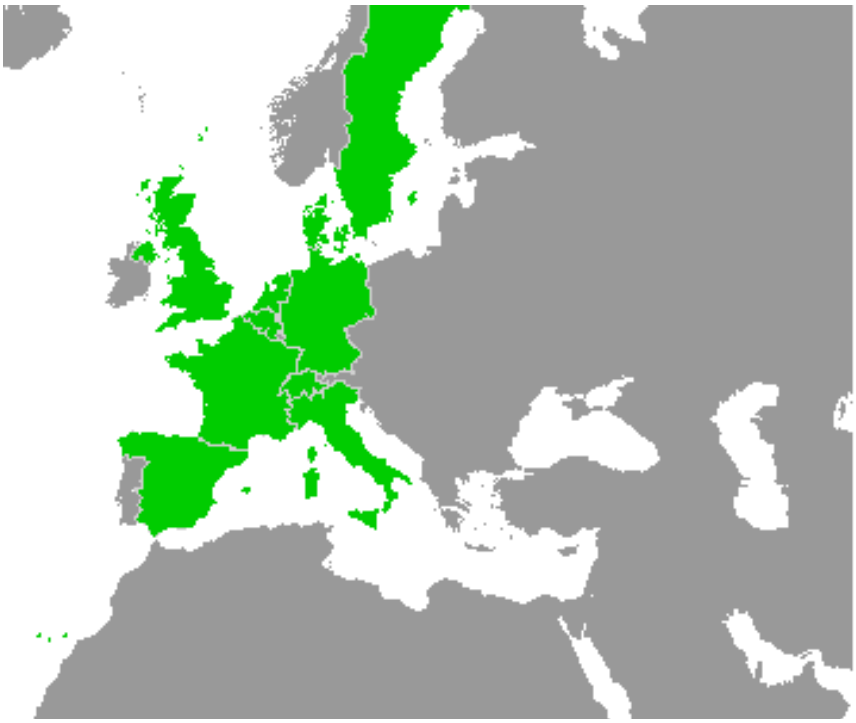
ITEA 2

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

openETCS



WP 5 Demonstrator - Functional Specification

Document characterisation

File name	Reference	Version and date	N° of Pages
WP5_Functional Specification_Demonstrator v03.doc	OETCS/SPEC/DEMONSTRATOR	1.0 of 02/10/2013	34

Approval

Last approved document version: 1.0			
Name	**Function / Company**	**Date**	**Signature**

Circulation

Company	Name/Function	Reason
openETCS		

***A** = Approval, **R** = **R**eview, **Q** = **Q**uality, **C** = a**C**tion, **I** = **I**nformation*

Description of changes

Revision	Date	Author	Comments	A*
0.1	20/06/2013	Patrick DEUTSCH	Creation	
0.2	30/06/2013	Patrick DEUTSCH	First draft	
0.3	30/06/2013	Patrick DEUTSCH	Second draft	
0.4	09/09/2013	Didier WECKMANN	Third draft	
1.0	02/10/2013	Patrick DEUTSCH	First official version	✓

**A(Approved) if this indication is marked, the corresponding version has been approved for circulation*

References

No	Author Company	Reference	Version and date	Title/Comments
/1/	ERA-UNISIG-EEUG	Subset-026	3.3.0 of 07/03/2012	System Requirement Specification
/2/	ERA	ERA_ERTMS_015560	3.3.0 of 01/02/2013	ETCS Driver Machine Interface
/3/	ERA-UNISIG-EEUG	Subset-027	3.0.0 of 02/03/2012	FIS Juridical Recording
/4/	UNISIG-EEUG-UNIFE	Subset-034	3.0.0 of 03/03/2012	FIS Train Interface
/5/	UNISIG	Subset-035	3.0.0 of 29/02/2012	Specific Transmision Module FFFIS
/6/	UNISIG	Subset-036	3.0.0 of 24/02/2012	FFFIS for Eurobalise
/7/	UNISIG	Subset-037	3.0.0 of 02/03/2012	EuroRadio FIS
/8/	UNISIG	Subset-044	2.4.0 of 29/02/2012	FFFIS for Euroloop
/9/	UNISIG	Subset-058	3.0.0 of 02/03/2012	FFFIS STM Application Layer
/10/	ERA	Subset-094	2.2.0 of 13/06/2013	FUNCTIONAL REQUIREMENTS FOR AN ON-BOARD REFERENCE TEST FACILITY
/11/	UNISIG	Subset-076	19/10/2009	Test Sequences

Table of contents

Illustrations

1 INTRODUCTION

The purpose of this document is to provide a functional specification of the demonstrator included in WP5.

The Demonstrator includes two main parts:

- a simulation of the ETCS on-board equipment including:
  - EVC kernel software;
  - peripheral equipment interfacing with the EVC kernel;
- a test environment for the on-board simulation.

This Functional Specification will be the main input for the development of the Demonstrator.

The content of this document is preliminary.

2 INTERACTION WITH OTHER WORKPACKAGES

This WP interacts with some other WPs of the project as follows:

2.1 WP 1

This WP shall provide data coming from Railway projects. Those data shall consist of track and signalling configurations and also train parameters. They shall be used to perform the data preparation phase of the Demonstrator.

2.2 WP 2

This WP shall provide a functional description of the API, which is the interface by which the EVC kernel shall communicate with the on-board peripherals.

2.3 WP 3 AND/OR WP 7

This WP shall provide source code of the EVC kernel in C language produced by the chain of tools. For a seamless integration the code should be fragmented into blocks that can be easily integrated into the non formal existing EVC kernel.

2.4 WP 6

This WP shall receive results to be disseminated.

3 OVERVIEW OF ETCS

3.1 ARCHITECTURE

ETCS includes an on-board and a trackside part. An overview of the ETCS on-board and its interfaces is provided in §2 ofsubset-026 (see /1/).

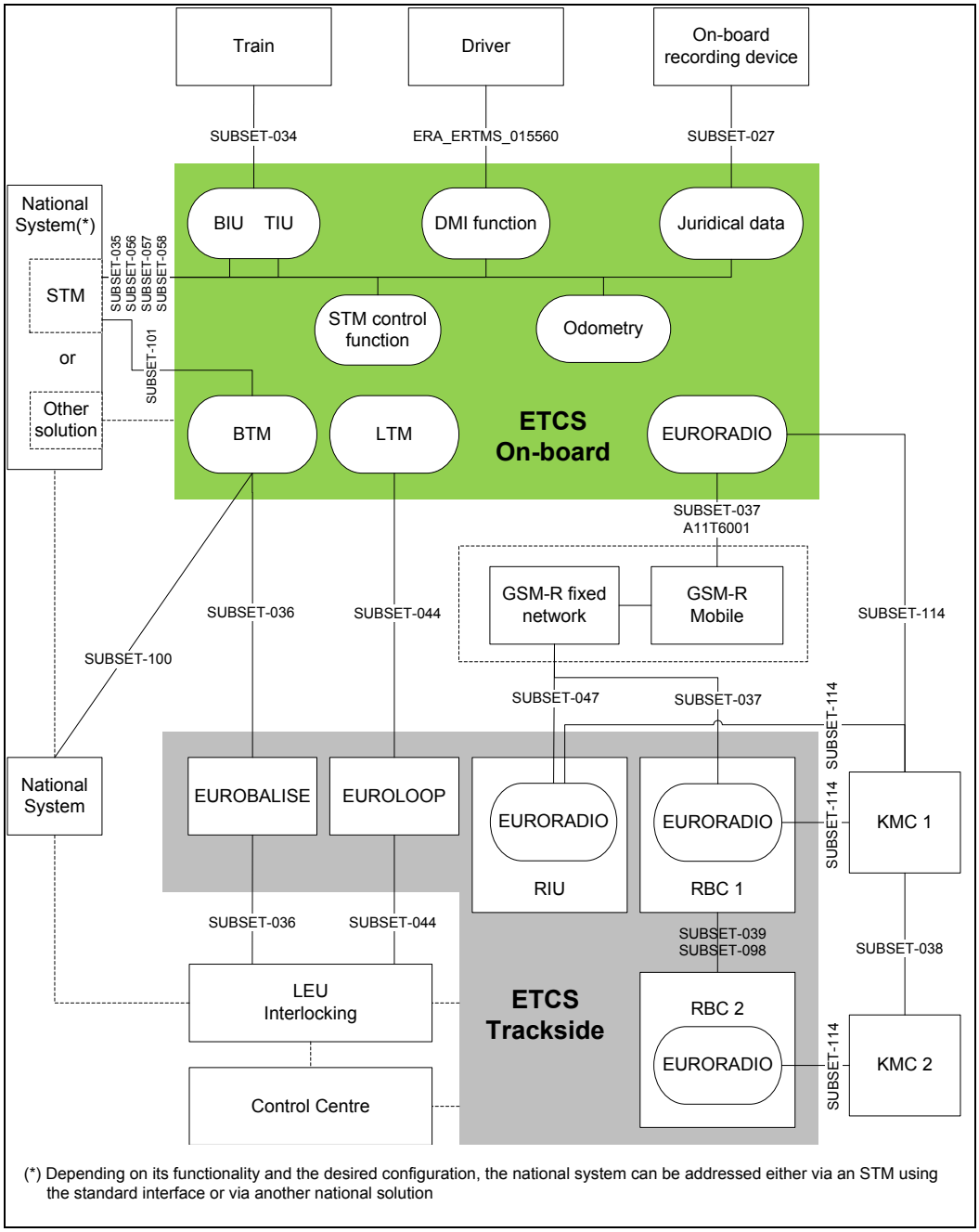


Figure 1: ERTMS/ETCS system and its interfaces (subset-026)

3.2 ETCS ON-BOARD PART

The purpose of the openETCS project is to implement as open source the core of the ETCS on-board, also named EVC kernel. It does not cover the implementation of the following components, which are peripheral resources used by the EVC kernel:

- Balise Transmission Module (BTM)
- Loop Transmission Module (LTM)
- EuroRadio
- Driver Machine Interface (DMI)
- Odometer
- Train interface
- ...

For testing purpose this components may be simulated. However, the interface to communicate to them (API) shall be normalised and published.

3.3 PRINCIPLE OF API

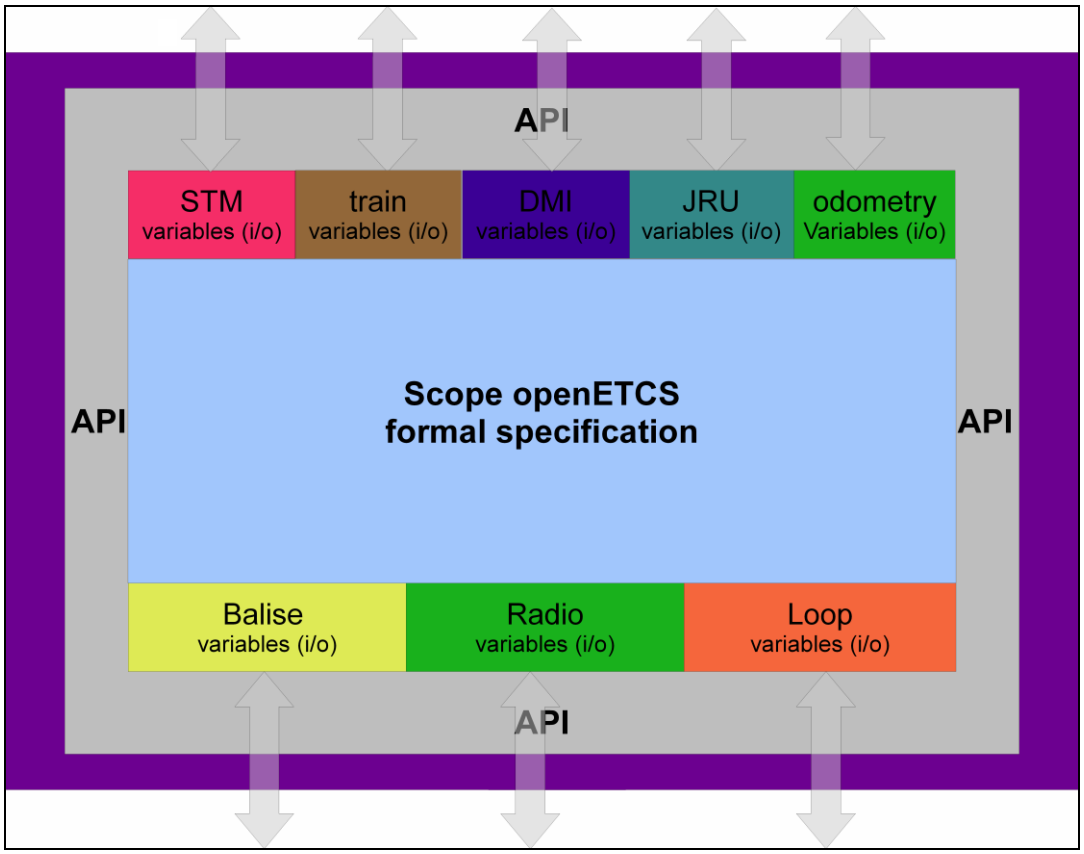


Figure 2: openETCS Application interfaces

4 OBJECTIVES OF THE DEMONSTRATOR

4.1 PROJECT AIMS

The Demonstrator provided by ERSA will be a hosting platform for the results produced by other WPs, especially the EVC code generated by the chain of tools produced in WP 3 and/or WP 7.

The Demonstrator will be validated by using the EVC code for baseline 3 produced by ERSA as the first EVC application hosted in the Demonstrator.

The EVC application developed by ERSA is a full functional implementation of SUBSET 026 (SRS 3.0.0), which helps the sector for the creation, test and consolidation of the Test Sequences which are part of SUBSET 076.

EVC code based on formal methods will be produced in the project. This code shall replace the equivalent ERSA EVC code. Such operation shall be performed in several steps, keeping the same structure.

The Demonstrator shall be used throughout the project and beyond the project as a test platform for the continuous integration of newly generated code.

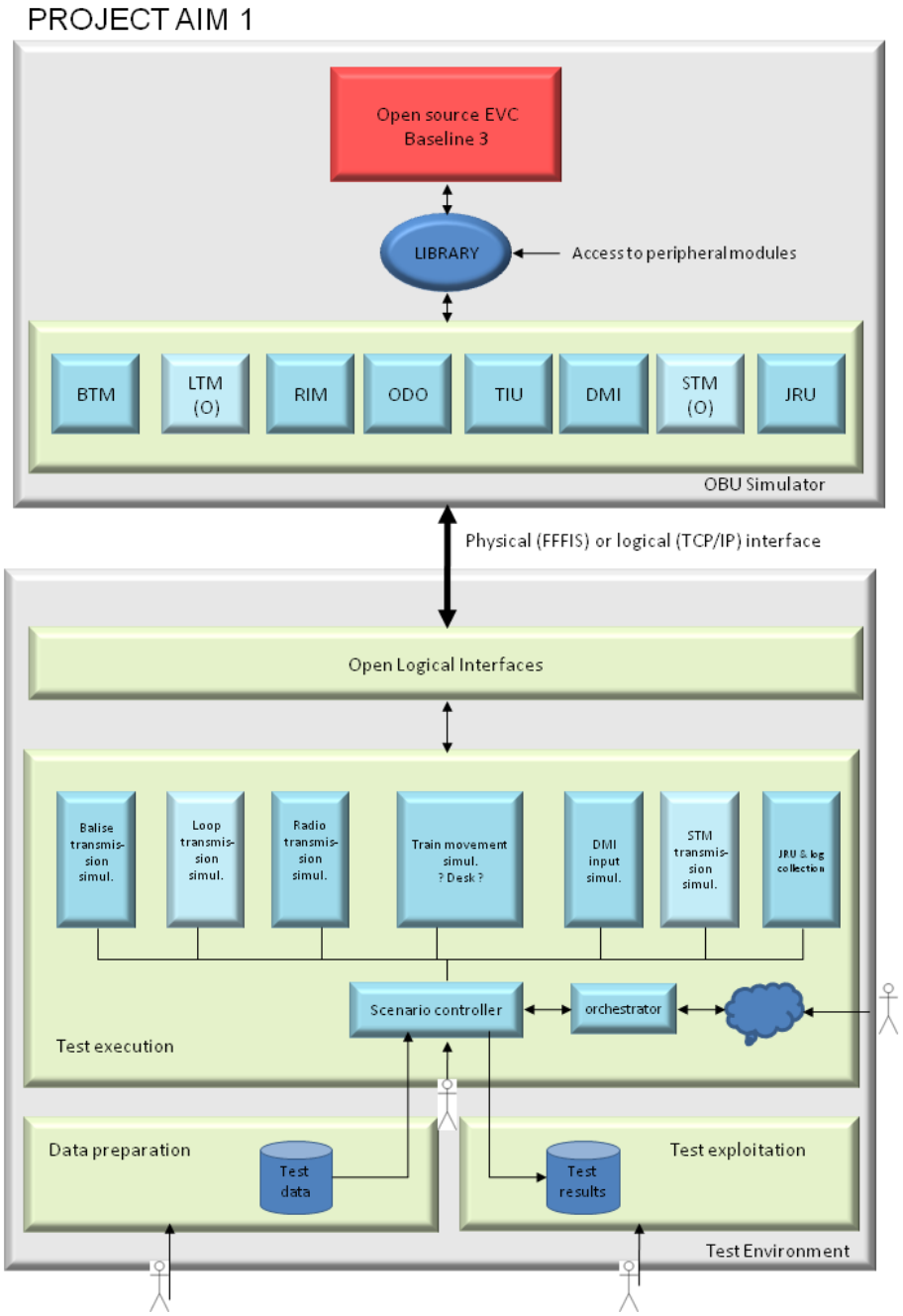


Figure 3: Project aim 1

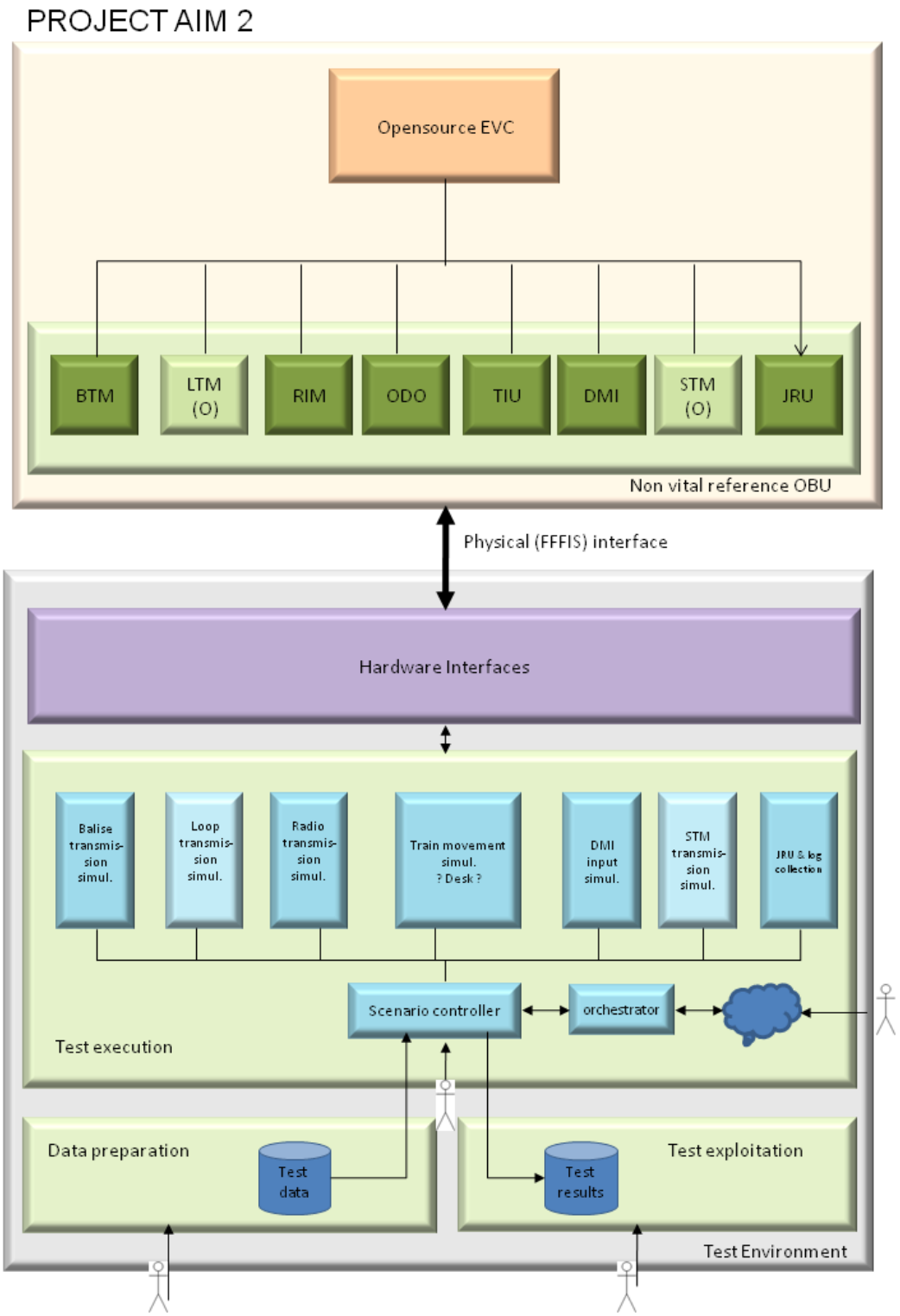


Figure 4: Project aim 2

4.2 IMPLEMENTATION STEPS

4.2.1 Principles

In a first step ERSA will use its own implementation of the EVC kernel baseline 3. At the moment SRS 3.3.0 is implemented. ERSA will use its API to interface with the hardware (PC based) and its resources and to communicate with the peripheral on-board equipments (BTM, RTM, TIU, ODO, DMI, etc...).

ERSA has developed its EVC kernel and API for baseline 3 in C language. The intention is to reuse as much as possible existing work and experience in order to achieve concrete results for the project in accordance with the FPP.

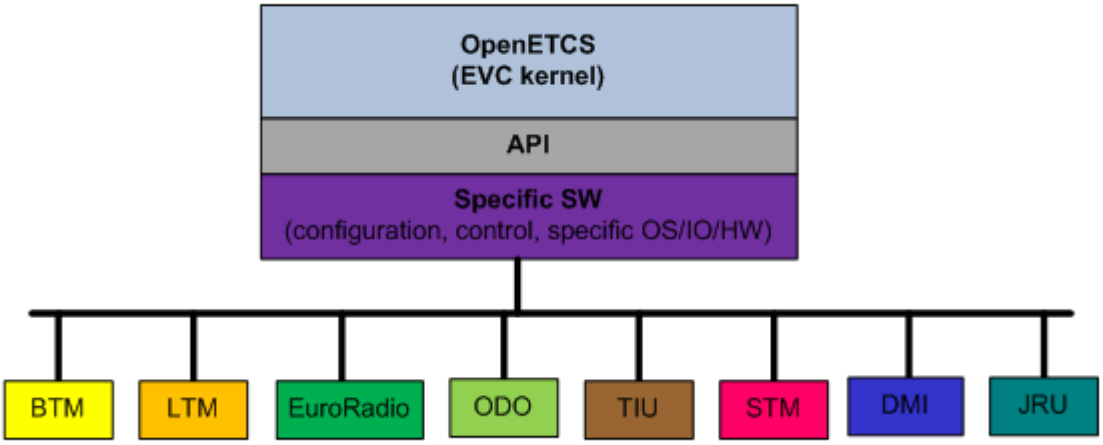


Figure 5: Use of API in the Demonstrator

4.2.2 Implementation step 1

There should be a high level (abstract) definition of the API, independent from any specific software language implementation. This abstract definition should then result in several APIs, for all languages and hardware platforms of interest (ADA, C, ...).

ERSA would then convert the abstract API into a specific one, usable within the Demonstrator.

The simulated On-board components and the test environment shall be adapted to use the new API definition. Using the test environment, new scenarios can then be created and executed to ensure a proper implementation of the API.

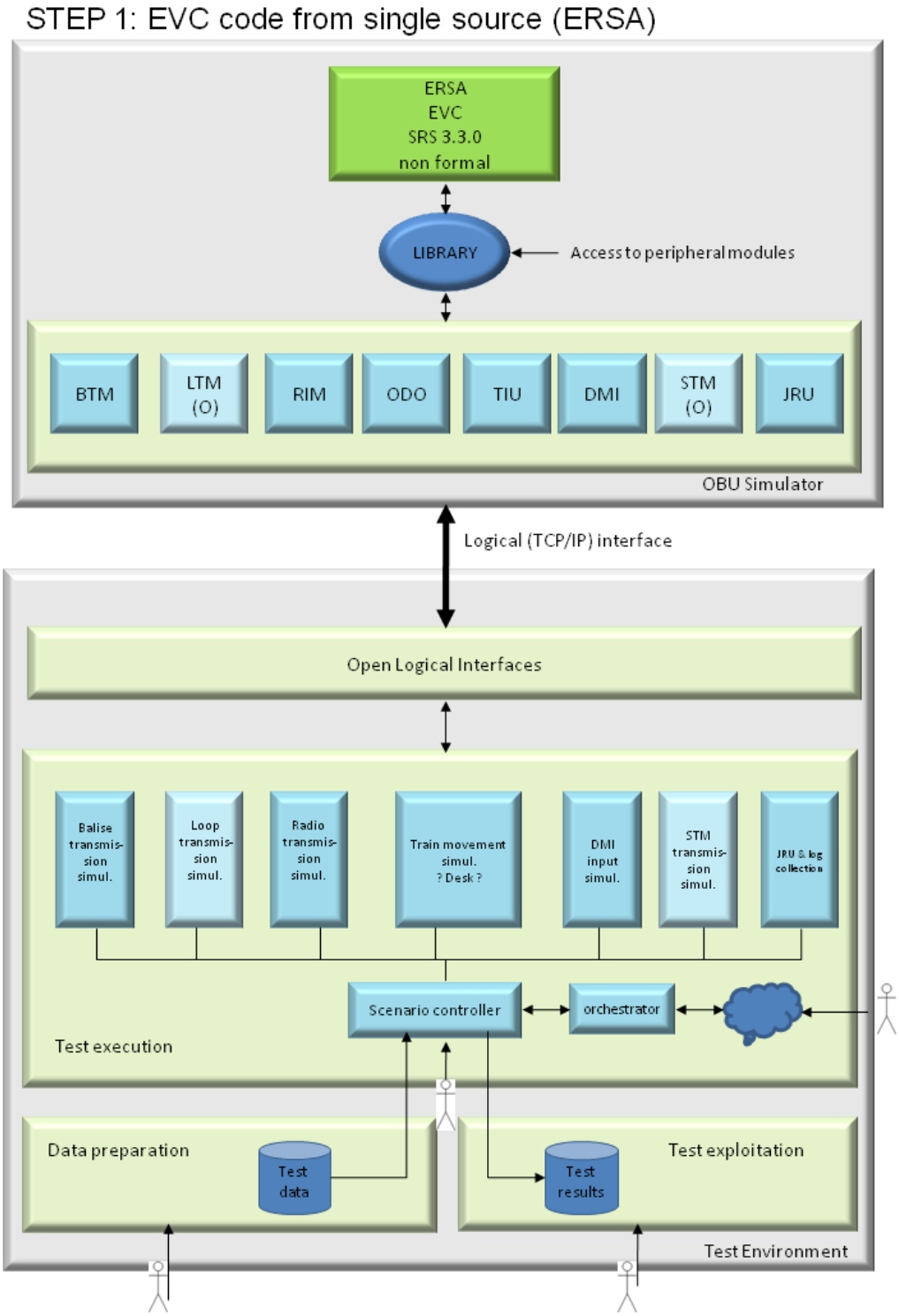


Figure 6: Implementation step 1

4.2.3 Implementation step 2

Once “formal code” blocks become available, they should be integrated progressively within the non-formal EVC kernel code. Modification of existing code may be needed to achieve the process, so each integration should be done with a validation using the test environment.

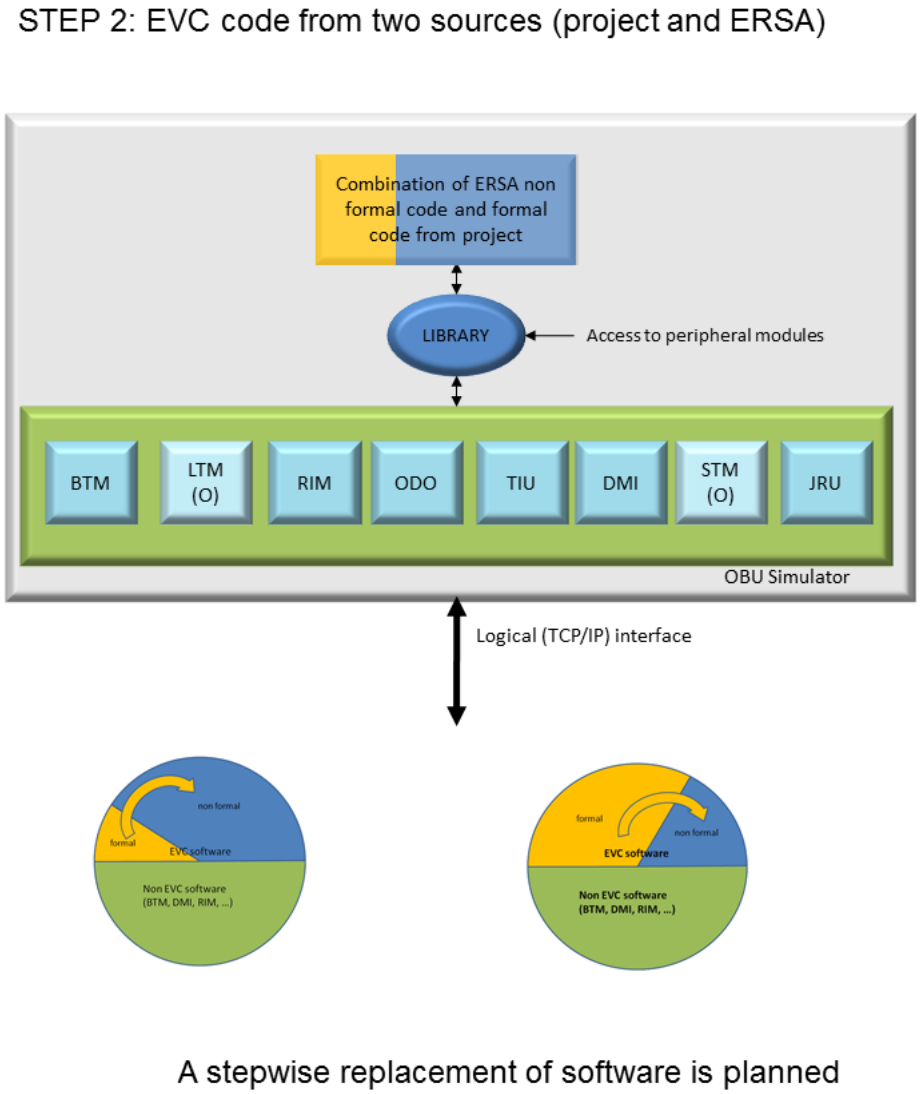


Figure 7: Implementation step 2

4.2.4 Implementation step 3 (outside project)

At the end of the process, it is expected that all EVC kernel code will be generated from the chain of tools. Eventually the openETCS EVC kernel shall replace the ERSA EVC kernel.

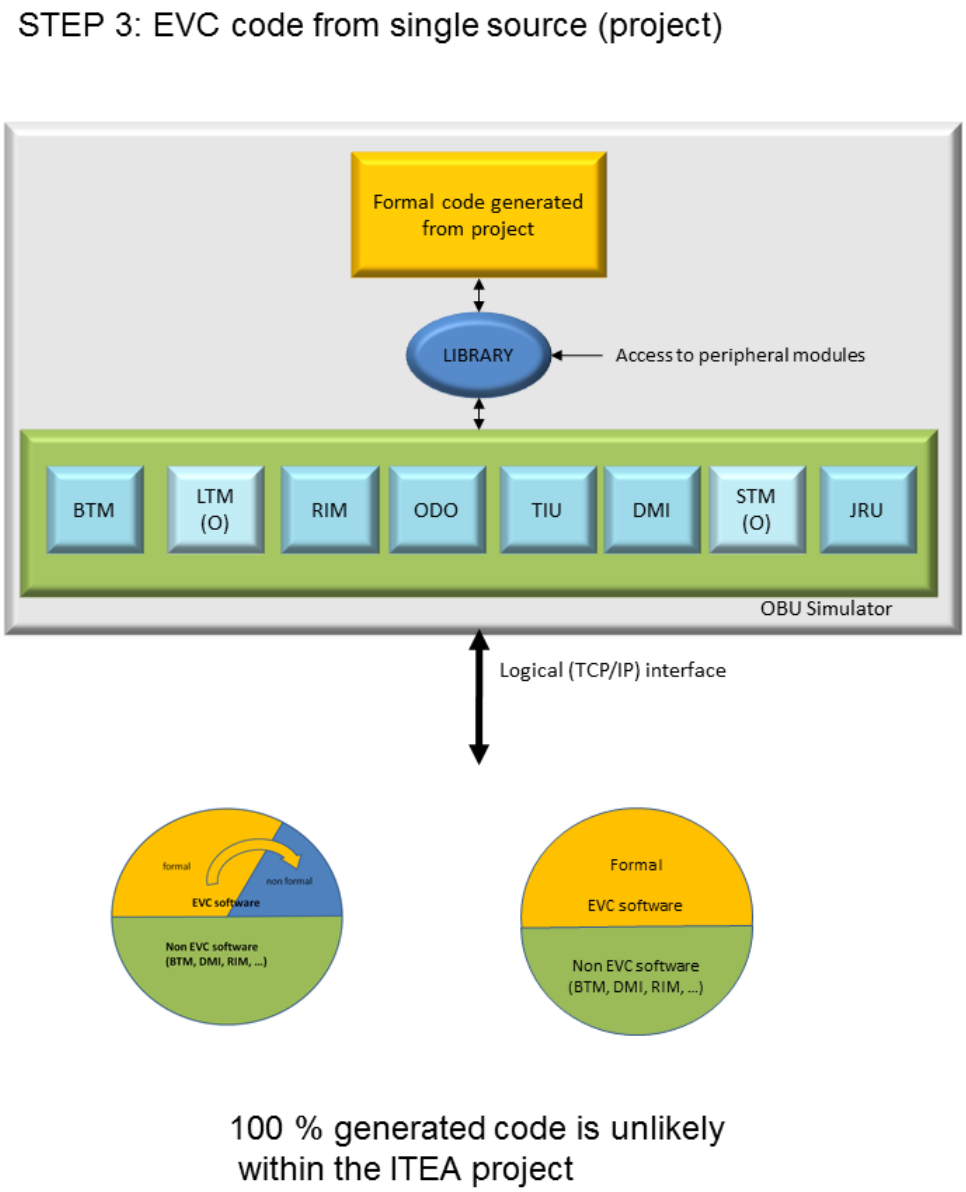


Figure 8: Implementation step 3

4.2.5 Implementation step Implementation step 2A

While integrating code produced from formal methods, a reference non-vital platform shall be provided using industrial hardware. This platform shall host the EVC kernel and its peripherals, the communication between both being performed through the API defined in an earlier step.

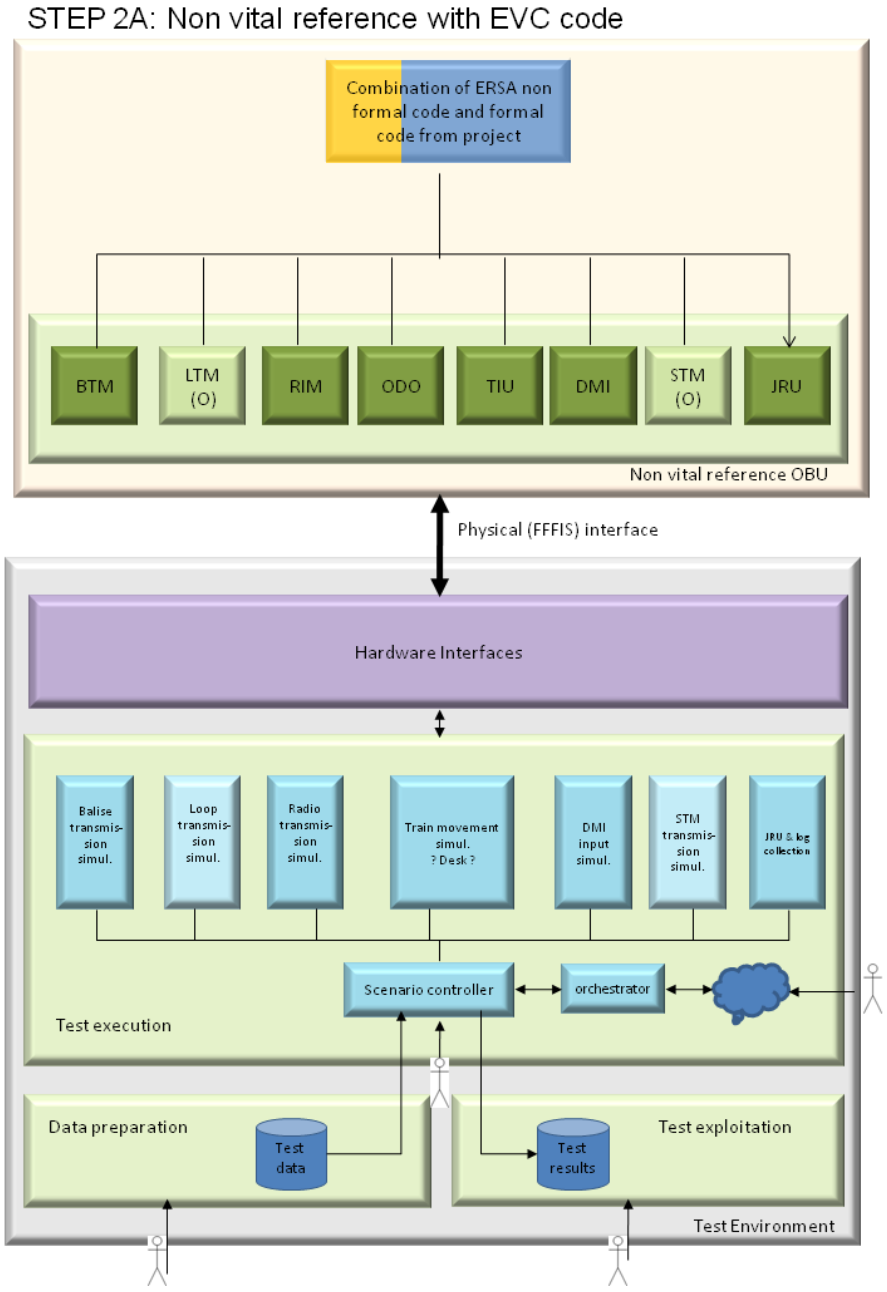


Figure 9: Implementation step 2A

5 ARCHITECTURE OF THE DEMONSTRATOR

The Demonstrator shall include two main parts: the on-board part and the test environment.

5.1 ON BOARD PART

The on-board parts include the EVC kernel and all its peripherals. The EVC kernel shall be clearly separated from the peripherals and shall communicate with them via the API.

The following peripherals will be simulated:

- DMI function;
- BTM;
- LTM (option);
- Euroradio;
- BIU & TIU;
- Odometry;
- STM Control Function (option);
- Juridical Data.

The on-board part will be installed on a PC.

5.2 ENVIRONMENT PART

The environment part includes an off line part and an on line part.

The off line part includes:

- Tools for data preparation; they are used for the creation of test scenarios;
- Tools for data exploitation.

The on line part includes:

- A set of tools for execution of scenarios.

The execution of a scenario consists of the stimulation of the on-board peripherals through standard interfaces.

The environment will be installed on a PC.

5.3 COMMUNICATION BETWEEN ON-BOARD PART AND ENVIRONMENT PART

The communication is a logical interface over ETHERNET TCP/IP.

6 TEST ENVIRONMENT FOR THE ON-BOARD

6.1 OFF LINE TOOLS

6.1.1 Data preparation

A template for test scenarios shall be created. This can be based on a combination of experience and the result of other projects.

At the end the data collected from projects shall be converted into that template. The data shall be stored in a database.

6.1.2 Data exploitation

During execution of test scenarios, the on-board part will generate data to be stored in the JRU. At the same time, the scenario execution tools will also generate log data. Both types of data shall be stored in a database.

The data exploitation shall consist of the analysis of such stored data. This analysis shall include criteria for defining whether tests are passed and shall define a template for test reports.

6.2 TEST EXECUTION TOOLS

A Scenario Controller shall be used to select and execute a test.

The Balise Transmission Simulator shall be used to transmit a balise message to the on board part.

The Loop Transmission Simulator shall be used to transmit a balise message to the on board part (option).

The Radio Transmission Simulator shall be used to exchange radio telegrams with the on board part.

The STM Transmission Simulator shall be used to exchange STM messages with the on board part (option).

The Train Movement Simulator shall be used to send odometry information to the on board part.

The Train Interface Simulator shall be used to exchange TIU & BIU information with the on board part.

The DMI Input Simulator shall be used to exchange DMI information with the on board part.

The JRU Collector shall be used to receive JRU information from the on board part.

7 ABSTRACT INTERFACE DESCRIPTION

The following data should be exchanged between the EVC kernel and its environment through the API.

7.1 CONFIGURATION

Some data are necessary to initialise and configure the EVC kernel:

- ETCS Identity,
- Available train equipment (radio equipment, brake systems, traction systems, STMs, ...)
- Braking data,
- On-board correction factors,
- National values
- Default data
- EVC memory (stored data, ...)

This preliminary list shall be refined.

7.2 ODOMETRY INTERFACE

As the implementation of Odometer functionalities is out of scope of openETCS, only relevant data exchanged with the Odometry functions are described.

7.2.1 Inputs:

- Time and movement data
  - Reference time
  - Movement direction (Standstill, Nominal, Reverse)
  - Estimated position
  - Position accuracy
  - Estimated speed
  - Speed accuracy

  - Estimated acceleration
- Odometer status

7.2.2 Outputs:

- Recalibration data.

This data shall be confirmed.

7.3 Brake / Train interface

The inputs and outputs with the train are described in the subset-034 (see /4/).

7.3.1 Inputs

- ETCS Main Switch status
- Cab status
- Direction controller
- Train Integrity status
- Sleeping Permission
- Non Leading Permission
- Passive Shunting Permission
- Regenerative Brake status
- Magnetic Shoe Brake status
- Eddy Current Brake status
- Electro Pneumatic (EP) status
- Additional brake status
- Train data information
- Type of train data entry
- Traction status
- National System Isolation status
- Brake pressure

7.3.2 Outputs

- Service Brake command
- Emergency Brake command
- Traction Cut-Off command
- Regenerative Brake inhibition
- Magnetic Shoe Brake inhibition
- Eddy Current Brake for SB inhibition
- Eddy Current Brake for EB inhibition
- Change of traction system
- Pantograph command
- Air Tightness command
- Main Power Switch command
- Isolation status
- Station location
- Allowed current consumption

7.4 BALISE INTERFACE

As the implementation of BTM functionalities is out of scope of openETCS, only relevant data exchanged with the BTM are described.

7.4.1 Inputs

- Reception of balise message:
  - Odometer stamp of the centre of the balise and accuracy
  - Time stamp of reception
  - Balise message (according to §7,8 of subset-026)

- BTM status information
  - Failure, alarm

7.4.2 Outputs

None

7.5 LOOP INTERFACE (OPTION)

As the implementation of LTM functionalities is out of scope of openETCS, only relevant data exchanged with the LTM are described.

7.5.1 Inputs

- Reception of loop message
  - Time stamp of reception
  - Loop message (according to §7,8 of subset-026)
- LTM status information
  - Failure, alarm

7.5.2 Outputs

- Configuration of loop antenna (spread spectrum code)

7.6 RADIO INTERFACE

As the implementation of EuroRadio functionalities is out of scope of openETCS, only relevant data exchanged with the EuroRadio subsystem are described.

The management of radio network / safe connection and exchange of radio messages is performed via service primitives as described in subset-037 (see /7/).

7.6.1 Service primitives

- Safe connection set-up
  - Request

  - Indication
  - Response
  - Confirmation
- Data transfer
  - Request
  - Indication
- Connection release
  - Request
  - Indication
- Error reporting
  - Indication
- High priority data
  - Request
  - Indication
- Network registration
  - Request
  - Indication
- Network permission
  - Request
  - Indication

7.7 STM INTERFACE (OPTION)

The data (STM messages) exchanged with the National Systems are described in subset-035 (see /5/) and subset-058 (see /9/).

7.8 JRU INTERFACE

The recording of Juridical data is deeply linked to the ETCS core functionalities and should be part of the openETCS application.

7.8.1 Inputs

None

7.8.2 Outputs

All juridical information described in subset-027 (see /3/) should be transmitted to the Recording Unit:

- General message
- Train data
- Emergency brake command state
- Service brake command state
- Message to radio infill unit
- Telegram from balise
- Message from euroloop
- Message from radio infill unit
- Message from RBC
- Message to RBC
- Driver's actions
- Balise group error
- Radio error
- STM information
- Information from cold movement detector
- Start displaying fixed text message
- Stop displaying fixed text message
- Start displaying plain text message
- Stop displaying plain text message
- Speed and distance monitoring information
- DMI symbol status
- DMI sound status
- DMI system status message
- Additional data
- SR speed/distance entered by the driver
- NTC selected

- Safety critical fault in mode SL, NL or PS
- Virtual balise cover set by the driver
- Virtual balise cover removed by the driver
- Sleeping input
- Passive shunting input
- Non leading input
- Regenerative brake status
- Magnetic shoe brake status
- Eddy current brake status
- Electro pneumatic brake status
- Additional brake status
- Cab status
- Direction controller position
- Traction status
- Type of train data
- National system isolation
- Traction cut off command state
- ETCS on-board proprietary juridical data

7.9 DRIVER INTERFACE

As the implementation of DMI functionalities is out of scope of openETCS, only relevant data exchanged with the DMI will be described.

The description of the data flow should cover the functionalities related to the data displayed and driver actions on DMI as described in ERA_ERTMS_015560 (see /2/).

7.9.1 Input

- Activity status of the DMI
- Driver Request or Acknowledgement (other than text)
- Text Message Acknowledgment
- Train Data Acknowledgment (Validation)

- Version information of the DMI
- Icon Acknowledgment
- Indication of audible information on DMI
- Set Virtual Balise Cover
- Remove Virtual Balise Cover
- Entered radio network
- VBC data (set) acknowledgement
- VBC data (remove) acknowledgement
- Output information related to NTC

7.9.2 Output

- Dynamic Data, like current train speed, target data…
- Request to enable/disable driver menus and buttons
- Request to input certain data (driver id, train data…)
- EVC Coded Train Data to be validated by EVC
- Predefined or Plain Text Message
- Description of track (speed and gradient profile…)
- Request for the DMI version information
- Request to display one or more icon(s) in any area.
- Display the EVC operated system version
- State of DMI display (cabin activation).
- List of available levels.
- List of available radio network
- List of VBCs stored on-board
- Coded VBC data (set) to be validated by driver
- Coded VBC data (remove) to be validated by driver
- Input information related to NTC
- Description of NTC data entry window

7.9.3 Two-way packets

- Default or Entered Driver Identifier
- Default or Entered Train Running Number
- Default or Entered Staff Responsible Data
- Default or Entered Train Data
- Default or Entered Adhesion Factor Data
- Default or entered ETCS Level
- Default or entered RBC contact info (RBC data and radio network ID)

This list shall be refined.

8 TESTS

8.1 INPUT

Track and train data shall be collected from concrete projects.

8.2 TEST CASE CREATION

Templates for test cases and/or test scenarios shall be defined. The input data shall be converted into test scenarios.

8.3 TEST SCENARIO IMPORTATION

Test scenarios shall be imported in the test execution environment.

8.4 TEST EXECUTION

Test scenarios shall be executed, and the results shall be stored. Stored information will include data from the EVC kernel (JRU files) and data from the test tools.

8.5 TEST RESULTS EXPORTATION

The results of the executed tests shall be exported to templates to be defined in the project.

8.6 TEST EXPLOITATION

The results shall be analysed and a report shall be generated, according to a template to be defined in the project.

9 CREATION OF NON VITAL PLATFORM

9.1 INPUT

The NON VITAL platform shall be provided by a project partner. It shall be delivered with all means to enable the implementation of software:

- development tools for C language and support to their use;
- peripheral on-board modules, APIs and drivers.

9.2 SOFTWARE IMPLEMENTATION

The EVC software which shall be installed on this platform cannot be defined at this stage. It is likely that an hybrid version of the EVC kernel based on ERSA code and project code shall be implemented in the platform.

9.3 AVAILABILITY OF PHYSICAL INTERFACES

The platform shall be equipped with physical interfaces, to enable their stimulation via ERTMS test tools owned by ERTMS Test Laboratories and/or ERSA.

It is not planned to create a complete Test Environment from scratch.

10 TESTS WITH NON VITAL PLATFORM

10.1 TEST CASE CREATION

The test data produced in the first part of the project (see chapter 8) shall be reused for performing tests. Only a subset shall be used.

10.2 TEST EXECUTION

Tests shall be performed and results stored in a database. The same principles as defined in chapter 8 shall be reused.

10.3 TEST RESULTS ANALYSIS

The test data shall be analysed and conclusions shall be drawn. The templates defined in chapter 8 shall be reused. Only a subset will be needed.

APPENDIX A: GLOSSARY

Term	Abb.	Description