applies to the initialisation of the *NIC* driver (see Figure 11). Once the necessary configuration data is retrieved, the *FrameworkManager* orders to initialise the driver thorough the *IRTDriver* interface.

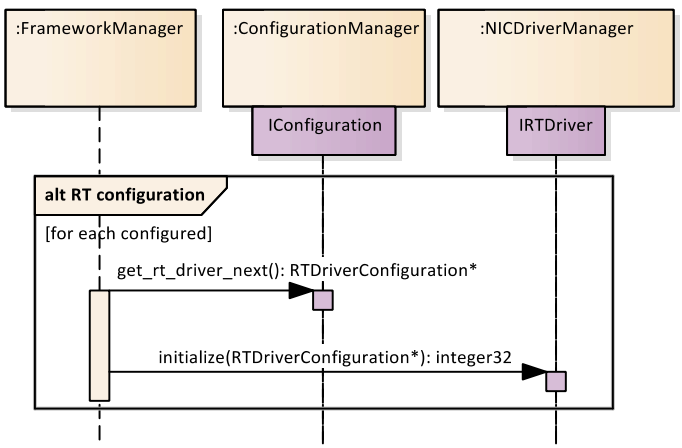

Figure 11. Initialization of NIC driver.

In the case of the Watchdog driver, WDDriverManager's IWDDriver is the interface used to initialise the Watchdog, as can be seen in Figure 12.

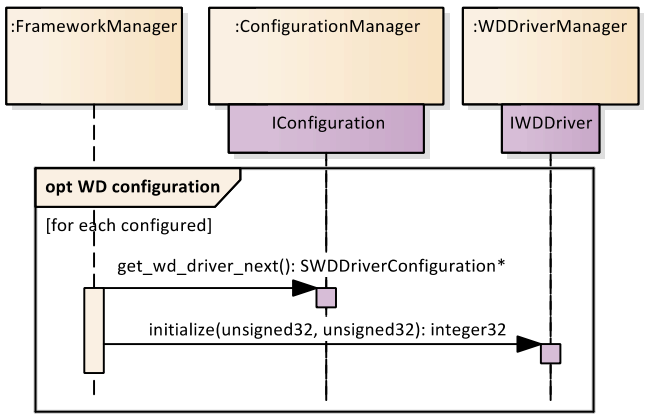

Figure 12. Initialization of watchdog drivers.

Finally, the interface used for the initialisation of ECU Driver is IECUDriver, as shown in the figure below.

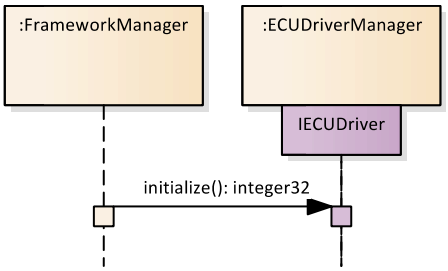

Figure 13. Initialization of ECU driver.

2.4.2.2 Data initialisation

Data needs to be initialised too before starting with the cyclic execution. With data variables, messages, topology and any other data structure with a given semantic are understood (see Figure 14).

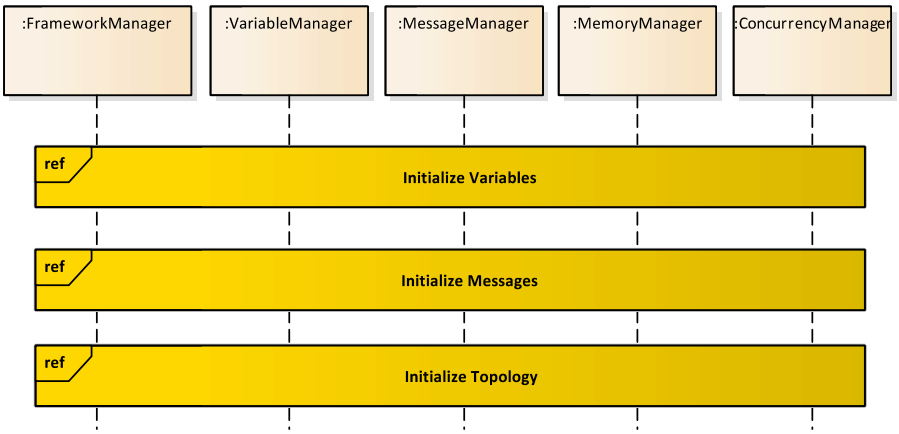

Figure 14. Data initialisation phase.

When initialising the messages, firstly the corresponding configuration is retrieved for every existing message store. With this information, the *FrameworkManager* by the use of *shm_open* function opens a shared memory through the *ISharedMemory* interface. If this memory is mirrored, then the mirrored shared memory is also opened. On the other hand, as shown in

Figure 15, "mmap" function is executed in the case that inter-process or if shared memory mirror identifier is not empty. Further, mutexes are initialised a shared whenever concurrent access occurs.

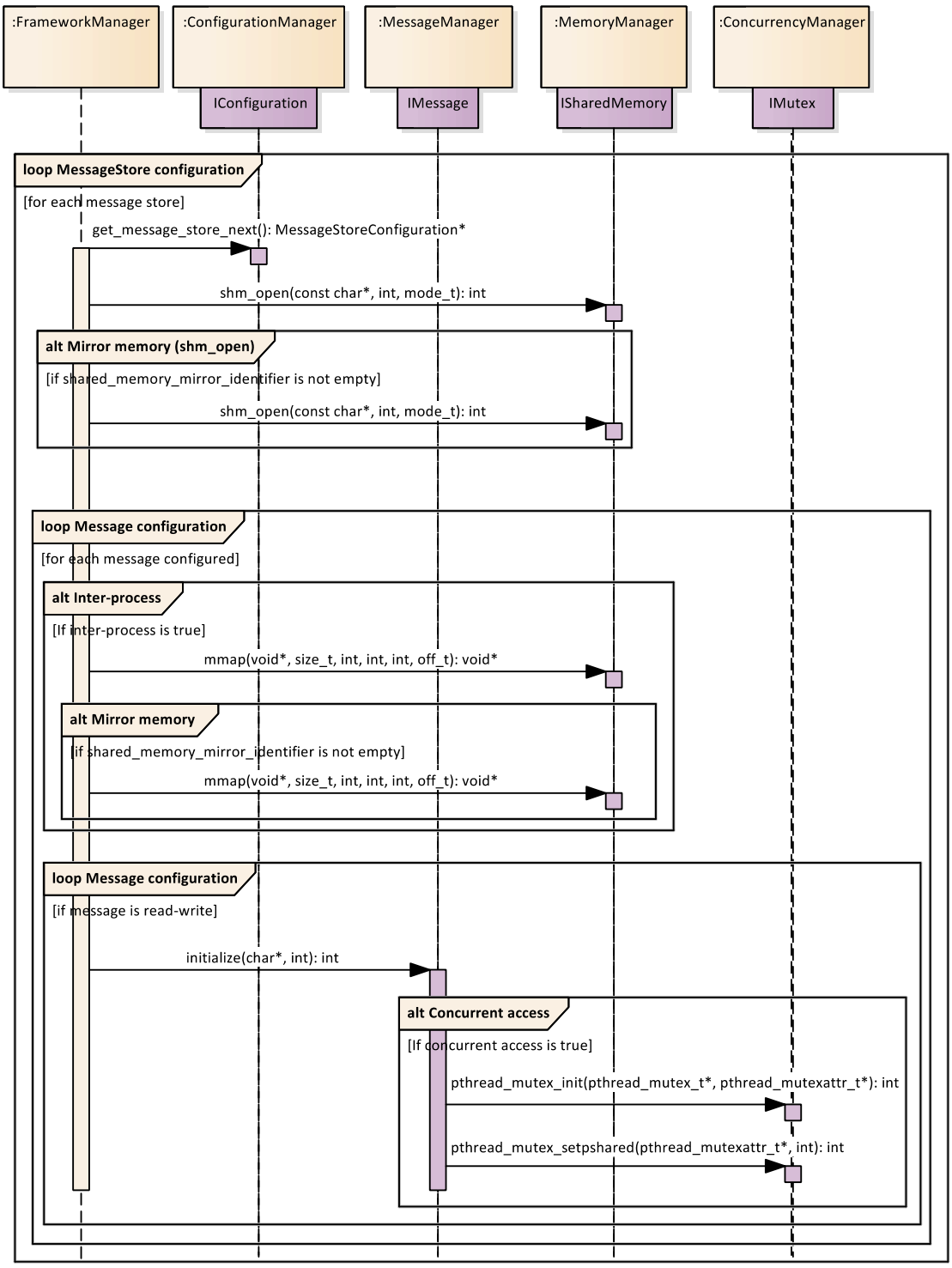

Figure 15. Initialization of Messages.

Similarly, when initialising variables, firstly, firstly the corresponding configuration is retrieved for every existing variable store. With this information, the *FrameworkManager* by the use of *shm_open* function opens a shared memory through the *ISharedMemory* interface and performs the mapping of the memory using the *mmap* function. In the case that the memory is mirrored, then the mirrored shared memory is also opened and mapped. In addition, for each variable configured and if the variable is intended to be read or write, the loop message

initialisation is carried out, initialising and configuring mutexes whenever concurrent access occurs.

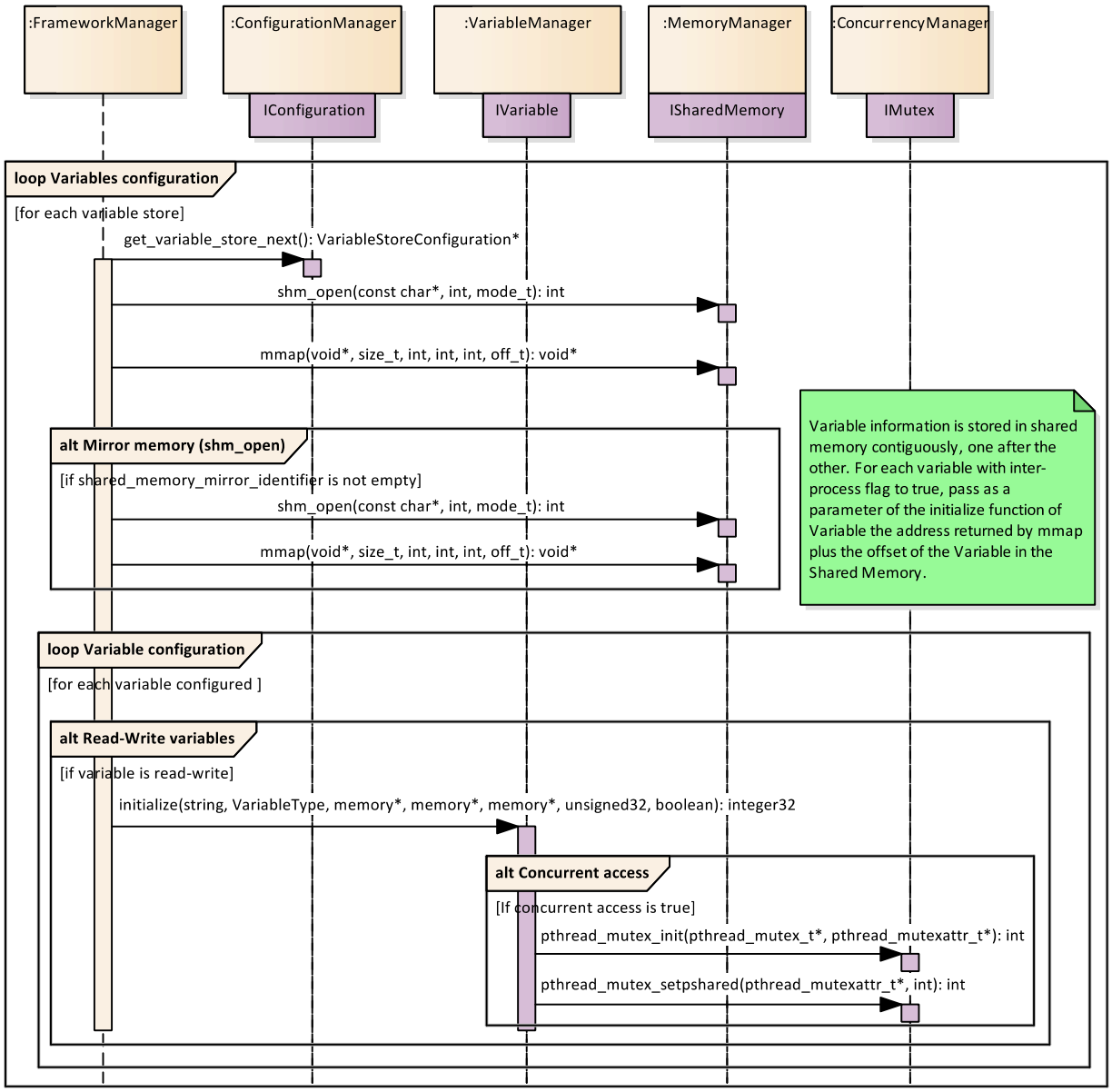

Figure 16. Initialization of Variables.

Figure 17 shows the initialisation of the topology manager that retrieves the corresponding configuration for every existing topology store. Then, the *FrameworkManager* opens the shared memory using the *shm_open* function and maps the topology object to the shared memory address.

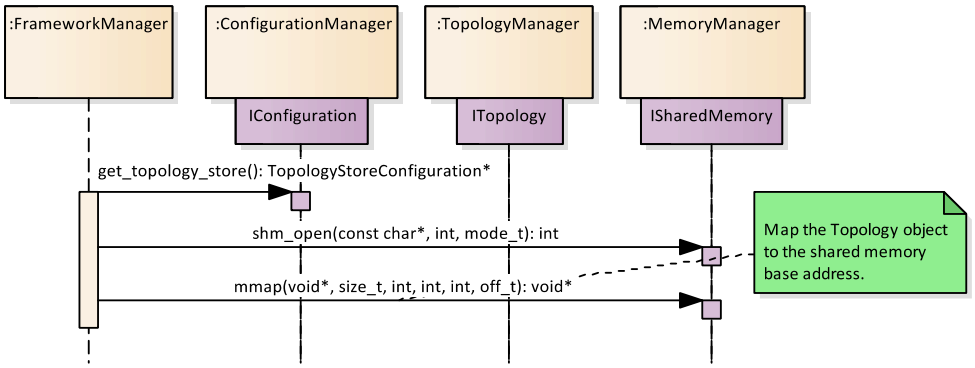

Figure 17. Initialization of Topology objects.

2.4.2.3 Function initialisation

The missing initialisation task is responsible for the setup of functions (see Figure 18). The *FunctionSchedule* is retrieved, which contains each of the existing function in this concrete instance and then the *FunctionManager* is initialised. After this point, every function is initialized by its concrete configuration and once it is ready, it is registered on the *FunctionManager* so that it will be able to execute it.

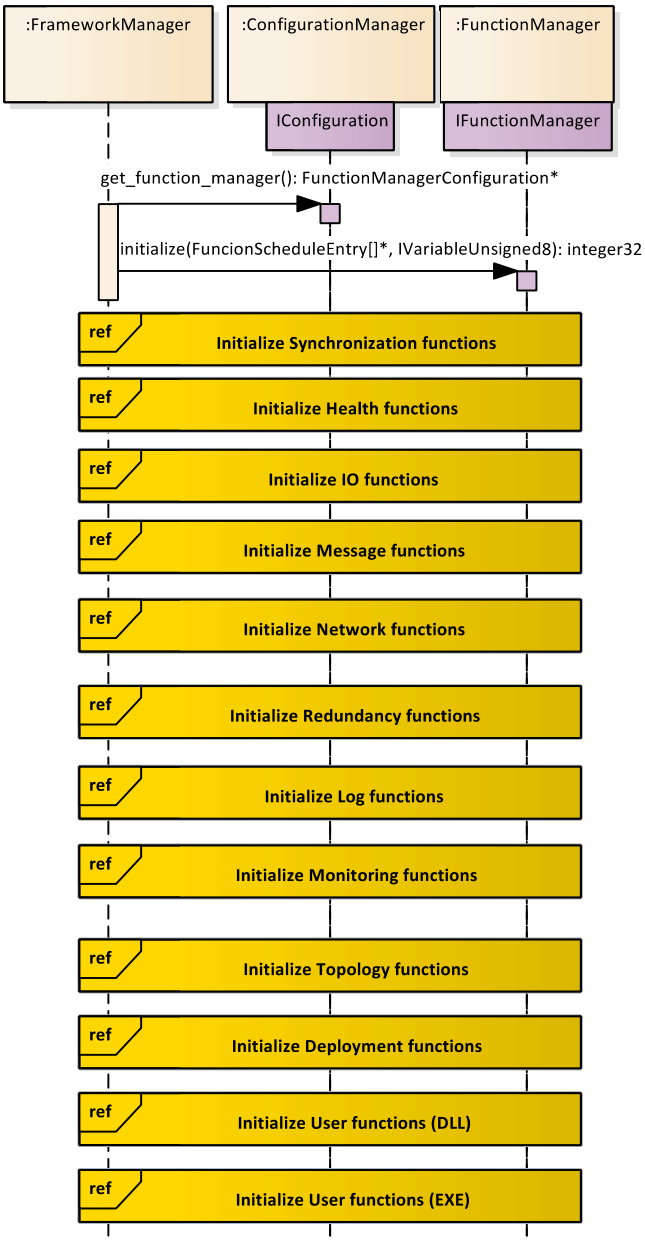

Figure 18. Function initialisation phase.

As mentioned before, if a *synchronisation* function has been configured (see Figure 19), its corresponding configuration is loaded and the *ServiceFunction* is initialized. When this step is ready, the function is registered in the *FunctionManager* and this last gets the unique identifier of the registered function through the *ISynchronizationFunction* interface of the *SychronizationManager* component.

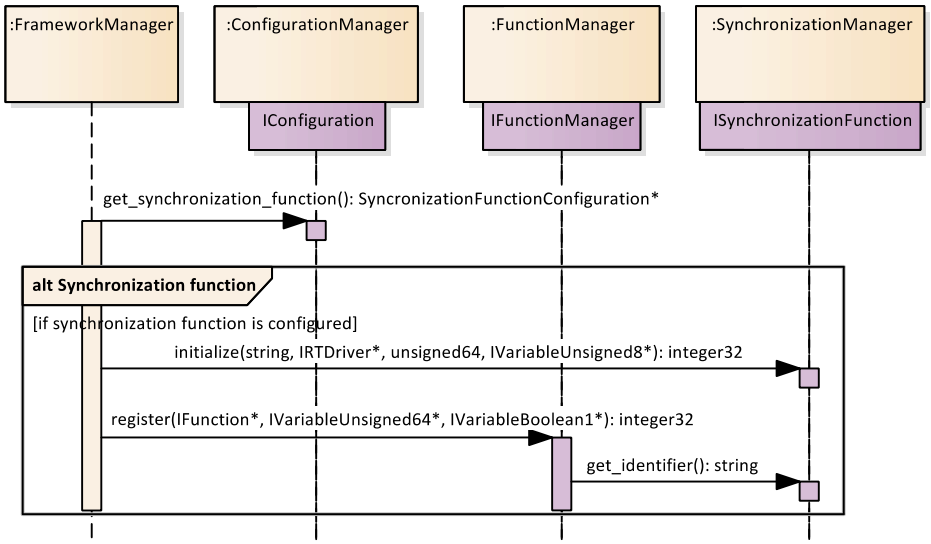

Figure 19. Initialization of Synchronization functions.

The initialisation of the health monitoring firstly configures the watchdog timer and then *ServiceFunction* is initialised (see Figure 20). When this step is completed, the function is registered in the *FunctionManager* and this last gets the unique identifier of the registered function through the *IWDFunction* interface of the *HealthManager* component. Finally, the *FrameworkManager* initialises the *ITemperatureFunction* interface and registers the temperature function in the *FunctionManager*.

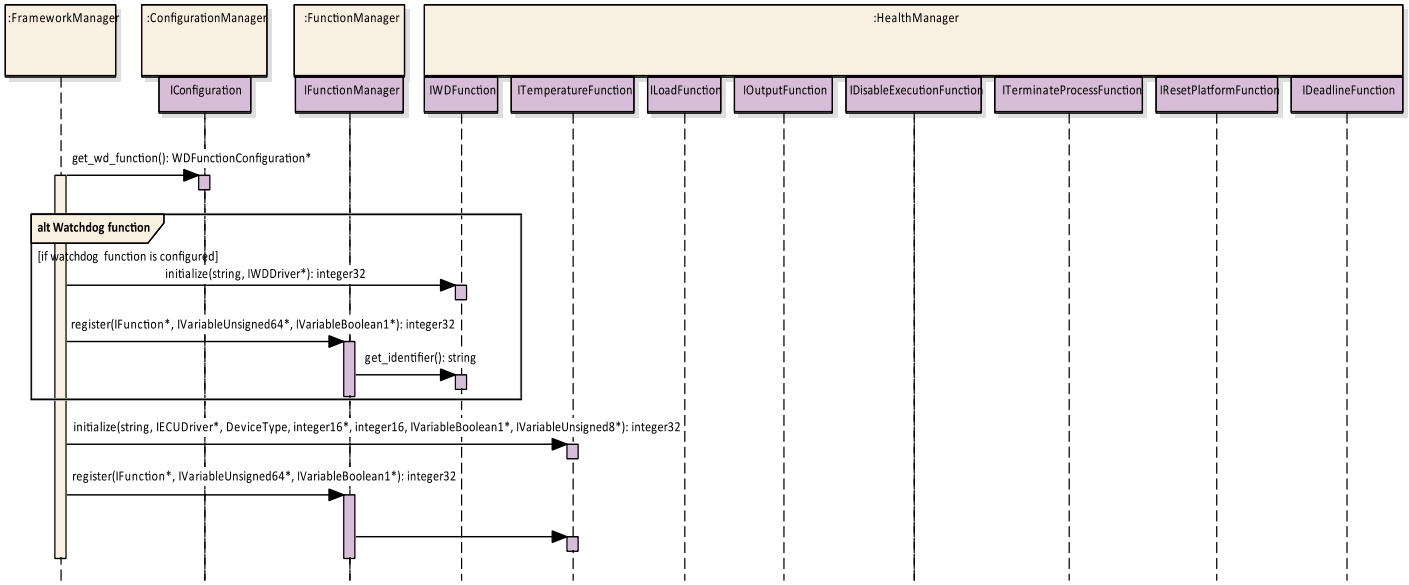

Figure 20. Initialization of Health functions.

As mentioned before in this document, the digital IO and analog input drivers shall also be initialized. To that end, the IO interface is configured at first instance, followed by the initialisation of the *ServiceFunction* (see Figure 21). After that, the function is registered in the *FunctionManager*, obtaining a unique identifier of the registered function through the *IOFunction* interface of the *IOManager* (digital or analog IO).

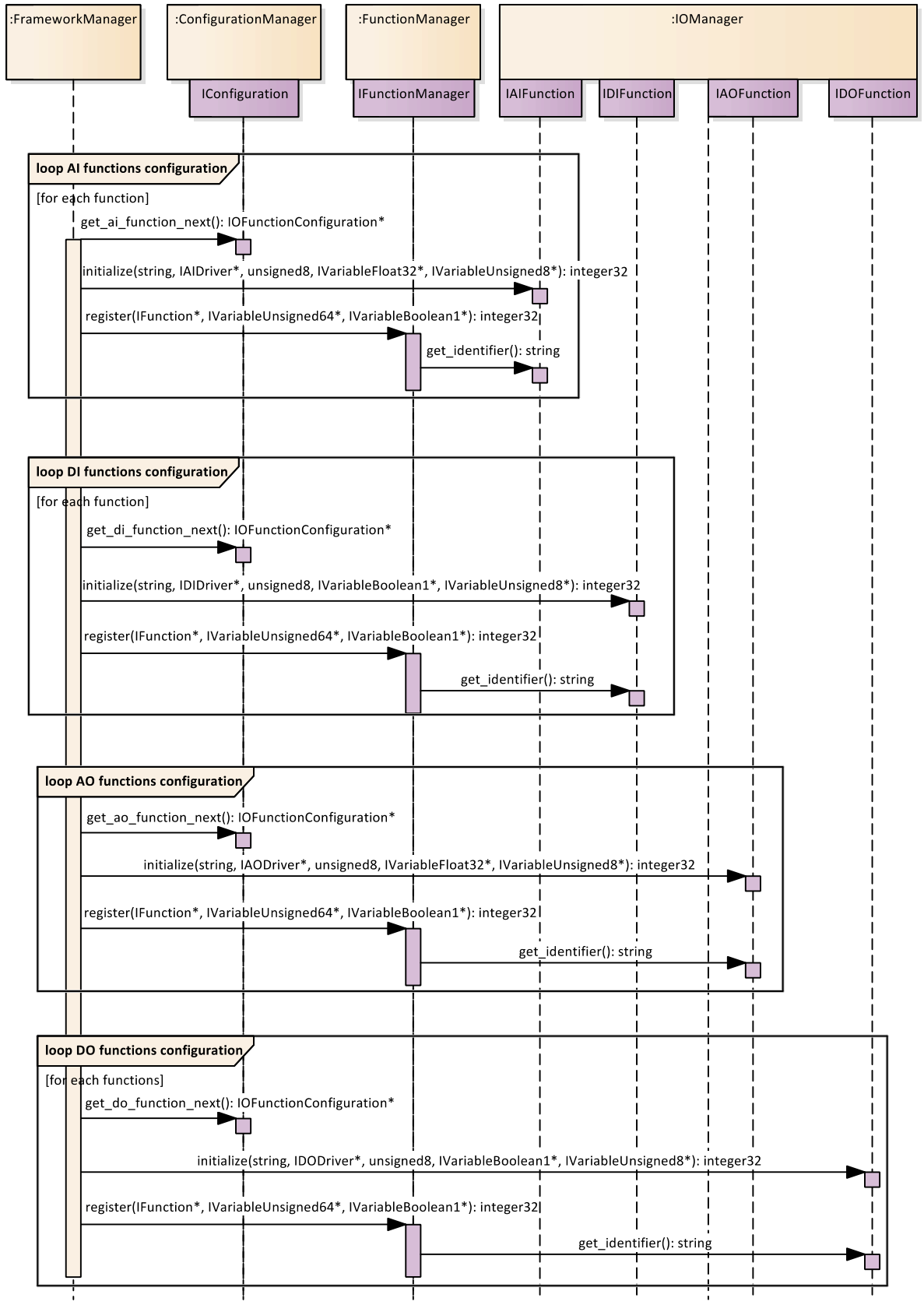

Figure 21. Initialization of IOFunctions.

The message function initialisation, first, the process retrieves the configuration of every parse function stored (see Figure 22). With this information, the *FrameworkManager* registers the

message function in the *FunctionManager*, obtaining a unique identifier of the registered function through the *IParseFunction* interface of the *MessageManager*.

Afterwards, the message function initialisation regains the configuration of the compose functions stored in the shared memory. This information is used by the *FrameworkManager* to register the compose function in the *FunctionManager* and to obtain a unique identifier through the *IComposeFunction* interface.

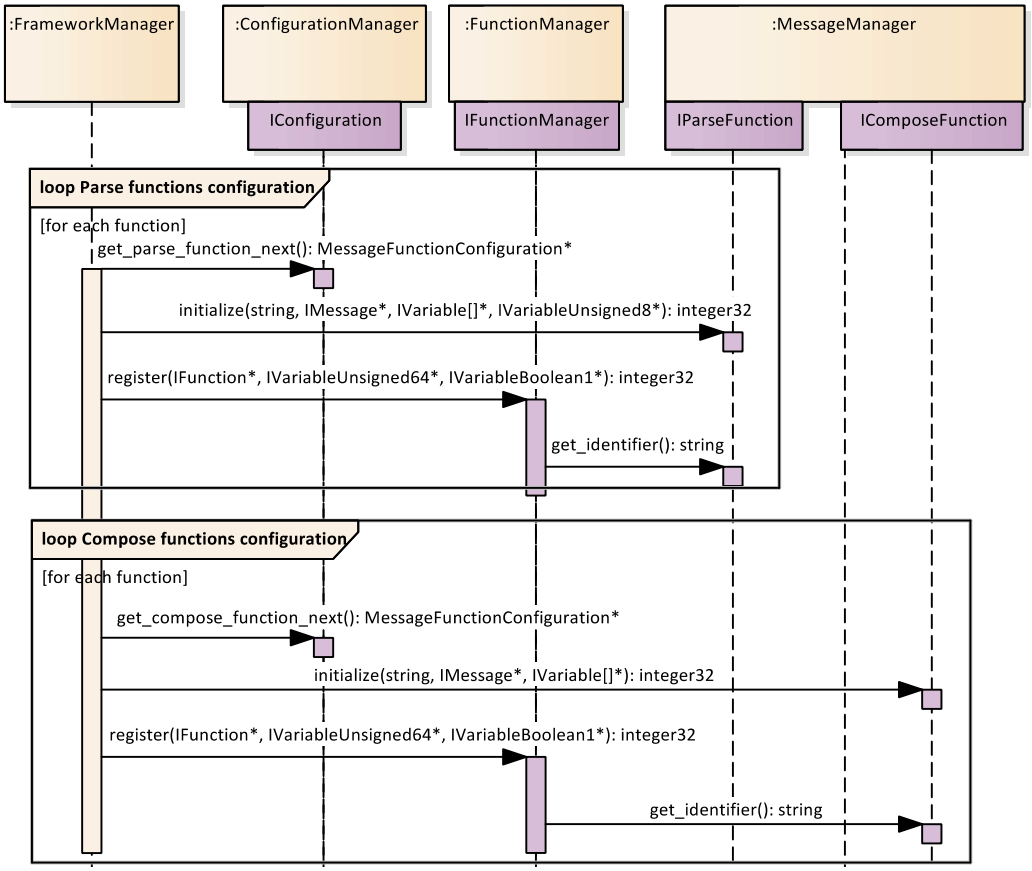

Figure 22. Initialization of MessageFunctions.

Figure 23 shows the initialisation process of the reception and transmission network functions. First of all, the configuration of those functions is retrieved and the socket function is called through the *SocketManager*. In the case of the receive function, a bind function is also executed. Then, the *FrameworkManager* uses the retrieved configuration of both functions to register them in the *FunctionManager*. Also, unique identifiers for those two functions are returned to the *IReceiveFunctions* and *ISendFunctions*.

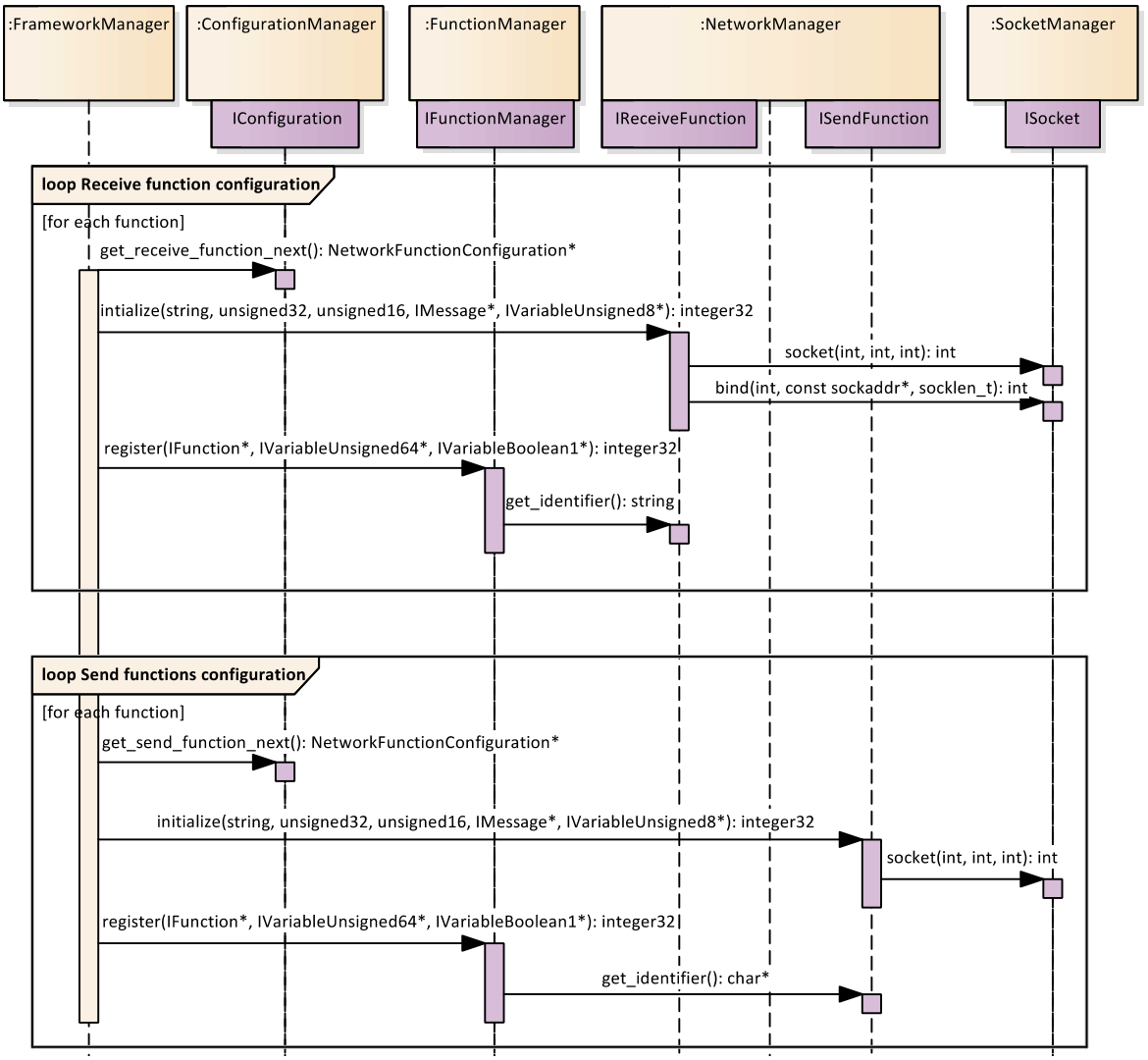

Figure 23. Initialization of NetworkFunctions.

The initialisation of the redundancy functions, first, retrieves the configuration of the redundancy function stored in the shared memory. Then, the function is registered in the *FunctionManager* by the *FrameworkManager* using obtained configuration, and the IRedundancyFunction obtains a unique identifier of the registered f*unction* (see Figure 24).

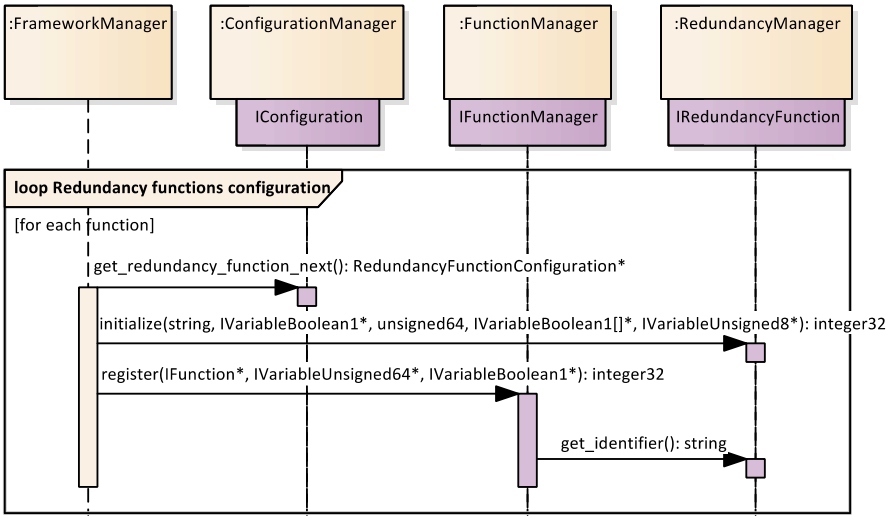

Figure 24. Initialization of RedundancyFunctions.

In this case, as shown in Figure 25, the initialization process of log functions includes different configurations of the logging interfaces. First, the configuration of the log configuration is retrieved. This configuration is used by the *FrameworkManager* to initialise the ILog interface. Then, the *mutex* interface provided by the *ConcurrencyManager* is initialized and configured.

On the other hand, the configuration of the log functions is also retrieved; the function is initialized and registered in the *FunctionManager*. Afterwards, a unique identifier is asked by the *FunctionManager* and provided to the *ILogFunction* interface.

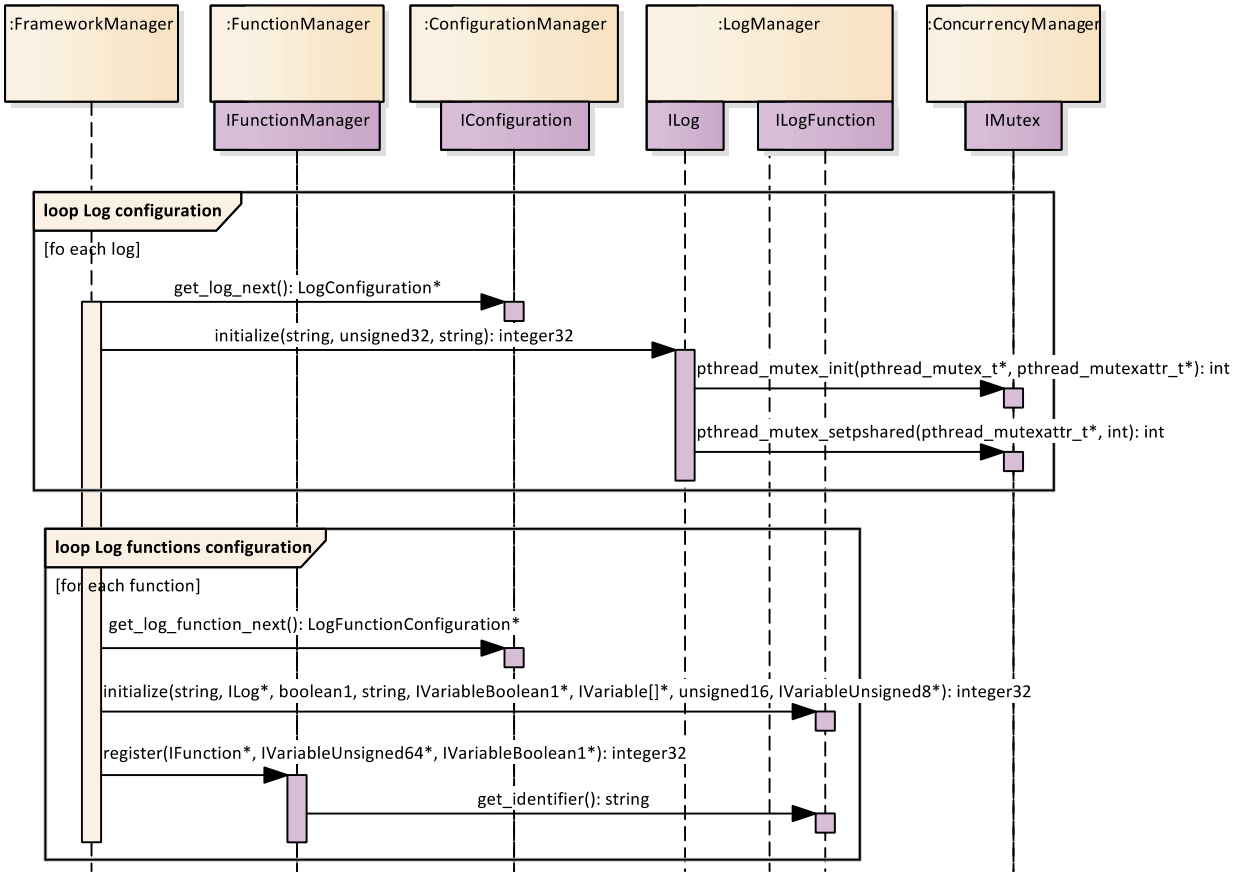

Figure 25. Initialization of LogFunctions.

Initializing monitoring functions consist of retrieving its configuration by the *FrameworkManager* and initialise the monitoring functions once it is configured (see Figure 26). Then, the socket function, bind and listen to the MonitoringFunction interface calls functions provided by the SocketManager. After configuring the sockets and using the configurations retrieved, the monitoring function is registered in the *FunctionManager,* and a unique identifier is obtained for the *MonitoringFunction* interface.

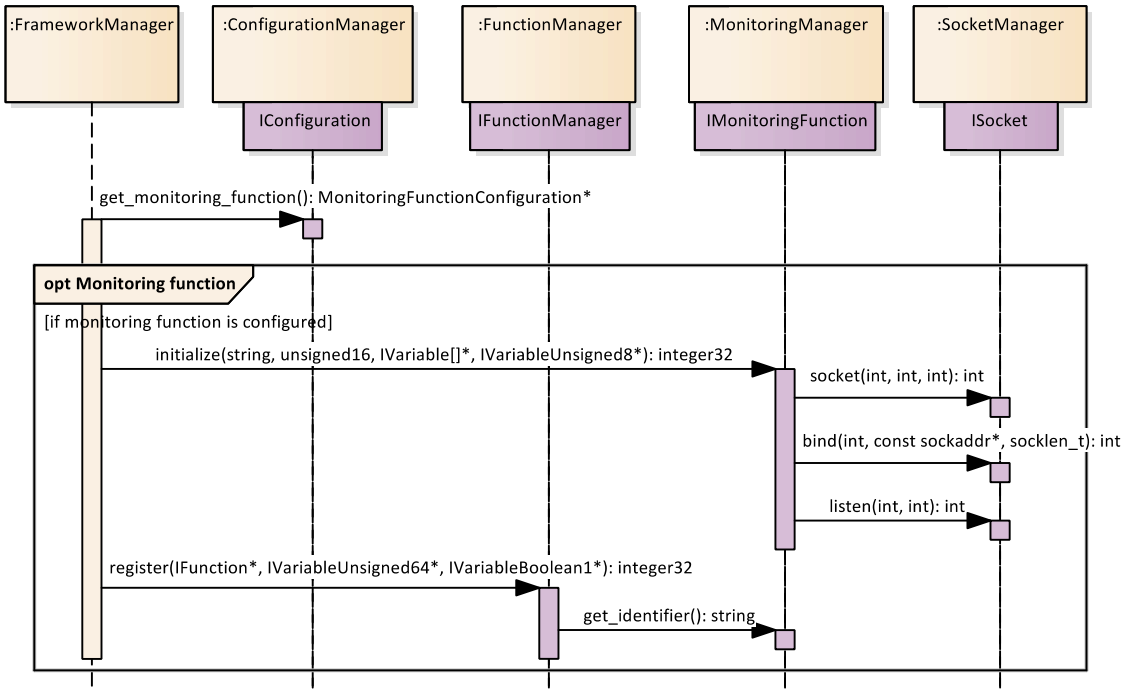

Figure 26. Initialization of MonitoringFunctions.

The last initialisation regards to the topology functions (see Figure 27). First of all, the topology's configuration is retrieved and used to configure topology function. Then, the topology of the socket is switched to listener mode (UDP), the *bind* function is called, and the socket's topology is again switched to client mode (TCP).

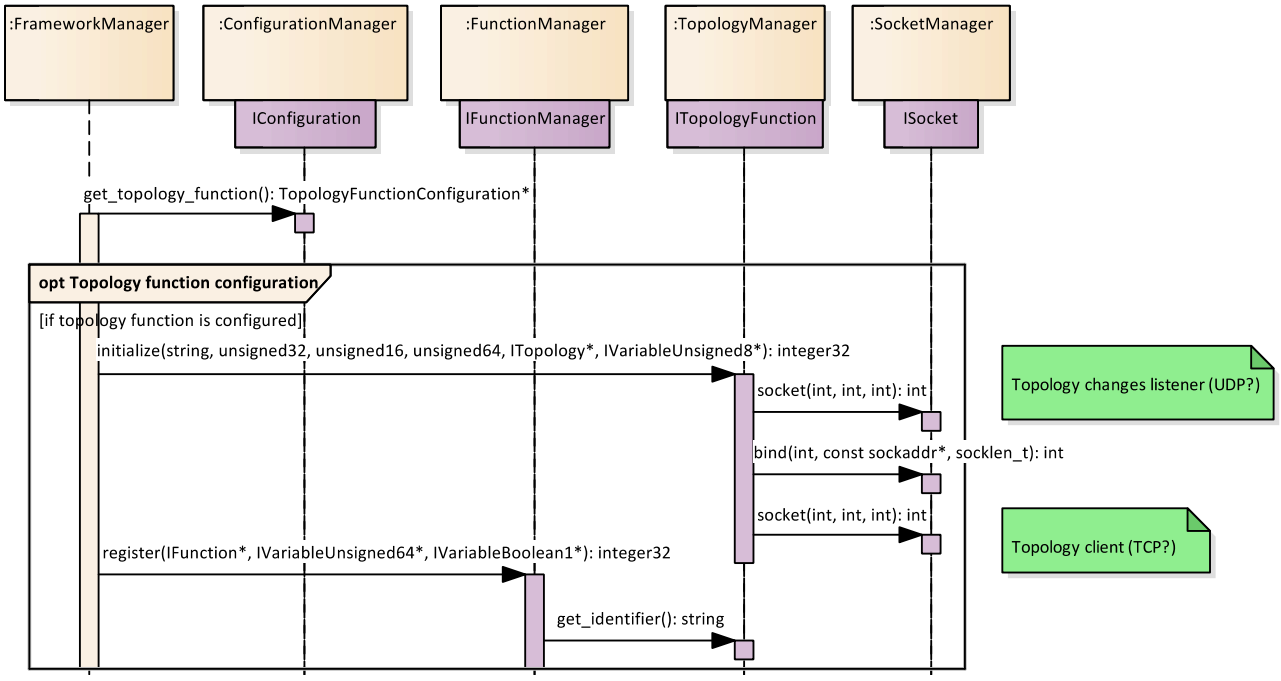

Figure 27. Initialization of TopologyFunctions.

The Deployment Function initialisation consists in getting first the necessary configuration. Once this is retrieved, the initialise function makes sockets be used to listen to traffic coming from the SFTP server.

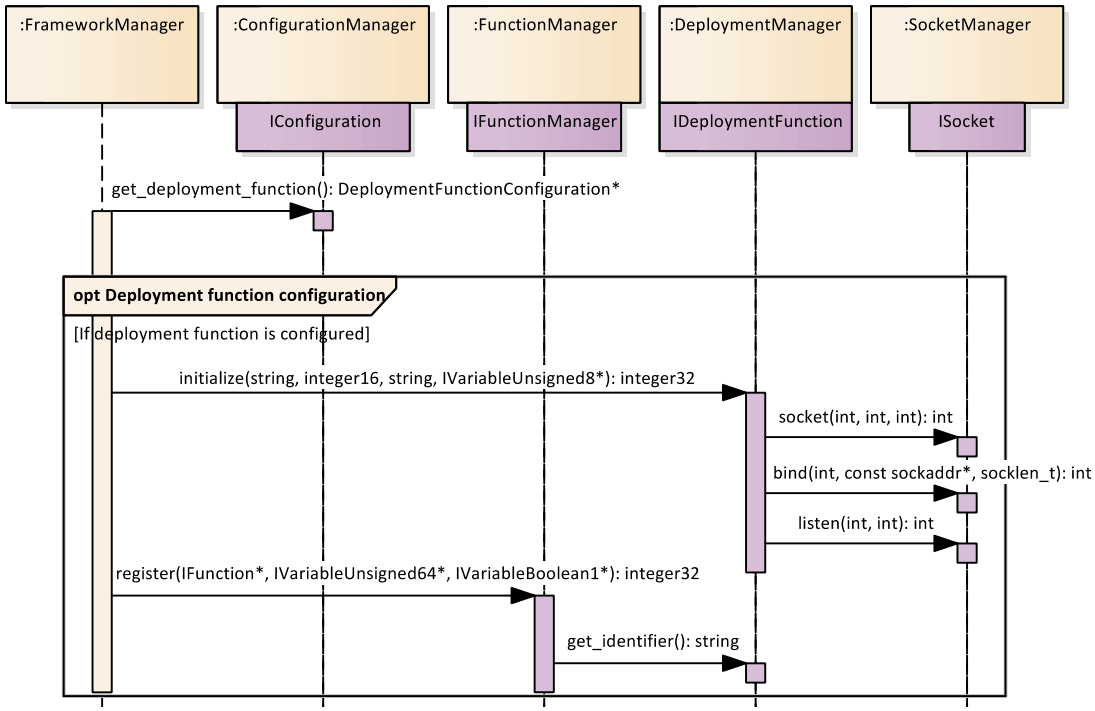

Figure 28. Initialization of Deployment functions.

The user function initialisation process, first of all, collects the variables configuration stored in the shared memory and then, restores also the topology-related configuration (see Figure 29). Once, those processes are completed, the user application interface is initialised and registers in the *FrameworkManager* and the *FunctionManager*, resulting in a unique identifier for the *ApplicationFunction* interface.

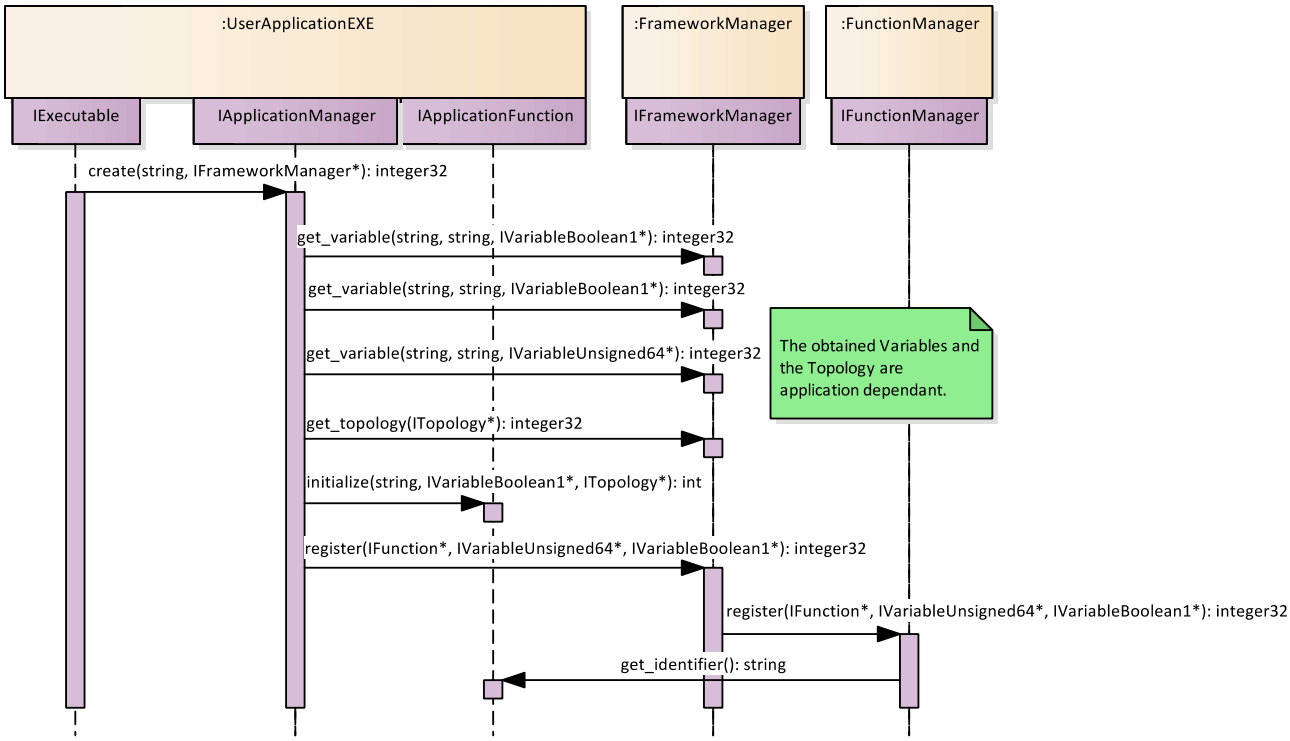

Figure 29. Initialization of UserFunctions (EXE).

The initialisation of DLL based user functions is almost the same as that with EXE files, except for the fact that the DLL libraries are opened before and closed after the initialisation and registering of the functions.

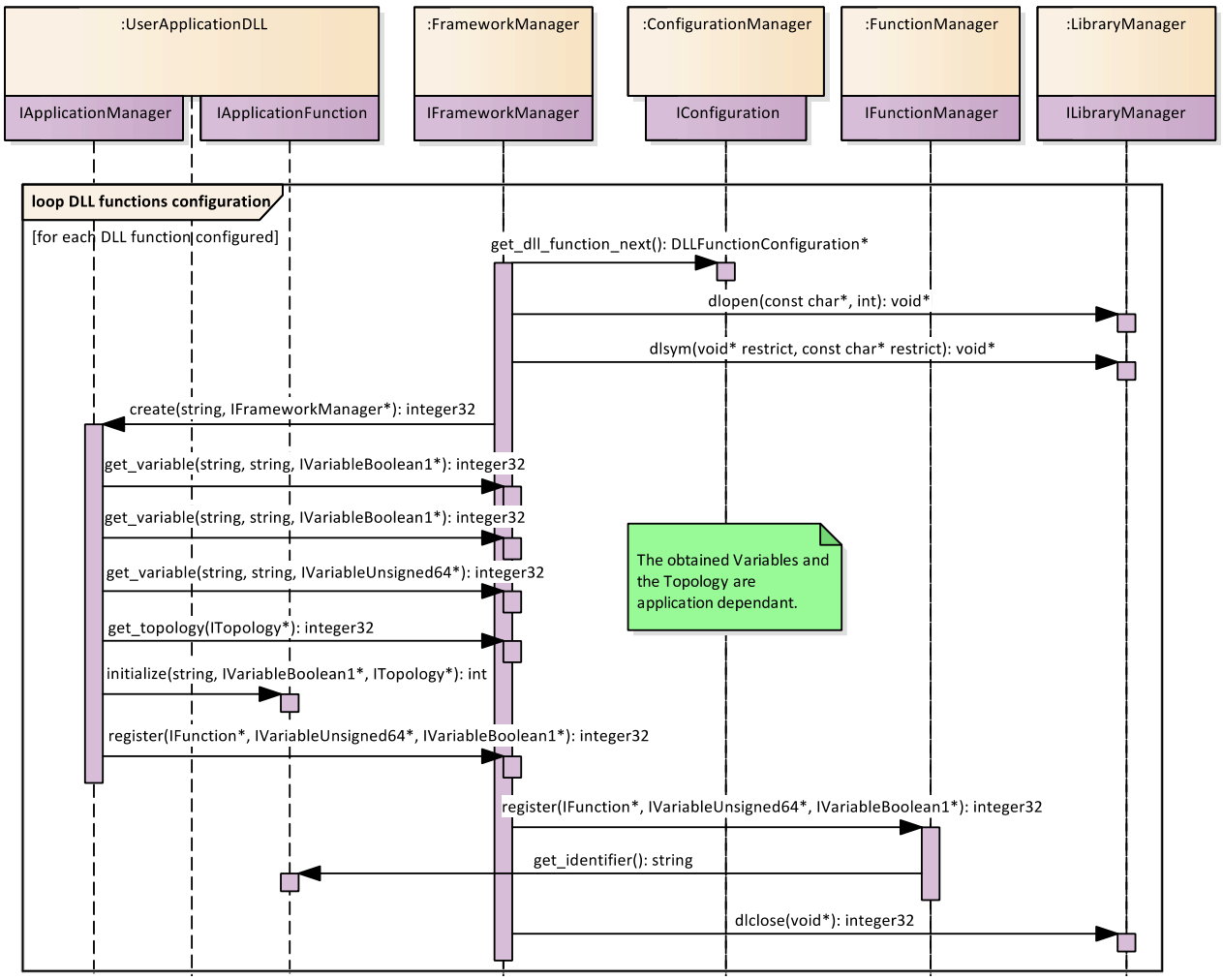

Figure 30. Initialization of UserFunctions (DLL).

2.4.3 *Execution phase*

Once the initialisation phase is over, the cyclic execution begins. In order to explain how the FDF behaves in this phase, a set of use cases has been selected. Before that, the general execution behaviour also depicted in Figure 31 is introduced in the following lines:

- User application executed the *FrameworkManager*
- The *FrameworkManager* executed a loop cycle where:
    - Configures the wait time of *ISemaphore* interface and posts the semaphore from the *ExecutionManager*.
    - Retrieves the values from the *VariableBoolean1* interface. If the value returned is "False" the system will not continue executing, else, if the value is "True", the system will enter to process execution flag
- In the case, that value returned is "True", the clock time will be asked to *IClock* interface, and Function manager will be executed.
- The *FunctionManager* will get the execution flag value from the *VariableBoolean1* interface.

- If the value returned is "True" the *FunctionManager* will get clock time, execute the function and re-ask for the clock time.

- For every value of the execution, flag returned the *FunctionManager* will continue measuring and saving the execution time of the function and setting the variable value.
- Finally, the *FrameworkManager* gets the clock time and measures and sets the execution time of the application and also sets the variable value.

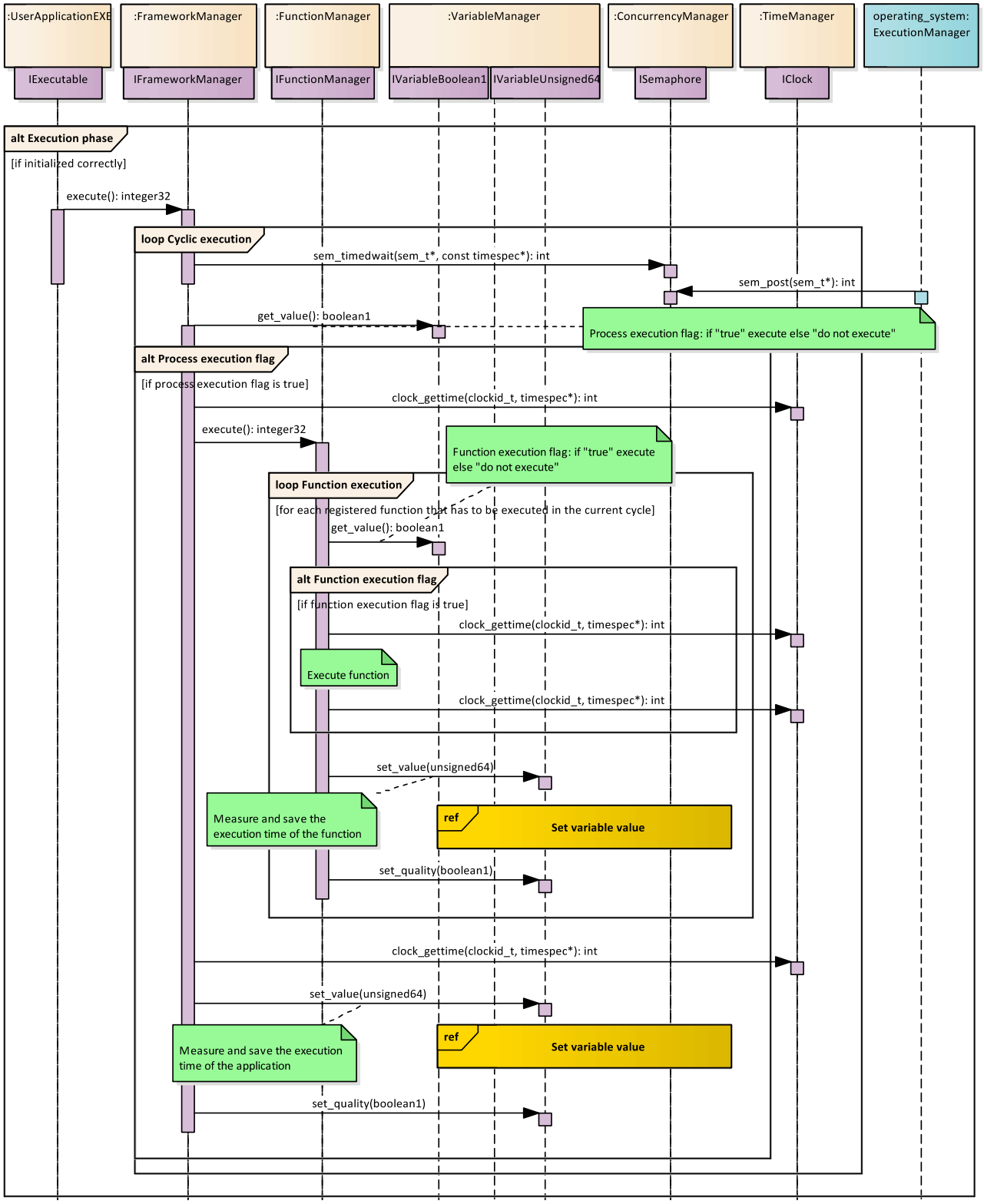

Figure 31. Cyclic execution phase.

In order to better explain the cyclic execution phase of the FDF, a set of specific Use Cases have been analyzed. These use cases are the following:

2.4.3.1 Data monitoring

Executing data monitoring (see Figure 32) starts with the *FunctionManager*. This manager calls to the execute function of the *MonitoringManager*. Then, the monitoring function accepts the socket offered by the *ISocket* interface and receives the socket message that consists of packets from the *IBEDriver* interface. After the reception, the monitoring manager gets the monitored variables from the *VariableManager* (this is done for each variable) and sends them through the socket connection to the *SocketManager*. Then, the socket manager sends the packets received to the *IBEDriver* interface of the *NICDriverManager*.

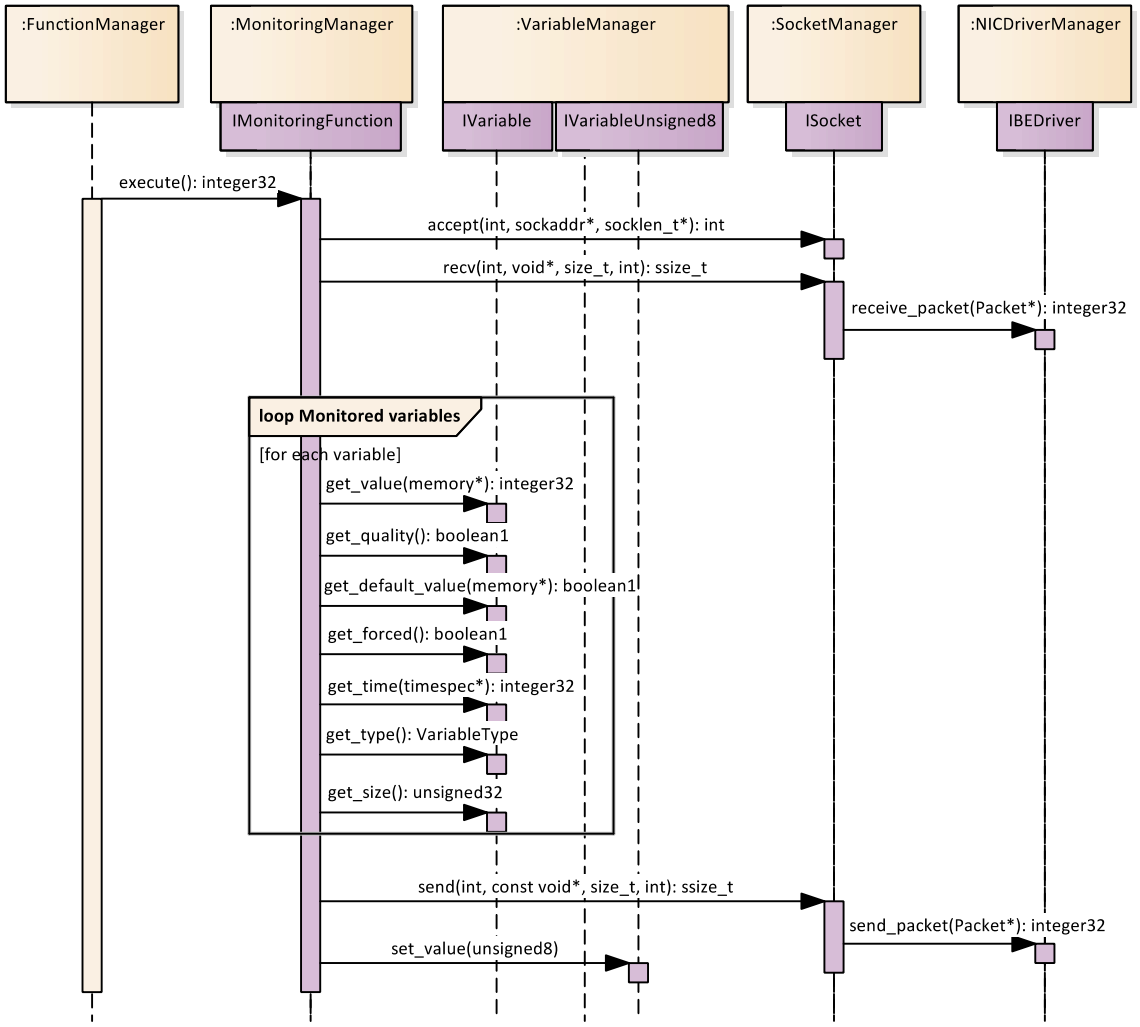

Figure 32. Data monitoring use case.

2.4.3.2 Data distribution

Data distribution execution composes of six sub-executions that include the message composing, parsing, value request, reception, transmission and value set, whose execution is explained in the following lines.

- Message Composing starts with the execution of the *IComposeFunction* interface from the *FunctionManager* (see Figure 33). The *IComposeManager* gets the value of

variables stored in memory and uses those values to build a message through the *IMessage* interface.

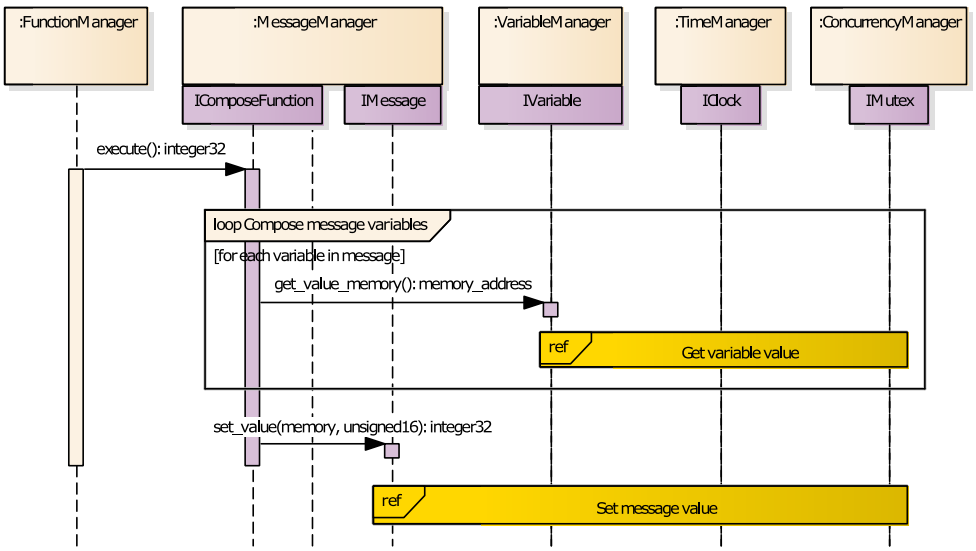

Figure 33. Message composing use case.

- Message Parsing execution is also initialised by the *FunctionManager* (see Figure 34). Then, the message manager gets the message variables through the *IMessage* interface (variables in messages) and sets those variables to a memory space (for each variable in the message).

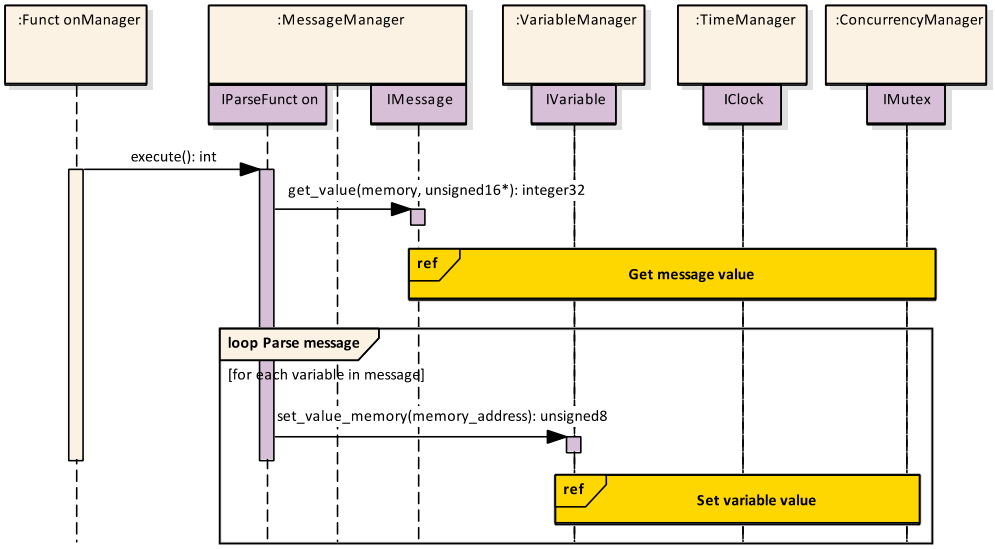

Figure 34. Message parsing use case.

- Get Message Value locks and unlocks the pthread *mutex* using the *IMutex* interface whenever concurrent access occurs. This execution sequence is shown in Figure 35.

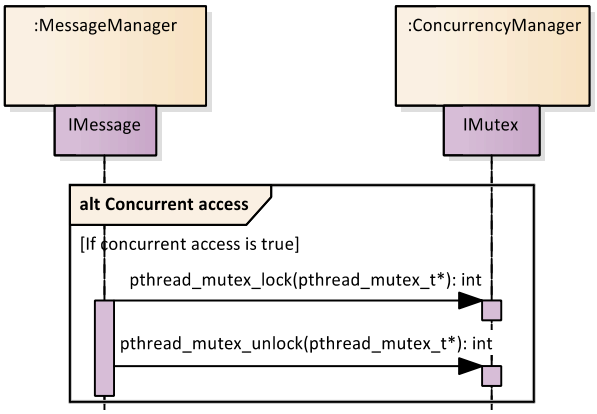

Figure 35. Get Message use case.

- Message Receiving execution sequence starts with the *FunctionManager* that executes the *IReceptionFunction* interface of the *NetworkManager*. Then, the reception interface request data reception to the *SocketManager*, which at the same time receives the data through the *IRTDriver* interface. Finally, values received by the *NetworkManager* are set to the *MessageManager*.

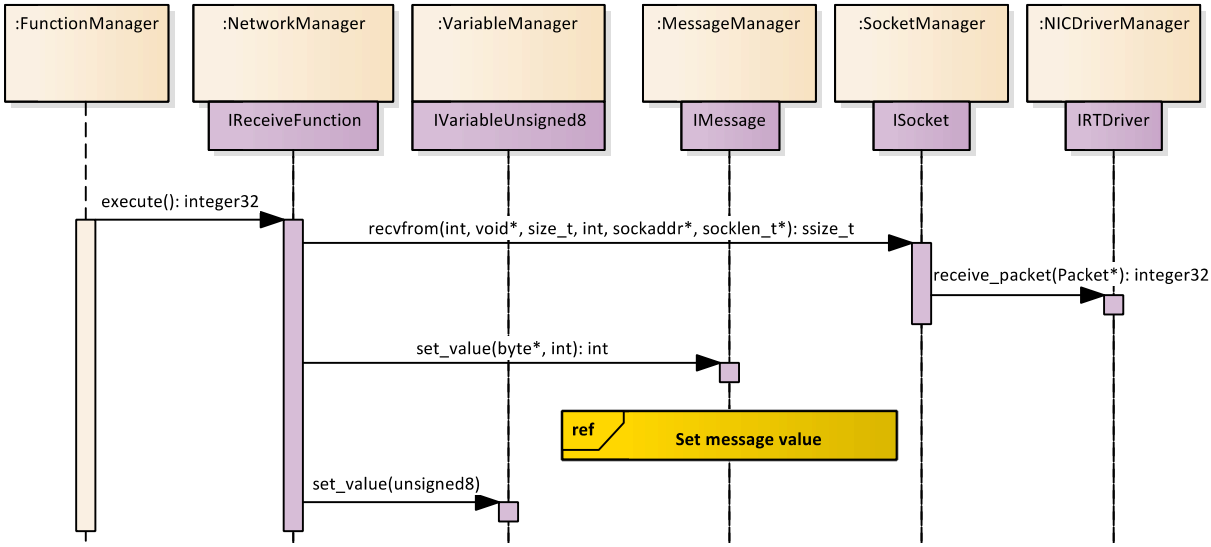

Figure 36. Receive Message use case.

- Message Sending sequence is also executed from the *FunctionManager* that executed the *NetworkManager* (see Figure 37). Then, the *ISendFunction* interface of the network manager gets the message values provided by the *MessageManager*. Finally, those values are sent to the *SocketManager* which send them in packets to the *NICDriverManager* in packets.

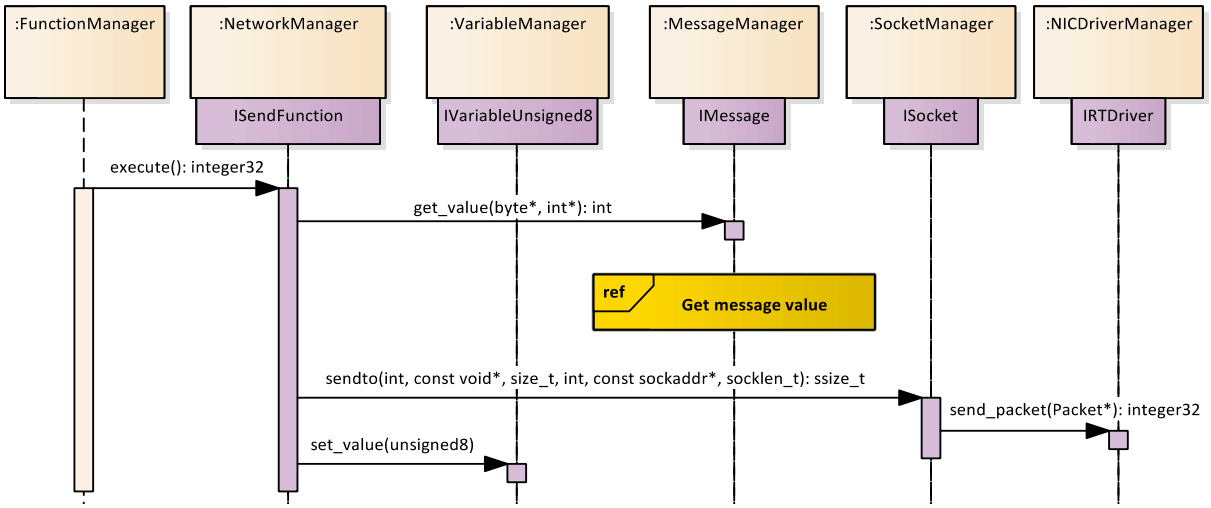

Figure 37. Send Message use case.

- Send Message Value execution sequence is started by the *MessageManager*, which requests the reception of the time clock specification to the *TimeManager* and locks and unlocks the *mutex* through the *IMutex* interface of the *CocurrencyManager* whenever concurrent access occurs (see Figure 38).

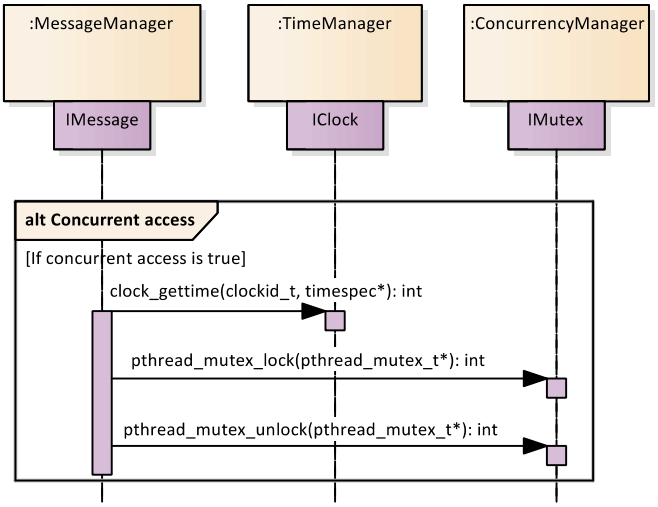

Figure 38. Set Message use case.

2.4.3.3 Global synchronisation

Global synchronisation is performed by the *SynchronizationManager* which is executed by the *FunctionManager* and sets the clocking time of the *TimeManager* with the *clocktime* received through the *IRTDriver* interface of the *NICDriverManager*. Figure 39 shows the sequential diagram of the execution the global synchronisation.

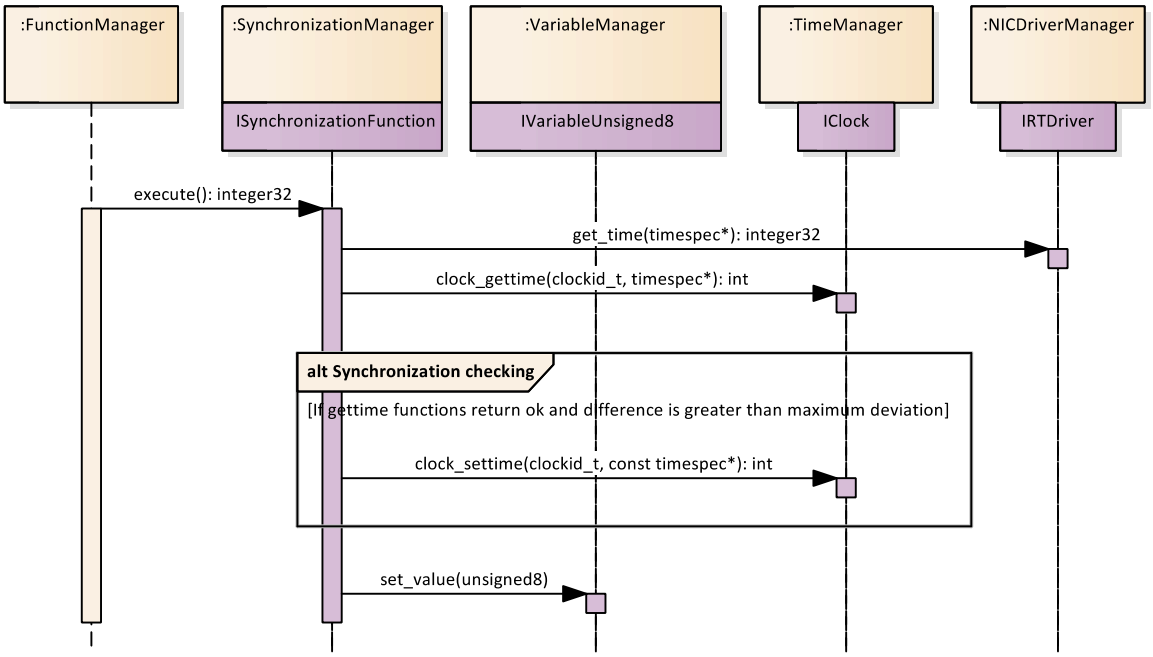

Figure 39. Global time synchronisation use case.

2.4.3.4 Watchdog refreshing

The watchdog timer refreshing is performed by the *HealthManager*, which execution is initialised by the *FunctionManager* (see Figure 40).

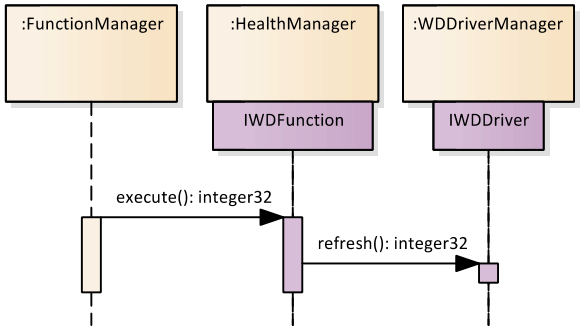

Figure 40. Refresh Watchdog use case.

2.4.3.5 Input reading

The following lines define the execution sequence diagrams for analog and digital input readings.

- Analog Input Reading starts with the execution of the *IAIFunction* interface of the *IOManager* (see Figure 41) and ends with the configuration of the *IVariableFloat32* interface using the data received from the *IAIDriver* interface.

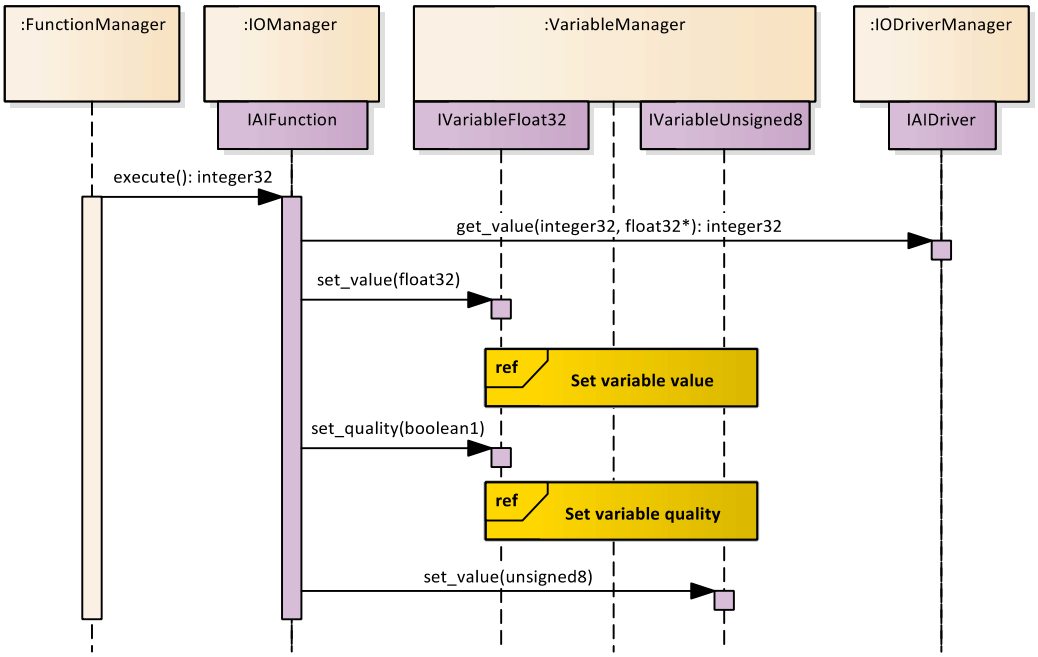

Figure 41. Read Analog Input use case.

- Digital Input Reading starts with the execution of the *IDIFunction* interface of the IOManager (see Figure 42) and ends with the configuration of the *IVariableBoolean1* interface using the data received from the *IDIDriver* interface.

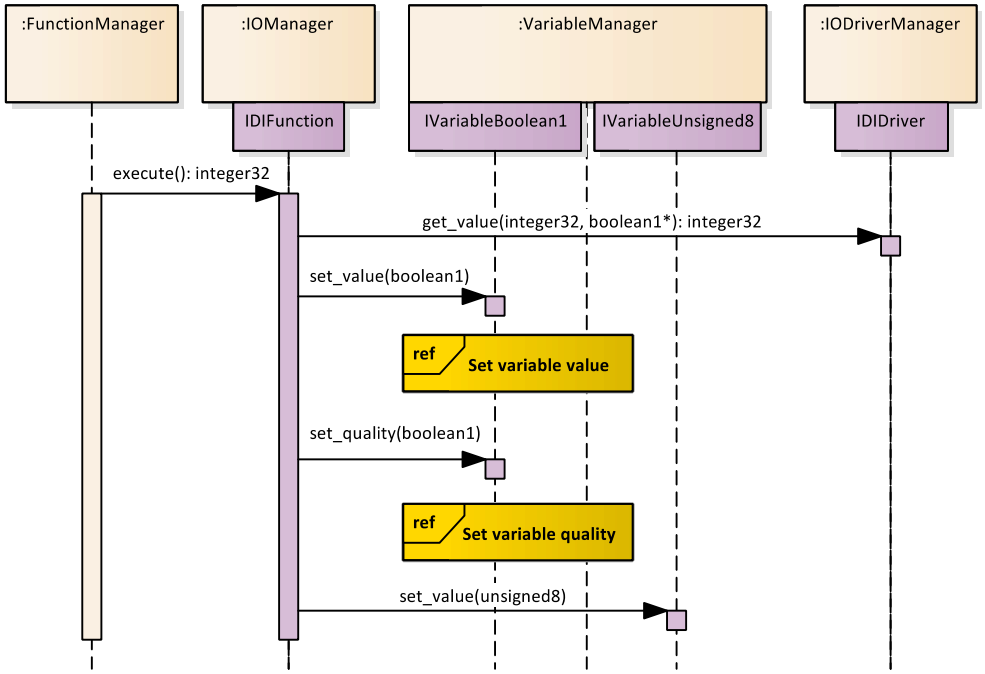

Figure 42. Read Digital Input use case.

2.4.3.6 Output writing

- Digital Output Writing: In this case, the sequence starts with the execution of the *IDOFunction* interface of the *IOManager* (see Figure 41) and ends with the configuration of the *IVariableBoolean1* interface using the data received from the *IDODriver* interface. States and qualities are taken into account before writing the outputs.

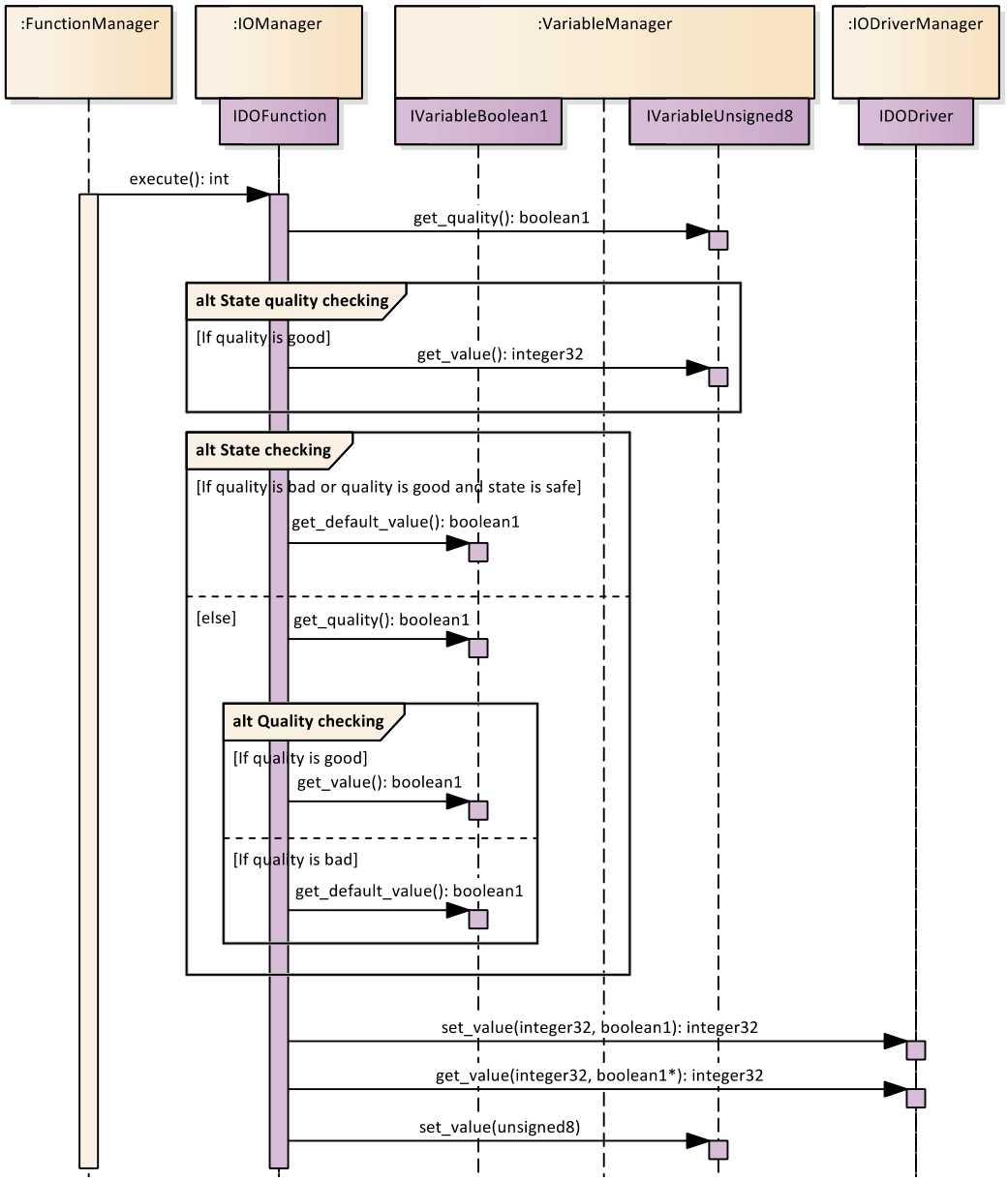

Figure 43. Write Digital Output use case.

- Analog Output Writing: This use case is identical to the digital output writing.

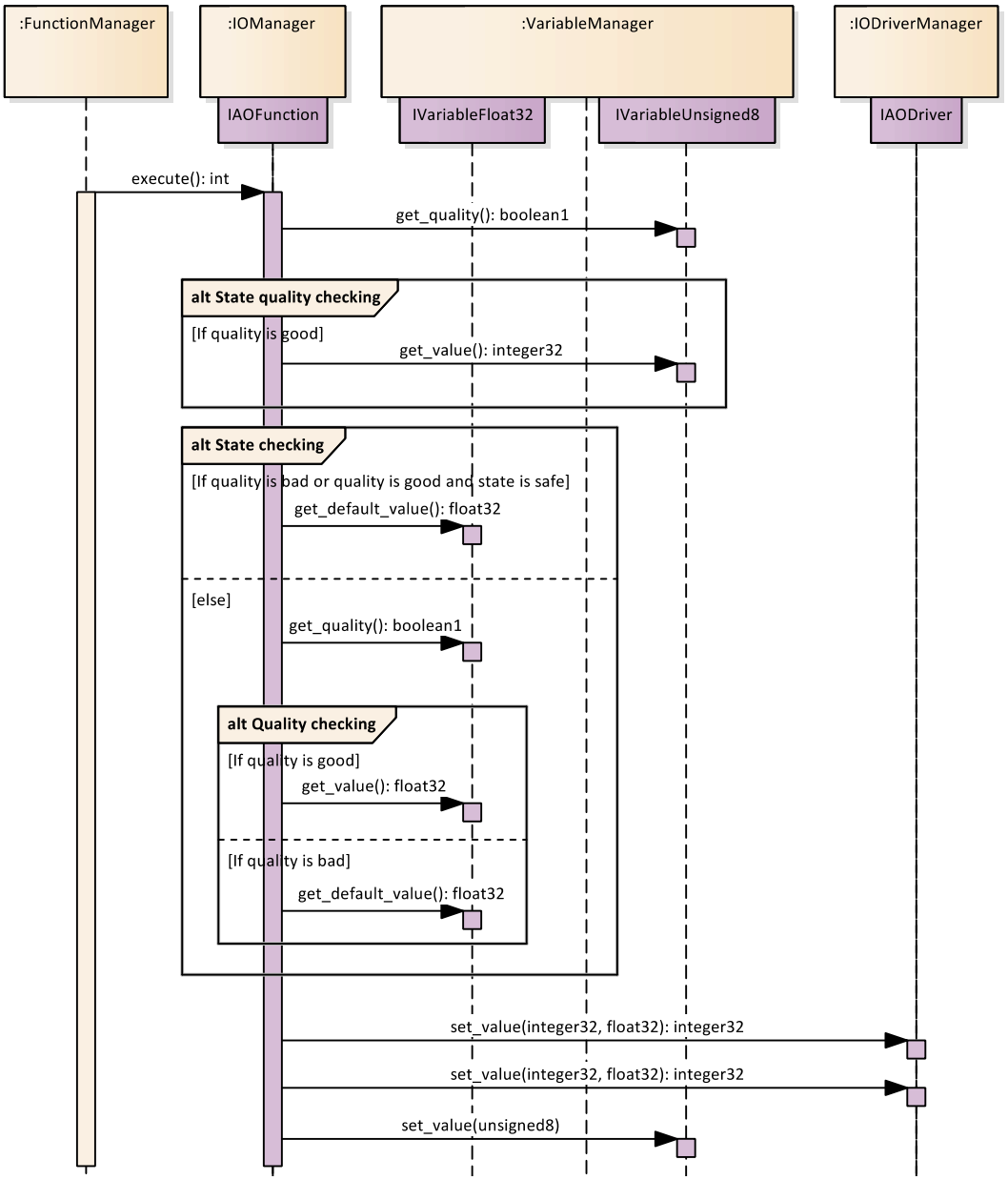

Figure 44. Write Analog Output use case.

2.4.3.7 Redundancy management

The redundancy manager is launched by the *FunctionManager*, following the configuration of the *VariableBoolean1* interface and time and concurrency managers (see Figure 45).

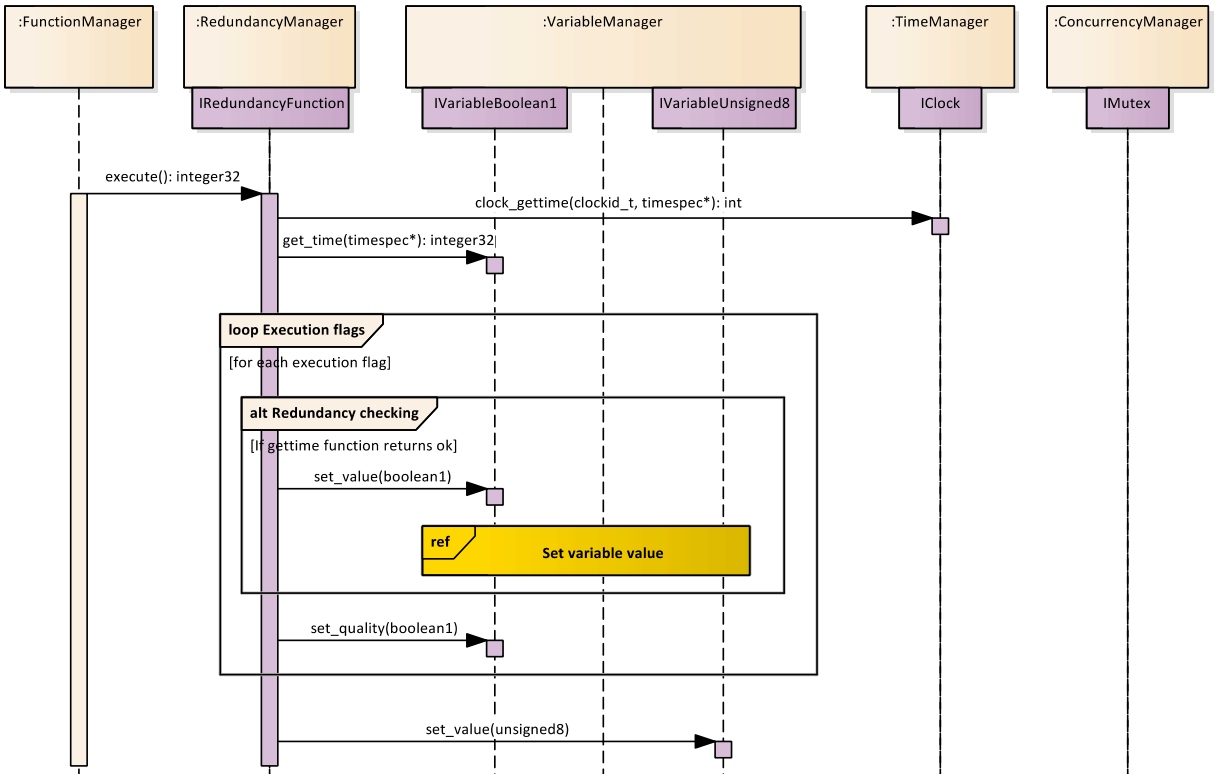

Figure 45. Redundancy Manager use case.

2.4.3.8 Data logging

Data logging execution sequence gets a string from the *VariableManager* for each variable and writes them to the *ILog* interface of the *LogManager*. After write process, the *ILog* interface gets the clock time through the *IClock* interface and locks the *pthread mutex* of the *ConcurrencyManager*. Further, the *ILog* interface opens a file, writes to it and closes it using the *IFile* interface. Finally, it unlocks the *ptherad mutex* of the concurrency manager. Figure 46 shows the sequence diagram of the data logging execution.

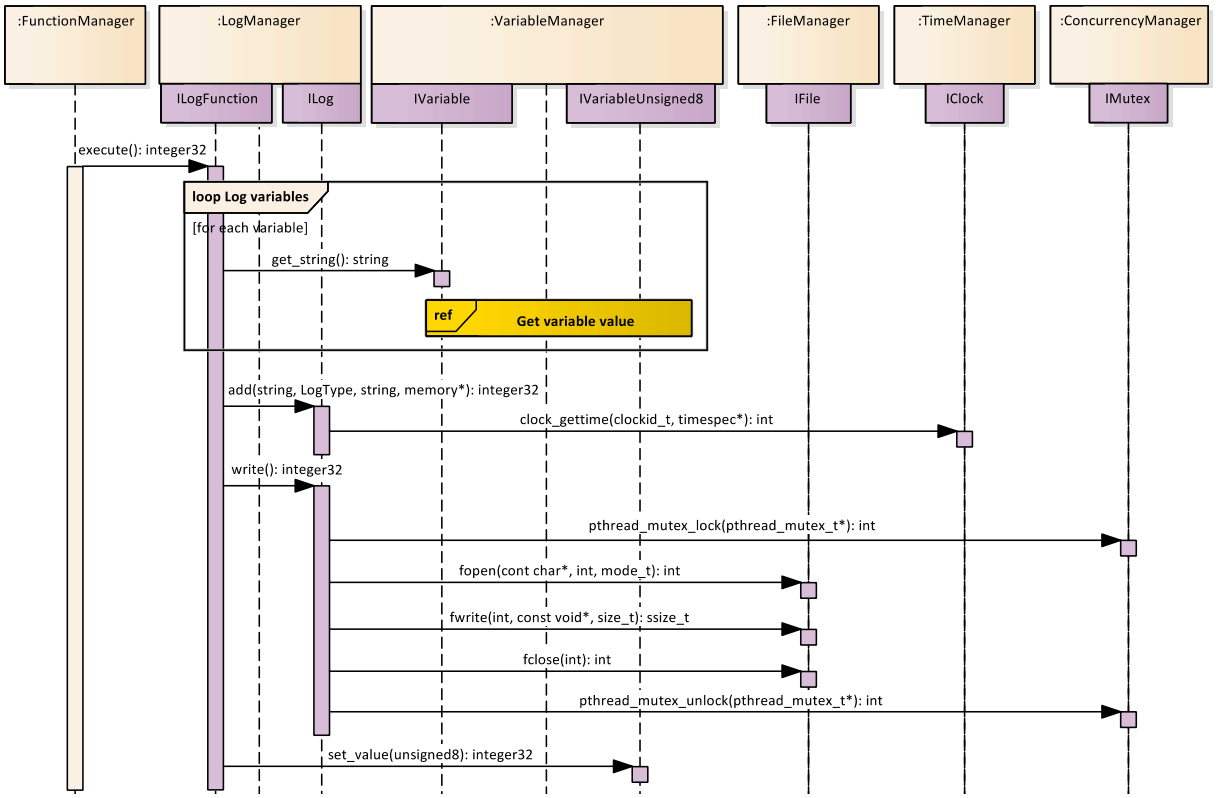

Figure 46. Data logging use case.

2.4.3.9 Data user function execution

The user application is launched by the *FunctionManager* as shown in Figure 47. After that, it gets for each input variables the *Boolean1* variable. This variable is provided through the *IVariableBoolean1* interface of the *VariableManager*. Finally, the user application sets for each output variable the *IVariableBoolean1* interface.

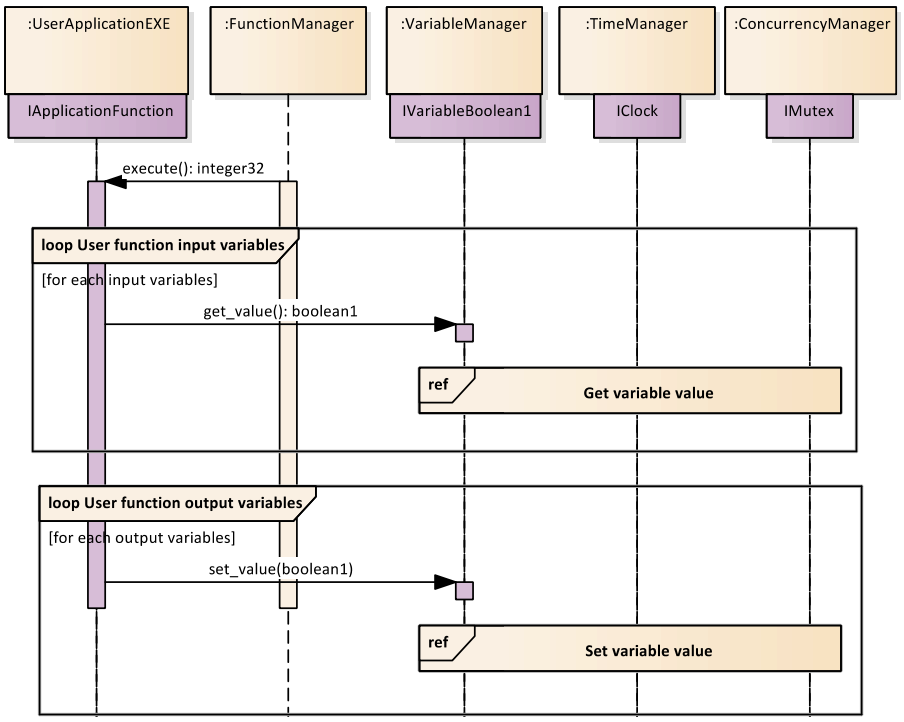

Figure 47. Execution of user application use case.

2.4.3.10 Data topology discovery

Data topology execution sequence (see Figure 48) is executed by the *FunctionManager*, which launches the *recvfrom* function of *ITopologyFunction* interface. *Recvfrom* function configures a socket communication from which receives and transmits data. Once *ITopologyFunctiion* interface sends and receives data through the socket, it initialises the *ITopology* interface and sets the quality of the topology. Finally, *ITopologyFunction* sets the values to *IVariableUnsigned8* interface.

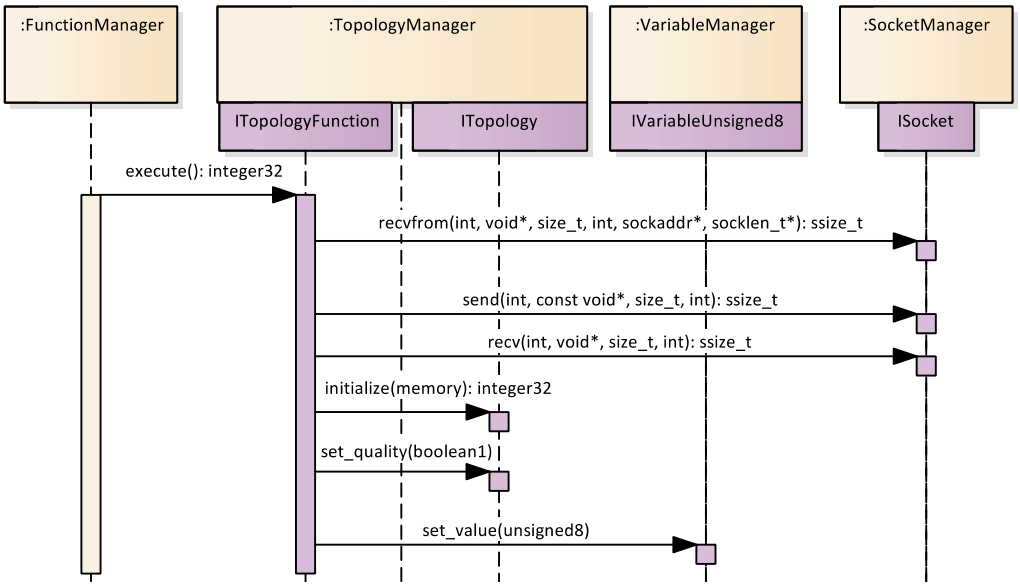

Figure 48. Data topology discovery use case.

2.4.3.11 Deadline checking

The deadline checking is also executed by the *FunctionManager (see* Figure 49*)*. Once *IDeadlineFunction* is launched, the *FunctionManager* gets the *unsigned64* and the quality values from the *VariableManager* and sets them to the *IVariableUnsigned8* interface.

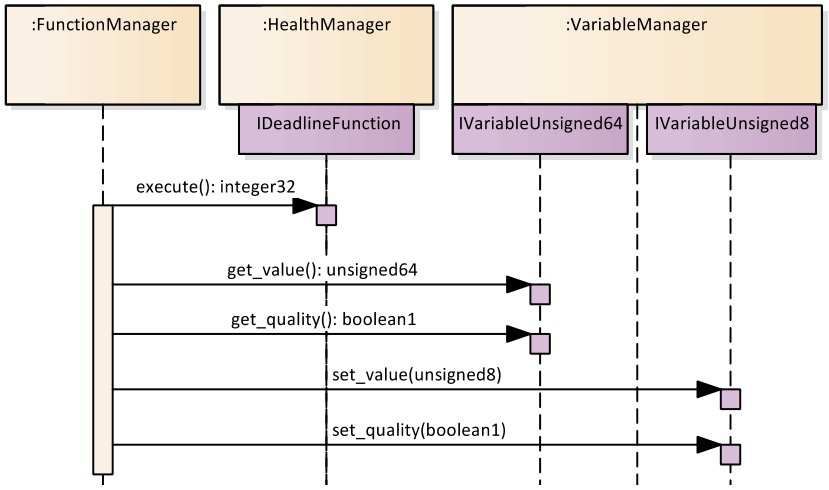

Figure 49. Deadline checking use case.

2.4.3.12 Disable execution

The execution sequence of *IDisableExecutionFunction* is executed by the *FunctionManager*, which gets the quality, value and default value from the *VariableManager* and then, sets those values to the *IVariableBoolean1* interface. This execution sequence is represented in Figure 50.

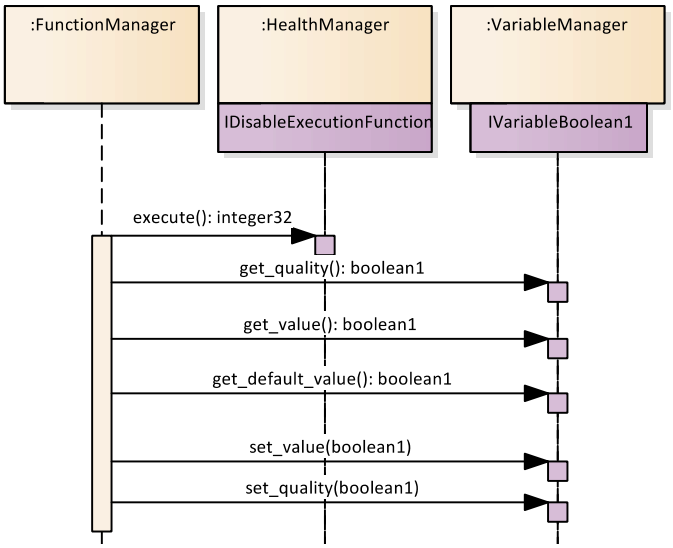

Figure 50. Disable execution use case.

2.4.3.13 Load checking

Once the *ILoadFunction* is launched (see Figure 51), it gets the load data from the *ECUDriverManager*. Further, the *FunctionManager* sets the value data and the quality of the *IvariableBoolean1* and *IVariableUnsigned8* interfaces.

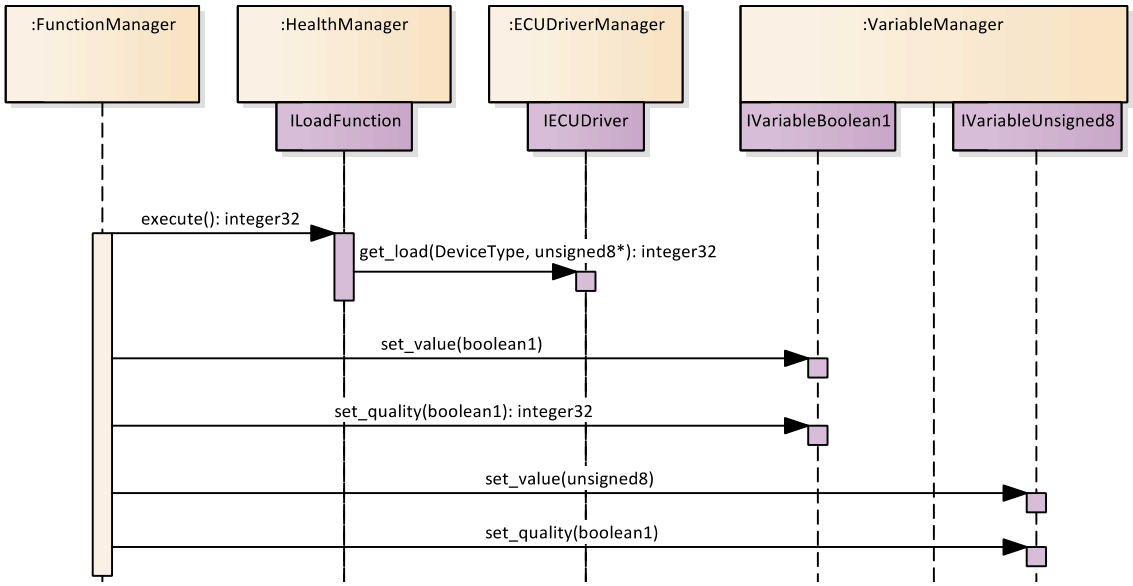

Figure 51. Load checking use case.

2.4.3.14 Output checking

The output checking interface is initialised by the *FunctionManager*. This manager also gets the time for each variable of *IVariable* interface and sets the value data and the quality of the

IVariableUnsigned8 interface of *VariableManager*. Figure 52 represents the sequence diagram of the *OutputChecking* sequence.

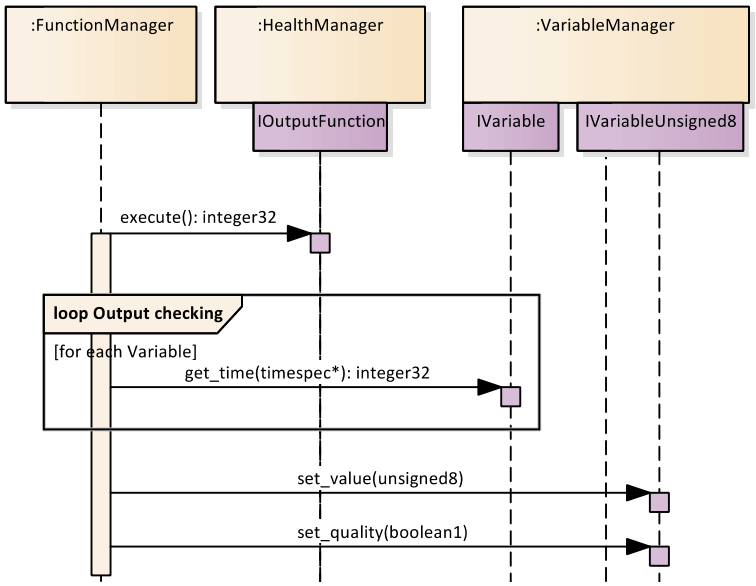

Figure 52. Output checking use case.

2.4.3.15 Temperature checking

Temperature checking interface gets the temperature values from the *IECUDriver* interface (see Figure 53). Then, the *FunctionManager* sets the value data and the quality of the *IVariableMemoryBoolean1* and *IVariabeUnsigned8* interfaces of the *VariableManager*.

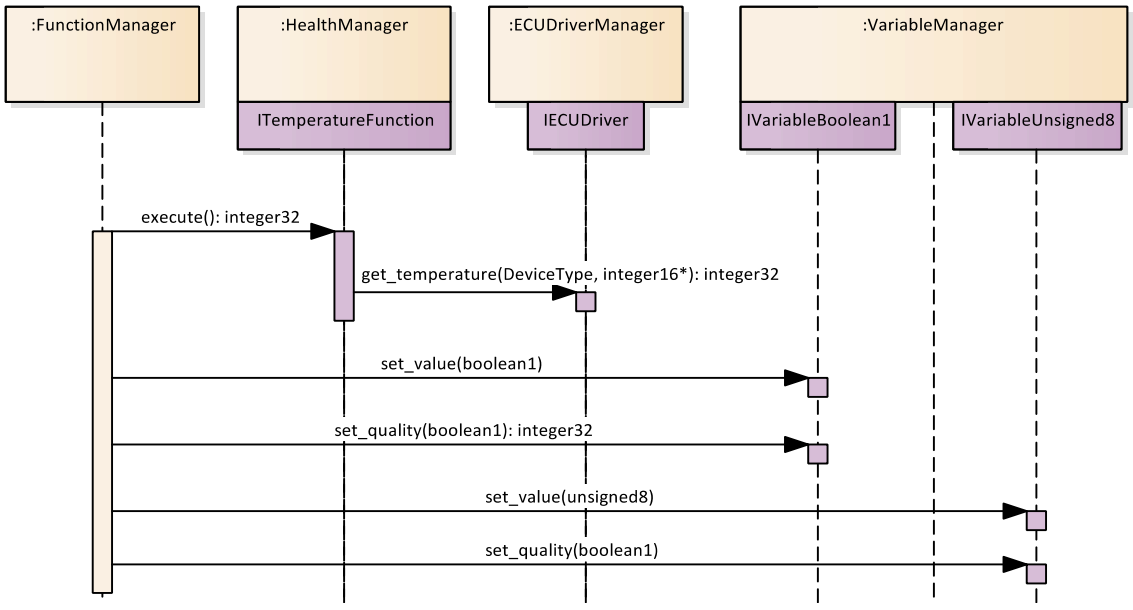

Figure 53. Temperature checking use case.

2.4.3.16 Reset platform

The reset interface of the *HealthManager* is launched by the *FunctionManager* (see Figure 54). This reset function gets the value, quality and default value data from the *IVariableBoolean1* interface for each variable and refreshes or not the *IWDDriverInterface* depending on the data received from the *VariableManager*.

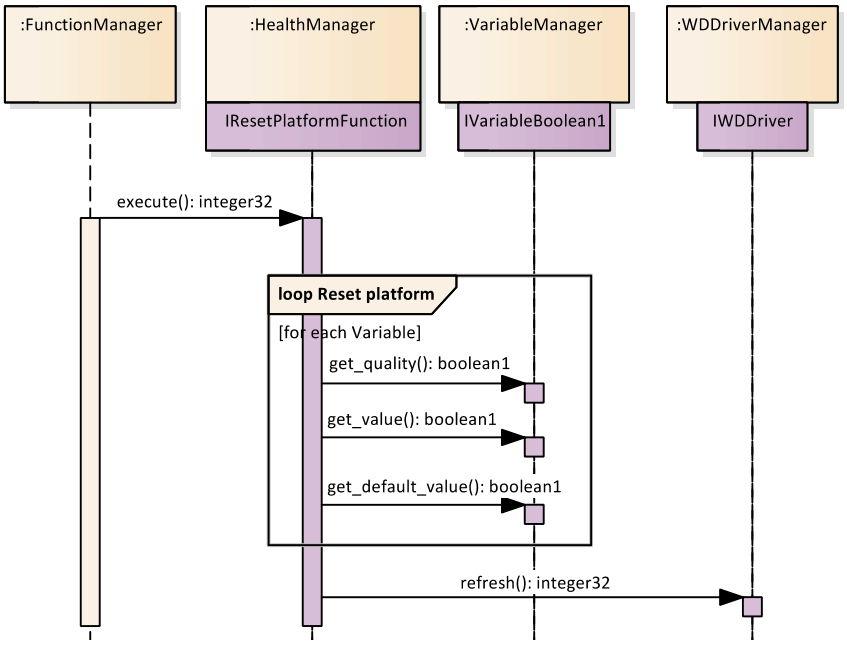

Figure 54. Reset platform use case.

2.4.3.17 Executable and configuration deployment

Once the deployment interface is launched (see Figure 55), it accepts the socket-based connection provided by the *ISocket* interface and requires the data reception through the *SocketManager*. The data transmitted through the socket becomes from the data packets provided by the *IBEDriver* interface. Afterwards, the *IDeployment* interface sends the new data values to the *ISocketInterface*, which updates the packet data values of the *NICDriverManager*. Finally, the deployment function sets the *IVariableUnsigner8* interface.

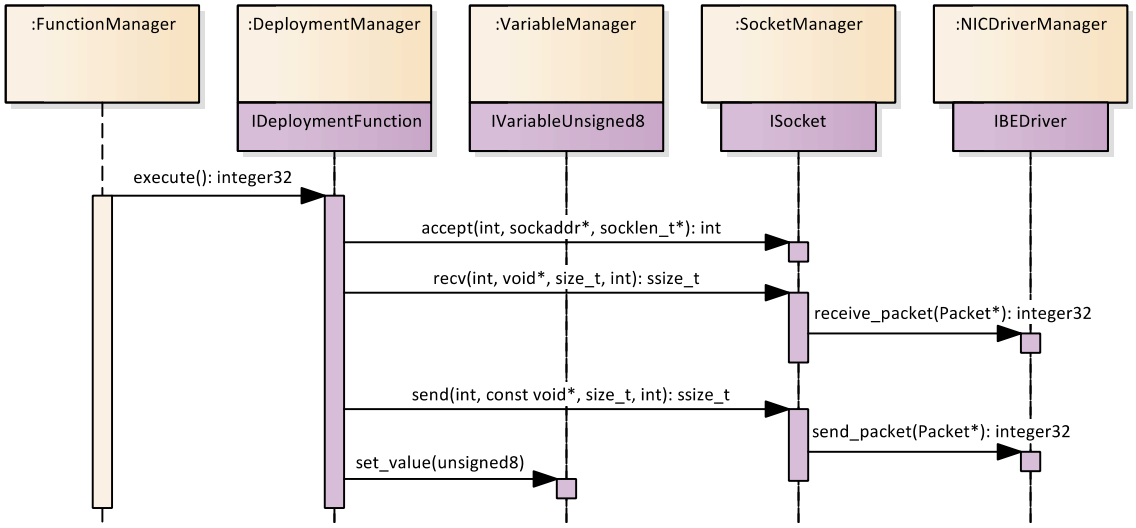

Figure 55. Executable and configuration deployment use case.

Chapter 3 Safety concept

The Safety concept of the TCMS Functional Distribution Framework (FDF) is defined by the set of measures required to assure the safe, functional operation of the hosted Application functions. These measures are elicited through a systematic approach, adopted for the development of the Preliminary Hazard Analysis (PHA) of the FDF.

The FDF PHA has the following main purposes:

- to assess systematically the potential functional deviations from the normal behaviour of the FDF and to identify the ones that can lead the hosted Applications to a condition contrary to safety;
- to assess the effects of these hazardous deviations and to group them into a set of reference (univocally identified) hazards;
- to specify the measures required to assure the safe, functional operation of the Functional Distribution Framework in spite of the postulated functional deviations, i.e. to reduce the related risk by preventing its occurrence and/or by mitigating its effects.

The following paragraphs provide:

- the specification of the functional model of the TCMS FDF taken as reference (see §3.1);
- a description of the methodology adopted to develop the FDF PHA (see §3.2).
- a summary of the main results obtained by the FDF PHA (see §3.3), and specifically the list of measures defining the FDF safety concept.

The Conceptual concept and the Safety concept are developed starting from a common "initial concept" of the Functional Distribution Framework, but with a degree of independence. The "Conceptual view" in chapter §2 describes all the physical and logical elements that interact within the FDF, as further refinement of the above "initial concept" of the Framework.

Further activity (out of the scope of this deliverable) will include the verification that the proposed physical and logical elements (i.e. Design concept) can implement the safety measures. Evidence will be provided by a traceability matrix between the FDF requirements and the safety measures identified during the Hazard Analysis, including countermeasures and recommendations. Any current misalignment between the lists of services/functions will be reconciled through at that stage.

3.1 FDF Functional model

The FDF PHA is based on a set of fundamental Functions implementing a set of fundamental Services provided to the hosted Applications.

Specifically, the PHA is focused on the fundamental FDF functions listed in Table 1.

Through these functions, FDF provides the fundamental services listed in Table 2 to the Applications.

Table 3 specifies the FDF Functions involved in the implementation of each Service. In general, a given deviation in the execution of a given function could lead to a relevant deviation of one or more Services provided by the Framework, and then to a hazardous deviation from the nominal behaviour of the Application function(s) using Service(s).

Communication	Transmission/reception of messages from/to Message Store to/from the network (remote functions)

Monitoring	Provision of SIL0 variables accessible remotely
Message function	Decomposition of messages (to share variables between remote functions) in variables (to share information between application functions) and composition of messages with variables
Input/Output function	Reading of input and updating of variables/setting of outputs according to variables
Time management	Dissemination of the global time from the external global clock
Framework management	Generation of variable stores, message stores and register functions as specified by the Configuration file. Offer API.
Configuration management	Reading, parsing, and loading of data in the configuration file
Functions management	Execution of registered Functions according to their scheduling plans
Fault management	Detection, isolation, notification and reaction to faults, and the recognition of system status with respect to errors and failures

Table 1: FDF's fundamental Functions

Initialization	It allows the generation and registration of the Application Functions in the Framework
Global clock synchronization	It allows the synchronization of the Application functions in the Framework
Scheduled execution of applications	It allows the execution of the Application functions in the Framework, according to their time-based scheduling plan
Data distribution	It allows the sharing (writing and reading) of variables among the Application functions , locally and through a safe and secure communication
IO reading and writing	It allows the Application functions in the Framework to acquire input for the updating of the related shared variable, and to provide output according to the related shared variable
Remote monitoring	It provides the values of shared variable to a given external client

Table 2: FDF's fundamental Services

Fundamental Services Vs Functions	**Initialization**	**Global clock synchronisation**	**Scheduled execution of applications**	**Data distribution**	**IO reading and writing**	**Remote monitoring**
Communication	X		X	X		X
Monitoring	X		X			X
Message function	X		X	X		
Input/Output function	X		X		X	
Time management		X	X			
Framework management	X					
Configuration management	X					
Functions management	X		X			
Fault management	X	X	X	X	X	X

Table 3: FDF's Functions and Services

3.2 FDF PHA Methodology

The Preliminary Hazard Analysis of the Functional Distribution Framework is developed in three main steps:

- identification of the functional deviations to be assessed (functional failure modes);
- evaluation of the effects of each functional deviation;
- specification of the measures required to assure the safe functional operation of the FDF and hosted Applications.

A HAZOP[1]-like approach is used for the **systematic identification of the functional deviations** from the nominal behaviour expected by the Functional Distribution Framework.

Deviations are defined through the application of guidewords to each function analysed. Table 4 provides the list of the guidewords and the deviations coming from their application to a generic function.

Guideword	Deviation
No	The function is not performed: the output is missed in spite of the input state.
Wrong	The function is not correctly performed: the output state is not the expected one for a defined input state.
Loss of / partially	The function is interrupted, only partially performed.
Undue	The function is correctly performed but when not required (undue output when there is not input)

Table 4: FDF PHA, Guidewords and deviations

The **effects of individual functional deviation** are evaluated with reference to the worst possible consequences of the Services provided by FDF (Local effect) and on a generic safety-related Application function (Final effect).

For each potentially hazardous deviation, all the measures required to assure the safe, functional operation of the FDF, i.e. to avoid, or preventing or mitigating the effects of the hazardous deviations in the execution of the Application functions, are specified.

The Functional Distribution Framework itself shall implement a subset of measures, i.e. they will be "covered" by the (safety) requirements of the FDF.

Countermeasures are classified into different categories, which are defined with reference to structure and content of the Technical Safety Report (section of the Safety case), as specified by the EN 50129 [4]:

- Assurance of functional operation, which concerns the correct operation of the TCMS FDF and hosted Applications under failure-free conditions;
- Detection of faults, which concerns the provisions to be implemented for the detection of a first fault, in a time sufficiently "short"[2];

[1] HAZard and OPerability analysis (HAZOP) is a structured and systematic methodology for the assessment of the deviations from the nominal behaviour of a given system. These deviations are defined by the application of "guide-words"", e.g. to the parameter (pressure-temperature-flow) of a process plant or to the functions implemented by an equipment / system.

[2] I.e. the detection-plus-negation time meets the safety target.

- Action following detection, which concerns the provisions to be implemented for the normalisation into a safe state (after the detection of the first fault), in a time sufficiently short2;
- Independence of items, which concerns internal and external (functional) influences;
- Systematic and Random faults, which concerns the specification of the Safety Integrity Level (SIL) required to the FDF's functions and services.

A further set of set of measures - **Application conditions** - shall be exported to the Application / remote functions and the interfaced external technical system(s). Their fulfilment is essential to guarantee the safe functional operation and behaviour under faults of the Functional Distribution Framework.

A last set of measures - **Recommendations** – provides non-mandatory indications about the implementation of the above countermeasures (e.g. architectural insights …).

The form used for the PHA of the Functional Distribution Framework is provided in the following tables. It is composed by three parts, which are focused on the results coming from the three steps of the analysis: Functional failure modes, Failure effects, Measures Specification.

Table 5 shows the first two parts of the FDF PHA form.

FUNCTIONAL FAILURE MODE					FAILURE EFFECTS		
ID	**Sub-function**	**Description**	**Guide-word**	**Deviation** (Functional Failure mode)	**Local effect**	**Final effect**	**System Hazard ID**

Table 5: FDF PHA form, Functional failure mode and Failure effects

Table 6 shows the third part of the FDF PHA form.

MEASURES SPECIFICATION													
Correct functional operation		**Detection of faults**		**Action following Detection**		**Independence of Items**		**Systematic & Random faults**		**Application conditions**		**Recommendations**	
ID	**Description**	**ID**	**Description**		**Description**		**Description**		**Description**	**ID**	**Description**	**ID**	**Description**

Table 6: FDF PHA form, Measures specification

3.3 FDF PHA Results

The results coming from the PHA developed for the Functional Distribution Framework can be consulted in ANNEX A: FDF Process Hazard Analysis.

It includes the following sheets: FDF_PHA; FDF_System hazards; FDF Countermeasures; FDF Countermeasures_pivot; FDF_Application conditions list; FDF_Recommendations list. The first sheet provides the FDF PHA form in Table 5 and Table 6, filled-in with the results coming from the failure assessment. The remaining sheets provide summaries of these results.

The following paragraphs provide the list of "system hazards" identified during the FDF PHA and the list of Countermeasures, Application conditions and Recommendations defining the TCMS FDF safety concept.

3.3.1 *System Hazards*

Table 7 provides the list of "System hazards" identified for the TCMS Functional Distribution Framework. "System hazards" represent the potential hazardous conditions in the execution of a generic safety-related Application function, due to deviation(s) in the execution of the FDF Functions and Services. Specifically, "System hazards" come from the (Final) effect described in the FDF PHA for each assessed functional failure mode (i.e. deviation). The same table also specifies the FDF Function(s) and the relevant deviation(s) able to produce each given system hazard.

ID	Hazard	FDF function	Deviation (Functional Failure mode)
FDF_SH_01	Potential unsafe behaviour of the Platform in the execution of the safety-related Application functions due to error(s) in the provision of data required by remote function(s) (missed, delay, incorrect data).	Monitoring	Delayed in the provision of variables to remote function(s)
			Incorrect provision of variables to remote function(s) (incorrect value)
			Incorrect provision of variables to remote function(s) (incorrect variable)
			No provision of variables to remote function(s)
FDF_SH_02	Potential unsafe behaviour of the Platform in the execution of the safety-related Application functions due to incorrect management of fault condition(s).	Fault management	Ineffective reaction to a detected fault
			Interaction of the Fault management services with other Service or Application functions.
			Missed detection of faults during the generation of the application software code
			Missed detection of faults during the run-time execution of the application software code
			Missed detection of faults of (hardware) resources used by Service and Application functions.
FDF_SH_03	Potential unsafe behaviour of the Platform due to a wrong timing in the execution of the safety-related Application functions	Functions management	Delayed execution of registered Function(s) with respect to the scheduling plan(s)
			Error in the execution of the function(s) with respect to the scheduling plan(s) and processes priority.
			Incomplete execution of registered Function(s) with respect to the scheduling plan(s)
			Undue execution of registered Functions, when not required by the scheduling plan(s)
		Time management	Incorrect dissemination of the global time from the external global clock (to all nodes or a subset of them)
			Missed update of the global time (i.e. according to the external clock)

ID	Hazard	FDF function	Deviation (Functional Failure mode)
			No dissemination of the global time from the external global clock
FDF_SH_04	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to a missed or incorrect acquisition of controls (input) from the interfaced object(s).	Input/Output function	Incorrect reading of input and/or updating of variables (exchange variable)
			Incorrect reading of input and/or updating of variables (wrong value)
			Incorrect timing in the reading of input and/or updating of variables (delayed or too fast)
			No reading of input and/or updating of variables
FDF_SH_05	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	Communication	Delayed exchange of messages from remote functions
			Incorrect exchange of messages between remote functions (including any possible types of communication error)
			Missed exchange of messages between remote functions
			Undue exchange of messages between remote functions (when not required)
		Message function	Corruption of safety-related data within the messages exchanged between remote function.
			Deletion of messages exchanged between remote function including safety-related data.
			Incorrect composition of messages with variables or data corruption during reading
			Incorrect decomposition of messages into variables or wrong updating
			Insertion within the messages exchanged between remote function including safety-related data.
			Masquerade messages including safety-related data exchanged between remote function.
			No / partial composition of messages with variables
			No / partial decomposition of messages into variables
			Repetitions of messages exchanged between remote function including safety-related data.
			Resequencing of messages exchanged between remote function including safety-related data.
			Undue use of message containing non-safety related data, for safety-related applications.
FDF_SH_06	Potential unsafe behaviour of the Platform in the execution of	Functions management	No execution of registered function(s) required by the scheduling plan(s) and process priority

ID	Hazard	FDF function	Deviation (Functional Failure mode)
	safety-related processes due to a missed or incorrect setting of commands (output) toward the interfaced object(s).	Input/Output function	Incorrect setting of outputs according to variables (exchange variable)
			Incorrect setting of outputs according to variables (wrong value)
			Incorrect timing in the setting of outputs according to variables (delayed or too fast)
			No setting of outputs according to variables
FDF_SH_07	Potential unsafe behaviour of the Platform in the execution of the safety-related processes due to an incorrect generation or allocation of resources or management of partitions.	Framework management	Inadequate allocation of resources to partition, for the execution of the Application function(s) / process(es)
			Inadequate generation of partition and/or allocation of resources, for the execution of multiple instances of the Application function(s) / process(es)
			No or partial allocation of resources to partition, for the execution of the Application function(s) / process(es)
			No, partial or delayed generation of partition(s) (definition of memory space, variable stored, messages' structure, register functions) specified in the Configuration file.
			Undue access to variables, and related I/O, by Application function(s) without the required read/write privilege.
			Wrong assignment of read-write privileges and constraints to Application functions.
			Wrong generation of partition(s) (e.g. wrong address or size of memory, structure of message, stores and register functions) with respect to the Configuration file.
FDF_SH_08	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect configuration.	Configuration management	Data corruption during reading, parsing, or loading of data in the configuration file
			Error during reading, parsing, or loading of data in the configuration file
			No / missed / partial reading, parsing, or loading of data in the configuration file
			Reading, parsing, and loading of data from a false or corrupted configuration file
			Loading of data in the configuration file at a wrong time (e.g. while the FDF has already been configured).
FDF_SH_09	Potential unsafe behaviour during the execution of safety-related processes due to unintended interactions between the Operating system and the Application functions.	Framework management	Unintended interactions between the Operating system and the Application functions.

Table 7: FDF PHA, List of System Hazards and relevant FDF Functions and deviations

3.3.2 *Countermeasures*

The following tables provide the set of countermeasures identified during the Preliminary Hazard Analysis of the Functional Distribution Framework, assuring the proper functional operation, detection of faults, action following detection, independence of items and defence against systematic & random faults.

Each table provides the countermeasures specified for each given FDF function. Within each table, the countermeasures are grouped by the above categories (see §3.2).

Equivalently, the countermeasures can be grouped by different categories and then listed with reference to the function(s) involved.

Table 8 provides the countermeasures specified for the Communication function.

COMMUNICATION FUNCTION		
Classification	**ID**	**Countermeasures**
Correct functional operation	HA_COM_01	The Framework shall provide a communication service that makes received messages available to the Application functions within defined timely bounds (deterministic receiving).
	HA_COM_02	The Framework shall provide a communication service that allows sending messages within defined timely bounds and with defined periodicity, and receiving messages within defined maximum delay (deterministic communication).
	HA_COM_03	The Framework shall define, configure, assess and guarantee performance of communication channels, including priority, throughput, jitter, latency, response time.
	HA_COM_04	The Framework shall implement Communication service without any operation on the messages' safety layer content.
Detection of faults	HA_COM_05	The Framework shall monitor the communication between remote functions.
Action following Detection	HA_COM_06	The Framework shall inform the Application function(s) in case of loss of valid communication between remote functions.

Table 8: FDF PHA - countermeasures, Communication function

Table 9 provides the countermeasures specified for the Configuration management function.

CONFIGURATION MANAGEMENT		
Classification	**ID**	**Countermeasures**
Correct functional operation	HA_CONF_01	The Framework shall instantiate messages and variable according to the Configuration file, which specifies at least: messages' identifier, variables, to receive or to send, schedule, deadline; variables' identifier, type, range, default value, deadline.
	HA_CONF_08	The Framework shall load the Configuration file during the execution of the inauguration services and assure that any re-configuration (re-loading of the Configuration file or loading of a new Configuration file) is performed involving all the Application functions to be executed.
	HA_CONF_02	The Framework shall accept only certified remote Configuration file (coming from a verified source), protected against data corruption, e.g. by CRC.

CONFIGURATION MANAGEMENT		
Classification	**ID**	**Countermeasures**
Detection of faults	HA_CONF_03	The Framework shall verify the validity and integrity of the Configuration file, before and after the end of the inauguration services, e.g. by CRC, MD or signature created by tooling.
	HA_CONF_04	The Framework shall verify the validity of results coming from the inauguration (Train Topology Database or equivalent data structure) and their coherence with the Configuration file.
Action following Detection	HA_CONF_05	The Framework shall not execute the Application functions in case of any error detected in the Configuration file or non-valid results coming from the inauguration or undue operation on the Configuration data, and notify a (fatal) Fault condition to all the Application function(s) involved.
Independence of Items	HA_CONF_06	The Framework shall assure that re-configuration required for new or modified Application functions is performed involving all the Application functions to be executed, or anyway the existing configuration for the remaining Application functions is not altered.
Systematic & Random faults	HA_CONF_07	The Framework shall read, parse, load and check data in the Configuration file and configure the platform accordingly, with the same SIL assigned to the related Application function.

Table 9: FDF PHA - countermeasures, Configuration management

Table 10 provides the countermeasures specified for the Framework management function.

FRAMEWORK MANAGEMENT		
Classification	**ID**	**Countermeasures**
Correct functional operation	HA_FRM_01	The Framework shall generate Partitions according to the Configuration file of the Application functions to be executed (which specify the SIL, address and size of the memory space, and time window inside the global scheduling plan) and protect each partition's addressing space through specific memory protection mechanisms, e.g. by a hardware memory management unit, and management of access privilege and restrictions.
	HA_FRM_02	The Framework shall provide to the partition assigned to an Application functions the computational resources (e.g. CPU time, memory) required into the Configuration file in order to meet the (worst-case) timing requirements.
	HA_FRM_03	The Framework shall provide to the Application functions the read-write privilege only to variables (and related input/output, if any) they are allowed to publish and the read-only privilege to software code, parameters and variables (and related input, if any) they are subscribed to.
	HA_FRM_04	The Framework shall guarantee that Application functions read / write variables, managing consequently the related platform's I/O, only if the required privilege is provided.
	HA_FRM_05	The Framework shall call Services required for the scheduled execution of the Application functions.

FRAMEWORK MANAGEMENT		
Classification	**ID**	**Countermeasures**
	HA_FRM_06	The Framework shall be able to generate partitions and allocate resources for Application function(s) requiring multiple instances (for the implementation of a reliable-safe architecture).
Detection of faults	HA_FRM_07	The Framework shall detect an invalid operation in the partition attempts by the Application function(s), e.g. access to a Memory space without the required reading or writing privilege.
	HA_FRM_18	The Framework shall detect the unavailability of Services required for the scheduled executions of the Application functions and their incorrect call (different than scheduled)
Action following Detection	HA_FRM_08	The Framework shall notify a Fault condition, in case of invalid operation in the partition attempt (fatal Fault), to all the Application functions involved.
	HA_FRM_09	The Framework shall inform the Application functions in case of unavailability of services required for their scheduled execution, or in case of incorrect call (different than scheduled).
Independence of Items	HA_FRM_10	The Framework shall protect and guarantee the independence of multiple instances of an Application function (e.g. implementing reliable-safe architecture), e.g. by data diversity (e.g. different time-stamp guarantying data freshness), timing diversity (instances do not execute simultaneously the same safety-related software modules), independent (hardware) resources.
	HA_FRM_11	The Framework shall guarantee the spatial separation among Partition, in order to ensure that no process in one partition can modify (without authorization) software code or application data (i.e.. write to memory data sections, stacks and code) or manage the I/O assigned to another partition, e.g. through the protection of their memory addressing space and the management of privilege and restrictions for variables read / write and for access to I/O.
	HA_FRM_12	The Framework shall guarantee spatial separation between memory spaces containing read-only (including software code and parameters) and read-write variables, variables with different SIL, variables used by multiple independent instances of the Application function.
	HA_FRM_13	The Framework shall prevent any unintended interactions between the Operating system activities and the Application functions, through the definition of formal boundaries and interaction modalities and protecting the Operating System (data sections, stacks, and code) against undue calls from the Application and Services functions (e.g. with an invalid handle, object, address or out of range value; in the wrong context; without the necessary permissions).
Systematic & Random faults	HA_FRM_14	The Framework shall generate partitions and allocate resources with the same SIL assigned to the Application functions to be executed, including memories spaces storing data with the same (unique) SIL.
	HA_FRM_15	The Framework shall assign privileges for read-write access to a Memory space only to independent Application functions with the same SIL. Read-only access could be assigned to remaining Application functions, if data alteration during reading can be excluded.
	HA_FRM_16	The Framework shall guarantee the read-write access to memory spaces (according to the assigned privileges) with the same SIL assigned to the Application function(s) and variables stored.

FRAMEWORK MANAGEMENT		
Classification	**ID**	**Countermeasures**
	HA_FRM_17	The Framework shall guarantee the effectiveness of call(s) to Service function(s) with the same SIL assigned to the Application functions using Service(s).

Table 10: FDF PHA - countermeasures, Framework management function

Table 11 provides the countermeasures specified for the Functions management function.

FUNCTION MANAGEMENT		
Classification	**ID**	**Countermeasures**
Correct functional operation	HA_FNM_01	The Framework shall control the execution (start, stop, synchronizing to external trigger) of Application functions assigned to each individual partition, through the deterministic management of timers (for sequential execution) and semaphores (for sequential and concurrent execution), according to their scheduling plans and to processes priority.
	HA_FNM_02	The Framework shall execute an Application function, giving access to memory resources, only when required by its scheduling plan (and take away access otherwise).
	HA_FNM_03	The Framework shall implement Service functions whose response times allow the real-time execution of processes and the fulfilment of the most restrictive response time required by the Application functions to be executed.
	HA_FNM_04	The Framework shall implement mechanisms to ensure the execution of real-time processes in spite of transient temporal violations, e.g. due to inter-module communications acknowledgements, time-outs, access to memory, interrupts.
	HA_FNM_05	The Framework shall avoid interrupts or manage them through the Operating system only (even if triggered by the Application functions or by hardware), avoiding any disturb to the time partitioning, i.e. without any change of the time budget allocation.
Detection of faults	HA_FNM_06	The Framework shall monitor the execution (start, stop, synchronizing to external trigger) of processes with respect to defined timing bounds for (intra-partition and inter-partition) communication and processing.
Action following Detection	HA_FNM_07	The Framework shall notify a Fault condition, in case of error in the execution of processes according to the scheduling plans, including the violation of timing bounds (fatal Fault), to all the Application functions involved.
Independence of Items	HA_FNM_08	The Framework shall implement temporal partitioning, by ensuring that a process within a given time budget cannot be affected by the actions of any other task from other partitions, in terms of rate, latency, jitter and duration of the scheduled access.
Systematic & Random faults	HA_FNM_09	The Framework shall control the execution of processes and the transmission of messages (according to their scheduling plans) with the same SIL assigned to the involved Application functions.

Table 11: FDF PHA - countermeasures, Functions management

Table 12 provides the countermeasures specified for the Input/Output function.

INPUT/OUTPUT FUNCTION		
Classification	**ID**	**Countermeasures**
Correct functional operation	HA_IO_01	The Framework shall provide services that allow the Application function to read the last valid value stored into an exchange variable and to update this value according to the status of the related input (coming from the interfaced object).

INPUT/OUTPUT FUNCTION		
Classification	**ID**	**Countermeasures**
	HA_IO_02	The Framework shall provide services that allow the Application function to write a value into an exchange variable and to update accordingly to the status of the related output (toward the interfaced object).
	HA_IO_03	The Framework shall identify univocally each input / output interfacing external objects, each exchange variable, and each association between them, according to the Configuration file(s) of the Application function(s) using them.
	HA_IO_04	The Framework shall read and write all the I/O related to the executed Application function in one cycle only, guarantying that the current value of every input is stored in the associated exchange variable at the beginning of each cycle and the current value of every output is set according to the value stored in the associated exchange variable at the end of each cycle..
Detection of faults	HA_IO_05	The Framework shall detect inconsistency between the values stored into the exchange variables and the status pf the related platform's input and output.
Action following Detection	HA_IO_06	The Framework, in case of any inconsistency between the values stored into an exchange variable and the status of the related platform's input / output, shall inform the Application function(s) with read and/or write privilege on this variable.
Independence of Items	HA_IO_07	The Framework shall be able to provide independence between different (set of) input / output interfacing external objects (that can be request by Application function to implement reliable-safe architecture).
Systematic & Random faults	HA_IO_08	The Framework shall guarantee the updating of each exchange variable (according to the status of related input) and its reading with the SIL assigned to the Application function(s) involved and to the specific variable.
	HA_IO_09	The Framework shall guarantee the updating the status of each output (according to value stored into the related exchange variable) and its writing with the SIL assigned to the Application function(s) involved and to the specific variable.
	HA_IO_10	The Framework shall allow I/O Function to access only to memory space with the same SIL.

Table 12: FDF PHA - countermeasures, Input/Output function

Table 13 provides the countermeasures specified for the Message function.

MESSAGE FUNCTION		
Classification	**ID**	**Countermeasures**
Correct functional operation	HA_MSG_01	The Framework shall ensure the integrity of safety-related data exchanged by communication protocol(s) implementing a safety layer (i.e. a safety code) with source and/or destination identifiers, information that the transmitter is operating properly, redundancy field allowing error detection and assuring data integrity.
	HA_MSG_02	The Framework shall ensure the timeliness and sequence of data exchanged and results of safety algorithms, e.g. by sequence number and/or time stamps generated by unique identifier related to the cycle (or equivalent measures).
	HA_MSG_03	The Framework shall protect the communication of safety-related data against cyber-attack, ensuring data authenticity and confidentiality, e.g. by software and/or hardware security mechanisms (e.g. cryptographic mechanisms, control of access to data).
	HA_MSG_04	The Framework shall use protocols for diagnostic, maintenance, configuration and communication of non-safety related data with different structures than one(s) used for the communication of safety-related data.

MESSAGE FUNCTION		
Classification	**ID**	**Countermeasures**
	HA_MSG_05	The Framework shall guarantee that Message Function read and write the required variables in a safe way, i.e. variables are read without altering their value and written according to specification (set during configuration).
Detection of faults	HA_MSG_06	The Framework shall check the integrity (i.e. information is complete and not altered) of incoming messages containing safety.
	HA_MSG_07	The Framework shall check the timeliness and sequence of messages containing safety-data, exchanged between remote functions.
	HA_MSG_08	The Framework shall check the authenticity of incoming message containing safety data, exchanged between remote functions.
Action following Detection	HA_MSG_09	The Framework and Application functions shall ignore the content and discharge a message (containing safety-data) when a communication error is identified through the messages authenticity, integrity, timeliness or sequence checks.
Independence of Items	HA_MSG_10	The Framework shall implement reactions against errors in the communication of safety-related data that are functionally independent by any non-trusted transmission.
Systematic & Random faults	HA_MSG_11	The Framework shall guarantee the validity of safety related data exchanged between remote functions, through messages composing and decomposing into variables carried out by the Message Function, with the same SIL assigned to the Application function(s) using messages and variables involved.
	HA_MSG_12	The Framework shall allow Message Function to access to memory space(s) containing messages and to memory space(s) containing variables with the same SIL.

Table 13: FDF PHA - countermeasures, Message function

Table 14 provides the countermeasures specified for the Monitoring function.

MONITORING FUNCTION		
Classification	**ID**	**Countermeasures**
Correct functional operation	HA_MON_01	The platform shall assign to the Monitoring Function privilege for read-only the variables stored into SIL0 Memory spaces, or to all the Memory spaces if data alteration during reading can be excluded, and execute Monitoring services without any disturb or unintended effects due to other Service and Application functions.

Table 14: FDF PHA - countermeasures, Holding Brake countermeasures

Table 15 provides the countermeasures specified for the Time management function.

TIME MANAGEMENT		
Classification	**ID**	**Countermeasures**
Correct functional operation	HA_TM_01	The Framework shall synchronize the local computer clock with the external global clock source and keep it synchronized with a maximum defined deviation fixed.

TIME MANAGEMENT		
Classification	**ID**	**Countermeasures**
	HA_TM_02	The Framework shall not finalize the inauguration and allow operation without a global time valid (i.e. aligned with the external global clock) and taken as unique reference by all Service and Application functions, independently from the partitions execution.
Detection of faults	HA_TM_03	The Framework shall monitor the alignment with the external global clock, the effectiveness of the global time dissemination and functions synchronization.
Action following Detection	HA_TM_04	The Framework shall notify a Fault condition, in case of error in the global time synchronisation (fatal Fault), to all the Application functions involved.
Independence of Items	HA_TM_05	The Framework shall synchronize the local computer clock with the external global clock source and keep it synchronized independently from the execution of the different partitions' processes.
Systematic & Random faults	HA_TM_06	The Framework shall disseminate the global time and/or detect any misalignment against the external reference time, with the highest SIL assigned to the Application functions to be executed.

Table 15: FDF PHA - countermeasures, Time management function

Table 16 provides the countermeasures specified for the Fault management function.

FAULT MANAGEMENT		
Classification	**ID**	**Countermeasures**
Correct functional operation	HA_FLT_01	The Framework shall provide services for the detection of faults of (hardware) resources used by Service and Application functions, at the power up (i.e. during the initialization) and periodically during the operation (nominal and degraded phases), e.g. test memories containing safety related data are totally tested at the initialization phase and at any new allocation and cyclically at run-time.
	HA_FLT_02	The Framework shall provide services for the detection of faults during the installation of the Applications software (otherwise, to be required to the Applications).
	HA_FLT_03	The Framework shall provide services for the detection of faults during the run-time execution of the Application function code (otherwise, to be required to the Application function), e.g. by monitoring the process and data flow and comparing their state to configured constraints (Program Flow Monitoring), by checking variables values against predefined range and for plausibility, by detecting and correcting errors in sensitive information (Error Detecting and Correcting Codes).
	HA_FLT_04	The Framework shall execute services for Fault detection, isolation, notification and reaction processes with the highest priority, without any disturb or unintended effects due to other Service and Application functions.
	HA_FLT_05	The Framework shall provide services for Fault detection and isolation without any disturb or unintended effects on the execution and performance (e.g. latency/jitter, sampling rate or resource reservation) of other Service and Application functions.
Detection of faults	HA_FLT_06	The Framework shall verify the capability to notify a Fault condition under a representative set of failure scenarios.

FAULT MANAGEMENT		
Classification	**ID**	**Countermeasures**
Action following Detection	HA_FLT_07	The Framework shall inhibit the execution of the Application function in case of negative results of the initial code integrity check.
	HA_FLT_08	The Framework, after the detection of a condition that blocks or threats the proper execution of Service or Application functions (fatal Fault), shall notify a Fault condition to all the Application functions involved, in a time that is compatible with their timely transition into safe state (i.e. not later than the maximum time for failure detection and negation specified by the Applications).
Independence of Items	HA_FLT_09	The framework shall manage the interaction between Service and Application functions: _avoiding that Service functions can force the outputs independently from the Application function when active, during operation (normal and degraded phases); _preventing the access to any off-line service (e.g. validation and verification support) at the power up, and during the initialization and the operating (nominal and degraded) phases; _guarantying the retention of a safe state after a fatal Fault (i.e. condition that blocks or threats the proper execution of Service or Application functions).
Systematic & Random faults	HA_FLT_10	The Framework shall detect, isolate, notify and react to fault with the highest SIL assigned to the safety-related Application functions to be executed.

Table 16: FDF PHA - countermeasures, Fault management function

3.3.3 *Application conditions*

Table 17 provides the list of the Application conditions specified during the Preliminary hazard Analysis, i.e. measures to be met by the hosted Application functions and by the interfaced external technical systems, in order to guarantee the safe functional operation and behaviour under fault conditions of the TCMS Functional Distribution Framework.

ID	Application condition
PHA_AC_01	The remote functions exchanging safety-data with and within the Framework shall: _implement safety protection in the generation of safety-data to be exchange through the transmission system; _verify the incoming messages in order to detect erroneous information (transmitter identity, type, value errors) and time errors (timing, sequencing error); _discharge a message when a communication error is identified; _react to the loss of valid communication, including tolerance of message errors if any, as for the notification of a fatal Fault.
PHA_AC_02	The Application function shall react to the notification of a Fault condition due to error in the execution of processes according to the scheduling plans (fatal Fault), implementing tolerance (e.g. timing bounds violated for a limited number of times) if any, by the transition into the specific safe state.
PHA_AC_03	The Application function shall react to the notification of a Fault condition due to error in the global time dissemination or functions synchronization (fatal Fault), implementing tolerance (e.g. errors for a limited number of cycles) if any, by the transition into the specific safe state.
PHA_AC_04	The Application function shall react to the notification of a Fault condition due to invalid operation in the partition attempts (fatal Fault), by the transition into the specific safe state.

ID	Application condition
PHA_AC_05	The Application function shall react to the notification of a Fault condition due to inconsistency between the values stored into an exchange variable and the status of the related platform's input / output (fatal Fault), by the transition into the specific safe state.
PHA_AC_06	Remote functions shall not use variables provided by the Framework's Monitoring functions (but Messages) for the execution of safety-related algorithms.
PHA_AC_07	The Application function shall react to any fatal Fault notified by the Framework (i.e. condition that blocks or threat its proper execution) through the transition and retention into its safe state, by blocking its safety-related functions and maintaining all outputs to their restrictive state (typically de-energized), till the execution of a defined maintenance procedure.
PHA_AC_08	The Application function shall react to the notification of a Fault condition due to error detected in the Configuration file or non-valid results coming from the inauguration (fatal Fault), by the transition into the specific safe state.
PHA_AC_09	External system shall provide a trusted time reference (external global clock), to be taken as reference by the Framework in the time synchronization.

Table 17: FDF PHA, Application conditions

3.3.4 *Recommendations*

Table 18 provides the list of Recommendations specified during the Preliminary hazard Analysis of the Functional Distribution Framework, which give some (non-mandatory) indications for the implementation of the countermeasures listed in §3.3.2.

ID	Recommendation
PHA_REC_01	It is recommended the compliance of the communication between remote functions with the EN50159 technical standard on Safety-related communication in transmission systems, for a Category 3 transmission system risk of unauthorised access to the transmission system not negligible).
PHA_REC_02	It is recommended to implement safety-related application functions in compliance with the EN 50129 technical standard on Safety related electronic systems for communication, signalling and processing systems. Specifically about the admitted architecture, according to the SIL assigned to the application, it is recommended: _a dual electronic structure based on composite fail-safety with fail-safe comparison or inherent fail-safe (highly recommended for >SIL2 applications); _a single electronic structure with self-tests and supervision (recommended for SIL 1 and SIL2 applications).
PHA_REC_03	It is recommended to execute services for faults detection at physical (e.g. temperature, voltage, memories failures), temporal and logical (e.g. error detecting codes, program sequence monitoring), and functional (e.g. configuration data integrity, spatial separation between resources) levels, at the power up (i.e. during the initialization) and periodically during the operation (nominal and degraded phases).
PHA_REC_04	It is recommended to implement means for the recognition of system status with respect to errors and failures that might occur or have occurred, supporting faults isolation and graceful degradation, in order to maintain the more critical Application functions available despite failures by dropping the less critical functions.

ID	Recommendation
PHA_REC_05	It is recommended to implement Validation and verification support service that allows fault injection and reaction monitoring, including faults of non-safety related Service and Application functions, partitioning and isolation mechanism, communication (transmission, reception) and sharing of network and memory resources, output control, input monitoring, application execution (timing, memory access, start, stop, throttling).
PHA_REC_06	It is recommended to avoid dynamic reconfiguration of software after a failure , i.e. remapping the logical architecture back onto the restricted resources left functioning (highly recommended for SIL3-SIL4 Applications, EN 50128 Table A.3).
PHA_REC_07	It is recommended to assess the implementation of messages retry mechanism by each Application functions, to improve dependability (tolerance of errors before transition into safe state) within safety constraints.

Table 18: FDF PHA, Recommendations

3.3.5 *CONNECTA functional requirements mapping*

The following table provides a mapping of the FDF system requirements as set out by CONNECTA in "D4.1 – Requirement specification for each sub task", chapter 4, CTA-T4.1-D-BTD-002-09, Rev. 9 and the FDF Software requirements that are proposed in Safe4RAIL:

CTA functional requirements	FDF Software Component	Remarks
Partition and process execution (CTA-D4.1-94)	ExecutionManager	The ExecutionManager of the FDF is responsable for handling the execution of partitions and processes. This manager grants processing resources to the partitions according to their Partition scheduling plan and guarantees the spatial separation between partitions. It is in charge of executing a partition with a corresponding period and time-interval, parameters which are provided in the configuration. It also executes processes according to their Process scheduling plan, enabling the concurrent or sequential execution of such. Fault isolation and memory protection are also provided by this component.
I/O services (CTA-D4.1-102)	IOManager	IOManager provides access to local analog and digital inputs as well as analog and digital outputs. By the use of this software component, the inputs are updated only before the execution of each of the partitions and the outputs written only after the complete execution of such.
Time services (CTA-D4.1-110)	SynchronizationManager	The required time services are handled by this manager. Each FDF node is synchronized based on a global clock for each consist. In order to update the system clock in each of the FDFs, this manager is able to suspend the execution of a partition. Besides, partitions can obtain this time by making use of the Synchronization Manager.
Communication services (CTA-D4.1-114)	VariableManager & MessageManager & NetworkManager	The communication between processes and partitions in the same ECU is handled by the VariableManager component whereas that between remote partitions is managed by the MessageManager and NetworkManager.
Replicate local variables on consist	VariableManager & MessageManager & NetworkManager	The variables are replicated on the consist network by the use of the VariableManager to handle variables, MessageManager to compose messages before being sent through the network

network (CTA-D4.1-117)		and the NetworkManager to actually transport the message in the network.
Control local variables based on consist network variables (CTA-D4.1-120)	VariableManager & MessageManager & NetworkManager	The same Managers as in the case above are used in order to modify a local output variable based on data received from the consist network.
Configuration (CTA-D4.1-123)	ConfigurationManager	The ConfigurationManager is used to check and load the configuration in order to configure the FDF. This configuration contains every parameter needed in the FDF, for instance: I/O data, description of variables and their parameters as, for instance, their default value; partition and process execution parameters.
Internal state monitoring and diagnosis (CTA-D4.1-132)	MonitoringManager & LogManager & HealthManager	The FDF provides an interface to allow an external device monitor internal variables by means of its MonitoringManager. The LogManager is used to log internal execution errors and every set of relevant information. Finally, the HealthManager is responsible for performing HW Integrity checks, monitoring partitions and processes and their executions, handling the refreshing of the watchdog.
Partition debugging (CTA-D4.1-140)	MonitoringManager	The FDF provides an interface to allow an external device monitor internal variables by means of its MonitoringManager. By the use of this manager variables can be forced and unforced for testing purposes.
Safety layer for consist network communications (CTA-D4.1-147)	MessageManager	The MessageManager adds the SDT safety layer to the Messages before being sent throughout the network.

Table 19: CONNECTA requirements – FDF Software Components mapping

Chapter 4 Security concept

4.1 Introduction

This section will describe a security concept for the FDF. The analysis made in this document is a personalised version of the methodology proposed by OWASP foundation[3] which allows customising a security solution based on standard technologies, with the aim of covering the foundational security requirements described in IEC 62443-3-3 "System security requirements and security levels" [13] and defining countermeasures subject of these requirements.

According to IEC 62443-2-1 [12], a high and detailed risk assessment is required. This security concept covers the high-level risk assessment together with the definition of countermeasures. It consists of examining what might be the impact of vulnerabilities and the likelihood that a threat might exercise these vulnerabilities but does not consider particular instances of these vulnerabilities. A representative use-case will be used to show the usage of the FDF, their main components and their relationships. This use case covers a representative example to determine assets to be protected, threats to which those assets are exposed, estimation of the risk and countermeasures.

Although some security countermeasures are already included in the FDF design, for instance, encryption and access control, this security concept will not consider them initially, but at the end, they will be evaluated to ensure correspondence to the security level as define in IEC 62443-3-3. The assurance of requirements based on target security level is mapped with IEC 62443-4-2 [14] that is required to get an ISASecure certification. This is the standard corresponding to a 'Component' within the group of standards of IEC 62443, since FDF is considered a component by definition in IEC 62443.

This security concept will be assessed in next Chapter by TÜV to early identifying possible issues in terms of security.

4.2 Motivation

In any domain, avionics, automotive, railway an industrial, the related safety standards define a generic engineering process for the creation of safety mechanisms to mitigate systematic and random faults in a system. However, these standards do not cover malicious attacks that can make the system failed. Sometimes safety is described as "protecting people from the system", whereas security refers to "protecting the system from people".

The use of an integrated platform with centralization of functionality means that applications will share resources (i.e. memory, file system, network interfaces, among others) that were previously physically separated. All of these can have a great impact on maintaining security for the system and users. Some applications, for instance, can store sensitive data, such as train positioning that must not be accessed by other applications. In addition to it, if granted access, it must be ensured that one application cannot deprive other applications of access to share resources. If various functional subsystems inside a consist are running on the same hardware, if a threat actor can get access to one subsystem of the train, it may also be able to take over another subsystem that can dramatically impact on the availability or result in loss of human lives. Other security issues in FDF can be gaining full privileges to install malicious

[3] OWASP Risk Rating Methodology. Available: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

software on it and then the injection of false sensor data to affect a safety-related application, or how to avoid buffer overflow attacks.

These are only some examples to bring out the importance of protecting the FDF against malicious or unintentional attacks. Further, all threats will be analysed. This security concept will deal with identifying security issues related to FDF and proposing countermeasures that mitigate these issues.

4.3 Scope

This security concept will only cover the security issues for the FDF aiming to protect FDF against any attack. Of course, the key is to address security measures in all layers (Figure 56), that is, for internal network communication, gateways and external communications. CONNECTA's D3.3 Report on RAMS and Security Analysis describes the risk assessment and security measures that seem to be necessary for supporting a DbD TCMS compliant with IEC 62443-3, covering networking aspects.

Since the use case of the bogie monitoring system is a distributed application that runs on top of two FDFs, the concept of end-to-end communication among them is considered but without any network device between them.

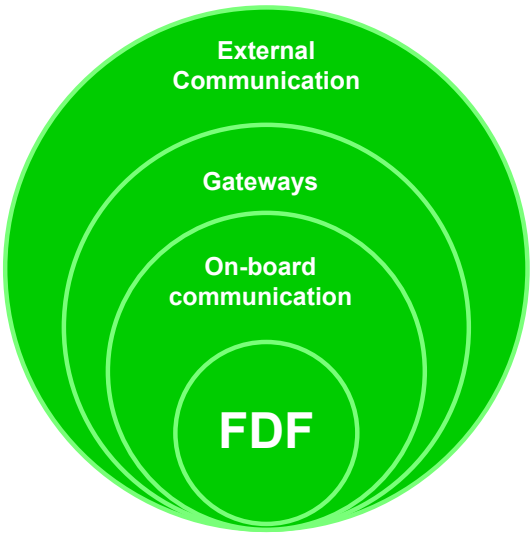

Figure 56. Logical View of the FDF with internal and external communication

Furthermore, only security aspects that cover the application and middleware services are considered. Security aspects for application development, operating systems including hypervisor solutions, hardware considerations, and other general security issues will not be covered. The following points will not be analysed:

- Application development: it is assumed that applications supplied from third parties will run on the same hardware, but these may contain vulnerabilities which can be used as attack vectors to the system.
- Patch management, that is, the maintenance and update of firmware, applications/FDF will be considered.
- Operating systems: there are three proof-of-concept instantiations: (1) INTEGRITY based, (2) PikeOS Hypervisor and (3) AUTOSAR-based [1]. (1) INTEGRITY is a real-time operating system and latest version is certified EAL 6+, High Robustness, (2) PikeOS achieved a Security Certification Evaluation according to French National Security Agency (but not Common Criteria EAL), in the case of (3) the AUTOSAR-based approach it depends on the Operating System host. The security concept will not address security issues related to the operating system or hypervisor.

- Hardware used for hosting the Operating System and FDF is beyond the scope.

An analysis of how safety measures can be potential sources of attacks is also beyond the scope of this security concept. For example, a safe-state condition can be exploitable by malicious attackers. When a safety-relevant failure condition is detected, the system shall go to a safe-state, and depending on the hazard nature, that safe-state can be from disabling a control function to applying emergency brakes to stop the train. Therefore, injecting a safety fault can launch a Denial-of-Service (DoS) attack.

4.4 Objective

The purpose of this security concept is to derive the justified set of security requirements, define countermeasures and assurance of 62443-4-2 requirements based on target security level. The mapping of these requirements with software components is also made in Annex B. A first step will be the identification of risks by analysing assets, the use case scenario and threats to obtain the security objectives. These security objectives are objectives needed to derive a secure design. The security objectives are based on an understanding of what risks the FDF might be exposed. These objectives will then be mapped to requirements. Next step will be assessing the severity of each risk if security is compromised. For each objective, an analysis of potential threats is performed together with an evaluation of the severity of risks based on Attack Potential and Damage Potential. Finally, security countermeasures will be proposed if risk exceeds the tolerable risk. IEC 62443-3-3 [13] will be follow since it provides guidance on countermeasures assigned to a category and a security level. Furthermore, coverage of the requirements versus countermeasures will be given.

The process is described in the following figure (Figure 57).

Figure 57. Steps in the security concept

Discovering threats is important but being able to estimate the associated severity of the risk is essential. The risk acceptance level is based on IKL's expertise for over around 7 years of experience. This methodology has been developed in collaboration with relevant European security actors and well-proven in many industrial projects. In Roll2Rail D2.4 and CONNECTA D3.3 a proposed methodology is given to measure the risk level. Although the parameters in our methodology are the same covered by these documents, the values assigned are quantified differently. Since results shall not be compromised at this time, we believe that as a first iteration, this well-proven methodology will be used and later if needed the required modifications will be addressed and a comparative analysis between both methodologies can be carried out.

By following this approach, it will be possible to estimate the severity of all risks and define countermeasures based on these rating risks: unacceptable risk (red), undesirable (orange), tolerable (yellow) and negligible (green).

4.5 Risk analysis – Security objectives

The purpose of this section is to define and describe the security objectives by identifying risks. This identification will be based upon the analysis of assets, use case scenarios and threats. Figure 58 shows the relation between assets and threats and how they are exposed to risks, based on likely damage and potential of occurrence.

Later, in the following section threats corresponding to each security objective will be assessed.

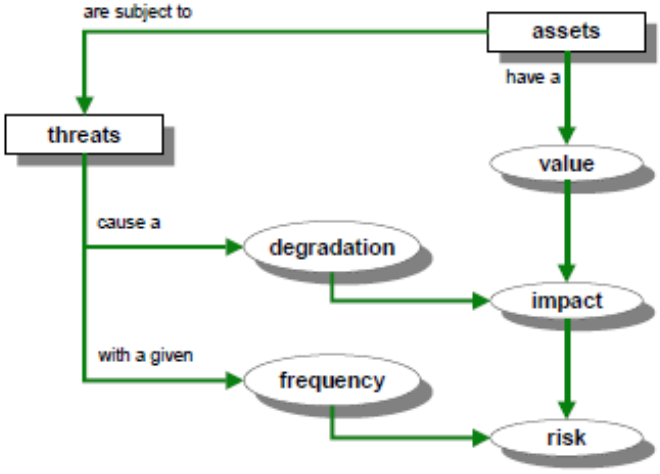

Figure 58. Elements for potential risk analysis. Source: Magerit [10].

4.5.1 FDF brief description

This section tries to give a brief summary of the FDF that serves as a basis for the understanding of what we need to protect. The idea behind it is that this security concept can be reviewed or assessed separately from the rest of this deliverable. Any reader already familiar with FDF may want to skip this section.

The goal of FDF is to offer an execution environment that enables:

- Hosting multiple TCMS application functions, safety-critical and non-safety functions
- Ensuring strict time/space partitioning
- Provision of abstraction from location, underlying network protocols and hardware

In order to achieve it, it provides the following internal services:

- Initialization
- Scheduled execution of applications
- Execution of applications of different SIL
- Safe local data distribution
- Safe and secure data distribution
- Transparent IO reading and writing
- Health monitoring
- Remote monitoring
- Maintenance

TCMS applications: TCMS provides a single point of control over all train subsystems related to three categories: Safety-related functions, Operator Oriented Services, and Customer Oriented Services, such as:

- Bogie Temperature Monitoring
- Door control
- Braking system
- HVAC
- Lightning
- CCTV
- Passenger information system
- Others

The final idea is that train manufacturers or subsystem function providers can build their one application and run it on top of the FDF. This FDF will provide the following services with a defined API:

- **FDF Services:** The code of these components is portable across different Platforms/OS because the Hardware Access Service and OS Service layers provide well-defined interfaces.
- **OS Services:** These components have the same interface but different implementations for each Platform/OS. They provide either a complete implementation of the services or an adapter to the services provided by the underlying OS.
- **HW Access Services:** These components have the same interface but different implementations for each IO and NIC Hardware. They provide either a complete implementation of the services or an adapter to the services provided by the underlying Drivers.

4.5.2 Security dimensions or attributes

A security dimension or attribute is an aspect that allows the value of an asset to be measured in the sense of the damage that would be caused by its loss of value. These dimensions cover not only the CIA triad (Confidentiality, Integrity and Availability), but also secondary dimensions like authenticity, accountability, and non-reputability.

- **Availability:** Readiness of the services and data to be used when necessary. Lack of availability causes an interruption of services.
- **Integrity:** Maintenance of completeness and correctness of data. Without integrity, information may appear to be altered, corrupt or incomplete.
- **Confidentiality:** Information must only reach authorised persons. Lack of confidentiality or secrecy could cause leaks of information as well as unauthorised accesses.
- **Authenticity** (who uses the data or services): An entity is who claims to be or guarantee the source from which the data originated. Against the authenticity of the information, we can have manipulation of origin or data. Against the authenticity of users/applications accessing services, we can have spoofing.
- **Accountability**: Guarantee that it will always be possible to determine who did what and when. Accountability is essential to analyse incidents, prosecute attackers and learn from experience. Accountability maps into the integrity of activity logs.
- **Non-repudiability:** also known as auditability, is the property of two entities not being able to deny processing of information sent/received.

All these attributes may or may not be required to be met depending on the situation. The risk analysis will determine the effort is required to achieve them.

4.5.3 System assets

The objective of this section is to identify the assets composing the system. An asset[4] is a component or function that may be subject to deliberate or accidental attacks that may have consequences for the organisation. Assets include information, data, services, applications (software), equipment (hardware), communications, media, facilities, and personnel. There are several ways to characterise assets.

In this security concept, two types of assets will be only considered: primary assets and secondary or supporting assets. Critical functions/services and data offered by an application over the FDF are primary assets; whereas, these are handled or run by supporting assets, such as software and hardware devices. Generally, secondary assets are the ones that will potentially receive security countermeasures to protect the primary ones (Figure 59).

PRIMARY SYSTEM ASSETS		
Name	**Functional description**	**Physical location**
non-critical and critical functions	This is related to application(s) that run on FDF either safety critical (i.e. SIL-4 braking system, SIL-2 door control, SIL-2 boggie monitoring) or non-critical (i.e. multimedia services, HVAC function)	FDF/Vehicle Control Unit (VCU)
non-critical and critical data	Data used by the safety-critical function(s)	FDF
SECONDARY SYSTEM ASSETS		
Name	**Functional description**	**Physical location**
Computing	Computing platform for procesing tasks in FDF	CPU or hardware
Communication	Send/receive messages from/to network	wired bus or Ethernet
Communication interfaces	communication interfaces for sensing and actuating on the system (I/O)	USB, Ethernet, Serial Port
Inputs	Get measurements from the environment	wired sensors
Outputs	Actuate on the system	wired actuators
Storage	variables, databases, data files (log, historical, configuration,..)	Memory (RAM, ROM, flash)
Application software	Implemented application on FDF	Software and data
FDF software	FDF services provided for development of applications (i.e. FDF access to hardware, operating system,...)	Software and data
Cryptographic keys	cryptographic keys for authentication and de/Encryption	Memory or specific hardware (hardware
External equipment	Power supply	Power cable and facilities
Engineering computer	PC or similar dedicated to build the FDF with different applications and load it into the final hardware	PC or workstation
Personnel	Configuration or maintenance services	Site

Figure 59. Primary and secondary system assets

4.5.4 Use case: Bogie Monitoring System

The analysis of use cases is helpful to state possible threats and then determining the security objectives. The bogie monitoring system has been selected as an example of an application running on top of the FDF. This application fulfils the European standard EN15437-2 [27] for onboard continuous temperature monitoring according to detect failures in axlebox bearings.

[4] [UNE 71504:2008]

4.5.4.1 General Description

This onboard temperature monitoring is a SIL2 functionality to improve safety and train operation reliability based on EN15437-2. According to the standard, "failed wheelset bearings on rolling stock create a hazard to the safe operation of the railway. If the bearing fails while rolling stock is in service there is the potential for a catastrophic event. One indication that a bearing is about to fail is a rise in the heat generated by the bearing." To mitigate the mentioned hazard, train manufacturers install on-board temperature monitoring systems that detect hazardous temperature levels and activate predefined alarms accordingly.

This system consists of some sets of temperature sensors, located in wheelset bearings, which send temperature data to a processing unit. In this unit, temperatures are compared to predefined thresholds and different alarms are activated according to the hazard severity.

4.5.4.2 Operational Description

There is no a unique way of implementing this bogie monitoring system in the FDF. The chosen implementation covers a complex and representative example of a distributed application running in two different control units. Figure 60 describes a logical and physical view of the Bogie Monitoring System Application (BMSA). The upper part of the figure shows how ECU1 and ECU2 could be placed at the consist or the train backbone level.

Each partition is mapped to a SIL, and it allocates only processes with that SIL, based on requirements of the architecture. In this case, for the sake of simplicity, it is assumed that only one process will run in a partition, but both SIL0 processes could have been within a SIL0 partition.

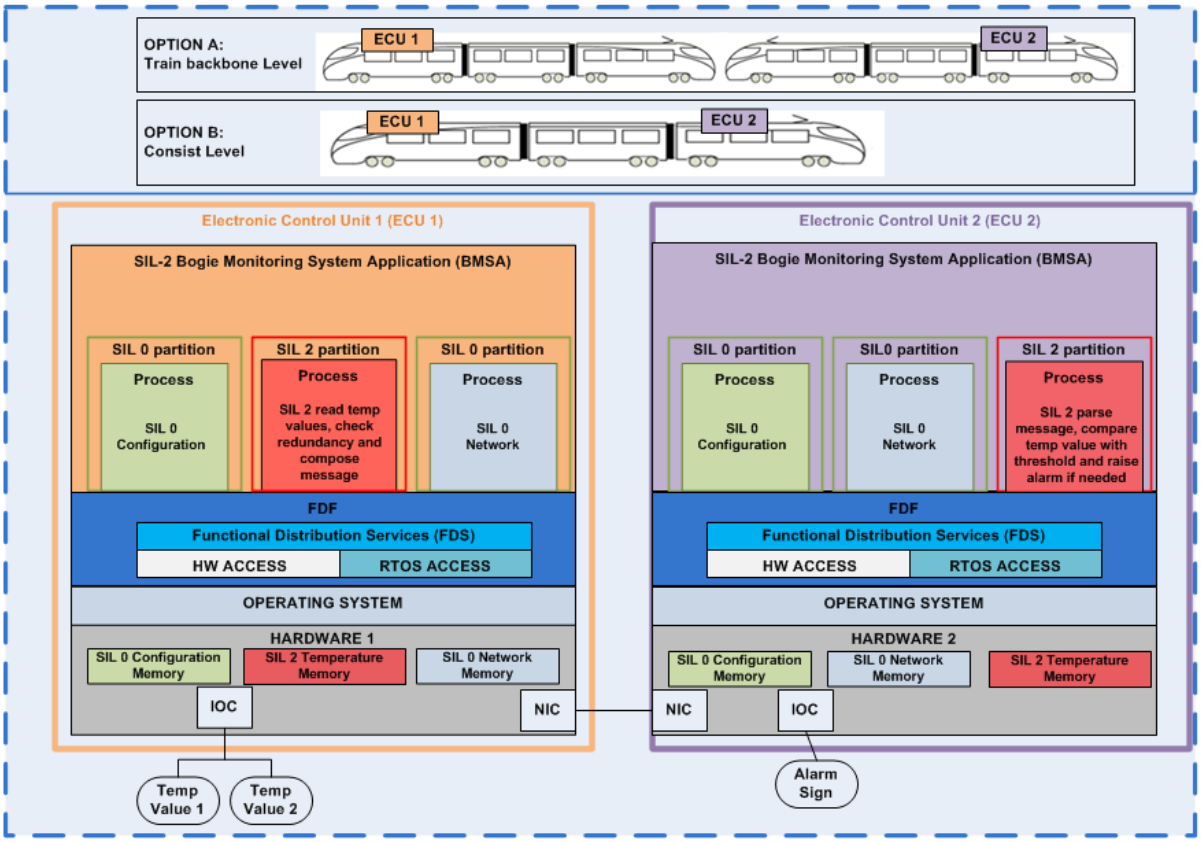

Figure 60. Logical and Physical view of the bogie monitoring systems (BMSA).

Processes are represented in order of execution in both ECUs, and each process is mapped to a piece of memory. Here a brief description starting from left, ECU1:

- SIL0 Configuration: this process is in charge of the reading configuration file for this application (i.e. variable names, functions ...), which will be saved in "SIL 0 Configuration Memory". This is valid for both ECUs. This configuration is used for configuring applications the first time. Then, these processes stay in 'idle' mode.
- ECU1's SIL2 Read temperature values check redundancy and compose a message: The temperature of bearings is measured using two redundant sensors in ECU1, the SIL2 process reads and checks these values, and save them in "SIL 2 temperature memory" and composes a message to be sent.
- ECU1's SIL0 Network: this process is in charge of sending this message to the ECU2.
- ECU2's SIL0 Network: receives the message.
- ECU2's SIL2 parses the message, save the value in "SIL2 Temperature Memory" compares it with given thresholds and based on the result, a "warm alarm" or a "hot alarm" can be caused depending on temperature range.

4.5.4.3 Assets Used

Two temperature sensors, shared memory, messages, FDF software components, ECU1 and ECU2, are the main assets for this use case.

Both ECU1 and ECU2 are applications with a SIL2 partition and SIL0 partition, respectively. In ACU, the SIL2 partition is responsible for reading and storing temperature values from both sensors in variables and the SIL0 partition consists of a process to send the corresponding message to the network. In VCU, the process running on SIL0 partition will decompose this message in two variables that the Application function will compare these variables together with threshold temperature values and necessary alarms will be fired if required. The following table shows each asset involved in this use case together with a description.

Name	Functional Description
Computing	Two control units ECU1 and ECU2
Communication	Remote data distribution with messages between ECU1 and ECU2
Communication interfaces	communication interfaces for getting temperature measures
Inputs	temperature measures
Outputs	Warm or hot alarms
Storage	• variables and messages in Shared memory (TemRedVarStrSIL2, TemRedMsgStr, TemAlrVarStrSIL2, TemAlrMsgStr), • Configuration files for different processes: TemConfCfg, TemMainCfg and TemMsgCfg
Application software	BMSA in both ECUs
FDF software	FDF services provided: IOManager, IODriverManager, NICDriverManager, FrameworkManager, MemoryManager, ExecutionManager, MessageManager, VariableStoreManager, Configuration Manager, FunctionManager, and more

Table 20: Security concept - Assets used

4.5.4.4 Possible Threats/Attacks

This section describes the different threat scenarios in the FDF that occur due to the nature of the FDF and the end-to-end communication between different FDFs without any

networking devices. Security objectives have been derived by analysing FDF architecture, the use case and threats and they are introduced in the following subsection (Section 4.5.5). Each threat will be mapped to the corresponding security objective, affected asset and dimensions. The access type for all of them is direct physical access.

Tamper FDF Data

With no authentication, tampering can be done by many means of:

- A compromised application with access to the network to send messages with fake data. This could tamper with temperature values for attacking the ACU. This could make the VCU think that the bearings temperature is high and fired a hot alarm to stop the train.
  - Security objectives: SO1: Authorized use of the FDF, SO2: Restricted access to ECU instructions.
  - Assets: outputs, storage, communication
  - Dimensions: Integrity, Availability
- Modify system data, for example, temperature thresholds
  - Security objectives: SO1: Authorized use of the FDF, SO4: Data authentication and encryption
  - Assets: storage, outputs, Application software
  - Dimensions: Integrity, Availability
- Change FDF configuration by modifying corresponding configuration files
  - Security objectives: SO1: Authorized use of the FDF, SO4: Data authentication and encryption
  - Assets: storage, outputs, Application software
  - Dimensions: Integrity, Availability
- Installing an application and corrupting assigned memory space can lead to access to the system or interfere with safety functions
  - Security objectives: SO1: Authorized use of the FDF, SO3: Application isolation
  - Assets: Application software, storage, outputs
  - Dimensions: Integrity, Availability
- A compromised application tampers with stored logging/auditable data
  - Security objectives: SO1: Authorized use of the FDF, SO3: Application isolation
  - Assets: Application software, storage, outputs
  - Dimensions: Authenticity, Accountability, non-repudiability

Denial of service attacks (DoS)

In general, the following denials of service attacks are envisioned:

- Reduce bandwidth by sending dummy data: using third-party tools it is possible to bring the system down by flooding it with large amounts of traffic. By overloading the network with dummy data and clogging up, the communication channels could affect messaging among other SIL applications. However, the countermeasures will

be from the TSN-based DbD, so that excess data sent does not affect other critical flows. In any case, an authorised use of FDF is required.

- Security objectives: SO1: Authorized use of the FDF
- Assets: Communication
- Dimensions: Availability

- Breaking down the system by sending huge data Again, this countermeasure will be given by the DbD, but authorized use of the FDF shall be ensured.
  - Security objectives: SO1: Authorized use of the FDF
  - Assets: Communication
  - Dimensions: Availability
- Unauthorized application can delete key storage files; the use of FDF services to delete sensitive data shall be authorized.
  - Security objectives: SO1: Authorized use of the FDF
  - Assets: Storage, outputs
  - Dimensions: Availability, Integrity
- Unauthorized application can shut down or modify clock synchronisation enabling a DoS attack
  - Security objectives: SO1: Authorized use of the FDF, SO2: Restricted access to ECU instructions
  - Assets: FDF software, outputs
  - Dimensions: Availability

Man-in-the-middle attack (MITM)

This refers to intercepting information sent between two FDFs. In general, using third-party tools, it is possible to listening, intercepting, altering, injecting or replacing messages between ACU and VCU. FDF should be able to authenticate the source of the communicated data, encrypt/decrypt outgoing/incoming data and validate that received data has not been modified in transit.

- Security objectives: SO1: Authorized use of the FDF, SO5: Trusted Message Exchange
- Assets: communication, storage
- Dimensions: Authentication,

Collect sensitive information

If an authorised application, or by means of accessing memory or intercepting messages, can track train's location, train configuration, collect keys and passwords, audit or log data. The information disclosure threat of non-sensitive data on the VCU or ACU does not result in any injuries or have any operational impact.

- Security objectives: SO4: Data authentication and encryption, SO5: Trusted Message Exchange
- Assets: communication, storage
- Dimensions: Confidentiality

Buffer overflow attack

Since there is no authentication, implementation errors can be exploited like buffer overflow, and if an application got affected by this type of attacks, this application should not affect the others either running on the same FDF or in another one.

- Security objectives: SO3: Application isolation
- Assets: communication, storage
- Dimensions: Availability

CPU manipulation attack

A third-party tool or an application can have access to CPU instructions to shut down the system, changing the clock, or to manipulate time-based resources to ensure 99% of total CPU cycles are used by a blocking application that can affect SIL functions.

- Security objectives: SO2: Restricted access to ECU instructions
- Assets: computing
- Dimensions: Availability

False alerts

The use of fake sensors can represent an abnormal behaviour of the system.

- Security objectives: SO6: Trusted input/output devices
- Assets: inputs
- Dimensions: Integrity

4.5.5 Security objectives

The security objectives consist of a set of short and clear statements to get a high-level solution to the security problem. The following security objectives have been derived from the analysis of assets, use cases and threads.

- **SECURITY OBJECTIVE 1: Authorized use of the FDF:** all the applications running over FDF needs to be authorised since this lack of authentication and authorisation represents the largest attack surface.

- **SECURITY OBJECTIVE 2: Restricted access to ECU instructions:** that for example can change global ECU state, ECU shutdown, clock synchronisation, process isolation shall be guaranteed. Only selected applications are allowed to request certain instructions (OS and HW services provided by FDF) that can dramatically affect to the system, for example, the creation of partitions, application installation, and so on.

- **SECURITY OBJECTIVE 3: Application isolation:** secure memory partitioning must enable running different applications in separate memory partitions and avoiding interference between them. Therefore, protection of data and state of functions during applications execution will be achieved. In that way, safety applications will not be corrupted or interfered by non-safety related applications. However, extra measures must be taken to ensure that an application cannot corrupt its own assigned memory space. The reason is that an attacker may be able to gain access to the system by this corrupted memory.

- **SECURITY OBJECTIVE 4: Data authentication and encryption** are needed for providing security operation. Configuration files, variables, messages and other data shall be encrypted. In addition to it, message authentication is needed to confirm that a message comes from a certain sender and encrypted for confidentiality.

- **SECURITY OBJECTIVE 5: Trusted Message Exchange:** any data exchange operation shall be carried out assuring that involved actors are allowed, and in addition to it, message authentication is needed to confirm that a message comes from a certain sender and encrypted for confidentiality. Temperature values sent between ACU and VCU shall be securitized.

- **SECURITY OBJECTIVE 6: Trusted input/output devices:** restricting access to memory and memory-mapped hardware shall be used for controlling hardware peripherals by reading from and writing to registers or memory blocks mapped to system memory. Physically disabling or removing connection ports and I/O devices help prevent disclosure of information or the introduction of malicious code into the system.

4.6 Security requirements

In this chapter, the security requirements of the FDF are defined. These requirements have been extracted from D2.5 "Report on requirements of next-generation TCMS framework", and these will be mapped with the security objectives already defined and new ones will be derived in Section 4.8.

Id	Text	SIL	SL
S4R_FDF_409	The framework shall operate accordingly/with regards to confidentiality • Ensure that data inside the framework cannot be read by an unauthorised entity: ensure non-disclosure of information/data towards entities (i.e. users, processes, and device) unless a successful access authorisation.	N/A	N/A
S4R_FDF_410	The framework shall operate accordingly/with regards to authenticity • Assurance of entities' identity • Ensure/verify data source: information/data comes from a verified and trusted entity (sender) • Information collected by the framework should be authentic with respect to origin and time if the framework performs actions based on that information • The author of the message, respectively the origin sending entity of the information/data, shall be evident and traceable at any time (with regards to non-repudiation)	N/A	N/A
S4R_FDF_411	The framework shall operate accordingly/with regards to data integrity • Support/offer mechanism(s) in order to ensure data integrity for information collected within the framework. • Ensure that the information has/have not been modified either in transit or storage on the route from the sender's entity to the receiver's entity.	N/A	N/A
S4R_FDF_412	The framework shall provide cryptographic mechanisms and handle cryptographic objects • Ensure framework's security as well as framework's communication channel (receiving and transmitting role) by means of secure cryptographic algorithms • Management of cryptographic keys (creation, deletion and retention) • Calculation of cryptographic functions (digital signatures, MACs, encryption/decryption)	N/A	N/A

S4R_FDF_413	The framework shall provide a Public Key Infrastructure (PKI) • Support/ensure the authentication process of entities (with regards to authenticity) • Management of certificates (retention and update)	N/A	N/A
S4R_FDF_414	The framework shall secure the incoming/outgoing communication (channel) to the ECUs (Electronic Control Units) against security threats with regards to confidentiality, authenticity, integrity and availability while respecting real-time constraints (i.e. predictable latency and low jitter).	N/A	N/A
S4R_FDF_415	Support/availability of access control in the network to ensure robustness to DoS attacks as well as side-channel attacks.	N/A	N/A
S4R_FDF_416	The framework shall protect stored data against adversaries (with regards to confidentiality, authenticity and data integrity).	N/A	N/A
S4R_FDF_417	The framework shall include a mechanism in order to prevent unknown/unexpected traffic (i.e. admission and access control).	N/A	N/A
S4R_FDF_418	The framework shall support secure storage for the key(s) and trust anchor(s) for secure authentication and communication (with regards to security services and authenticity).	N/A	N/A
S4R_FDF_419	The framework shall operate with authenticated entities (ECUs, SW/HW components) only (with regards to authenticity) • The framework shall enforce authenticity and integrity of the ECUs in order to meet/fulfil framework's security requirements. • The framework shall enforce authenticity and integrity of the software components in order to meet/fulfil framework's security requirements.	N/A	N/A
S4R_FDF_420	The framework shall accomplish the need of protecting the data and state of the functions during execution on an ECU.	N/A	N/A
S4R_FDF_421	The framework shall accomplish the need of protecting the data and state of the functions during execution within software components.	N/A	N/A
S4R_FDF_422	The framework shall ensure the data isolation between different partitions created and maintained by the framework so that the data in a partition is accessible only by code running in that partition.	N/A	N/A
S4R_FDF_423	The framework shall ensure the isolation of the resource between different partitions created and maintained by the framework so that the resources exported by the framework into a partition are accessible only by code running in that partition.	N/A	N/A
S4R_FDF_424	The framework shall provide information flow control that enforces strict partition isolation so that only explicitly configured interaction is allowed.	N/A	N/A
S4R_FDF_425	The framework shall ensure that a failure in one partition is not propagated to other partitions.	N/A	N/A
S4R_FDF_426	The framework shall ensure that an attack affecting one partition is not propagated to other partitions.	N/A	N/A
S4R_FDF_427	The framework shall ensure that security policy enforcement functions are non-bypassable.	N/A	N/A
S4R_FDF_428	The framework shall ensure that security policy enforcement functions are always invoked.	N/A	N/A

S4R_FDF_429	The framework shall ensure that security policy enforcement functions and the data that configures them cannot be modified without authorisation.	N/A	N/A
S4R_FDF_430	The framework shall provide the capability to detect, generate and export audit records for security relevant auditable events.	N/A	N/A

Table 21: Security Requirements of FDF.

This table is a proof that the security objectives are in line with the system requirements and cover them. It also demonstrates that there is no requirement for trusted control of input and output devices.

SO	Requirements
SO.1: Authorized use of the FDF	S4R_FDF_410, S4R_FDF_422, S4R_FDF_423, S4R_FDF_424, S4R_FDF_427, S4R_FDF_428, S4R_FDF_429, S4R_FDF_430
SO.2: Restricted access to ECU instructions	S4R_FDF_414, S4R_FDF_415, S4R_FDF_425, S4R_FDF_426
SO.3: Application isolation	S4R_FDF_410, S4R_FDF_413, S4R_FDF_419, S4R_FDF_427, S4R_FDF_428
SO.4: Data authentication and encryption	S4R_FDF_409, S4R_FDF_410 S4R_FDF_414, S4R_FDF_416, S4R_FDF_418
SO.5: Trusted Message exchange	S4R_FDF_414, S4R_FDF_417
SO.6: Trusted input/output devices	No requirements

Table 22: Security objective coverage.

4.7 Risk assessment

Before providing any specific solution, a classification is needed to rate and evaluate each of the risks. Discovering threats is important, but being able to estimate the associated severity of the risk is as essential. The risk acceptance level is based on IKL's expertise for over around 7 years of experience. This methodology has been developed in collaboration with relevant European security actors and well-proven in many industrial projects. In Roll2Rail D2.4 and CONNECTA D3.3, a proposed methodology is given to measure the risk level. Although the parameters in our methodology are the same covered by these documents, the values assigned are quantified differently. Since results shall not be compromised at this time, we believe that as a first iteration, this well-proven methodology will be used and later if needed the required modifications will be addressed and a comparative analysis between both methodologies can be carried out.

Ideally, there would be a universal risk rating system that would accurately estimate all risks for all organizations. However, a vulnerability that is critical to one organization may not be very important to another that is why a customized 'Risk Rating Method' is used to make the risk estimation.

For every Security Objective, a list of threats are defined and analysed with two main aspects in mind, attack potential and damage potential. As mentioned before, categorised values are based on the normal set of prerequisites used in this kind of developments and how are in general configured this kind of hardware/software artefacts by system architects.

From the attack potential's point of view, the following characteristics have been evaluated: the elapsed time to perform an attack, the needed expertise, type of information obtained, ease of access to the target, and the needed equipment/resources to perform the attack. From the damage potential point of view, the following characteristics have been evaluated: personal damage level, operative damage and financial damage.

After the risk rating analysis, which applies certain rules and formulas, security estimation is obtained for both the attack potential and the damage potential. Combining them, a risk value is obtained for every threat previously identified. The following table shows the results of a preliminary analysis.

Colours are also used to identify and prioritise potential risk at first sight. The red colour will identify a non-desired situation where actions should be taken urgently in order to avoid or mitigate it. For example, the usage of a cryptographic library version that is already known breakable. Orange is used to mark undesirable situations, and although the potential risk is high, it can be considered that the system could be safe in normal running. For instance, hard drive failure, where actions should be taken adding a mirror or redundant disk solutions, but we can consider that hard drive failure is statistically calculated and hardly happened under normal circumstances. Yellow is used to identify tolerable situations, for example, not using session timeout could be used by a hacker to take control over the system modifying or deleting data, stopping vital processes etc. but if we consider that to access the device terminal is quite difficult because someone has to pass some security doors inside a guarded building, it is not crucial to have a terminal in session closed each time people takes a break. Finally, green colour is used to identify normal and trivial situations.

4.7.1 *Security Level Target*

The security level target (SL-T) is the level of protection a system must provide against the threats to a system, and it is a measure of confidence that the system is free from vulnerabilities and functions in the intended manner. These are the security levels proposed by IEC 62443.

- **SL1** - protection against casual or coincidental violation
- **SL2** - protection against intentional violation using simple means with low resources, generic skills and low motivation
- **SL3** - protection against intentional violation using sophisticated means with moderate resources, specific skills and moderate motivation
- **SL4** - protection against intentional violation using sophisticated means with extended resources, specific skills and high motivation

The asset owner should define the security level target based on the risk level that is considered tolerable. In this case, the analysis of risks (Figure 61) shows that the type of attacker and means derive a SL3 or SL4. Now an analysis of appropriate countermeasures corresponding to that SL will be described.

4.7.2 *Determination of the severity of the risk*

Damage potential for each threat is been estimated, but train manufacturers from CONNECTA should verify this assesment, and maybe risk value could changed.

Cybersecurity		Attack Potential (worst case)					Damage Potential (worst case)			Cybersecurity Risk Estimation				
Security Objetive	Attack	Elapsed Time	Expertise	Information about the target	Access to Target	Equipment	Personal Damage	Operative Damage	Financial Damage	Attack Potential		Damage Potential		Risk Value
SO.1 Authorized use of FDF	Session hijacking	Months	Multiple Experts	Critical	Difficult	Specialized	Severe and life -threatening injuries (survival possibl	Maintenance require	< 100.000 euros	Beyond High-Rare	43	Catastrophic	1020	Undesirable
	FDF manipulation	Months	Expert	Critical	Difficult	Multiple Bespoke	Severe and life -threatening injuries (survival possibl	Unusable	< 1.000.000 euros	Beyond High-Rare	46	Catastrophic	1200	Undesirable
	Authorization/Privileges modification	Months	Expert	Critical	Difficult	Specialized	Severe and life -threatening injuries (survival possibl	Maintenance require	< 100.000 euros	Beyond High-Rare	41	Catastrophic	1020	Undesirable
	Passwork hack	Months	Proficient	Restricted	Moderate	Standard	Severe and life -threatening injuries (survival possibl	Comfort affected	< 100.000 euros	High	20	Catastrophic	1011	Undesirable
	Power failure	Hours	Layman	Public	Difficult	Standard	No effect	Maintenance require	< 10.000 euros	Enhanced-Basic	11	Medium	10	Undesirable
	Hard drive failure	Hours	Expert	Public	Difficult	Standard	No effect	Unusable	< 10.000 euros	Moderate	17	Critical	100	Undesirable
	User supplantation	Weeks	Proficient	Restricted	Moderate	Standard	No effect	Comfort affected	< 100.000 euros	Moderate	14	Medium	11	Undesirable
SO.2 Restricted access to ECU instructions	CPU manipulation	Months	Expert	Critical	Difficult	Multiple Bespoke	Severe and life -threatening injuries (survival possibl	Unusable	< 1.000.000 euros	Beyond High-Rare	46	Catastrophic	1200	Undesirable
	Data injection/deletion	Months	Expert	Critical	Difficult	Multiple Bespoke	Severe and life -threatening injuries (survival possibl	Unusable	< 100.000 euros	Beyond High-Rare	46	Catastrophic	1110	Undesirable
SO.3 Application isolation	Data corruption	Months	Proficient	Restricted	Difficult	Multiple Bespoke	Severe and life -threatening injuries (survival possibl	Maintenance require	< 100.000 euros	Beyond High-Rare	35	Catastrophic	1020	Undesirable
	Network flooding	Years	Expert	Sensitive	Moderate	Multiple Bespoke	Severe and life -threatening injuries (survival possibl	Maintenance require	< 100.000 euros	Beyond High-Rare	46	Catastrophic	1020	Undesirable
SO.4 Data authentication and encryption	Breach of cryptography	Years	Multiple Experts	Critical	Difficult	Specialized	Severe and life -threatening injuries (survival possibl	Maintenance require	< 100.000 euros	Beyond High-Rare	53	Catastrophic	1020	Undesirable
	Collect sensitive information (keys, logs)	Months	Expert	Critical	Difficult	Specialized	Severe and life -threatening injuries (survival possibl	Maintenance require	< 100.000 euros	Beyond High-Rare	41	Catastrophic	1020	Undesirable
SO. 5 Trusted message exchange	Message injection	Months	Expert	Critical	Difficult	Multiple Bespoke	Severe and life -threatening injuries (survival possibl	Comfort affected	< 100.000 euros	Beyond High-Rare	46	Catastrophic	1011	Undesirable
	Man-in-the-Middle	Months	Multiple Experts	Sensitive	Moderate	Multiple Bespoke	Severe and life -threatening injuries (survival possibl	Comfort affected	< 10.000 euros	Beyond High-Rare	38	Catastrophic	1001	Undesirable
SO.6 Trusted input/output devices	Peripheral device supplantation	Months	Multiple Experts	Public	Difficult	Bespoke	Severe and life -threatening injuries (survival possibl	Maintenance require	< 10.000 euros	Beyond High-Rare	35	Catastrophic	1010	Undesirable
	Port tampering	Hours	Expert	Public	Difficult	Bespoke	No effect	Maintenance require	< 10.000 euros	High	24	Medium	10	Tolerable
	Ethernet tampering	Hours	Expert	Public	Difficult	Bespoke	No effect	Maintenance require	< 10.000 euros	High	24	Medium	10	Tolerable
	Repudiation	Hours	Expert	Public	Difficult	Bespoke	No effect	Maintenance require	< 10.000 euros	High	24	Medium	10	Tolerable

Figure 61. Severity of risk.

4.8 Security countermeasures

Once the security objectives and security level target have been identified, selected countermeasures are introduced in order to overcome previously described quantified risks addressing already defined security requirements and based on security level target. Attacks identified involved getting physical access to the FDF and high expertise to perform them. Therefore all countermeasures shall be aligned with a high-Security Level 3 (SL) intentional with moderate resources and moderate motivation or SL4 intentional using sophisticated means with extended resources and high motivation. In this case, SL3 has been selected, but the asset owner should define which security level target based on what needs to be protected.

Initially, all countermeasures will be defined, and in next section, they will be mapped with the assigned countermeasures defined in 62443-4-2 for Embedded Device Security Assurance (EDSA) certification.

Each countermeasure is linked to specific requirements and security objectives to see coverage.

4.8.1 *Countermeasure 1: Trusted Platform Module (TPM)*

C1: A hardware security chip or Trusted Platform Module (TPM) is a tamper-resistance computing chip that can securely store artefacts used to authenticate, such as, passwords, certificates and cryptographic keys. The countermeasure would be used in combination with a crypto USB or smartcard token in which personnel and applications certificates can be stored to be used for public key authentication, PIN support, user-defined key restriction (i.e. one-time password, a limited number of usage) and key audit counter (i.e. counts down with each key usage).

In this website, a list of certified TPMs can be found https://trustedcomputinggroup.org/membership/certification/tpm-certified-products/ . It is important to pay attention to the following features to select one:

- at least SHA-256 for hashing because MD5 and SHA1 are broken
- Cryptographic algorithms AES (preferable) or 3DES.
- OTP memory: one-time programmable non-volatile memory, non-volatile storage for cryptographic keys, secrets, and authorisation information
- Crypto Accelerator
- RNG: Random Number Generator – a strong password generator is required with high randomness
- UID: Unique Identifier Designator that guarantees to be unique among all identifiers
- Antitamper: with anti-tampering measures (i.e. active shields, anti-DPA) and in general anti-SCA (Side-Channel Attacks).
- Certification: FIPS or Common Criteria

FDF can use this technology for identification and authentication ECUs and applications, encryption, secure key storage and integrity verification.

So, due to C1:

- S4R_FDF_409, S4R_FDF_410, S4R_FDF_411, S4R_FDF_412, S4R_FDF_413, S4R_FDF_414, S4R_FDF_415, S4R_FDF_416, S4R_FDF_418, S4R_FDF_419, requirements are fulfilled.
- S4R_FDF_409 related to confidentiality, data encryption is used with this countermeasure
- S4R_FDF_410 and S4R_FDF_411 related to authenticity and data integrity within FDF, this can be achieved by means of a MAC or digital signature. A MAC is an algorithm that mathematically combines a key with a hash function to simultaneously verify both data integrity and message authentication.
- S4R_FDF_412 is related to providing cryptographic mechanisms and handle cryptographic objects. With the TPM, secure cryptographic functions (digital signatures, MACs, encryption/decryption) will be used together with secure storage and management of cryptographic keys (creation, deletion and retention).
- S4R_FDF_413 related to the use of a PKI. A USB token can be used by certification and registration authorities to generate user and application certificates using an external PKI.
- S4R_FDF_414 about securing the incoming/outgoing communication (channel) to the ECUs
- S4R_FDF_415 related to access control can be used with USB or smartcard-based tokens.
- S4R_FDF_416 related to protect stored data against adversaries
- S4R_FDF_418 related to secure storage for key(s) and trust anchor(s) for secure authentication and confidentiality
- S4R_FDF_419 related to operate with authenticated entities (ECUs, SW/HW components
- SO1: Authorized use of the FDF: only previously identified, validated and correctly authorised user or applications will be the only ones that can use the FDF
- SO2: Restricted access to ECU instructions: TPM will ensure user and application authentication checking and validating certificates.
- SO4: Data and file authentication and encryption: is also fulfilled because sessions will be end-to-end protected from the very beginning stages of the communication, authenticating parties using their personal certificates stored in the TPM, and in the ongoing phase, encrypting the used data transfer channel using the shared key certificates.

4.8.2 *Countermeasure 2: Password policy*

Username and password are required worldwide in order to avoid any user impersonation and to login a system and communicate between software components. Password robustness is also required to avoid any password hacking method. Detection of this attack method, for instance blocking the system when a fixed number of wrong passwords are typed, is also a way of improving security. Instead of username and password, there could also be used certificates as credentials in order to demonstrate who it is, person or application component.

Therefore, covering this aspect of security that is, limiting access to trusted users only to the FDF/OS with robust passwords, and as consequence restricting and tailoring the

accessible functions to them, the global security NIST recommendation[5] for digital identity guidelines shall be ensured.

It is recommended to enable password expiration.

This measure contributes to:

- Requirements: new requirement "USER MANAGEMENT: All users need to be identified and authenticated for all access to the FDF. Authentication of the identity of such entities should be accomplished by using methods such as stronger passwords, tokens or location (physical or logical)." This requirement is mapped to 62443-4-2 requirements related to Access Control (Section 4.9).
- SO1: Authorized use of the FDF: only previously identified, validated and correctly authorised user or applications will be the only ones that can use the FDF

NIST provides some guidelines for password policies https://pages.nist.gov/800-63-3/sp800-63b.html?ncid=txtlnkusaolp00000618. For example, some of them:

- The username and password texts in the security component are case-sensitive to increase the security level.
- The username in the security component must only comprise letters and numbers and must start with a letter to prevent possible issues with different usage environments
- The password in the security component can include letters, numbers and punctuation signs to increase the security level
- The password in the security component must include at least one letter and one number to increase the security level
- The password text in the security component shall not be visible on the user interface as this might mean a security vulnerability
- Unsuccessful login attempts shall be considered.

4.8.3 *Countermeasure 3: User profiles and application profiles policies*

Access to different services and data (including file systems) offered by FDF shall be restricted based on user and application profiles. Therefore, rules to determine which actions they are allowed to perform and their restrictions to access resources such as hardware (e.g., memory, network) or software (execution of programs or commands) should be taken into account to define and assign proper permissions to different user or application.

The system must implement a security policy that specifies who or what may access a file system, and type of access permitted: for example, R-Read, W-Write, X-Execute and Supervisor/User mode. Moreover, there could be policies to enable: runtime, deployment, and so on.

The least privilege shall be applied.

Moreover, the following features should be used:

[5] 800-63-3 Digital identity guidelines, published June 2017, https://doi.org/10.6028/NIST.SP.800-63-3

1. Session timeout: On the other hand, once the system has granted one session, it should control if it continues online in the long term, and if not, the session should be closed after the established time-out for inactivity is triggered.
2. Concurrent session control: Concurrent session control for our use case amounts to controlling the number of sessions a user can have at the same time.
3. User/Application expiration: The security administrator can indicate if a user profile expires or not. By means of a smartcard or USB token this renewal can be performed easily.

This measure contributes to:

- Requirements: S4R_FDF_409
- SO1: Authorized use of the FDF: only previously identified, validated and correctly authorized user or applications will be the only ones that can use the FDF.
- SO2 : Restricted access to ECU instructions

4.8.4 *Countermeasure 4: Role-based access control (RBAC)*

A role-based access control shall be used to restricting of FDF access to only authorized users based on roles and permission. User roles can be assigned depending on specific operations, such as FDF admin, operator, application function deployer, maintenance person, and so on. Each role will have different permissions/privileges, for example, the FDF administrator will have rights to edit system files, access network, edit user profiles and application profiles, and edit configuration files; whereas the operator will only have access to diagnostics data.

Roles such as, administrator with full privileges, and other with fewer privileges, such as, application developer, operator and maintenance person shall be considered. Roles have to be assigned to users so upon successful authentication of the user, they are authorizes as having the privileges associated with the assigned role.

Administrator user role shall be able to create other user accounts and manage their privileges, always applying the least privilege philosophy.

Applications shall be also configured with different privileges, for example, to restrict network, hardware, operating system based on application's role.

Users and applications have to be categorised in roles allowing a RBAC security paradigm, and the least privilege shall be applied.

This measure contributes to:

- Requirements: S4R_FDF_414, S4R_FDF_415, S4R_FDF_416, S4R_FDF_421, S4R_FDF_427, S4R_FDF_428, S4R_FDF_429
- SO1: Authorized use of the FDF: only previously identified, validated and correctly authorized user or applications will be the only ones that can use the FDF.
- SO2 : Restricted access to ECU instructions

4.8.5 *Countermeasure 5: Encryption*

Apart from using secure channels to transfer data, the transferred sensitive data itself should be encrypted prior to send it. In that way, a double security level is achieved in data

transfer channels between an FDF and another system or FDF. If the secured channel is compromised, as data is encrypted, it could be almost impossible to interpret the data.

In the case of FDF, it needs to be considered whether all data stored and messages shall be encrypted due to performance reasons, or whether only confidential or sensitive data that is susceptible of being compromised shall be encrypted.

This measure contributes to:

- Requirements: S4R_FDF_412, S4R_FDF_414, S4R_FDF_416
- SO4 Data authentication and encryption, SO5 Trusted Message Exchange

4.8.6 *Countermeasure 6: Session bindings*

Once authentication has taken place, it is desirable to continue using application/services over time without requiring authentication. To facilitate this behaviour, a *session* may be started in response to an authentication event, and continue the session until such time that it is terminated. Session management is preferable over the continual presentation of credentials. There are several mechanisms for managing a session over time; in this case, a session binding seems to be desirable. A session secret is shared between application and service being accessed. This secret binds the two ends of the session, allowing the application to continue using the service over time. This secret can be given using the TPM.

This measure contributes to:

- Requirements: S4R_FDF_414, S4R_FDF_417
- SO1: Authorized use of the FDF, SO2 : Restricted access to ECU instructions, SO5 Trusted Message Exchange

4.8.7 *Countermeasure 7: Network limited bandwidth*

Usually, the first barrier used where data transfer is carried out in some kind of network is a firewall. A firewall can help filtering connections from known and unknown sources to reduce the incoming traffic to the system. Nevertheless, due to hardware and/or software restrictions and specifically in embedded devices, it is not possible to install and use a firewall as in a desktop computer.

The measure that can be used is to enforce bandwidth limitation at the application or FDF level, together with the corresponding limitation of bandwidth at the network components. The use of internal network ports should be tailored and restricted (closed) as well as a firewall does create specific rules for incoming data. Whitelisting can also be defined to accept communications from different applications, but everything else is denied. If the communication does not appear on the whitelist, the communication is rejected. It is preferable to deny all traffic and permit only that traffic that is necessary. This security model is known as Deny All Permit Exception. In general, this is a more secure posture than using a blacklist that permits everything and blocks only traffic that someone decides is bad. All allowed traffic shall be logged for audit purposes. Although some comments address the network level, as stated this is beyond the scope of this security concept.

By means of Ethernet TSN, the monitoring and control of traffic is achieved to secure and protect critical traffic, together with physical network segmentation.

This measure contributes to:

- Requirements: S4R_FDF_417

- SO5 Trusted Message Exchange

4.8.8 *Countermeasure 8: Asset inventory*

An asset inventory of all systems connected to FDF, including inputs, outputs, network, network devices, network addresses, machine names, purpose of each system, asset owner responsible for each of them. Authentication of all these devices shall be performed, for example to network level to determine authorised versus unauthorised systems.

Furthermore, restricting access to memory and memory-mapped hardware shall be used for controlling hardware peripherals by reading from and writing to registers or memory blocks mapped to system memory. Physically disabling or removing connection ports and I/O devices help prevent disclosure of information or the introduction of malicious code into the system.

This measure contributes to:

- Requirements: S4R_FDF_415
- SO6 Trusted input/output devices

4.8.9 *Countermeasure 9: Software-based memory protection unit*

The FDF shall prevent read/write access to an application's memory from non-trusted applications. Moreover, FDF may prevent non-trusted applications from executing code.

A solution can be on memory partitioning based on Memory Protection Units. INTEGRITY and PikeOS provide these mechanisms to isolate special and temporal partitions.

- Requirements: S4R_FDF_415
- SO3 Application isolation

4.9 Functional Security Assessment Requirements

Initially, all countermeasures will be defined, and in next section they will be mapped with the assigned countermeasures defined in 62443-4-2,. This standard applies to FDF because it satisfies clause 3.1.15 "special purpose device running embedded software designed to monitor, control or actuate an industrial process directly".

ISASecure Embedded Device Security Assurance (EDSA) is a certification program that includes:

- Communication Robustness Testing (CRT)
- Functional Security Assessment (FSA)
- Software Development Security Assessment (SDSA)

These countermeasures will cover only FSA.

IEC 62443 defines seven foundational requirements

- **Access control (AC):** guarantees that all users (people, software processes and devices) must successfully identify and authenticate so that they are allowed to access the system.

- **Use control (UC):** The use of a certain equipment, information, or both will be monitored to prevent unauthorised operation of the device or unauthorised use of information.
- **Data integrity (DI):** The integrity of data in certain communication channels is safeguarded to prevent unauthorised data exchange.
- **Data confidentiality (DC):** Communication channels are protected from eavesdropping, in order to guarantee the confidentiality of specific data.
- **Restricted dataflow (RDF):** The data flow in communication channels is restricted in order to prevent unauthorised lowering before the disclosure of information.
- **Timely response (TRE):** In case of violations of IT security there will be a reply through a notification within a defined period, and corrective activities are initiated.
- **Resource availability (RA):** The availability of all network resources is ensured for protecting against service attacks denials.

The complete table of Functional Security Assessment Requirements can be found in ANNEX B: Functional Security Assessment Requirements table. For each foundational requirement, a set of functional security assessments is given for FSA (Figure 62). Each requirement is assigned with a SL to fulfil. At this time, there is no evidence of any component certified for security level 3 by ISA.

Reference ID and Name	*ISASecure*™ Level
Access Control	
└— FSA-AC-1 Access Control Authorization	NA
└— FSA-AC-1.1 Role Based Access	>1
└— FSA-AC-1.2 Dual Approval Access	>1
└— FSA-AC-1.3 Least Privilege Default Access	>1
└— FSA-AC-1.4 Administrator User Role	>1
└— FSA-AC-1.5 Administrator Support Functions	>2
└— FSA-AC-2 User Authentication	NA
└— FSA-AC-2.1 Authentication by User ID and Password	NA
└— FSA-AC-2.1.1 User Management of Password	All
└— FSA-AC-2.1.2 Monitor Unsuccessful Login Attempts	All
└— FSA-AC-2.1.3 Record Successful Logins	All
└— FSA-AC-2.1.4 Display Previous Login History	>2
└— FSA-AC-2.1.5 Password Modification Reminder	>2
└— FSA-AC-2.1.6 Password Strength Enforcement	>2
└— FSA-AC-2.1.7 Action for High Number of Unsuccessful Login	All
└— FSA-AC-2.1.8 Minimum Password Capability	All
└— FSA-AC-2.1.9 Clear Text Passwords	All
└— FSA-AC-2.1.10 Cryptographic Password Protection	>2
└— FSA-AC-2.1.11 Access Control for All Exposed Services	All
└— FSA-AC-2.2 Other Authentication Methods	Not required
└— FSA-AC-2.3 Two Factor Authentication (local network)	>2
└— FSA-AC-2.3 Two Factor Authentication (remote)	All
└— FSA-AC-2.5 Authentication Feedback	All
└— FSA-AC-3 System Use Notification	All
└— FSA-AC-4 Local Session Locking Timeout	>1
└— FSA-AC-5 Remote Session Termination Timeout	>1
Use Control	
└— FSA-UC-1 Wireless Access	NA
└— FSA-UC-1.1 Physical Disable Wireless Access	All
└— FSA-UC-2 Device Authentication	NA
└— FSA-UC-2.1 Failures in Cryptology Services	All
└— FSA-UC-2.2 Basic Device Authentication	>1
└— FSA-UC-2.3 Cryptographic Device Authentication	>2
└— FSA-UC-3 Creation of Audit Trail	NA
└— FSA-UC-3.1 Configuration of Audit Events	>2
└— FSA-UC-3.2 Content of Audit Record	NA
└— FSA-UC-3.2.1 Time Stamp for Audit	>1
└— FSA-UC-3.2.2 Information for Non-repudiation	>2
└— FSA-UC-3.2.3 Additional Content for Audit Record	>2
└— FSA-UC-3.3 Protection of Audit Information	NA
└— FSA-UC-3.3.1 Audit Fault Warning	>2
└— FSA-UC-3.3.2 Basic Protection of Audit Information	>1
└— FSA-UC-3.3.3 Crypto Protection of Audit Information	>2
└— FSA-UC-3.4 System Wide Audit	>2
└— FSA-UC-3.5 Audit Report Generation	>1

Figure 62. Excerpt of Functional Security Assessment requirements for ISASecure certification for two functional requirements 'Access Control' and 'Use Control'.

In ANNEX B: Functional Security Assessment Requirements table, each defined requirement is mapped to:

- Existence of a requirement in D2.5.
- Countermeasure that is applied to fulfil it.
- ISASecure Level that is required for each system requirement
- Software component that shall cover the associated requirement, or any clarification.
- Security Objectives

4.10 Conclusions and next steps

This section has provided a process to analyse and assess risks to protect FDF, including security objectives. After this identification and assesment, countermeasures were described to address all security objectives. In some countermeasures, a selection of criteria to choose between commercially available solutions was described.

The most important countermeasure is that a certified Trusted Platform Module (TPM) should be included in the hardware platform to securely store encryption keys, passwords, FDF authentication, or any other sensitive data.

Apart from that, Section 4.9 describes all system assurance requirements needed to satisfy level 2-3-4 (there is no distinction among them), and they were mapped to already defined requirements in D5.2, countermeasures, software components to be in charge of, and security objectives.

This assurance is not all covered by already defined requirements in D2.5, they were classified like 'None'. An update of that deliverable is needed.

As a result of this analysis, new software components were created:

- User Account Manager: the lack of authentication and authorisation represents the largest attack surface, so there is a need of user account management.
- Crypto Manager: responsible for encryption, decryption, key generation and management, encoding, decoding.
- Security Monitoring Manager: takes care of the most security functions namely: user authentication, access authorization, application deployment, session control, reporting, recovery, and so on.

Furthermore, it was shown that the security objectives would not cover the cases of system backup and recovery.

Chapter 5 Assessment of the safety and security concepts

Within this chapter the assessment of the two concepts related to safety (see Chapter 3) and security (see Chapter 4) is provided. Section 5.1 contains information about the requirements and expectations on the FDF design to support safety and security aspects. In part 5.2 the assessment of the safety concept is shown and in section 5.3 the results for the security concept is displayed.

5.1 Requirements given to the FDF design

Section 5.1.1 contains basic information about the functionality of the FDF and the expectation provided by CONNECTA in form of system requirements. The capabilities of the FDF design derived from the standards is displayed in section 5.1.2 for safety aspects and paragraph 5.1.3 for security.

5.1.1 *General*

The FDF design shall support and provide all characteristics and functionality that is necessary to implement train control and monitoring systems (TCMS) on an integrated modular platform (IMP), like:

- Execution of applications
- Communication between applications
- Control of input/ output lines
- Deployment of applications
- Debugging of applications
- Testing of applications
- Certification of applications

In supporting and providing the above characteristics and functionality the FDF design shall fulfil the system requirements as set out by CONNECTA in "D4.1 – Requirement specification for each sub task", chapter 4, CTA-T4.1-D-BTD-002-09, Rev. 9.

The functional requirements of CTA-T4.1-D-BTD-002-09 relate to:

- Partition and process execution (CTA-D4.1-94)
- I/O services (CTA-D4.1-102)
- Time services (CTA-D4.1-110)
- Communication services (CTA-D4.1-114)
- Replicate local variables on consist network (CTA-D4.1-117)
- Control local variables based on consist network variables (CTA-D4.1-120)
- Configuration (CTA-D4.1-123)
- Internal state monitoring and diagnosis (CTA-D4.1-132)
- Partition debugging (CTA-D4.1-140)
- Safety layer for consist network communications (CTA-D4.1-147)

The non-functional requirements of CTA-T4.1-D-BTD-002-09 relate to:

- Implementation support for functions with a safety integrity level (SIL) up to 4 according to IEC 61508 standard (CTA-D4.1-151).
- Implementation support for functions with a security level (SL) up to 4 according to IEC 62443-3-3 standard (CTA-D4.1-152 ff.)

5.1.2 *Safety*

To comply with the non-functional safety requirements (CTA-D4.1-151) the FDF design shall be capable to fulfil the requirements set up by software safety standards that are common to the railway domain. The FDF design shall therefore follow or provide the highly recommended general techniques and means with the rigidity SIL4.

The highly recommended general techniques and means from EN 50657:2017 [11], EN 50128:2011 [3] and IEC 61508-3:2010 [6] applicable to a software framework as FDF can be summarized as follows:

- Fault detection and diagnosis shall be used and supported.
- Error detecting codes shall be used and supported.
- Failure assertion programming shall be used and supported.
- Divers programming shall be used or at least supported.
- Backward and forward recovery shall not be used.
- The memorizing of execution paths and the detection and reaction on unlicensed execution paths shall be supported.
- Artificial intelligence fault correction shall not be used.
- Dynamic reconfiguration of software shall not be used.
- Graceful degradation in the event of failure shall be used and/ or supported.
- Software interfaces shall be fully defined.
- Information encapsulation shall be used and supported. The software design shall follow a modular approach.
- The timely behaviour of the software shall be guaranteed. This shall be achieved ether by a cyclic behaviour with guaranteed maximum cycle time or by a time-triggered architecture. An event-driven architecture shall not be used.
- Resources shall be allocated statically
- Access to shared resources shall be synchronized and the synchronization shall be configured statically.

5.1.3 *Security*

To comply with the non-functional security requirements (CTA-D4.1-152 ff.) the FDF design shall be capable to fulfil the system requirements that are given in the IEC 62443-3-3 standard up to security level capability 4 (SL-C 4). The system requirements relate to:

- Identification and authentication control (CTA-D4.1-256)
- Use control (CTA-D4.1-260)
- System integrity (CTA-D4.1-262)
- Data confidentiality (CTA-D4.1-261)

- Restricted date flow (CTA-D4.1-259)
- Timely response to events (CTA-D4.1-258)
- Resource availability (CTA-D4.1-257)

5.2 Assessment of the safety concept

This section focuses on the assessment of the safety concept of the TCMS Functional Distribution Framework (FDF). In the first section 5.2.1 the expectation on the concept is provided. In the following chapter 5.2.2 the approach to fulfil the single requirement or question as identified during the assessment is shown. The result of the evaluation for each question is provided in paragraph 5.2.3.

5.2.1 *Requirements*

Based on the requirements mentioned in section 5.1.1 and 5.1.2 the main questions for a technical safety concept are:

QSAF1:

Are the safety requirements that are defined sufficient (on a concept level), are they in line with the system requirements and do they cover the requirements given by the relevant safety standards?

QSAF2:

Is the safety architecture and design concept capable to fulfil the safety requirements?

5.2.2 *Approach and findings*

For the safety concept (see Chapter 3), a preliminary hazard analyses (PHA) of the FDF has been performed. The functional model of the FDF the analyses was based on a set of dedicated fundamental safety related FDF services (see Table 1) and FDF functions (see Table 2). System hazards (see Table 7) have been identified, allocated to the different FDF functions of the functional model and for each function countermeasures have been specified (see section 3.3.2). As additional result of the PHA application conditions (see section 3.3.3) and recommendations (see section 3.3.4) have been defined. A mapping of CTA FDF system requirements to FDF software components has been established, but there is no direct link to the fundamental FDF services used in the PHA. The safety requirements and the safety architecture is implicitly available in form of the defined countermeasures, application conditions and recommendations.

Evaluation on a representative example:

The PHA lead to a potential hazard which has been identified as "FDF_SH_04" (missed or incorrect input), which is allocated to the FDF "Input/Output function" (see Table 7: FDF PHA, List of System Hazards and relevant FDF Functions and deviations). Within Table 12: FDF PHA - countermeasures, Input/Output function the measure to mitigate the risks like "FDF_SH_04" of the FDF input/output function are specified. In Table 19: CONNECTA requirements – FDF Software Components mapping the relevant FDF software component for the FDF input/output function is mapped to the related system requirement to provide information that the coverage is given.

5.2.3 *Appraisal*

Ad. QSAF1:

The safety requirements, available as countermeasures and application conditions, are defined sufficient on a concept level. They are in line with the system requirements and cover the requirements given by the relevant safety standards to a sufficient degree for a concept.

Ad. QSAF2:

The mitigation measures are a plausible set of measures to achieve the safety goals. The architecture is capable to provide a basis for the implementation of the measures compliant to the safety standards and to fulfil the safety requirements. The assessment did not identify safety related problems in the available concept data.

5.3 Assessment of the security concept

This section focuses on the assessment of the security concept of the TCMS Functional Distribution Framework (FDF). In the first section 5.3.1 the expectation on the concept is provided. In the following chapter 5.3.2 the approach to fulfil the single requirement or question as identified during the assessment is shown. The result of the evaluation for each question is provided in paragraph 5.3.3.

5.3.1 *Requirements*

Based on the requirements mentioned in section 5.1.1 and 5.1.3 the main questions for a technical safety concept are:

QSEC1:

Are the security requirements that are defined sufficient (on a concept level), are they in line with the system requirements and do they cover the requirements given by the relevant security standards?

QSEC2:

Is the security architecture and design concept capable to fulfil the security requirements?

5.3.2 *Approach and findings*

For the security concept (see Chapter 4), the use case "boogie monitoring system" has been analysed (see section 4.5.4) to identify possible threats and attacks (see section 4.5.4.4). The outcome of the analyses was used to derive the relevant security objectives (see section 4.5.5). With help of the existing security requirements of deliverable D2.5 the newly identified security objectives have been validated (see section 4.6). In the following a risk assessment to evaluate the potential severity of the security objective (see section 4.7) was performed. From the preceding activities, countermeasures (see section 4.8) have been defined and security requirements established (see section 4.9).

Evaluation on representative examples:

The analyses of the use case lead to a potential threat which has been identified is "Tamper FDF Data". Two security objectives have been allocated to this item: "SO1: Authorized use of the FDF" and "SO2: Restricted access to ECU instructions" (see 4.5.4.4). According to Table 22: Security objective coverage., several requirements of D2.5 "Report on requirements of next-generation TCMS framework" can be allocated to the threat. As a result of the risk assessment (see Figure 61. Severity of risk.) both security objectives are mainly rated as "catastrophic" and "undesirable". The proposed countermeasures are as defined in chapter 4.8

"Trusted Platform Module (TPM)" (C1: see 4.8.1), "Password policy" (C2: see 4.8.2), "User profiles and application profiles policies" (C3: see 4.8.3), "Role-based access control (RBAC)" (C4: see 4.8.4), "Software update policy" (C5: see 4.8.5), and "Session bindings" (C7: see 4.8.6). Following the information in chapter 4.9 the countermeasures Cx are partly covered by the set of existing requirements and several new ones have to be established to comply with the standard and to fulfil the system requirements.

The analysis of the use case lead to a potential threat which has been identified is "False alerts". One security objective has been allocated to this item: "SO6: Trusted input/output devices" (see 4.5.4.4). According to Table 22: Security objective coverage., no requirement of D2.5 "Report on requirements of next-generation TCMS framework" can be allocated to the threat. As a result of the risk assessment (see Figure 61. Severity of risk.) the security objective is mainly rated as "medium" and "tolerable". The proposed countermeasure is as defined in chapter 4.8 "Asset inventory" (C9: see 4.8.8). Following the information in chapter 4.9 the countermeasure C9 is not covered by an existing requirement and at least a new one has to be established to comply with the standard and to fulfil the system requirements.

5.3.3 *Appraisal*

Ad. QSEC1:

The security requirements that are defined are sufficient on a concept level. They are in line with the system requirements and cover the requirements given by the relevant security standards to a sufficient degree for a concept.

Ad. QSEC2:

The countermeasures are a plausible set of security measures to achieve the security goals. The architecture is capable to provide a basis for the implementation of the measures compliant to the security standards and to fulfil the security requirements. The assessment did not identify security related problems in the available concept data.

In general:

As stated above, the security requirements are sufficient to meet a concept level. Gaps were discovered in the course of concept development based on the specific use case. Although the use case chosen is a representative example since it is covered the most complex case, in which an application runs in two different control units, the recommendation is to analyse further use cases in order to discover and close further gaps.

Chapter 6 Integration of the Framework in the IMP

In the end, the FDF can be understood as an abstraction layer under the applications which provides, among other functionalities, memory and execution partitioning, local and remote data distribution and Input and Output Management. This layer needs to collaborate with the Drive-by-Data technology, in order to achieve a global synchronisation of the different ECUs in the train and be able to send and receive packets by the use of Real-Time communication, mainly for critical data, and Best-Effort communication. Both technologies together conform what we call the Integrated Modular Platform (IMP) and, in this relationship, the FDF needs to provide every instance's configuration to the DbD, which will then suppose a constraint for this last. DbD will have to collect the configuration of every single FDF instance connected to the network and design a plan to satisfy every ECUs needs, i.e., concrete data at exact points in time.

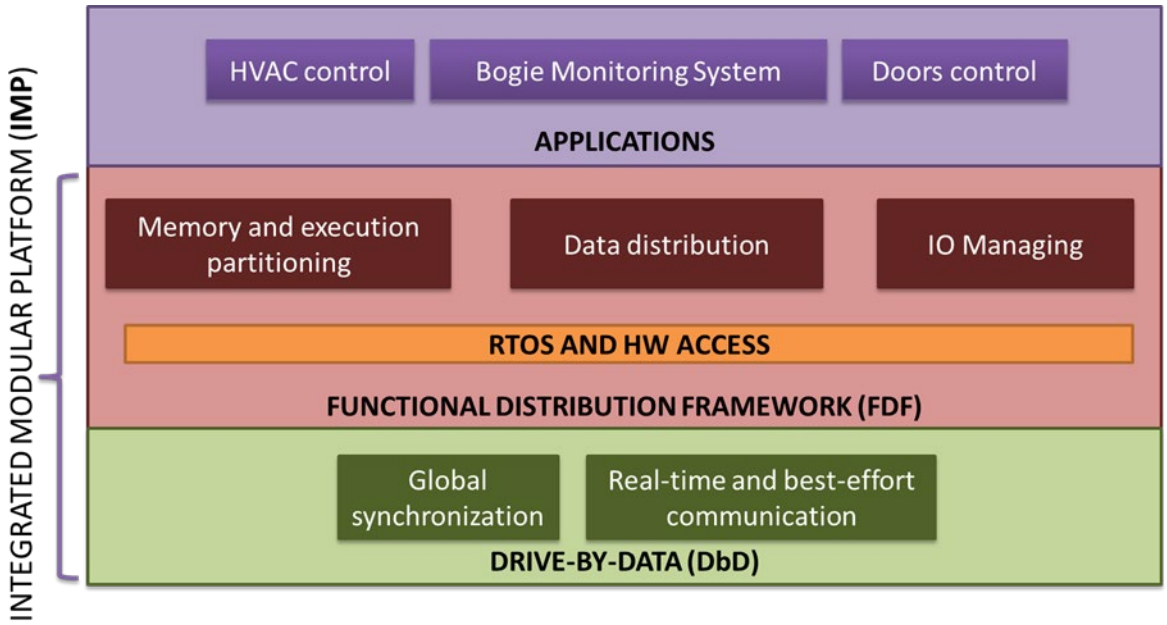

Figure 63. Integrated modular platform overview.

Apart from the configuration dependency, the real challenge when integrating FDF with DbD will reside in ensuring that the correct operation of the DbD Hardware is achieved when coupling it with the corresponding End Device or ECU containing the FDF, as well as making the necessary adaptations to get the correct interaction between the DbD driver and the FDF, i.e., making both technologies actually be able to communicate with each other. This means the interfaces must be well-defined and, concretely, the DbD driver must correctly communicate with the FDF. In such context, it must be given a special attention to the integration of specific communication middleware (e.g. TRDP), in the end, FDF's NetworkManager, with the NICDriverManager component, which will be a wrapper function providing the services with the DbD driver will offer.

Chapter 7 Summary and conclusion

As expressed before, this document focuses on the Design, Safety and Security concept proposed for the TCMS Functional Distribution Framework (FDF). The main goal is to provide the "Functional Distribution" architecture concept for a mixed criticality embedded platform, offering an execution environment for distributed TCMS safe and secure applications up to SIL4.

Chapter 1 explains the motivation for creating the Functional Distribution Framework and explains its main characteristics of the same. Chapter 2 describes the proposed "Design concept", from conceptual, structural and behavioural points of view.

In such a context, starting from a common "initial concept" of the TCMS Framework, Chapter 3 provides the results coming from the Preliminary Hazard Analysis. They include the list the potential hazardous conditions in the execution of a generic safety-related Application function, due to deviation(s) in the execution of the FDF's Functions and Services (i.e. System hazards). The set of measures required to assure safe functional operation of the hosted Application functions and safe behaviour under fault conditions, defines the Safety concept of the TCMS Functional Distribution Framework against the above "System hazards".

A set of "Countermeasures" has to be implemented by the Framework in order to guarantee its proper functional operation, detection of faults, action following detection, independence of items and defence against systematic & random faults. A further set of (non-mandatory) "Recommendations" provides indications for the implementation of Countermeasures. Further activity (out of the scope of this deliverable) will verifiy that the proposed physical and logical elements (i.e. the FDF Design concept) can implement the safety measures (i.e. Countermeasures and Recommendations). Besides, a final set of "Application conditions" has to be implemented by the hosted Application functions and by the interfaced external technical systems.

Chapter 4 comes next, where by the use of a security concept, after analyzing the risks and assessing them, it has been identified what is necessary to protect. For the security concept, the Bogie Monitoring System was chosen as a representative example since covered the most complex case, that is, a distributed application running in two different control units. The security concept is still valid for simpler TCMS applications. The security objectives give us an idea about what we need to protect based on the use case and assets analyzed. The security level target, based on attacker expertise and means, should be 3 or 4, although so far there is no evidence of any component certified by 62443-4-2 with such a level, level 2 is the maximum. Countermeasures were introduced to cover security requirements and objectives. The assurance of these requirements and countermeasures for IEC 62443 certification was also addressed. As a result, new requirements were placed in D2.5, by the use of the DOORS tool at Ikerlan ensuring a correct version management and traceability (as described in D2.5 Chapter 2). For instance, new software components such as, "User Account Manager", "Crypto Manager" and "Security Monitoring Manager" were introduced in the design and in the design instantiation (D2.4).

The results of the assessment of the safety concept and the security concept which are provided in Chapter 3 and Chapter 4, respectively, are shown in Chapter 5. During the activity, it was evaluated if the concepts and results sufficiently covered the given system requirements and applicable standards. The assessment did not identify any safety or security related problem in the available concept data.

On the whole, the Functional Distribution Framework aims to have isolated but integrated applications instead of dedicated equipment for each train function as well as make possible to run applications up to SIL4. The benefits are the reduction of the number and complexity of equipment, the abstraction from the Hardware and communication and the interoperability,

which facilitates an easier certification. In fact, this safe functional distribution architecture aims to achieve an interference freeness which would enable the capability of a modular certification. The evidence of such capability will be detailed inD2.7, which is confidential. . The proposed architecture for the FDF allows for portability across different platforms by providing well-defined interfaces so that the different groups of Software Components can communicate with each other and also makes the interaction with the Drive-by-Data technology possible in order to compose the Integrated Modular Platform. Moreover, the architecture is capable to provide a basis for the implementation of the measures compliant to the safety and security standards and to fulfill the corresponding safety and security requirements. It is important to stress that the proposed reference architecture and its Safety and Security concepts were shared and contrasted with CONNECTA project, made out of representative railway manufacturers, in order to be in line with their needs and expectations. The members were constantly informed about the progress of these activities and provided valuable feedback on them [6].

[6] Evidence of such feedback can be found in deliverables that are confidential

Chapter 8 List of Abbreviations

ACU	Application Control Unit
API	Application Programming Interface
BIT	Built-in-shelf-Test
BMSA	Bogie Monitoring System Application
BSP	Board Support Package
CAPEX	CAPital EXpenses (initial investment)
CIA	Confidentiality, Integrity and Availability
COTS	Commercial off-the-shelf
CRC	Cyclic Redundancy Check
DbD	Drive-by-Data
DI	Data Integrity
DNR	Domain Name Resolver
DoS	Denial of Service
EAL	Evaluation Assurance Level
ECN	Ethernet Consist Network
ECU	Electronic Control Unit
ED	End Device
EDS	Embedded Device Security Assurance
ERTMS	European Railway Traffic Management System
ETB	Ethernet Train Backbone
ETBN	ETB Node (also referred to as Train Switch)
ETCS	European Train Control System
FDF	Functional Distribution Framework
FSA	Functional Safety Assessment
IMP	Integrated Modular Platform
IO	Input/Output

IP	Internet Protocol
LLDP	Link Layer Discovery Protocol
MAC	Media Access Control
MD	Message Data
MMU	Memory Management Unit
MVB	Multifunction Vehicle Bus
NIST	National Institute of Standards and Technology
NWIP	New Work Item Proposal
OEM	Original Equipment Manufacturer
OS	Operating System
PD	Process Data
PHA	Process hazard Analysis
PTP OC	Precision Time Protocol
RA	Resource Availability
RAMS	Reliability, Availability, Maintainability and Safety
RBAC	Role-Based Access Control
RBAC	Role-Based Access Control
RDF	Restricted Data Flow
RO	Read-Only
RW	Read/Write
SDT	Safe Data Transmission
SIL	Safety Integrity Level
SL	Security Level
SL-T	Security Level Target
SO	Security Objective
TCMS	Train Control and Management System
TCP	Transmission Control Protocol

THR	Tolerable Hazard Rate
TLV	Type, Length Value
TPM	Trusted Platform Module
TPM	Trusted Platform Module
TRDP	Train Real Time Data Protocol
TRE	Timely Response
TSN	Time Sensitive Network
TTDB	Train Topology Database
TTI	Train Topology Information
UC	Use Control
UDP	User Datagram Protocol
V&V	Verification & Validation
VCU	Vehicle Control Unit
WD	WatchDog
WDT	WatchDog Timer
WTB	Wire Train Bus
XML	Extended Markup Language

Table 23: List of Abbreviations

Chapter 9 Bibliography

References from introduction section:

[1] AUTOSAR, AUTomotive Open System Architecture. http://www.autosar.org/

[2] CENELEC, EN50126: Railway applications - The specification and demonstration of Reliability." Availability, Maintainability and Safety (RAMS) (1999).

[3] CENELEC, EN50128: Railway applications – Communications, signalling and processing systems – Software for railway control and protection systems (2011).

[4] CENELEC, EN50129: Railway applications - Communication, signalling and processing systems - Safety related electronic systems for signalling." (2003).

[5] CONNECTA project website, https://shift2rail.org/projects/connecta/, retrieved Aug 2017

[6] IEC 61508: Functional safety of electrical/electronic/programmable electronic safety related systems, 2010.

[7] Safe4RAIL project website, https://safe4rail.eu/, retrieved Aug 2017

[8] W. Steiner, G. Bauer, B. Hall, and M. Paulitsch, "TTEthernet: Time-Triggered Ethernet," in Time-Triggered Communication, R. Obermaisser, Ed. CRC Press, Aug 2011.

[9] TSN, Time-Sensitive Networking Task Group, http://www.ieee802.org/1/pages/tsn.html, retrieved Aug 2017

[10] Magerit, https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/rm-ra-methods/m_magerit.html

[11] CENELEC, EN50657: Railways Applications - Rolling stock applications - Software on Board Rolling Stock, 2017

[12] IEC62443-2-1: Security for industrial automation and control systems - Part 2-1: Industrial automation and control system security management system, 2015

[13] IEC62443-3-3: Security for industrial automation and control systems - Part 3-3: System security requirements and security levels, 2013

[14] IEC62443-4-2: Security for industrial automation and control systems - Part 4-2: Technical security requirements for IACS components, 2015

[15] CENELEC, "EN 50159:2011. Railway applications - communication, signaling and processing systems - safety-related communication in transmission systems.," 2011.

[16] SAE International, "http://standards.sae.org/as6802/," SAE Standards, Warrendale, PA, 2011.

[17] IEEE 1588 WG, "1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," IEEE Instrumentation and Measurement Society.

[18] DIN, "DIN VDE V 0831-104. Electric signaling systems for railways - part 104: It security guideline based on IEC 62443, draft. October, 2015.," 2015.

[19] DIN, "DIN VDE V 0831-102. Electric signaling systems for railways - part 102: Protection profile for technical functions in railway signaling, draft. December, 2013.," 2013.

[20] ISO, "ISO/IEC 15408-1. Information technology - security techniques - evaluation criteria for it security - part 1: Introduction and general model.".

[21] IEEE802, "802.1AB-2009 - IEEE Standard for Local and Metropolitan Area Networks--Station and Media Access Control Connectivity Discovery".

[22] UNISIG, "STM FFFIS Safe Time Layer - SUBSET-056," UNISIG, 2016.

[23] CENELEC, "EN 50126-1:2015. Railway applications - the specification and demonstration of reliability, availability, maintainability and safety (rams)," 2015.

[24] CENELEC, "EN 50129:2016. Railway applications - communication, signaling and processing systems - safety related electronic systems for signaling," 2016.

[25] S. Morris and D. Nicholls, Reliability Toolkit: Commercial Practices Edition: A Practical Guide for Commercial Products and Military Systems Under Acquisition Reform, Rome NY: Reliability Analysis Center, 1995.

[26] "D4.1 – Requirement specification for each sub task", CTA-T4.1-D-BTD-002-09, Rev. 9.

[27] EN15437-2. "Railway applications. Axlebox condition monitoring. Interface and design requirements. Performance and design requirements of on-board systems for temperature monitoring", 2012.

ANNEX A: FDF Process Hazard Analysis

This annex contains the complete table of the Functional Distribution Framework Process Hazard Analysis. Despite the size, it can be zoomed in order to check the table more closely.

Table 24: FDF Process Hazard Analysis

FUNCTIONAL FAILURE MODE				FAILURE EFFECTS			COUNTERMEASURES SPECIFICATION															
							Correct functional operation		Detection of faults		Action following Detection		Independence of Items		Systematic & Random faults		Application conditions		Recommendations	
Sub-function	Description	Guide-word	Deviation / Functional Failure mode	Local effect	Final effect	Hazard ID	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description
Communication	transmission / reception of messages from/to Message Store to/from network (remote functions)	No / loss of / partial	Missed exchange of messages between remote functions	Missed exchange of data between remote functions during the Application functions execution or for remote monitoring.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	FDF_SH_05	HA_COM_03	The Framework shall define, configure, assess and guarantee performance of communication channels, including priority, throughput, jitter, latency, response time.	HA_COM_05	The Framework shall monitor the communication between remote functions.	HA_COM_06	The Framework shall inform the Application function(s) in case of loss of valid communication between remote functions.	HA_COM_04	The Framework shall implement Communication service without any operation on the messages' safety layer content.	-	-			PHA_REC_01	It is recommend the compliance of the communication between remote functions with the EN50159 technical standard on Safety-related communication in transmission systems, for a Category 3 transmission system risk of unauthorised access to the transmission system not negligible).
Communication	transmission / reception of messages from/to Message Store to/from network (remote functions)	Wrong	Incorrect exchange of messages between remote functions (including any possible types of communication error)	Exchange of incorrect data between remote functions during the Application functions execution or for remote monitoring.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	FDF_SH_05	HA_MSG_01	The Framework shall ensure the integrity of safety-related data exchanged by communication protocol(s) implementing a safety layer (i.e. a safety code) with source and/or destination identifiers, information that the transmitter is operating properly, redundancy field allowing error detection and assuring data integrity.	HA_MSG_06	The Framework shall check the integrity (i.e. information is complete and not altered) of incoming messages containing safety.	HA_COM_06	The Framework shall inform the Application function(s) in case of loss of valid communication between remote functions.	HA_COM_04	The Framework shall implement Communication service without any operation on the messages' safety layer content.	-	-				
Communication	transmission / reception of messages from/to Message Store to/from network (remote functions)	Delayed	Delayed exchange of messages from remote functions	Delayed exchange of data between remote function during the application functions execution.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	FDF_SH_05	HA_COM_02	The Framework shall provide a communication service that allows sending messages within defined timely bounds and with defined periodicity, and receiving messages within defined maximum delay (deterministic communication).	HA_MSG_07	The Framework shall check the timeliness and sequence of messages containing safety-data, exchanged between remote functions.	HA_COM_06	The Framework shall inform the Application function(s) in case of loss of valid communication between remote functions.	HA_COM_04	The Framework shall implement Communication service without any operation on the messages' safety layer content.	-	-				
Communication	transmission / reception of messages from/to Message Store to/from network (remote functions)	Undue	Undue exchange of messages between remote functions (when not required)	Undue exchange or remote distribution of data to remote functions, not required by the execution of the Application functions or for remote monitoring.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	FDF_SH_05	HA_MSG_02	The Framework shall ensure the timeliness and sequence of data exchanged and results of safety algorithms, e.g. by sequence number and/or time stamps generated by unique identifier related to the cycle (or equivalent measures).	HA_MSG_07	The Framework shall check the timeliness and sequence of messages containing safety-data, exchanged between remote functions.	HA_MSG_09	The Framework and Application functions shall ignore the content and discharge a message (containing safety-data) when a communication error is identified through the messages authenticity, integrity, timeliness or sequence checks.	HA_COM_04	The Framework shall implement Communication service without any operation on the messages' safety layer content.	-	-				
Monitoring	provision of SIL0 variables accessible remotely	No / partial	No provision of variables to remote function(s)	Remote function cannot be properly executed due to missed data	Potential unsafe behaviour of the Platform in the execution of the safety-related Application functions due to error(s) in the provision of data required by remote function(s) (missed, delay, incorrect data).	FDF_SH_01	HA_MON_01	The platform shall assign to the Monitoring Function privilege for read-only the variables stored into SIL0 Memory spaces, or to all the Memory spaces if data alteration during reading can be excluded, and execute Monitoring services without any disturb or unintended effects due to other Service and Application functions.	-	-	-	-	-	-	-	-	PHA_AC_06	Remote functions shall not use variables provided by the Framework's Monitoring functions (but Messages) for the execution of safety-related algorithms.		
Monitoring	provision of SIL0 variables accessible remotely	Wrong	Incorrect provision of variables to remote function(s) (incorrect value)	Remote function cannot be properly executed due to incorrect data	Potential unsafe behaviour of the Platform in the execution of the safety-related Application functions due to error(s) in the provision of data required by remote function(s) (missed, delay, incorrect data).	FDF_SH_01	HA_MON_01	The platform shall assign to the Monitoring Function privilege for read-only the variables stored into SIL0 Memory spaces, or to all the Memory spaces if data alteration during reading can be excluded, and execute Monitoring services without any disturb or unintended effects due to other Service and Application functions.	-	-	-	-	-	-	-	-	PHA_AC_06	Remote functions shall not use variables provided by the Framework's Monitoring functions (but Messages) for the execution of safety-related algorithms.		
Monitoring	provision of SIL0 variables accessible remotely	Delayed	Delayed in the provision of variables to remote function(s)	Remote function cannot be properly executed due to missed data	Potential unsafe behaviour of the Platform in the execution of the safety-related Application functions due to error(s) in the provision of data required by remote function(s) (missed, delay, incorrect data).	FDF_SH_01	HA_MON_01	The platform shall assign to the Monitoring Function privilege for read-only the variables stored into SIL0 Memory spaces, or to all the Memory spaces if data alteration during reading can be excluded, and execute Monitoring services without any disturb or unintended effects due to other Service and Application functions.	-	-	-	-	-	-	-	-	PHA_AC_06	Remote functions shall not use variables provided by the Framework's Monitoring functions (but Messages) for the execution of safety-related algorithms.		
Monitoring	provision of SIL0 variables accessible remotely	Undue	Incorrect provision of variables to remote function(s) (incorrect variable)	Remote function cannot be properly executed due to incorrect data	Potential unsafe behaviour of the Platform in the execution of the safety-related Application functions due to error(s) in the provision of data required by remote function(s) (missed, delay, incorrect data).	FDF_SH_01	HA_MON_01	The platform shall assign to the Monitoring Function privilege for read-only the variables stored into SIL0 Memory spaces, or to all the Memory spaces if data alteration during reading can be excluded, and execute Monitoring services without any disturb or unintended effects due to other Service and Application functions.	-	-	-	-	-	-	-	-	PHA_AC_06	Remote functions shall not use variables provided by the Framework's Monitoring functions (but Messages) for the execution of safety-related algorithms.		

FUNCTIONAL FAILURE MODE				FAILURE EFFECTS			COUNTERMEASURES SPECIFICATION													
							Correct functional operation		Detection of faults		Action following Detection		Independence of Items		Systematic & Random faults		Application conditions		Recommendations	
Sub-function	Description	Guide-word	mode	Local effect	Final effect	Hazard ID	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description
Message function	decomposition of messages (to share variables between remote functions) in variables (to application functions) and composition of messages with variables	No / loss of / partial	No / partial decomposition of messages into variables	Missed or partial updating of variables according to the incoming messages and missed or incorrect execution of Application function(s).	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	FDF_SH_05	HA_MSG_05	The Framework shall guarantee that Message Function read and write the required variables in a safe way, i.e. variables are read without altering their value and written according to specification (set during configuration).	HA_MSG_07	The Framework shall check the timeliness and sequence of messages containing safety-data, exchanged between remote functions.	HA_MSG_09	The Framework and Application functions shall ignore the content and discharge a message (containing safety-data) when a communication error is identified through the messages authenticity, integrity, timeliness or sequence checks.	HA_MSG_10	The Framework shall implement reactions against errors in the communication of safety-related data that are functionally independent by any non-trusted transmission.	HA_MSG_11	The Framework shall guarantee the validity of safety related data exchanged between remote functions, through messages composing and decomposing into variables carried out by the Message Function, with the same SIL assigned to the Application function(s) using messages and variables involved.	PHA_AC_01	The remote functions exchanging safety-data with and within the Framework shall: _ implement safety protection in the generation of safety-data to be exchange through the transmission system; _ verify the incoming messages in order to detect erroneous information (transmitter identity, type, value errors) and time errors (timing, sequencing error); _discharge a message when a communication error is identified; _ react to the loss of valid communication, including tolerance of message	PHA_REC_01	It is recommend the compliance of the communication between remote functions with the EN50159 technical standard on Safety-related communication in transmission systems, for a Category 3 transmission system risk of unauthorised access to the transmission system not negligible).
Message function	decomposition of messages (to share variables between remote functions) in variables (to application functions) and composition of messages with variables	No / loss of / partial	No / partial composition of messages with variables	Missed exchange of data incorrect execution of the Application function(s).	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	FDF_SH_05	HA_MSG_05	The Framework shall guarantee that Message Function read and write the required variables in a safe way, i.e. variables are read without altering their value and written according to specification (set during configuration).	HA_MSG_07	The Framework shall check the timeliness and sequence of messages containing safety-data, exchanged between remote functions.	HA_MSG_09	The Framework and Application functions shall ignore the content and discharge a message (containing safety-data) when a communication error is identified through the messages authenticity, integrity, timeliness or sequence checks.	HA_MSG_10	The Framework shall implement reactions against errors in the communication of safety-related data that are functionally independent by any non-trusted transmission.	HA_MSG_11	The Framework shall guarantee the validity of safety related data exchanged between remote functions, through messages composing and decomposing into variables carried out by the Message Function, with the same SIL assigned to the Application function(s) using messages and variables involved.	PHA_AC_01	The remote functions exchanging safety-data with and within the Framework shall: _ implement safety protection in the generation of safety-data to be exchange through the transmission system; _ verify the incoming messages in order to detect erroneous information (transmitter identity, type, value errors) and time errors (timing, sequencing error); _discharge a message when a communication error is identified; _ react to the loss of valid communication, including tolerance of message		
Message function	decomposition of messages (to share variables between remote functions) in variables (to application functions) and composition of messages with variables	No / loss of / partial (Msg Deletion)	Deletion of messages exchanged between remote function including safety-related data.	Incorrect data distribution and execution of the Application function(s).	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	FDF_SH_05	HA_MSG_02	The Framework shall ensure the timeliness and sequence of data exchanged and results of safety algorithms, e.g. by sequence number and/or time stamps generated by unique identifier related to the cycle (or equivalent measures).	HA_MSG_07	The Framework shall check the timeliness and sequence of messages containing safety-data, exchanged between remote functions.	HA_MSG_09	The Framework and Application functions shall ignore the content and discharge a message (containing safety-data) when a communication error is identified through the messages authenticity, integrity, timeliness or sequence checks.	HA_MSG_10	The Framework shall implement reactions against errors in the communication of safety-related data that are functionally independent by any non-trusted transmission.	HA_MSG_11	The Framework shall guarantee the validity of safety related data exchanged between remote functions, through messages composing and decomposing into variables carried out by the Message Function, with the same SIL assigned to the Application function(s) using messages and variables involved.	PHA_AC_01	The remote functions exchanging safety-data with and within the Framework shall: _ implement safety protection in the generation of safety-data to be exchange through the transmission system; _ verify the incoming messages in order to detect erroneous information (transmitter identity, type, value errors) and time errors (timing, sequencing error); _discharge a message when a communication error is identified; _ react to the loss of valid communication, including tolerance of message		
Message function	decomposition of messages (to share variables between remote functions) in variables (to application functions) and composition of messages with variables	Wrong	Incorrect decomposition of messages into variables or wrong updating	Incorrect updating of variables according to the incoming messages and missed or incorrect execution of Application function(s).	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	FDF_SH_05	HA_MSG_05	The Framework shall guarantee that Message Function read and write the required variables in a safe way, i.e. variables are read without altering their value and written according to specification (set during configuration).	HA_MSG_07	The Framework shall check the timeliness and sequence of messages containing safety-data, exchanged between remote functions.	HA_MSG_09	The Framework and Application functions shall ignore the content and discharge a message (containing safety-data) when a communication error is identified through the messages authenticity, integrity, timeliness or sequence checks.	HA_MSG_10	The Framework shall implement reactions against errors in the communication of safety-related data that are functionally independent by any non-trusted transmission.	HA_MSG_11	The Framework shall guarantee the validity of safety related data exchanged between remote functions, through messages composing and decomposing into variables carried out by the Message Function, with the same SIL assigned to the Application function(s) using messages and variables involved.	PHA_AC_01	The remote functions exchanging safety-data with and within the Framework shall: _ implement safety protection in the generation of safety-data to be exchange through the transmission system; _ verify the incoming messages in order to detect erroneous information (transmitter identity, type, value errors) and time errors (timing, sequencing error); _discharge a message when a communication error is identified; _ react to the loss of valid communication, including tolerance of message		
Message function	decomposition of messages (to share variables between remote functions) in variables (to application functions) and composition of messages with variables	Wrong	Incorrect composition of messages with variables or data corruption during reading	Incorrect data distribution and execution of the Application function(s).	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	FDF_SH_05	HA_MSG_05	The Framework shall guarantee that Message Function read and write the required variables in a safe way, i.e. variables are read without altering their value and written according to specification (set during configuration).	HA_MSG_07	The Framework shall check the timeliness and sequence of messages containing safety-data, exchanged between remote functions.	HA_MSG_09	The Framework and Application functions shall ignore the content and discharge a message (containing safety-data) when a communication error is identified through the messages authenticity, integrity, timeliness or sequence checks.	HA_MSG_10	The Framework shall implement reactions against errors in the communication of safety-related data that are functionally independent by any non-trusted transmission.	HA_MSG_11	The Framework shall guarantee the validity of safety related data exchanged between remote functions, through messages composing and decomposing into variables carried out by the Message Function, with the same SIL assigned to the Application function(s) using messages and variables involved.	PHA_AC_01	The remote functions exchanging safety-data with and within the Framework shall: _ implement safety protection in the generation of safety-data to be exchange through the transmission system; _ verify the incoming messages in order to detect erroneous information (transmitter identity, type, value errors) and time errors (timing, sequencing error); _discharge a message when a communication error is identified; _ react to the loss of valid communication, including tolerance of message		
Message function	decomposition of messages (to share variables between remote functions) in variables (to application functions) and composition of messages with variables	Wrong (Msg Repetitions)	Repetitions of messages exchanged between remote function including safety-related data.	Incorrect data distribution and execution of the Application function(s).	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	FDF_SH_05	HA_MSG_02	The Framework shall ensure the timeliness and sequence of data exchanged and results of safety algorithms, e.g. by sequence number and/or time stamps generated by unique identifier related to the cycle (or equivalent measures).	HA_MSG_07	The Framework shall check the timeliness and sequence of messages containing safety-data, exchanged between remote functions.	HA_MSG_09	The Framework and Application functions shall ignore the content and discharge a message (containing safety-data) when a communication error is identified through the messages authenticity, integrity, timeliness or sequence checks.	HA_MSG_10	The Framework shall implement reactions against errors in the communication of safety-related data that are functionally independent by any non-trusted transmission.	HA_MSG_11	The Framework shall guarantee the validity of safety related data exchanged between remote functions, through messages composing and decomposing into variables carried out by the Message Function, with the same SIL assigned to the Application function(s) using messages and variables involved.	PHA_AC_01	The remote functions exchanging safety-data with and within the Framework shall: _ implement safety protection in the generation of safety-data to be exchange through the transmission system; _ verify the incoming messages in order to detect erroneous information (transmitter identity, type, value errors) and time errors (timing, sequencing error); _discharge a message when a communication error is identified; _ react to the loss of valid communication, including tolerance of message		
Message function	decomposition of messages (to share variables between remote functions) in variables (to application functions) and composition of messages with variables	Wrong (Msg Resequencing)	Resequencing of messages exchanged between remote function including safety-related data.	Incorrect data distribution and execution of the Application function(s).	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	FDF_SH_05	HA_MSG_02	The Framework shall ensure the timeliness and sequence of data exchanged and results of safety algorithms, e.g. by sequence number and/or time stamps generated by unique identifier related to the cycle (or equivalent measures).	HA_MSG_07	The Framework shall check the timeliness and sequence of messages containing safety-data, exchanged between remote functions.	HA_MSG_09	The Framework and Application functions shall ignore the content and discharge a message (containing safety-data) when a communication error is identified through the messages authenticity, integrity, timeliness or sequence checks.	HA_MSG_10	The Framework shall implement reactions against errors in the communication of safety-related data that are functionally independent by any non-trusted transmission.	HA_MSG_11	The Framework shall guarantee the validity of safety related data exchanged between remote functions, through messages composing and decomposing into variables carried out by the Message Function, with the same SIL assigned to the Application function(s) using messages and variables involved.	PHA_AC_01	The remote functions exchanging safety-data with and within the Framework shall: _ implement safety protection in the generation of safety-data to be exchange through the transmission system; _ verify the incoming messages in order to detect erroneous information (transmitter identity, type, value errors) and time errors (timing, sequencing error); _discharge a message when a communication error is identified; _ react to the loss of valid communication, including tolerance of message		
Message function	decomposition of messages (to share variables between remote functions) in variables (to application functions) and composition of messages with variables	Wrong (Msg Corruption)	Corruption of safety-related data within the messages exchanged between remote function .	Incorrect data distribution and execution of the Application function(s).	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	FDF_SH_05	HA_MSG_01	The Framework shall ensure the integrity of safety-related data exchanged by communication protocol(s) implementing a safety layer (i.e. a safety code) with source and/or destination identifiers, information that the transmitter is operating properly, redundancy field allowing error detection and assuring data integrity.	HA_MSG_06	The Framework shall check the integrity (i.e. information is complete and not altered) of incoming messages containing safety.	HA_MSG_09	The Framework and Application functions shall ignore the content and discharge a message (containing safety-data) when a communication error is identified through the messages authenticity, integrity, timeliness or sequence checks.	HA_MSG_10	The Framework shall implement reactions against errors in the communication of safety-related data that are functionally independent by any non-trusted transmission.	HA_MSG_11	The Framework shall guarantee the validity of safety related data exchanged between remote functions, through messages composing and decomposing into variables carried out by the Message Function, with the same SIL assigned to the Application function(s) using messages and variables involved.	PHA_AC_01	The remote functions exchanging safety-data with and within the Framework shall: _ implement safety protection in the generation of safety-data to be exchange through the transmission system; _ verify the incoming messages in order to detect erroneous information (transmitter identity, type, value errors) and time errors (timing, sequencing error); _discharge a message when a communication error is identified; _ react to the loss of valid communication, including tolerance of message		
Message function	decomposition of messages (to share variables between remote functions) in variables (to application functions) and composition of messages with variables	Undue (Msg Insertion)	Insertion within the messages exchanged between remote function including safety-related data.	Incorrect data distribution and execution of the Application function(s).	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	FDF_SH_05	HA_MSG_01	The Framework shall ensure the integrity of safety-related data exchanged by communication protocol(s) implementing a safety layer (i.e. a safety code) with source and/or destination identifiers, information that the transmitter is operating properly, redundancy field allowing error detection and assuring data integrity.	HA_MSG_07	The Framework shall check the timeliness and sequence of messages containing safety-data, exchanged between remote functions.	HA_MSG_09	The Framework and Application functions shall ignore the content and discharge a message (containing safety-data) when a communication error is identified through the messages authenticity, integrity, timeliness or sequence checks.	HA_MSG_10	The Framework shall implement reactions against errors in the communication of safety-related data that are functionally independent by any non-trusted transmission.	HA_MSG_11	The Framework shall guarantee the validity of safety related data exchanged between remote functions, through messages composing and decomposing into variables carried out by the Message Function, with the same SIL assigned to the Application function(s) using messages and variables involved.	PHA_AC_01	The remote functions exchanging safety-data with and within the Framework shall: _ implement safety protection in the generation of safety-data to be exchange through the transmission system; _ verify the incoming messages in order to detect erroneous information (transmitter identity, type, value errors) and time errors (timing, sequencing error); _discharge a message when a communication error is identified; _ react to the loss of valid communication, including tolerance of message		
Message function	decomposition of messages (to share variables between remote functions) in variables (to application functions) and composition of messages with variables	Undue (Msg Masquerade)	Masquerade messages including safety-related data exchanged between remote function .	Incorrect data distribution and execution of the Application function(s).	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	FDF_SH_05	HA_MSG_03	The Framework shall protect the communication of safety-related data against cyber-attack, ensuring data authenticity and confidentiality, e.g. by software and/or hardware security mechanisms (e.g. cryptographic mechanisms, control of access to data).	HA_MSG_08	The Framework shall check the authenticity of incoming message containing safety data, exchanged between remote functions.	HA_MSG_09	The Framework and Application functions shall ignore the content and discharge a message (containing safety-data) when a communication error is identified through the messages authenticity, integrity, timeliness or sequence checks.	HA_MSG_10	The Framework shall implement reactions against errors in the communication of safety-related data that are functionally independent by any non-trusted transmission.	HA_MSG_11	The Framework shall guarantee the validity of safety related data exchanged between remote functions, through messages composing and decomposing into variables carried out by the Message Function, with the same SIL assigned to the Application function(s) using messages and variables involved.	PHA_AC_01	The remote functions exchanging safety-data with and within the Framework shall: _ implement safety protection in the generation of safety-data to be exchange through the transmission system; _ verify the incoming messages in order to detect erroneous information (transmitter identity, type, value errors) and time errors (timing, sequencing error); _discharge a message when a communication error is identified; _ react to the loss of valid communication, including tolerance of message		
Message function	decomposition of messages (to share variables between remote functions) in variables (to application functions) and composition of messages with variables	Undue	Undue use of message containing non-safety related data, for safety-related applications.	Incorrect data distribution and execution of the Application function(s).	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect exchange of data between remote functions.	FDF_SH_05	HA_MSG_04	The Framework shall use protocols for diagnostic, maintenance, configuration and communication of non-safety related data with different structures than one(s) used for the communication of safety-related data.	HA_MSG_08	The Framework shall check the authenticity of incoming message containing safety data, exchanged between remote functions.	HA_MSG_09	The Framework and Application functions shall ignore the content and discharge a message (containing safety-data) when a communication error is identified through the messages authenticity, integrity, timeliness or sequence checks.	HA_MSG_10	The Framework shall implement reactions against errors in the communication of safety-related data that are functionally independent by any non-trusted transmission.	HA_MSG_12	The Framework shall allow Message Function to access to memory space(s) containing messages and to memory space(s) containing variables with the same SIL.	PHA_AC_01	The remote functions exchanging safety-data with and within the Framework shall: _ implement safety protection in the generation of safety-data to be exchange through the transmission system; _ verify the incoming messages in order to detect erroneous information (transmitter identity, type, value errors) and time errors (timing, sequencing error); _discharge a message when a communication error is identified; _ react to the loss of valid communication, including tolerance of message errors if any, as for the notification of a fatal Fault.		

FUNCTIONAL FAILURE MODE				FAILURE EFFECTS			COUNTERMEASURES SPECIFICATION													
							Correct functional operation		Detection of faults		Action following Detection		Independence of Items		Systematic & Random faults		Application conditions		Recommendations	
Sub-function	Description	Guide-word	Deviation / Functional Failure mode	Local effect	Final effect	Hazard ID	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description
Input/Output function	reading of input and updating of variables / setting of outputs according to variables	No / loss of	No reading of input and/or updating of variables	Unavailability of the updated control(s) from the interfaced object(s), required for the proper execution of the Application functions.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to a missed or incorrect acquisition of controls (input) from the interfaced object(s).	FDF_SH_04	HA_IO_01	The Framework shall provide services that allow the Application function to read the last valid value stored into an exchange variable and to update this value according to the status of the related input (coming from the interfaced object).	HA_IO_05	The Framework shall detect inconsistency between the values stored into the exchange variables and the status pf the related platform's input and output.	HA_IO_06	The Framework, in case of any inconsistency between the values stored into an exchange variable and the status of the related platform's input / output, shall inform the Application function(s) with read and/or write privilege on this variable.	-	-	HA_IO_08	The Framework shall guarantee the updating of each exchange variable (according to the status of related input) and its reading with the SIL assigned to the Application function(s) involved and to the specific variable.	PHA_AC_05	The Application function shall react to the notification of a Fault condition due to inconsistency between the values stored into an exchange variable and the status of the related platform's input / output (fatal Fault), by the transition into the specific safe state.		
Input/Output function	reading of input and updating of variables / setting of outputs according to variables	No / loss of / partial	No setting of outputs according to variables	Missed updating of command(s) toward the interfaced object(s), required for the proper execution of the Application functions.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to a missed or incorrect setting of commands (output) toward the interfaced object(s).	FDF_SH_06	HA_IO_02	The Framework shall provide services that allow the Application function to write a value into an exchange variable and to update accordingly to the status of the related output (toward the interfaced object).	HA_IO_05	The Framework shall detect inconsistency between the values stored into the exchange variables and the status pf the related platform's input and output.	HA_IO_06	The Framework, in case of any inconsistency between the values stored into an exchange variable and the status of the related platform's input / output, shall inform the Application function(s) with read and/or write privilege on this variable.	-	-	HA_IO_09	The Framework shall guarantee the updating the status of each output (according to value stored into the related exchange variable) and its writing with the SIL assigned to the Application function(s) involved and to the specific variable.	PHA_AC_05	The Application function shall react to the notification of a Fault condition due to inconsistency between the values stored into an exchange variable and the status of the related platform's input / output (fatal Fault), by the transition into the specific safe state.		
Input/Output function	reading of input and updating of variables / setting of outputs according to variables	Wrong	Incorrect reading of input and/or updating of variables (wrong value)	Incorrect control(s) coming from the interfaced object(s), i.e. different with respect to the current status, used for the proper execution of the Application functions.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to a missed or incorrect acquisition of controls (input) from the interfaced object(s).	FDF_SH_04	HA_IO_01	The Framework shall provide services that allow the Application function to read the last valid value stored into an exchange variable and to update this value according to the status of the related input (coming from the interfaced object).	HA_IO_05	The Framework shall detect inconsistency between the values stored into the exchange variables and the status pf the related platform's input and output.	HA_IO_06	The Framework, in case of any inconsistency between the values stored into an exchange variable and the status of the related platform's input / output, shall inform the Application function(s) with read and/or write privilege on this variable.	-	-	HA_IO_08	The Framework shall guarantee the updating of each exchange variable (according to the status of related input) and its reading with the SIL assigned to the Application function(s) involved and to the specific variable.	PHA_AC_05	The Application function shall react to the notification of a Fault condition due to inconsistency between the values stored into an exchange variable and the status of the related platform's input / output (fatal Fault), by the transition into the specific safe state.		
Input/Output function	reading of input and updating of variables / setting of outputs according to variables	Wrong	Incorrect setting of outputs according to variables (wrong value)	Incorrect command(s) toward the interfaced object(s), i.e. different than required for the proper execution of the Application functions.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to a missed or incorrect setting of commands (output) toward the interfaced object(s).	FDF_SH_06	HA_IO_02	The Framework shall provide services that allow the Application function to write a value into an exchange variable and to update accordingly to the status of the related output (toward the interfaced object).	HA_IO_05	The Framework shall detect inconsistency between the values stored into the exchange variables and the status pf the related platform's input and output.	HA_IO_06	The Framework, in case of any inconsistency between the values stored into an exchange variable and the status of the related platform's input / output, shall inform the Application function(s) with read and/or write privilege on this variable.	HA_IO_07	The Framework shall be able to provide independence between different (set of) input / output interfacing external objects (that can be request by Application function to implement reliable-safe architecture).	HA_IO_09	The Framework shall guarantee the updating the status of each output (according to value stored into the related exchange variable) and its writing with the SIL assigned to the Application function(s) involved and to the specific variable.	PHA_AC_05	The Application function shall react to the notification of a Fault condition due to inconsistency between the values stored into an exchange variable and the status of the related platform's input / output (fatal Fault), by the transition into the specific safe state.		
Input/Output function	reading of input and updating of variables / setting of outputs according to variables	Delayed / undue	Incorrect timing in the reading of input and/or updating of variables (delayed or too fast)	Incorrect control(s) coming from the interfaced object(s), i.e. different with respect to the current status, used for the proper execution of the Application functions.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to a missed or incorrect acquisition of controls (input) from the interfaced object(s).	FDF_SH_04	HA_IO_04	The Framework shall read and write all the I/O related to the executed Application function in one cycle only, guarantying that the current value of every input is stored in the associated exchange variable at the beginning of each cycle and the current value of every output is set according to the value stored in the associated exchange variable at the end of each cycle..	HA_IO_05	The Framework shall detect inconsistency between the values stored into the exchange variables and the status pf the related platform's input and output.	HA_IO_06	The Framework, in case of any inconsistency between the values stored into an exchange variable and the status of the related platform's input / output, shall inform the Application function(s) with read and/or write privilege on this variable.	HA_IO_07	The Framework shall be able to provide independence between different (set of) input / output interfacing external objects (that can be request by Application function to implement reliable-safe architecture).	HA_IO_08	The Framework shall guarantee the updating of each exchange variable (according to the status of related input) and its reading with the SIL assigned to the Application function(s) involved and to the specific variable.	PHA_AC_05	The Application function shall react to the notification of a Fault condition due to inconsistency between the values stored into an exchange variable and the status of the related platform's input / output (fatal Fault), by the transition into the specific safe state.		
Input/Output function	reading of input and updating of variables / setting of outputs according to variables	Delayed / undue	Incorrect timing in the setting of outputs according to variables (delayed or too fast)	Incorrect command(s) toward the interfaced object(s), i.e. different than required for the proper execution of the Application functions.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to a missed or incorrect setting of commands (output) toward the interfaced object(s).	FDF_SH_06	HA_IO_04	The Framework shall read and write all the I/O related to the executed Application function in one cycle only, guarantying that the current value of every input is stored in the associated exchange variable at the beginning of each cycle and the current value of every output is set according to the value stored in the associated exchange variable at the end of each cycle..	HA_IO_05	The Framework shall detect inconsistency between the values stored into the exchange variables and the status pf the related platform's input and output.	HA_IO_06	The Framework, in case of any inconsistency between the values stored into an exchange variable and the status of the related platform's input / output, shall inform the Application function(s) with read and/or write privilege on this variable.	HA_IO_07	The Framework shall be able to provide independence between different (set of) input / output interfacing external objects (that can be request by Application function to implement reliable-safe architecture).	HA_IO_09	The Framework shall guarantee the updating the status of each output (according to value stored into the related exchange variable) and its writing with the SIL assigned to the Application function(s) involved and to the specific variable.	PHA_AC_05	The Application function shall react to the notification of a Fault condition due to inconsistency between the values stored into an exchange variable and the status of the related platform's input / output (fatal Fault), by the transition into the specific safe state.		
Input/Output function	reading of input and updating of variables / setting of outputs according to variables	Undue	Incorrect reading of input and/or updating of variables (exchange variable)	Incorrect control(s) coming from the interfaced object(s), i.e. different with respect to the current status, used for the proper execution of the Application functions.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to a missed or incorrect acquisition of controls (input) from the interfaced object(s).	FDF_SH_04	HA_IO_03	The Framework shall identify univocally each input / output interfacing external objects, each exchange variable, and each association between them, according to the Configuration file(s) of the Application function(s) using them.	HA_IO_05	The Framework shall detect inconsistency between the values stored into the exchange variables and the status pf the related platform's input and output.	HA_IO_06	The Framework, in case of any inconsistency between the values stored into an exchange variable and the status of the related platform's input / output, shall inform the Application function(s) with read and/or write privilege on this variable.	HA_IO_07	The Framework shall be able to provide independence between different (set of) input / output interfacing external objects (that can be request by Application function to implement reliable-safe architecture).	HA_IO_10	The Framework shall allow I/O Function to access only to memory space with the same SIL.	PHA_AC_05	The Application function shall react to the notification of a Fault condition due to inconsistency between the values stored into an exchange variable and the status of the related platform's input / output (fatal Fault), by the transition into the specific safe state.		
Input/Output function	reading of input and updating of variables / setting of outputs according to variables	Undue	Incorrect setting of outputs according to variables (exchange variable)	Incorrect command(s) toward the interfaced object(s), i.e. different than required for the proper execution of the Application functions.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to a missed or incorrect setting of commands (output) toward the interfaced object(s).	FDF_SH_06	HA_IO_03	The Framework shall identify univocally each input / output interfacing external objects, each exchange variable, and each association between them, according to the Configuration file(s) of the Application function(s) using them.	HA_IO_05	The Framework shall detect inconsistency between the values stored into the exchange variables and the status pf the related platform's input and output.	HA_IO_06	The Framework, in case of any inconsistency between the values stored into an exchange variable and the status of the related platform's input / output, shall inform the Application function(s) with read and/or write privilege on this variable.	HA_IO_07	The Framework shall be able to provide independence between different (set of) input / output interfacing external objects (that can be request by Application function to implement reliable-safe architecture).	HA_IO_10	The Framework shall allow I/O Function to access only to memory space with the same SIL.	PHA_AC_05	The Application function shall react to the notification of a Fault condition due to inconsistency between the values stored into an exchange variable and the status of the related platform's input / output (fatal Fault), by the transition into the specific safe state.		
Time management	dissemination of the global time from the external global clock	No / missed	No dissemination of the global time from the external global clock	Different time references are used by the different Application functions, with an ineffective scheduled execution of applications.	Potential unsafe behaviour of the Platform due to a wrong timing in the execution of the safety-related Application functions	FDF_SH_03	HA_TM_02	The Framework shall not finalize the inauguration and allow operation without a global time valid (i.e. aligned with the external global clock) and taken as unique reference by all Service and Application functions, independently from the partitions execution.	HA_TM_03	The Framework shall monitor the alignment with the external global clock, the effectiveness of the global time dissemination and functions synchronization.	HA_TM_04	The Framework shall notify a Fault condition, in case of error in the global time synchronization (fatal Fault), to all the Application functions involved.	HA_TM_05	The Framework shall synchronize the local computer clock with the external global clock source and keep it synchronized independently from the execution of the different partitions' processes.	HA_TM_06	The Framework shall disseminate the global time and/or detect any misalignment against the external reference time, with the highest SIL assigned to the Application functions to be executed.	PHA_AC_03	The Application function shall react to the notification of a Fault condition due to error in the global time dissemination or functions synchronization (fatal Fault), implementing tolerance (e.g. errors for a limited number of cycles) if any, by the transition into the specific safe state.	PHA_REC_07	It is recommended to assess the implementation of messages retry mechanism by each Application functions, to improve dependability (tolerance of errors before transition into safe state) within safety constraints.
Time management	dissemination of the global time from the external global clock	Wrong / Undue	Incorrect dissemination of the global time from the external global clock (to all nodes or to a subset of them)	Wrong global clock synchronization and consequent disturb to the scheduled execution of applications	Potential unsafe behaviour of the Platform due to a wrong timing in the execution of the safety-related Application functions	FDF_SH_03	HA_TM_02	The Framework shall not finalize the inauguration and allow operation without a global time valid (i.e. aligned with the external global clock) and taken as unique reference by all Service and Application functions, independently from the partitions execution.	HA_TM_03	The Framework shall monitor the alignment with the external global clock, the effectiveness of the global time dissemination and functions synchronization.	HA_TM_04	The Framework shall notify a Fault condition, in case of error in the global time synchronization (fatal Fault), to all the Application functions involved.	HA_TM_05	The Framework shall synchronize the local computer clock with the external global clock source and keep it synchronized independently from the execution of the different partitions' processes.	HA_TM_06	The Framework shall disseminate the global time and/or detect any misalignment against the external reference time, with the highest SIL assigned to the Application functions to be executed.	PHA_AC_03	The Application function shall react to the notification of a Fault condition due to error in the global time dissemination or functions synchronization (fatal Fault), implementing tolerance (e.g. errors for a limited number of cycles) if any, by the transition into the specific safe state.		
Time management	dissemination of the global time from the external global clock	Loss of / partially / delayed	Missed update of the global time (i.e. according to the external clock)	Potential drift of the global time (with respect to the external clock) and consequent disturb to the scheduled execution of applications	Potential unsafe behaviour of the Platform due to a wrong timing in the execution of the safety-related Application functions	FDF_SH_03	HA_TM_01	The Framework shall synchronize the local computer clock with the external global clock source and keep it synchronized with a maximum defined deviation fixed.	HA_TM_03	The Framework shall monitor the alignment with the external global clock, the effectiveness of the global time dissemination and functions synchronization.	HA_TM_04	The Framework shall notify a Fault condition, in case of error in the global time synchronization (fatal Fault), to all the Application functions involved.	HA_TM_05	The Framework shall synchronize the local computer clock with the external global clock source and keep it synchronized independently from the execution of the different partitions' processes.	HA_TM_06	The Framework shall disseminate the global time and/or detect any misalignment against the external reference time, with the highest SIL assigned to the Application functions to be executed.	PHA_AC_03	The Application function shall react to the notification of a Fault condition due to error in the global time dissemination or functions synchronization (fatal Fault), implementing tolerance (e.g. errors for a limited number of cycles) if any, by the transition into the specific safe state.		

FUNCTIONAL FAILURE MODE				FAILURE EFFECTS			COUNTERMEASURES SPECIFICATION													
							Correct functional operation		Detection of faults		Action following Detection		Independence of Items		Systematic & Random faults		Application conditions		Recommendations	
Sub-function	Description	Guide-word	Deviation / Functional Failure mode	Local effect	Final effect	Hazard ID	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description
Framework management	generation of variable stores, message stores and register functions as specified by the Configuration file. Offer API.	No / partial / Delayed	No, partial or delayed generation of partition(s) (definition of memory space, variable stored, messages' structure, register functions) specified in the Configuration file.	Impossible or incorrect scheduled execution of Application function(s), access to variables (for reading and/or writing), and/or management of messages (decomposing into variables and composing from variables).	Potential unsafe behaviour of the Platform in the execution of the safety-related processes due to an incorrect generation or allocation of resources or management of partitions.	FDF_SH_07	HA_FRM_01	The Framework shall generate Partitions according to the Configuration file of the Application functions to be executed (which specify the SIL, address and size of the memory space, and time window inside the global scheduling plan) and protect each partition's addressing space through specific memory protection mechanisms, e.g. by a hardware memory management unit, and management of access privilege and restrictions.	HA_FRM_07	The Framework shall detect an invalid operation in the partition attempts by the Application function(s), e.g. access to a Memory space without the required reading or writing privilege.	HA_FRM_08	The Framework shall notify a Fault condition, in case of invalid operation in the partition attempt (fatal Fault), to all the Application functions involved.	HA_FRM_11	The Framework shall guarantee the spatial separation among Partition, in order to ensure that no process in one partition can modify (without authorization) software code or application data (i.e.. write to memory data sections, stacks and code) or manage the I/O assigned to another partition, e.g. through the protection of their memory addressing space and the management of privilege and restrictions for variables read / write and for access to I/O.	HA_FRM_14	The Framework shall generate partitions and allocate resources with the same SIL assigned to the Application functions to be executed, including memories spaces storing data with the same (unique) SIL.	PHA_AC_04	The Application function shall react to the notification of a Fault condition due to invalid operation in the partition attempts (fatal Fault), by the transition into the specific safe state.	PHA_REC_02	It is recommended to implement safety-related application functions in compliance with the EN 50129 technical standard on Safety related electronic systems for communication, signalling and processing systems. Specifically about the admitted architecture, according to the SIL assigned to the application, it is recommended: _a dual electronic structure based on composite fail-safety with fail-safe comparison or inherent fail-safe (highly recommended for >SIL2 applications); _a single electronic structure with self-tests and supervision (recommended for
Framework management	generation of variable stores, message stores and register functions as specified by the Configuration file. Offer API.	No / partial	No or partial allocation of resources to partition, for the execution of the Application function(s) / process(es)	Impossible or incorrect scheduled execution of Application function(s), access to variables (for reading and/or writing), and/or management of messages (decomposing into variables and composing from variables).	Potential unsafe behaviour of the Platform in the execution of the safety-related processes due to an incorrect generation or allocation of resources or management of partitions.	FDF_SH_07	HA_FRM_02	The Framework shall provide to the partition assigned to an Application functions the computational resources (e.g. CPU time, memory) required into the Configuration file in order to meet the (worst-case) timing requirements.	HA_FRM_07	The Framework shall detect an invalid operation in the partition attempts by the Application function(s), e.g. access to a Memory space without the required reading or writing privilege.	HA_FRM_08	The Framework shall notify a Fault condition, in case of invalid operation in the partition attempt (fatal Fault), to all the Application functions involved.			HA_FRM_14	The Framework shall generate partitions and allocate resources with the same SIL assigned to the Application functions to be executed, including memories spaces storing data with the same (unique) SIL.				
Framework management	generation of variable stores, message stores and register functions as specified by the Configuration file. Offer API.	Wrong	Wrong generation of partition(s) (e.g. wrong address or size of memory, structure of message, stores and register functions) with respect to the Configuration file.	Incorrect scheduled execution of Application function(s), access to variables (for reading and/or writing), and/or management of messages (decomposing into variables and composing from variables).	Potential unsafe behaviour of the Platform in the execution of the safety-related processes due to an incorrect generation or allocation of resources or management of partitions.	FDF_SH_07	HA_FRM_01	The Framework shall generate Partitions according to the Configuration file of the Application functions to be executed (which specify the SIL, address and size of the memory space, and time window inside the global scheduling plan) and protect each partition's addressing space through specific memory protection mechanisms, e.g. by a hardware memory management unit, and management of access privilege and restrictions.	HA_FRM_07	The Framework shall detect an invalid operation in the partition attempts by the Application function(s), e.g. access to a Memory space without the required reading or writing privilege.	HA_FRM_08	The Framework shall notify a Fault condition, in case of invalid operation in the partition attempt (fatal Fault), to all the Application functions involved.	HA_FRM_11	The Framework shall guarantee the spatial separation among Partition, in order to ensure that no process in one partition can modify (without authorization) software code or application data (i.e.. write to memory data sections, stacks and code) or manage the I/O assigned to another partition, e.g. through the protection of their memory addressing space and the management of privilege and restrictions for variables read / write and for access to I/O.	HA_FRM_14	The Framework shall generate partitions and allocate resources with the same SIL assigned to the Application functions to be executed, including memories spaces storing data with the same (unique) SIL.	PHA_AC_04	The Application function shall react to the notification of a Fault condition due to invalid operation in the partition attempts (fatal Fault), by the transition into the specific safe state.		
Framework management	generation of variable stores, message stores and register functions as specified by the Configuration file. Offer API.	Wrong	Wrong assignment of read-write privileges and constraints to Application functions.	Application function(s) can access to variables (for reading and/or writing) unduly (i.e. when it should be not possible or it is not scheduled). Application functions can interfere in the management of variables and related I/O.	Potential unsafe behaviour of the Platform in the execution of the safety-related processes due to an incorrect generation or allocation of resources or management of partitions.	FDF_SH_07	HA_FRM_03	The Framework shall provide to the Application functions the read-write privilege only to variables (and related input/output, if any) they are allowed to publish and the read-only privilege to software code, parameters and variables (and related input, if any) they are subscribed to.	HA_FRM_07	The Framework shall detect an invalid operation in the partition attempts by the Application function(s), e.g. access to a Memory space without the required reading or writing privilege.	HA_FRM_08	The Framework shall notify a Fault condition, in case of invalid operation in the partition attempt (fatal Fault), to all the Application functions involved.	-	-	HA_FRM_15	The Framework shall assign privileges for read-write access to a Memory space only to independent Application functions with the same SIL. Read-only access could be assigned to remaining Application functions, if data alteration during reading can be excluded.	PHA_AC_04	The Application function shall react to the notification of a Fault condition due to invalid operation in the partition attempts (fatal Fault), by the transition into the specific safe state.		
Framework management	generation of variable stores, message stores and register functions as specified by the Configuration file. Offer API.	Wrong	Inadequate allocation of resources to partition, for the execution of the Application function(s) / process(es)	Incorrect execution of the scheduled Application function(s) due to limitation of resources used.	Potential unsafe behaviour of the Platform in the execution of the safety-related processes due to an incorrect generation or allocation of resources or management of partitions.	FDF_SH_07	HA_FRM_02	The Framework shall provide to the partition assigned to an Application functions the computational resources (e.g. CPU time, memory) required into the Configuration file in order to meet the (worst-case) timing requirements.	HA_FRM_07	The Framework shall detect an invalid operation in the partition attempts by the Application function(s), e.g. access to a Memory space without the required reading or writing privilege.	HA_FRM_08	The Framework shall notify a Fault condition, in case of invalid operation in the partition attempt (fatal Fault), to all the Application functions involved.	-	-	HA_FRM_14	The Framework shall generate partitions and allocate resources with the same SIL assigned to the Application functions to be executed, including memories spaces storing data with the same (unique) SIL.	PHA_AC_04	The Application function shall react to the notification of a Fault condition due to invalid operation in the partition attempts (fatal Fault), by the transition into the specific safe state.		
Framework management	generation of variable stores, message stores and register functions as specified by the Configuration file. Offer API.	Wrong	Inadequate generation of partition and/or allocation of resources, for the execution of multiple instances of the Application function(s) / process(es)	Incorrect execution of multiple instances of the scheduled Application function(s).	Potential unsafe behaviour of the Platform in the execution of the safety-related processes due to an incorrect generation or allocation of resources or management of partitions.	FDF_SH_07	HA_FRM_06	The Framework shall be able to generate partitions and allocate resources for Application function(s) requiring multiple instances (for the implementation of a reliable-safe architecture).	HA_FRM_07	The Framework shall detect an invalid operation in the partition attempts by the Application function(s), e.g. access to a Memory space without the required reading or writing privilege.	HA_FRM_08	The Framework shall notify a Fault condition, in case of invalid operation in the partition attempt (fatal Fault), to all the Application functions involved.	HA_FRM_10	The Framework shall protect and guarantee the independence of multiple instances of an Application function (e.g. implementing reliable-safe architecture), e.g. by data diversity (e.g. different time-stamp guarantying data freshness), timing diversity (instances do not execute simultaneously the same safety-related software modules), independent (hardware) resources.	HA_FRM_14	The Framework shall generate partitions and allocate resources with the same SIL assigned to the Application functions to be executed, including memories spaces storing data with the same (unique) SIL.	PHA_AC_04	The Application function shall react to the notification of a Fault condition due to invalid operation in the partition attempts (fatal Fault), by the transition into the specific safe state.		
Framework management	generation of variable stores, message stores and register functions as specified by the Configuration file. Offer API.	Undue	Unintended interactions between the Operating system and the Application functions.	Impossible or incorrect scheduled execution of Application function(s).	Potential unsafe behaviour during the execution of safety-related processes due to unintended interactions between the Operating system and the Application functions.	FDF_SH_09	HA_FRM_05	The Framework shall call Services required for the scheduled execution of the Application functions.	HA_FRM_18	The Framework shall detect the unavailability of Services required for the scheduled executions of the Application functions and their incorrect call (different than scheduled)	HA_FRM_09	The Framework shall inform the Application functions in case of unavailability of services required for their scheduled execution, or in case of incorrect call (different than scheduled).	HA_FRM_13	The Framework shall prevent any unintended interactions between the Operating system activities and the Application functions, through the definition of formal boundaries and interaction modalities and protecting the Operating System (data sections, stacks, and code) against undue calls from the Application and Services functions (e.g. with an invalid handle, object, address or out of range value; in the wrong context; without the necessary permissions).	HA_FRM_17	The Framework shall guarantee the effectiveness of call(s) to Service function(s) with the same SIL assigned to the Application functions using Service(s).	PHA_AC_04	The Application function shall react to the notification of a Fault condition due to invalid operation in the partition attempts (fatal Fault), by the transition into the specific safe state.		
Framework management	generation of variable stores, message stores and register functions as specified by the Configuration file. Offer API.	Undue	Undue access to variables, and related I/O, by Application function(s) without the required read/write privilege.	Application functions can interfere in the management of variables and related I/O.	Potential unsafe behaviour of the Platform in the execution of the safety-related processes due to an incorrect generation or allocation of resources or management of partitions.	FDF_SH_07	HA_FRM_04	The Framework shall guarantee that Application functions read / write variables, managing consequently the related platform's I/O, only if the required privilege is provided.	HA_FRM_07	The Framework shall detect an invalid operation in the partition attempts by the Application function(s), e.g. access to a Memory space without the required reading or writing privilege.	HA_FRM_08	The Framework shall notify a Fault condition, in case of invalid operation in the partition attempt (fatal Fault), to all the Application functions involved.	HA_FRM_12	The Framework shall guarantee spatial separation between memory spaces containing read-only (including software code and parameters) and read-write variables, variables with different SIL, variables used by multiple independent instances of the Application function.	HA_FRM_16	The Framework shall guarantee the read-write access to memory spaces (according to the assigned privileges) with the same SIL assigned to the Application function(s) and variables stored.	PHA_AC_04	The Application function shall react to the notification of a Fault condition due to invalid operation in the partition attempts (fatal Fault), by the transition into the specific safe state.		

FUNCTIONAL FAILURE MODE				FAILURE EFFECTS			COUNTERMEASURES SPECIFICATION															
							Correct functional operation		Detection of faults		Action following Detection		Independence of Items		Systematic & Random faults		Application conditions		Recommendations	
Sub-function	Description	Guide-word	Deviation / Functional Failure mode	Local effect	Final effect	Hazard ID	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description
Configuration management	reading, parsing, and loading of data in the configuration file	No / missed / partial / delayed	No / missed / partial reading, parsing, or loading of data in the configuration file	Incomplete Platform initialization.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect configuration.	FDF_SH_08	HA_CONF_01	The Framework shall instantiate messages and variable according to the Configuration file, which specifies at least: messages' identifier, variables, to receive or to send, schedule, deadline; variables' identifier, type, range, default value, deadline.	HA_CONF_04	The Framework shall verify the validity of results coming from the inauguration (Train Topology Database or equivalent data structure) and their coherence with the Configuration file.	HA_CONF_05	The Framework shall not execute the Application functions in case of any error detected in the Configuration file or non-valid results coming from the inauguration or undue operation on the Configuration data, and notify a (fatal) Fault condition to all the Application function(s) involved.	-	-	HA_CONF_07	The Framework shall read, parse, load and check data in the Configuration file and configure the platform accordingly, with the same SIL assigned to the related Application function.	PHA_AC_08	The Application function shall react to the notification of a Fault condition due to error detected in the Configuration file or non-valid results coming from the inauguration (fatal Fault), by the transition into the specific safe state.		
Configuration management	reading, parsing, and loading of data in the configuration file	Wrong	Error during reading, parsing, or loading of data in the configuration file	Incorrect Platform initialization.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect configuration.	FDF_SH_08	HA_CONF_01	The Framework shall instantiate messages and variable according to the Configuration file, which specifies at least: messages' identifier, variables, to receive or to send, schedule, deadline; variables' identifier, type, range, default value, deadline.	HA_CONF_04	The Framework shall verify the validity of results coming from the inauguration (Train Topology Database or equivalent data structure) and their coherence with the Configuration file.	HA_CONF_05	The Framework shall not execute the Application functions in case of any error detected in the Configuration file or non-valid results coming from the inauguration or undue operation on the Configuration data, and notify a (fatal) Fault condition to all the Application function(s) involved.	-	-	HA_CONF_07	The Framework shall read, parse, load and check data in the Configuration file and configure the platform accordingly, with the same SIL assigned to the related Application function.	PHA_AC_08	The Application function shall react to the notification of a Fault condition due to error detected in the Configuration file or non-valid results coming from the inauguration (fatal Fault), by the transition into the specific safe state.		
Configuration management	reading, parsing, and loading of data in the configuration file	Wrong	Data corruption during reading, parsing, or loading of data in the configuration file	Incorrect Platform initialization.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect configuration.	FDF_SH_08	HA_CONF_01	The Framework shall instantiate messages and variable according to the Configuration file, which specifies at least: messages' identifier, variables, to receive or to send, schedule, deadline; variables' identifier, type, range, default value, deadline.	HA_CONF_03	The Framework shall verify the validity and integrity of the Configuration file, before and after the end of the inauguration services, e.g. by CRC, MD or signature created by tooling.	HA_CONF_05	The Framework shall not execute the Application functions in case of any error detected in the Configuration file or non-valid results coming from the inauguration or undue operation on the Configuration data, and notify a (fatal) Fault condition to all the Application function(s) involved.	HA_CONF_06	The Framework shall assure that re-configuration required for new or modified Application functions is performed involving all the Application functions to be executed, or anyway the existing configuration for the remaining Application functions is not altered.	HA_CONF_07	The Framework shall read, parse, load and check data in the Configuration file and configure the platform accordingly, with the same SIL assigned to the related Application function.	PHA_AC_08	The Application function shall react to the notification of a Fault condition due to error detected in the Configuration file or non-valid results coming from the inauguration (fatal Fault), by the transition into the specific safe state.		
Configuration management	reading, parsing, and loading of data in the configuration file	Wrong/undue	Loading of data in the configuration file at a wrong time (e.g. while the FDF has already been configured).	Use of incorrect data after the Platform initialization.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect configuration.	FDF_SH_08	HA_CONF_08	The Framework shall load the Configuration file during the execution of the inauguration services and assure that any re-configuration (re-loading of the Configuration file or loading of a new Configuration file) is performed involving all the Application functions to be executed.	HA_CONF_03	The Framework shall verify the validity and integrity of the Configuration file, before and after the end of the inauguration services, e.g. by CRC, MD or signature created by tooling.	HA_CONF_05	The Framework shall not execute the Application functions in case of any error detected in the Configuration file or non-valid results coming from the inauguration or undue operation on the Configuration data, and notify a (fatal) Fault condition to all the Application function(s) involved.	HA_CONF_06	The Framework shall assure that re-configuration required for new or modified Application functions is performed involving all the Application functions to be executed, or anyway the existing configuration for the remaining Application functions is not altered.	HA_CONF_07	The Framework shall read, parse, load and check data in the Configuration file and configure the platform accordingly, with the same SIL assigned to the related Application function.	PHA_AC_08	The Application function shall react to the notification of a Fault condition due to error detected in the Configuration file or non-valid results coming from the inauguration (fatal Fault), by the transition into the specific safe state.		
Configuration management	reading, parsing, and loading of data in the configuration file	Undue	Reading, parsing, and loading of data from a false or corrupted configuration file	Incorrect Platform initialization.	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to an incorrect configuration.	FDF_SH_08	HA_CONF_02	The Framework shall accept only certified remote Configuration file (coming from a verified source), protected against data corruption, e.g. by CRC.	HA_CONF_03	The Framework shall verify the validity and integrity of the Configuration file, before and after the end of the inauguration services, e.g. by CRC, MD or signature created by tooling.	HA_CONF_05	The Framework shall not execute the Application functions in case of any error detected in the Configuration file or non-valid results coming from the inauguration or undue operation on the Configuration data, and notify a (fatal) Fault condition to all the Application function(s) involved.	-	-	HA_CONF_07	The Framework shall read, parse, load and check data in the Configuration file and configure the platform accordingly, with the same SIL assigned to the related Application function.	PHA_AC_08	The Application function shall react to the notification of a Fault condition due to error detected in the Configuration file or non-valid results coming from the inauguration (fatal Fault), by the transition into the specific safe state.		
Functions management	execution of registered Functions according to their scheduling plans	No / missed	No execution of registered function(s) required by the scheduling plan(s) and process priority	Missed or partial execution of Application function(s).	Potential unsafe behaviour of the Platform in the execution of safety-related processes due to a missed or incorrect setting of commands (output) toward the interfaced object(s).	FDF_SH_06	HA_FNM_01	The Framework shall control the execution (start, stop, synchronizing to external trigger, ...) of Application functions assigned to each individual partition, through the deterministic management of timers (for sequential execution) and semaphores (for sequential and concurrent execution), according to their scheduling plans and to processes priority.	HA_FNM_06	The Framework shall monitor the execution (start, stop, synchronizing to external trigger, ...) of processes with respect to defined timing bounds for (intra-partition and inter-partition) communication and processing.	HA_FNM_07	The Framework shall notify a Fault condition, in case of error in the execution of processes according to the scheduling plans, including the violation of timing bounds (fatal Fault), to all the Application functions involved.	HA_FNM_08	The Framework shall implement temporal partitioning, by ensuring that a process within a given time budget cannot be affected by the actions of any other task from other partitions, in terms of rate, latency, jitter and duration of the scheduled access.	HA_FNM_09	The Framework shall control the execution of processes and the transmission of messages (according to their scheduling plans) with the same SIL assigned to the involved Application functions.	PHA_AC_02	The Application function shall react to the notification of a Fault condition due to error in the execution of processes according to the scheduling plans (fatal Fault), implementing tolerance (e.g. timing bounds violated for a limited number of times) if any, by the transition into the specific safe state.		
Functions management	execution of registered Functions according to their scheduling plans	Wrong	Error in the execution of function(s) with respect to the scheduling plan(s) and processes priority.	Missed execution of Application function(s). Undue execution of Application function(s) when not required, with potential disturb to the time partitioning.	Potential unsafe behaviour of the Platform due to a wrong timing in the execution of the safety-related Application functions	FDF_SH_03	HA_FNM_05	The Framework shall avoid interrupts or manage them through the Operating system only (even if triggered by the Application functions or by hardware), avoiding any disturb to the time partitioning, i.e. without any change of the time budget allocation.	HA_FNM_06	The Framework shall monitor the execution (start, stop, synchronizing to external trigger, ...) of processes with respect to defined timing bounds for (intra-partition and inter-partition) communication and processing.	HA_FNM_07	The Framework shall notify a Fault condition, in case of error in the execution of processes according to the scheduling plans, including the violation of timing bounds (fatal Fault), to all the Application functions involved.	HA_FNM_08	The Framework shall implement temporal partitioning, by ensuring that a process within a given time budget cannot be affected by the actions of any other task from other partitions, in terms of rate, latency, jitter and duration of the scheduled access.	HA_FNM_09	The Framework shall control the execution of processes and the transmission of messages (according to their scheduling plans) with the same SIL assigned to the involved Application functions.	PHA_AC_02	The Application function shall react to the notification of a Fault condition due to error in the execution of processes according to the scheduling plans (fatal Fault), implementing tolerance (e.g. timing bounds violated for a limited number of times) if any, by the transition into the specific safe state.		
Functions management	execution of registered Functions according to their scheduling plans	Loss of / partially	Incomplete execution of registered Function(s) with respect to the scheduling plan(s)	Missed or partial execution of Application function(s).	Potential unsafe behaviour of the Platform due to a wrong timing in the execution of the safety-related Application functions	FDF_SH_03	HA_FNM_01	The Framework shall control the execution (start, stop, synchronizing to external trigger, ...) of Application functions assigned to each individual partition, through the deterministic management of timers (for sequential execution) and semaphores (for sequential and concurrent execution), according to their scheduling plans and to processes priority.	HA_FNM_06	The Framework shall monitor the execution (start, stop, synchronizing to external trigger, ...) of processes with respect to defined timing bounds for (intra-partition and inter-partition) communication and processing.	HA_FNM_07	The Framework shall notify a Fault condition, in case of error in the execution of processes according to the scheduling plans, including the violation of timing bounds (fatal Fault), to all the Application functions involved.	HA_FNM_08	The Framework shall implement temporal partitioning, by ensuring that a process within a given time budget cannot be affected by the actions of any other task from other partitions, in terms of rate, latency, jitter and duration of the scheduled access.	HA_FNM_09	The Framework shall control the execution of processes and the transmission of messages (according to their scheduling plans) with the same SIL assigned to the involved Application functions.	PHA_AC_02	The Application function shall react to the notification of a Fault condition due to error in the execution of processes according to the scheduling plans (fatal Fault), implementing tolerance (e.g. timing bounds violated for a limited number of times) if any, by the transition into the specific safe state.		
Functions management	execution of registered Functions according to their scheduling plans	Delayed	Delayed execution of registered Function(s) with respect to the scheduling plan(s)	Execution of Application function(s) with an excessive response time (i.e. not compatible with specific constraints).	Potential unsafe behaviour of the Platform due to a wrong timing in the execution of the safety-related Application functions	FDF_SH_03	HA_FNM_03	The Framework shall implement Service functions whose response times allow the real-time execution of processes and the fulfilment of the most restrictive response time required by the Application functions to be executed.	HA_FNM_06	The Framework shall monitor the execution (start, stop, synchronizing to external trigger, ...) of processes with respect to defined timing bounds for (intra-partition and inter-partition) communication and processing.	HA_FNM_07	The Framework shall notify a Fault condition, in case of error in the execution of processes according to the scheduling plans, including the violation of timing bounds (fatal Fault), to all the Application functions involved.	HA_FNM_08	The Framework shall implement temporal partitioning, by ensuring that a process within a given time budget cannot be affected by the actions of any other task from other partitions, in terms of rate, latency, jitter and duration of the scheduled access.	HA_FNM_09	The Framework shall control the execution of processes and the transmission of messages (according to their scheduling plans) with the same SIL assigned to the involved Application functions.	PHA_AC_02	The Application function shall react to the notification of a Fault condition due to error in the execution of processes according to the scheduling plans (fatal Fault), implementing tolerance (e.g. timing bounds violated for a limited number of times) if any, by the transition into the specific safe state.		
Functions management	execution of registered Functions according to their scheduling plans	Wrong	Error in the execution of function(s) with respect to the scheduling plan(s) and processes priority.	Missed execution of Application function(s) because of reduced time budget than initially allocated.	Potential unsafe behaviour of the Platform due to a wrong timing in the execution of the safety-related Application functions	FDF_SH_03	HA_FNM_04	The Framework shall implement mechanisms to ensure the execution of real-time processes in spite of transient temporal violations, e.g. due to inter-module communications acknowledgements, time-outs, access to memory, interrupts.	HA_FNM_06	The Framework shall monitor the execution (start, stop, synchronizing to external trigger, ...) of processes with respect to defined timing bounds for (intra-partition and inter-partition) communication and processing.	HA_FNM_07	The Framework shall notify a Fault condition, in case of error in the execution of processes according to the scheduling plans, including the violation of timing bounds (fatal Fault), to all the Application functions involved.	HA_FNM_08	The Framework shall implement temporal partitioning, by ensuring that a process within a given time budget cannot be affected by the actions of any other task from other partitions, in terms of rate, latency, jitter and duration of the scheduled access.	HA_FNM_09	The Framework shall control the execution of processes and the transmission of messages (according to their scheduling plans) with the same SIL assigned to the involved Application functions.	PHA_AC_02	The Application function shall react to the notification of a Fault condition due to error in the execution of processes according to the scheduling plans (fatal Fault), implementing tolerance (e.g. timing bounds violated for a limited number of times) if any, by the transition into the specific safe state.		
Functions management	execution of registered Functions according to their scheduling plans	Undue	Undue execution of registered Functions, when not required by the scheduling plan(s)	Undue execution of Application function(s) and access to memory when not required, with potential disturb to the time partitioning.	Potential unsafe behaviour of the Platform due to a wrong timing in the execution of the safety-related Application functions	FDF_SH_03	HA_FNM_02	The Framework shall execute an Application function, giving access to memory resources, only when required by its scheduling plan (and take away access otherwise).	HA_FNM_06	The Framework shall monitor the execution (start, stop, synchronizing to external trigger, ...) of processes with respect to defined timing bounds for (intra-partition and inter-partition) communication and processing.	HA_FNM_07	The Framework shall notify a Fault condition, in case of error in the execution of processes according to the scheduling plans, including the violation of timing bounds (fatal Fault), to all the Application functions involved.	HA_FNM_08	The Framework shall implement temporal partitioning, by ensuring that a process within a given time budget cannot be affected by the actions of any other task from other partitions, in terms of rate, latency, jitter and duration of the scheduled access.	HA_FNM_09	The Framework shall control the execution of processes and the transmission of messages (according to their scheduling plans) with the same SIL assigned to the involved Application functions.	PHA_AC_02	The Application function shall react to the notification of a Fault condition due to error in the execution of processes according to the scheduling plans (fatal Fault), implementing tolerance (e.g. timing bounds violated for a limited number of times) if any, by the transition into the specific safe state.		

FUNCTIONAL FAILURE MODE				FAILURE EFFECTS			COUNTERMEASURES SPECIFICATION													
							Correct functional operation		Detection of faults		Action following Detection		Independence of Items		Systematic & Random faults		Application conditions		Recommendations	
Sub-function	Description	Guide-word	Deviation / Functional Failure mode	Local effect	Final effect	Hazard ID	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description	ID	Description
Fault management	detection, isolation, notification and reaction to faults, and the recognition of system status with respect to errors and failures	No / missed	Missed detection of faults of (hardware) resources used by Service and Application functions.	Faults of (hardware) resources used by Service and Application functions can be latent (i.e. not detected) and lead (alone or with further concurrent faults) to an incorrect execution of Service and Application functions.	Potential unsafe behaviour of the Platform in the execution of the safety-related Application functions due to an incorrect management of fault condition(s).	FDF_SH_02	HA_FLT_01	The Framework shall provide services for the detection of faults of (hardware) resources used by Service and Application functions, at the power up (i.e. during the initialization) and periodically during the operation (nominal and degraded phases), e.g. test memories containing safety related data are totally tested at the initialization phase and at any new allocation and cyclically at run-time.	HA_FLT_06	The Framework shall verify the capability to notify a Fault condition under a representative set of failure scenarios.	HA_FLT_08	The Framework, after the detection of a condition that blocks or threats the proper execution of Service or Application functions (fatal Fault), shall notify a Fault condition to all the Application functions involved, in a time that is compatible with their timely transition into safe state (i.e. not later than the maximum time for failure detection and negation specified by the Applications).	-	-	HA_FLT_10	The Framework shall detect, isolate, notify and react to fault with the highest SIL assigned to the safety-related Application functions to be executed.	PHA_AC_07	The Application function shall react to any fatal Fault notified by the Framework (i.e. condition that blocks or threat its proper execution) through the transition and retention into its safe state, by blocking its safety-related functions and maintaining all outputs to their restrictive state (typically de-energized), till the execution of a defined maintenance procedure.	PHA_REC_03	It is recommended to execute services for faults detection at physical (e.g. temperature, voltage, memories failures), temporal and logical (e.g. error detecting codes, program sequence monitoring), and functional (e.g. configuration data integrity, spatial separation between resources) levels, at the power up (i.e. during the initialization) and periodically during the operation (nominal and degraded phases).
Fault management	detection, isolation, notification and reaction to faults, and the recognition of system status with respect to errors and failures	No / missed	Missed detection of faults during the generation of the application software code	Faults during the generation of the software code can be latent (i.e. not detected) and lead to an incorrect execution of the Application function.	Potential unsafe behaviour of the Platform in the execution of the safety-related Application functions due to an incorrect management of fault condition(s).	FDF_SH_02	HA_FLT_02	The Framework shall provide services for the detection of faults during the installation of the Applications software (otherwise, to be required to the Applications).	HA_FLT_06	The Framework shall verify the capability to notify a Fault condition under a representative set of failure scenarios.	HA_FLT_07	The Framework shall inhibit the execution of the Application function in case of negative results of the initial code integrity check.			HA_FLT_10	The Framework shall detect, isolate, notify and react to fault with the highest SIL assigned to the safety-related Application functions to be executed.	PHA_AC_07	The Application function shall react to any fatal Fault notified by the Framework (i.e. condition that blocks or threat its proper execution) through the transition and retention into its safe state, by blocking its safety-related functions and maintaining all outputs to their restrictive state (typically de-energized), till the execution of a defined maintenance procedure.	PHA_REC_04	It is recommended to implement means for the recognition of system status with respect to errors and failures that might occur or have occurred, supporting faults isolation and graceful degradation, in order to maintain the more critical Application functions available despite failures by dropping the less critical functions.
Fault management	detection, isolation, notification and reaction to faults, and the recognition of system status with respect to errors and failures	No / missed	Missed detection of faults during the run-time execution of the application software code	Faults during the execution of the software code can lead to an incorrect execution of the Application function.	Potential unsafe behaviour of the Platform in the execution of the safety-related Application functions due to an incorrect management of fault condition(s).	FDF_SH_02	HA_FLT_03	The Framework shall provide services for the detection of faults during the run-time execution of the Application function code (otherwise, to be required to the Application function), e.g. by monitoring the process and data flow and comparing their state to configured constraints (Program Flow Monitoring), by checking variables values against predefined range and for plausibility, by detecting and correcting errors in sensitive information (Error Detecting and Correcting Codes).	HA_FLT_06	The Framework shall verify the capability to notify a Fault condition under a representative set of failure scenarios.	HA_FLT_08	The Framework, after the detection of a condition that blocks or threats the proper execution of Service or Application functions (fatal Fault), shall notify a Fault condition to all the Application functions involved, in a time that is compatible with their timely transition into safe state (i.e. not later than the maximum time for failure detection and negation specified by the Applications).			HA_FLT_10	The Framework shall detect, isolate, notify and react to fault with the highest SIL assigned to the safety-related Application functions to be executed.	PHA_AC_07	The Application function shall react to any fatal Fault notified by the Framework (i.e. condition that blocks or threat its proper execution) through the transition and retention into its safe state, by blocking its safety-related functions and maintaining all outputs to their restrictive state (typically de-energized), till the execution of a defined maintenance procedure.	PHA_REC_05	It is recommended to implement Validation and verification support service that allows fault injection and reaction monitoring, including faults of non-safety related Service and Application functions, partitioning and isolation mechanism, communication (transmission, reception) and sharing of network and memory resources, output control, input monitoring, application execution (timing, memory access, start, stop, throttling).
Fault management	detection, isolation, notification and reaction to faults, and the recognition of system status with respect to errors and failures	Wrong / Delay	Ineffective reaction to a detected fault	Missed or delayed transition into a safe state (in case of fault impacting safety-related Application function(s)).	Potential unsafe behaviour of the Platform in the execution of the safety-related Application functions due to an incorrect management of fault condition(s).	FDF_SH_02	HA_FLT_04	The Framework shall execute services for Fault detection, isolation, notification and reaction processes with the highest priority, without any disturb or unintended effects due to other Service and Application functions.	HA_FLT_06	The Framework shall verify the capability to notify a Fault condition under a representative set of failure scenarios.	HA_FLT_08	The Framework, after the detection of a condition that blocks or threats the proper execution of Service or Application functions (fatal Fault), shall notify a Fault condition to all the Application functions involved, in a time that is compatible with their timely transition into safe state (i.e. not later than the maximum time for failure detection and negation specified by the Applications).	HA_FLT_09	The framework shall manage the interaction between Service and Application functions: _avoiding that Service functions can force the outputs independently from the Application function when active, during operation (normal and degraded phases); _preventing the access to any off-line service (e.g. validation and verification support) at the power up, and during the initialization and the operating (nominal and degraded) phases; _guarantying the retention of a safe state after a fatal Fault (i.e. condition that blocks or threats the proper execution of Service or Application functions).	HA_FLT_10	The Framework shall detect, isolate, notify and react to fault with the highest SIL assigned to the safety-related Application functions to be executed.	PHA_AC_07	The Application function shall react to any fatal Fault notified by the Framework (i.e. condition that blocks or threat its proper execution) through the transition and retention into its safe state, by blocking its safety-related functions and maintaining all outputs to their restrictive state (typically de-energized), till the execution of a defined maintenance procedure.	PHA_REC_06	It is recommended to avoid dynamic reconfiguration of software after a failure , i.e. remapping the logical architecture back onto the restricted resources left functioning (highly recommended for SIL3-SIL4 Applications, EN 50128 Table A.3).
Fault management	detection, isolation, notification and reaction to faults, and the recognition of system status with respect to errors and failures	Undue	Interaction of the Fault management services with other Service or Application functions.	Incorrect execution of Service or Application functions due to Fault management services.	Potential unsafe behaviour of the Platform in the execution of the safety-related Application functions due to an incorrect management of fault condition(s).	FDF_SH_02	HA_FLT_05	The Framework shall provide services for Fault detection and isolation without any disturb or unintended effects on the execution and performance (e.g. latency/jitter, sampling rate or resource reservation) of other Service and Application functions.	HA_FLT_06	The Framework shall verify the capability to notify a Fault condition under a representative set of failure scenarios.	HA_FLT_08	The Framework, after the detection of a condition that blocks or threats the proper execution of Service or Application functions (fatal Fault), shall notify a Fault condition to all the Application functions involved, in a time that is compatible with their timely transition into safe state (i.e. not later than the maximum time for failure detection and negation specified by the Applications).	HA_FLT_09	The framework shall manage the interaction between Service and Application functions: _avoiding that Service functions can force the outputs independently from the Application function when active, during operation (normal and degraded phases); _preventing the access to any off-line service (e.g. validation and verification support) at the power up, and during the initialization and the operating (nominal and degraded) phases; _guarantying the retention of a safe state after a fatal Fault (i.e. condition that blocks or threats the proper execution of Service or Application functions).	HA_FLT_10	The Framework shall detect, isolate, notify and react to fault with the highest SIL assigned to the safety-related Application functions to be executed.	PHA_AC_07	The Application function shall react to any fatal Fault notified by the Framework (i.e. condition that blocks or threat its proper execution) through the transition and retention into its safe state, by blocking its safety-related functions and maintaining all outputs to their restrictive state (typically de-energized), till the execution of a defined maintenance procedure.		

ANNEX B: Functional Security Assessment Requirements table

This annex contains the complete table of the Functional Security Assessment Requirements for IEC 62443 certification mapped with already-defined requirements in D2.5, countermeasures and software components that will implement it.

Tabla 25. Security requirements

Requirement ID	Reference Name	Requirement Description	Previous Requirement	Countermea-sure	624443 - Level	SW component implementation /Clarification	Security Objectives
FSA-AC-1	Access Control Authorization	The IACS embedded device shall provide capability for user configured Access Control Functionality to facilitate automated enforcement of a site specific Access Control Policy based upon authenticated entities.			NA		
FSA-AC-1.1	Role Based Access	The IACS embedded device Access Control Functionally shall provide the capability to support role based access control policies.	None	C4	>1	"UserAccountManager":user management is required, which prevents non-authorized access on FDF services and sensitive data	SO1, SO2
FSA-AC-1.2	Dual Approval Access	The IACS embedded device Access Control Functionally shall provide the capability to support dual-approval mechanisms as an access control option for user modification or control of critical parameters or actions.	None	C1, C2	>1	"SecurityMonitoringManager" Approval for access can be granted by the system and by notifying administrator	SO1, SO2, SO4
FSA-AC-1.3	Least Privilege Default Access	New Access Accounts for the Access Control shall be created by default based on least privileges requiring explicit action by the account administrator to raise privilege level.	None	C3, C4	>1	"UserAccountManager"shall apply least privilege philosophy	SO1
FSA-AC-1.4	Administrator User Role	The IACS embedded device Access Control Functionally shall provide support for an administrator user role which has the ability to create user accounts and manage the privileges of other users	None	C3, C4	>1	"UserAccountManager" only administrator user can create new user accounts and manage their privileges. The authorization will come from "SecurityMonitoringManager"	SO1
FSA-AC-1.5	Administrator Support Functions	The IACS embedded device shall provide the administrator the ability list of all current user accounts and login history such as time of last login	None	NA	>2	"SecurityMonitoringManager" shall provide a way to list of user accounts and login history	SO1
FSA-AC-2.1	Authentication by User ID and Password	The IACS embedded device shall support user authentication via entry of user ID and password.			NA	"SecurityMonitoringManager"	SO1
FSA-AC-2.1.1	User Management of Password	The IACS embedded device shall provide the capability for [IACS Administrator] or the user to modify password within their control without impacting normal operation	None	C1, C2	ALL	"SecurityMonitoringManager"	SO1
FSA-AC-2.1.2	Monitor Unsuccessful Login Attempts	The IACS embedded device shall monitor and record the number and time of unsuccessful login attempts per user id since the last successful login.	None	NA	ALL	"SecurityMonitoringManager": takes care of the most security functions namely: User authentication, access authorization, application deployment, and continuous security monitoring.	SO1
FSA-AC-2.1.3	Record Successful Logins	The IACS embedded device shall monitor and record the date and time of the last successful login.	None	NA	ALL	"SecurityMonitoringManager"	SO1
FSA-AC-2.1.4	Display Previous Login History	Following successful user authentication the IACS embedded device shall display the date and time of the last successful login plus the number of unsuccessful login attempts for this user ID since that time.	None	NA	>2	"SecurityMonitoringManager"	SO1

Requirement ID	Reference Name	Requirement Description	Previous Requirement	Countermea-sure	624443 - Level	SW component implementation /Clarification	Security Objectives
FSA-AC-2.1.5	Password Modification Reminder	Following successful user authentication the IACS embedded device shall provide the capability for automated reminder of need to modify user password after [[IACS administrator defined time] has passed since the last password modification.	None	NA	>2	"SecurityMonitoringManager"	SO1
FSA-AC-2.1.6	Password Strength Enforcement	The IACS embedded device shall provide the capability to only accept user requested password updates for passwords that meet [[IACS administrator configured] criteria for strong passwords based on minimum length, use of upper / lower case and non-alpha characters.	None	C2	>2	"SecurityMonitoringManager"	SO1
FSA-AC-2.1.7	Action for High Number of Unsuccessful Login	The IACS embedded device shall provide option to take [IACS administrator configured] action if the number of unsuccessful login attempts exceeds a user configured value with in a [user configured time period].	None	C2	ALL	"SecurityMonitoringManager"	SO1
FSA-AC-2.1.8	Minimum Password Capability	User authentication through manual login with password shall support a minimum of 6 character passwords to be used	None	C2	ALL	"SecurityMonitoringManager"	SO1
FSA-AC-2.1.9	Clear Text Passwords	The IACS embedded device shall not internally store or send password over shared networks in clear text format.	None	NA	ALL	"SecurityMonitoringManager"	SO5
FSA-AC-2.1.10	Cryptographic Password Protection	The IACS embedded device passwords shall have cryptographic protection for transmission over networks	S4R_FDF_412	C2, C5	>2	"CryptoManager"	SO5
FSA-AC-2.1.11	Access Control for All Exposed Services	The IACS embedded device user authentication shall cover access to all services supported by the device during normal operation	None	NA	ALL	"SecurityMonitoringManager"	SO1, SO2, SO3
FSA-AC-2.2	Other Authentication Methods	The IACS embedded device authentication may provide optional interfaces to support alternative user authentication methods.	S4R_FDF_419		Not required		
FSA-AC-2.3	Two Factor Authentication (local network)	The IACS embedded device Access Control Functionally shall support two factor authentication mechanisms.	None	C1, C2	>2	"SecurityMonitoringManager" Username and password and the USB or smartcard with user credentials	SO1, SO2
FSA-AC-2.4	Two Factor Authentication (remote)	The IACS embedded device Access Control Functionally shall support two factor authentication mechanisms for remote access.	None	NA	ALL	"SecurityMonitoringManager"if remote access is required	SO1, SO2
FSA-AC-2.5	Authentication Feedback	The IACS embedded device shall obscure feedback of authentication information during the authentication process to protect the information from possible exploitation/use by unauthorized individuals.	None	NA	ALL	"CryptoManager"	SO1, SO2
FSA-AC-3	System Use Notification	The IACS embedded device authentication shall provide option for presenting an [IACS administrator] provided "system use notification message" before granting system access informing potential users they are entering a redistricted area.	None	NA	ALL	"SecurityMonitoringManager"	
FSA-AC-4	Local Session Locking Timeout	The IACS embedded device authentication shall provide option for session locking after a [IACS administrator] specified period of time of inactivity for the session.	None	C6	>1	"SecurityMonitoringManager"	

Requirement ID	Reference Name	Requirement Description	Previous Requirement	Countermea-sure	624443 - Level	SW component implementation /Clarification	Security Objectives
FSA-AC-5	Remote Session Termination Timeout	The IACS embedded device authentication shall provide option for automated session termination for remote sessions after a user specified period of time of inactivity for the session.	None	C6	>1	"SecurityMonitoringManager"	
FSA-UC-1	Wireless Access	The IACS embedded device wireless access must be protected via sufficient authentication and encryption protection.	None	NA	NA	Not applicable	
FSA-UC-1.1	Physical Disable Wireless Access	IACS embedded products that provide wireless access must provide the option to be physically disabled by the end user of the product by a method unable to be overridden by SW or user soft configuration	None	NA	ALL	Not applicable, since wireless it is not considered	
FSA-UC-2	Device Authentication	The IACS embedded device shall provide authentication methods for device identification prior to establishing a connection to support Access Management and Use Control Functionality.	None	C1, C8	NA		
FSA-UC-2.1	Failures in Cryptography Services	IACS embedded device shall not be dependent on outside cryptography services that could result in denial of service for the embedded device if the service were no longer available	None	C1, C5	ALL	"CryptoManager" TPM by means of TPM	SO4, SO5, SO6
FSA-UC-2.2	Basic Device Authentication	The IACS embedded device shall provide at least basic measures for authentication of device identification	None	C1, C5	>1	"CryptoManager" by means of TPM	SO4, SO5, SO6
FSA-UC-2.3	Cryptographic Device Authentication	The IACS embedded device shall provide cryptographic measures for authentication of device identification	None	C1, C5	>2	"CryptoManager" by means of TPM	SO4, SO5, SO6
FSA-UC-3	Creation of Audit Trail	The IACS embedded device shall provide option for generation and storage of audit information for post security incident and process improvement activities.			NA		
FSA-UC-3.1	Configuration of Audit Events	The IACS embedded device shall provide for [IACS administrator] configuration of what events are included in list of auditable events.	S4R_FDF_430	NA	>2	"SecurityMonitoringManager"	SO1, SO2
FSA-UC-3.2	Content of Audit Record	The IACS embedded device shall provide for [IACS administrator] configuration of required information for each auditable event.	None	NA	NA	"SecurityMonitoringManager"	SO1, SO2
FSA-UC-3.2.1	Time Stamp for Audit	The IACS embedded device shall provide time stamps for use in audit record generation based on "system time".	None	NA	>1	"SecurityMonitoringManager"	SO1, SO2
FSA-UC-3.2.2	Information for Non-repudiation	The IACS embedded device or the responsible higher level component shall provide the capability to include in the audit trail which device or individual initiated or performed a particular action.	None	NA	>2	"SecurityMonitoringManager"	SO1, SO2
FSA-UC-3.2.3	Additional Content for Audit Record	The IACS embedded device shall provide the capability to include additional, more detailed information in the audit records for audit events identified by type, location, or subject.	None	NA	>2	"SecurityMonitoringManager"	SO1, SO2

Requirement ID	Reference Name	Requirement Description	Previous Requirement	Countermea-sure	624443 - Level	SW component implementation /Clarification	Security Objectives
FSA-UC-3.3	Protection of Audit Information	The IACS embedded device shall protect audit information and audit tools from unauthorized access, modification, and deletion.	None	C1, C2, C3, C4	NA	"SecurityMonitoringManager"	SO1, SO2
FSA-UC-3.3.1	Audit Fault Warning	The IACS embedded device or a higher level component shall alert appropriate organizational officials in the event of an auditprocessing failure and support additional configurable actions (e.g., overwrite oldest audit records, stop generating audit records).	None	NA	>2	"SecurityMonitoringManager"	SO1, SO2
FSA-UC-3.3.2	Basic Protection of Audit	The IACS embedded device shall provide basic noncryptographic measures for protection of audit information	None	NA	>1	"CryptoManager" -- Digital signature, digital message receipts, time stamps	SO1, SO2
FSA-UC-3.3.3	Cryptographic Protection of Audit Information	The IACS embedded device shall provide cryptographic measures for protection of audit information	None	C1, C5	>2	"CryptoManager"	SO4
FSA-UC-3.4	System Wide Audit	The IACS embedded device shall provide capability to pass [IACS administrator configurable] auditable events to another device for creation of a higher level consolidated audit log.	None	NA	>2	"SecurityMonitoringManager"	SO4
FSA-UC-3.5	Audit Report Generation	The IACS embedded device or the responsible higher level component shall provide an audit reduction and report generation capability for audit reduction, review, and reporting tools support after-the-fact investigations of security incidents without altering original audit records.	None	NA	>1	"CryptoManager"	SO4
FSA-DI-1	Integrity of Data in Transit	The IACS embedded device shall protect the integrity of transmitted information			NA		
FSA-DI-1.1	Insertion of Data Packets	The IACS embedded device shall protect the integrity of transmitted information against insertion of data packets not intended to be part of the transmitted data	S4R_FDF_410	C5	>1	"SecurityMonitoringManager"	SO5
FSA-DI-1.2	Deletion of Data Packets	The IACS embedded device shall protect the integrity of transmitted information against deletion of data packets	None	NA	>1	"SecurityMonitoringManager"	SO5
FSA-DI-1.3	Excessive Delay of Data Packets	The IACS embedded device shall protect the integrity of transmitted information against delay of data packets by more than tolerable by the intended application	None	NA	>1	"SecurityMonitoringManager"	SO5
FSA-DI-1.4	Re-sequencing or Replay of Data Packets	The IACS embedded device shall protect the integrity of transmitted information against re-sequencing or replay of data packets	None	NA	>1	"SecurityMonitoringManager"	SO5
FSA-DI-1.5	Basic Modification of Transmitted Data	The IACS embedded device shall employ basic mechanisms to recognize changes to information during communication independent of the basic communication protocol stack	None	NA	>1	From the functional safety, CRC can be used to recognize changes	SO5
FSA-DI-1.6	Modification of Transmitted Data	The IACS embedded device shall employ cryptographic mechanisms to recognize changes to information during communication	None	C1, C5	>1	"SecurityMonitoringManager"	SO5
FSA-DI-1.7	Point to point Communications	All point to point communication connections to the IACS embedded device shall provide sufficient security measures to insure communications only take place with properly authorized parties	S4R_FDF_410	C1, C5	>2	"SecurityMonitoringManager"	SO1, SO5

Requirement ID	Reference Name	Requirement Description	Previous Requirement	Countermea-sure	624443 - Level	SW component implementation /Clarification	Security Objectives
FSA-DI-1.7.1	Session Creation	All point to point communication connections to the IACS embedded device shall provide measures to properly identify and authenticate the other party prior to approving the connection	Nonce	C1, C6, C5	NA	"SecurityMonitoringManager"	SO1, SO5
FSA-DI-1.7.2	Basic Session Protection	All point to point communication connections to the IACS embedded device shall provide measures to protect the integrity of authorized sessions and prevent others from participating in or stealing the authorized session	S4R_FDF_411	C1, C6, C5	>1	"SecurityMonitoringManager"	SO1, SO5
FSA-DI-1.7.3	Crypto Session Protection	All point to point communication connections to the IACS embedded device shall provide cryptographic measures to protect the integrity of authorized sessions and prevent others from participating in or stealing the authorized session	None	C1, C6, C5	>1	"Network Manager" "Crypto Manager"	SO1, SO2, SO3, SO5
FSA-DI-1.7.4	Session Closure	All point to point communication connections to the IACS embedded device shall have a method to close the session when the purpose of the session has be completed or session is no longer required	None	C6	>2	"SecurityMonitoringManager"	SO1, SO2, SO3, SO5
FSA-DI-1.7.5	Session Timeout	All point to point communication connections to the IACS embedded device shall have a method to close the session when it has been open or inactive for longer than a [IACS administrator] configured time	None	C6	>2	"SecurityMonitoringManager"	SO1, SO2, SO3, SO4, SO5
FSA-DI-1.8	Multicast / Broadcast Communications	All multicast / broadcast communication connections to the IACS embedded device shall provide product measures for [IACS administrator] to manage security of broadcast communications or sufficient information disclosure and security measures to allow proper management of its capabilities	None	C5	NA		
FSA-DI-1.8.1	Multicast Restrictions	The IACS embedded device shall only use critical information from multicast transmissions for which it can properly validate the source and integrity of the transmission	None	NA	>2	"SecurityMonitoringManager"	SO1, SO5
FSA-DI-1.8.2	Multicast Reception Protection	The IACS embedded device using critical data from a multicast source shall verify multicast transmissions continue to originate from a properly validated source and verify integrity of the transmission	None	NA	>2	"SecurityMonitoringManager"	SO1, SO5
FSA-DI-1.8.3	Multicast Transmission Restrictions	The IACS embedded device multicast transmissions shall include measures to only allow properly authorized devices to subscribe to its multicast transmission or alternatively clearly document means for users to restrict the propagation of the multicast signal within a controlled region of the network	None	NA	>2	"SecurityMonitoringManager"	SO1, SO5
FSA-DI-1.9	Verify Input Data Syntax	The IACS embedded device shall check information for reasonability of values, completeness, validity, and authenticity	S4R_FDF_411		>1	"SecurityMonitoringManager"	SO1, SO5
FSA-DI-1.10	Handling Error Conditions	The IACS embedded device shall identify and handle error conditions in an expeditious manner without providing information that could be exploited by adversaries	None	NA	>1	"SecurityMonitoringManager"	SO1, SO5
FSA-DI-2	Integrity of Data at Rest Measures	The IACS embedded device shall protect the integrity of data stored within the device by measures independent of access control			NA		

Requirement ID	Reference Name	Requirement Description	Previous Requirement	Countermea-sure	624443 - Level	SW component implementation /Clarification	Security Objectives
FSA-DI-2.1	Protection of Static Data	The IACS embedded device shall protect against unauthorized changes to software and information	S4R_FDF_429	C1, C2, C3, C4	NA	"SecurityMonitoringManager" It is essential the protection of configuration files and sensitive data	SO1, SO2, SO3, SO4
FSA-DI-2.1.1	Disable Unused Ports	The IACS embedded device shall provide the user the capability to disable communication services and ports that are not required for normal online use for their particular application or not covered by Access Control measures	None	None	ALL	"SecurityMonitoringManager"	SO1, SO5, SO6
FSA-DI-2.1.2	Write Protection	The IACS embedded device shall have independent hardware and/or software measures to prevent writing to static data	None	C1, C2	>2	"SecurityMonitoringManager"	SO1, SO2, SO3, SO4
FSA-DI-2.2	Detection of Unauthorized Changes	The IACS embedded device shall employ mechanisms to automatically recognize changes to static data stored in memory able to be modified but not automatically modified during normal operation			NA		
FSA-DI-2.2.1	Executable Code Basic Mod Protection	The IACS embedded device shall implement basic means to detect modifications to executable code if susceptible to this type of threat within vendor published time interval	None	NA	>1	"SecurityMonitoringManager" together with "FrameworkManager" protection against malicious software, blacklisting, checker	SO1, SO2, SO3, SO4, SO5, SO6
FSA-DI-2.2.2	Executable Code Crypto Mod Protection	The IACS embedded device shall implement cryptographic means to detect modifications to executable code if susceptible to this type of threat within vendor published time interval	None	C1, C5	>2	"Security Manager"	SO1, SO2, SO3
FSA-DI-2.2.3	App Configuration Basic Protection	The IACS embedded device shall implement basic means to detect unauthorized modification, deletion or insertion of user application configuration data within vendor published time interval	None	NA	>1	"SecurityMonitoringManager" - blacklisting, whitelisting, checker	SO1, SO2, SO3, SO4
FSA-DI-2.2.4	App Configuration Crypto Protection	The IACS embedded device shall implement cryptographic means to detect unauthorized modification, deletion or insertion of user application configuration data within vendor published time interval	None	C1,C5	>2	"CryptoManager"	SO4
FSA-DI-2.2.5	Verify Application Specific Syntax	The IACS embedded device shall check application input and program configuration information for reasonability of values, completeness, validity, and correctness of syntax or include crypto protection of application against code modification (FSA-DI-2.2.4)	None	NA	>1	"SecurityMonitoringManager"	SO2, SO3
FSA-DI-2.2.6	OS Basic Configuration Protection	New Access Accounts for the Access Control shall be created by default based on least privileges requiring explicit action by the account administrator to raise privilege level.	None	NA	>1	Statically created accounts	SO2, SO3
FSA-DI-2.2.7	OS Crypto Configuration Protection	The IACS embedded device shall implement cryptographic means to detect modification, deletion or insertion of data that is capable of modifying the behavior or operation of the product's operating system such as exception vectors or scheduling, if an OS is used by the product.	None	C1, C5	>2	"CryptoManager"	SO2, SO3
FSA-DI-2.2.8	Basic Executable Code Insert Protection	The IACS embedded device shall implement means to prevent or detect insertion of malicious code within vendor published time interval	None	C1, C3,	>1	"SecurityMonitoringManager"	SO2, SO3

Requirement ID	Reference Name	Requirement Description	Previous Requirement	Countermea-sure	624443 - Level	SW component implementation /Clarification	Security Objectives
FSA-DI-2.2.9	Crypto Executable Code Insert Protection	The IACS embedded device shall implement means to prevent or cryptographic means to detect insertion of malicious code within vendor published time interval	None	C1, C3, C5	>2	"Security Manager"	SO2, SO3
FSA-DI-2.2.10	Non Execution of Data	The IACS embedded device shall have separate memory spaces for data versus executable code and have measures to prevent execution of code located in data space	None	C9	>2	"SecurityMonitoringManager"	SO2, SO3
FSA-DI-3	Auto Verify Security Functions	The IACS embedded device shall periodically verify the correct operation of security protection functions and notify system administrator when anomalies are discovered.	None	NA	>2	"SecurityMonitoringManager"	SO2, SO3
FSA-DC-1	Confidentiality of Data in Transit	The IACS embedded device shall protect the confidentiality of transmitted information			NA		
FSA-DC-1.1	No Clear Text in Data Transit	The IACS embedded device shall not send any data in clear text format for basic prevention of unauthorized disclosure of information during communication	None	C5	ALL	"CryptoManager"	SO1, SO5
FSA-DC-1.2	Cryptographic Protection for Data Confidentiality	The IACS embedded device shall employ cryptographic mechanisms to prevent unauthorized disclosure of information during communication	S4R_FDF_412	C5	>1	"CryptoManager" Due to time performance restrictions, it is to be considered only sensitive information to be encrypted or all data.	SO4
FSA-DC-1.3	Cryptographic Key Management	The IACS embedded device shall provide automated support for automation of cryptographic key management	S4R_FDF_412	C1	>2	"CryptoManager"	SO1, SO2, SO4, SO5
FSA-DC-2	Confidentiality of Data at Rest	The IACS embedded device shall provide measures to protect confidentiality of stored information			NA		
FSA-DC-2.1	Basic Confidentiality of Data at Rest	The IACS embedded device shall use storage in non clear text formats to provide measures for sensitive data storage to protect confidentiality of stored information	S4R_FDF_412	C1, C5	>1	"CryptoManager" base64 encoding	SO4
FSA-DC-2.2	Crypto Confidentiality of Data at Rest	The IACS embedded device shall provide cryptographic measures for sensitive data storage to protect confidentiality of stored information	S4R_FDF_412	C1, C5	>2	"CryptoManager"	SO4
FSA-DC-3	Cryptographic Mechanisms	The IACS embedded device shall document the cryptographic mechanisms used and any independent validation of the measures so that users can verify if the mechanisms used comply with applicable laws, directives, policies, regulations, standards, and guidance for their target market	S4R_FDF_412	C1	>1	"CryptoManager"	SO4
FSA-RDF-1	Information Flow Enforcement	The IACS embedded device shall provide means to enforce assigned authorizations for controlling the flow of information outside the embedded controller zone and between interconnected systems in accordance with user specific policy	None	C3, C4	ALL	"SecurityMonitoringManager"	SO1, SO2, SO3
FSA-RDF-2	Application Partitioning	The IACS embedded device shall separate data acquisition services, from management functionality	S4R_FDF_425, S4R_FDF_426	C9	>1	"SecurityMonitoringManager"	SO2, SO3
FSA-RDF-3	Security Function Isolation	The IACS embedded device shall isolate security functions from non-security functions by means of partitions, domains, etc., including control of access to and integrity of, the hardware, software, and firmware that perform those security functions	S4R_FDF_425, S4R_FDF_426, S4R_FDF_427	C4	>2	"SecurityMonitoringManager" Function is like an application in the FDF	SO3

Requirement ID	Reference Name	Requirement Description	Previous Requirement	Countermea-sure	624443 - Level	SW component implementation /Clarification	Security Objectives
FSA-RDF-4	Shared System Resources	The IACS embedded device shall prevent unauthorized and unintended information transfer via shared system resources where it supports connection sessions from users with different levels of access	S4R_FDF_409	C1, C3, C4	>2	"SecurityMonitoringManager"	SO1, SO2, SO3
FSA-TRE-1	Incident Response Support	The IACS embedded device may provide features to support configurable automated incident notification services to those not currently connected to the IACS	None	NA	>2	"SecurityMonitoringManager", by means of e-mail, text messages, or any other means	None
FSA-NRA-1	Denial of Service Protection	The IACS embedded device shall protect against or limit the effects of denial of service attacks	S4R_FDF_415	C7	ALL	"SecurityMonitoringManager", by filtering packets that can provoke a DoS	None
FSA-NRA-1.1	Data Flooding Protection	The IACS embedded device shall be capable of taking mitigating actions to attempt to maintain primary function communications while under standard DOS style attacks	S4R_FDF_417	C7	>2	"SecurityMonitoringManager"	SO2, SO3, SO5
FSA-NRA-1.2	Protocol Fuzzing Protection	The IACS embedded device communications shall be tolerant to standard protocol fuzzing attacks for protocols supported by the device	NA	C7	ALL	"SecurityMonitoringManager"	SO2, SO3, SO5
FSA-NRA-1.3	Deterministic Loss of Comm	The IACS embedded device communications shall provide documented or configurable default states for IO and other transmitted variable to be applied upon loss of communications	NA	NA	ALL	"SecurityMonitoringManager"	SO5, SO6
FSA-NRA-1.4	Notification of Attack	The IACS embedded device communications shall be able to notify the higher level system if experiencing heavy communication demands as would experience under DOS attack	NA	NA	>1	"SecurityMonitoringManager", by means of e-mail, text messages, or any other means or communicating to a higher system	None
FSA-NRA-1.5	Preservation of Essential Services	The IACS embedded device communications shall be able to maintain essential services under flooding attack, as defined in robustness testing specification	S4R_FDF_417	C7	ALL	"SecurityMonitoringManager", safety-critical applications shall be protected	SO2, SO3, SO5
FSA-NRA-2	IACS Backup	The IACS embedded device or its support utilities shall provide user functionality to facilitate creation of backups of user-level and system-level information (including system security state information) contained in the IACS	None	NA	ALL	"SecurityMonitoringManager" shall create a backup for recovery	None
FSA-NRA-3	IACS Recovery	The IACS embedded device shall provide user functionality to allow the IACS to be recovered and reconstituted to previously saved IACS Backup after a disruption or failure	None	NA	ALL	"SecurityMonitoringManager" shall be able to be recovered and reconstituted to previously version	Noe