OETCS/WP3/D??

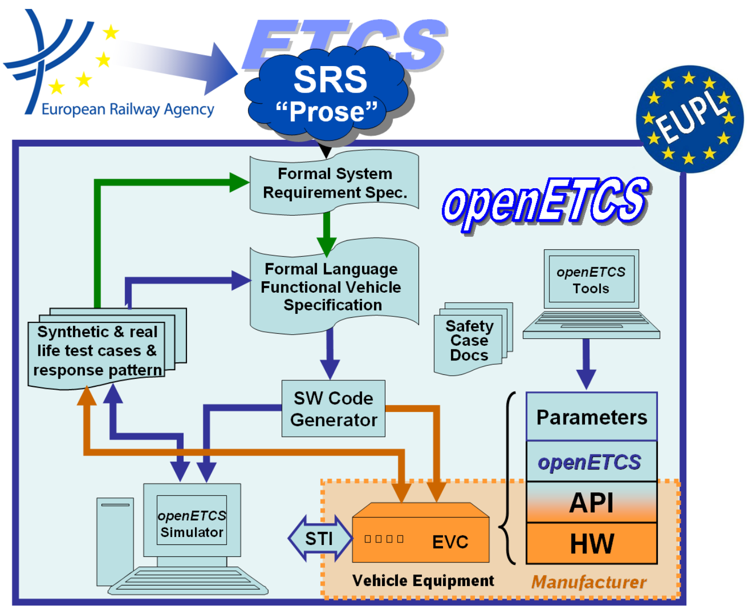
ITEA2 Project
Call 6 11025
2012 – 2015

openETCS@ITEA Work Package 3: “Modelling”

openETCS API description

Generic interface between openETCS application and overall system

Nicolas Boverie, David Mentré and Nicolas Van Landeghem June 2014



Funded by:

openETCS

This page is intentionally left blank

openETCS@ITEA Work Package 3: “Modelling” **OETCS/WP3/D??**
June 2014

openETCS API description

Generic interface between openETCS application and overall system

Document approbation

Lead author:	Technical assessor:	Quality assessor:	Project lead:
location / date	location / date	location / date	location / date
signature [creator name] ([affiliation])	signature [assessor name] ([affiliation])	signature [assessor name] ([affiliation])	signature Klaus-Rüdiger Hase (DB Netz)

Nicolas Boverie
Alstom Transport

David Mentré
Mitsubishi Electric R&D Centre Europe

Nicolas Van Landeghem
ERSA

Draft Report

Prepared for openETCS@ITEA2 Project

Abstract: FIXME

Modification History

Version	Section	Modification / Description	Author

Table of Contents

Figures and Tables

Figures

Tables

1 Introduction and context

1.1 Introduction

This document describes the "openETCS API". The openETCS API is an open, standardized, interface between a vendor specific platform and the openETCS application software. More details on the software architecture are given in chapter 3.

Main objectives

The main objectives of this document are to describe an API:

- Suitable for every partner of openETCS project;
- Compatible with Vendor specific API through an "Adaptation Layer";
- Making explicit all assumptions, including non-functional ones;
- Language agnostic, allowing interfacing with software written in C, Ada or other programming languages;
- Simple;
- Fulfilling safety objectives;
- Offering reasonable performance.

1.2 References

The requirements for this document are defined in (Baro and Welte, 2013, §7.1) and Welvaarts and Jacob (2014) documents.

Source of information and constraints for this document are following documents:

- Alstom API proposal defined in Boverie (2014), Alstom Transport API application layer (2014) and Alstom Transport API Appendix Functional Data Dictionary (2014);
- Comments on Alstom API proposal in Alstom Transport API comments (2014);
- The top level interfaces of openETCS SysML model (2014);
- SCADE openETCS model requirements on run-time defined in §2.2 and §3 of Steinke (2014);
- ERSA Simulator API FIXME: ref?? FIXME
- ETCS on-board interfaces as pointed out in Figure 1 of §2.5.3 in SUBSET-026 (2012).

1.3 Identified issues

This documents aims at reaching a consensus, satisfying relevant point of view of all openETCS project's partners.

As of now, we have identified following issues on which no consensus has been found (yet!):

1. Timing requirements are currently incompatible between SCADE model and Alstom's API definition
   (a) Sub-issue regarding cycle time definition
   (b) Sub-issue regarding ordering of events
2. The data flow definition in Alstom's API proposal is sometimes too far from SRS
3. No agreement on the overall behaviour of Basic software and Application software: when data flows are exchanged, how they are stored, constraints on them, ...
   (a) No agreement on general software architecture
   (b) What is in application Sw and generic software?
4. Define what is in or out of application (odometry, crc, ...)? Allow differentiation points between suppliers
5. No agreement on the way the various components are calling each other: order sequence, constraints, ...
6. How to manage physical units in unambiguous way?
7. The API description should distinguish abstract and concrete parts:
   (a) Abstract parts: Dataflows, Datatypes, Input/Output, ...
   (b) Concrete parts: Given by value or reference, C or Ada language, where is made allocation and (possible) de-allocation, ...

1.4 Abbreviations

Abbreviations not described in this section are described in SUBSET-023 (2012).

API Application Programming Interface. An interface between two parts of a system, describing aspects that the two parts shall be compliant with and keeping other points undetermined

BIU FIXME: def??

JRU Juridical Recording Unit

2 Reference abstract hardware architecture

2.1 Why a reference abstract hardware architecture?

For proper understanding of openETCS API and of constraints imposed on both sides of the API, we need to define a *reference abstract hardware architecture*. This hardware architecture is “abstract” is the sense that the actual vendor specific hardware architecture might be totally different of the abstract architecture described in this chapter. For example, several units might be grouped together on the same processor.

However the actual vendor specific architecture shall fulfill all the requirements and constraints of this reference abstract hardware architecture and shall not request additional constraints.

2.2 Definition of the reference abstract hardware architecture

The reference abstract hardware architecture is shown in figure 1.

The reference abstract hardware architecture is made of a bus on which are connected *units*:

- EVC (European Vital Computer);
- BIU FIXME: def??; FIXME
- TIU (Train Interface Unit);
- Odometry;
- DMI (Driver Machine Interface);
- STM (Specific Transmission Module);
- BTM (Balise Transmission Module);
- LTM (Loop Transmission Module);

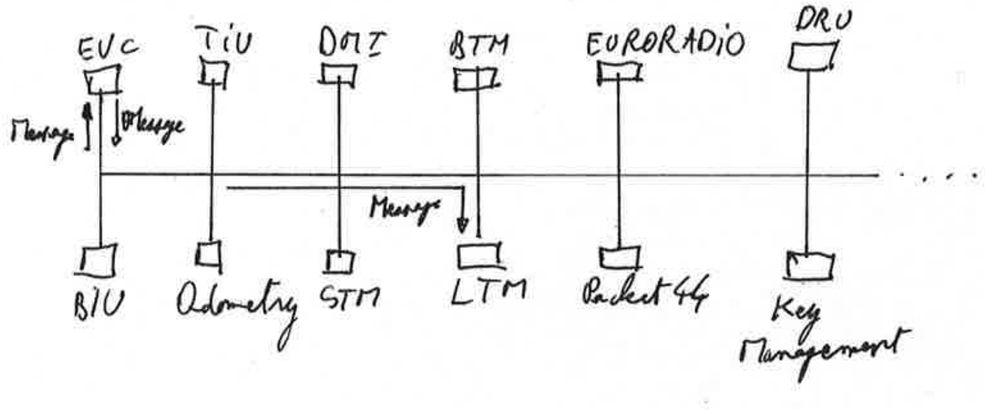


Figure 1. Reference abstract hardware architecture

- EURORADIO;
- JRU (Juridical Recording Unit);
- Packet 44 (Alstom Transport specific module, Boverie (2014));
- DRU (Diagnostic Recording Unit, Alstom Transport specific module, Boverie (2014));
- Key Management (Alstom Transport specific module, Boverie (2014));
- Other units.

A given instance of openETCS might not have all of above units. FIXME: Define a set of mandatory units?

Those units shall working concurrently. They shall exchange information with other units through asynchronous message passing.

3 Reference abstract software architecture

3.1 Overall architecture

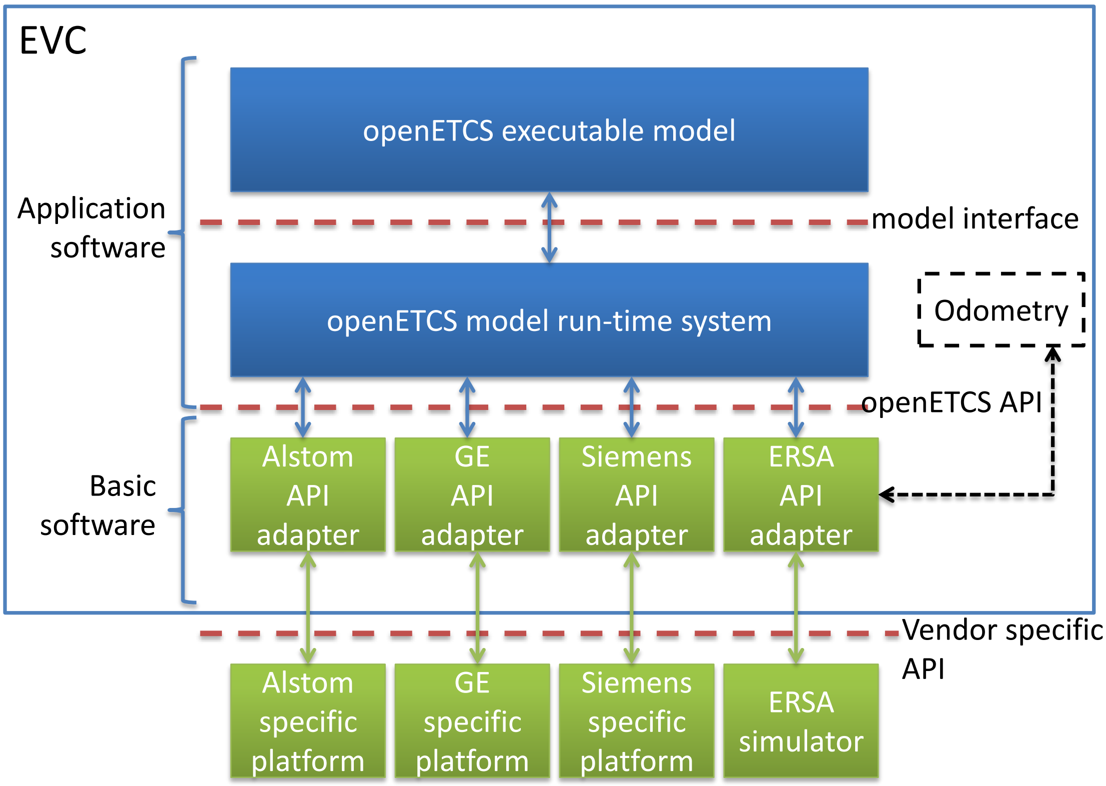


Figure 2. Reference abstract software architecture

The *reference abstract software architecture* is shown in figure 2. This architecture is made of following elements:

- *openETCS executable model* produced by the openETCS SCADE model (2014). It shall contain the program implementing core ETCS functions;
- *openETCS model run-time system* shall help the execution of the openETCS executable model by providing additional functions like encode/decode messages, proper execution of the model through appropriate scheduling, re-order or prioritize messages, etc. This block shall be described in another openETCS document. FIXME: ref? FIXME
- *Vendor specific API adapter* shall make the link between the Vendor specific platform and the openETCS model run-time system. It can buffer message parts, encode/decode messages, route messages to other EVC components, etc.
- All above three elements shall be included in the EVC;
- *Vendor specific platform* shall be all other elements of the system, bus and other units, as shown in figure 1.

We have thus three interfaces:

- *model interface* is the interface between openETCS executable model and openETCS model run-time system. It shall be described in another openETCS document FIXME: ref?;
- *openETCS API* is the interface between openETCS model run-time system and Vendor specific API adapter. It is described in this document;
- *Vendor specific API* is the interface between Vendor specific API adapter and Vendor specific platform. This interface is not publicly described.

The two blocks openETCS executable model and openETCS model run-time system are making the *Application software* part.

The Vendor specific API adapter is making the *Basic software* part.

3.2 Information exchange between blocks

At this level of description, we do not explain how the various blocks of above architecture are calling themselves. We only assume they are exchanging *messages* in an asynchronous way. A message is a set of information corresponding to an event of a particular unit, e.g. a balise received from the BTM. The possible kind of messages are described in chapter 5.

How the exchange of messages in implemented in actual software, e.g. function call, storage of data in a shared buffer, ..., is described in chapter 6.

3.3 Architectural variations

Please note that the reference abstract hardware and software architectures do not forbid architectural variations. For example, the Odometry function could be put within the EVC (see figure 2) instead of a separate hardware unit (as it was shown on figure 1). Such Odometry function would be part of the Application software. But communication between this Odometry function within EVC and the openETCS model run-time system shall be done through the openETCS API and shall follow its conventions.

As another example, part of Vendor specific platform could be on EVC and thus the Vendor specific API would be within the EVC.

4 Real-time and ordering constraints

For proper functioning of the system, the openETCS API specifies real-time and ordering constraints that both the Application software as well as the Basic software and remaining of the system shall ensure. Those constraints are described in this chapter.

Some of constraints defined below are coming from Alstom Transport (Boverie (2014)).

4.1 System-wide constraints

Overall, the constraints imposed on openETCS API shall ensure the ETCS Requirements on performance are fulfilled (see SUBSET-041 (2012) for details and complete reference):

- Delay between receiving of a balise message and applying the emergency brake < 1 s
- Delay between receiving of a balise message and initiating a communication session establishment < 1.5 s
- Delay between receiving of a balise message and reporting the resulting change of status on-board < 1.5 s
- Delay between receiving of a MA via radio (both from RBC and from radio in-fill) and the update of EOA on-board < 1.5 s
- Delay between receiving of a MA from Euroloop and the update of EOA on-board < 1.5 s
- Delay between receiving of a MA from Euroloop and the update of EOA on-board < 3 s
- Delay between receiving of an emergency message and applying the reaction on-board < 1 sec, for brake order; < 1.5 sec, for indication to the driver
- Delay between receiving of a radio message and initiating a communication session establishment < 1 sec
- Delay between passing an EOLM and decoding of the first loop message ≤ 4 s
- Delay between driver action and new window displayed ≤ 2s
- Desk becomes open to “enter Driver ID” is displayed ≤ 3 s
- Desk becomes open to SH mode is displayed ≤ 15 s
- Delay between passing an EOA/LOA and applying the emergency brake < 1 s
- Accuracy of distances measured on-board: for every measured distance s the accuracy shall be better or equal to ± (5m + 5% s)
- Accuracy of speed known on-board: ± 2 km/h for speed lower than 30 km/h, then increasing linearly up to ± 12 km/h at 500 km/h
- The position of the train front indicated in a position report shall be estimated less than 1 sec before the beginning of sending of the corresponding position report
- Safe clock drift: 0.1%

4.2 Cyclic execution of the Basic and Application software

Both Basic and Application software shall be initialized once, in an *initialization phase*, and then shall be executed cyclically in a sequence of *cycles*.

During initialization phase, all units of the system shall be initialized and be ready to proceed at the end of initialization phase.

4.3 Ordering constraints on message exchange

Between units of the system (DMI, EVC, ...), following constraints shall be ensured:

- No message shall be lost;
- Messages sent from one unit towards another one shall be received in emission order;
- Messages sent from two units towards a single one or received by a single units from two other units shall be received in any order.

4.4 Real-time constraints

4.4.1 Event propagation delay

The *external world* is the physical world out of the ETCS system. This external world sends and receives *events* to/from the ETCS system.

When an external world event is seen on a unit of the system (e.g. balise received in BTM), this event is processed and a message might be sent to another unit (e.g. EVC). In the reverse, a unit (e.g. EVC) might send a message to another unit (e.g. DMI) that might produce an event to the external world (e.g. display a message to the driver).

The system shall ensure following constraints:

- The minimum delay from an input event received of the external world until it is received (in a message) by the Application software shall be FIXME: XX ms;
- The maximum delay from an input event of the external world until it is received (in a message) by the Application software shall be FIXME: XX ms;
- The minimum delay from a message sent by the Application software until an event is sent to the external world shall be FIXME: XX ms;
- The maximum delay from a message sent by the Application software until an event is sent to the external world shall be FIXME: XX ms.

4.4.2 Event re-ordering

As a consequence of ordering constrains on message exchange (§4.3) and event propagation delay (§4.4.1), two messages corresponding to two events (e.g. reception of a balise and a radio message) on two different units (e.g. BTM and EURORADIO) send towards a third unit (e.g. EVC) might not be received in real time order, i.e. the message corresponding to the first real time event might arrive in second position on the reception unit.

4.4.3 Event time-stamping

The system is asynchronous: an event is received on a unit and some time is spent before seeing the corresponding message in another unit. In order to do its computations, the Application software needs to know at which real time the event was received. In order to do so, a time-stamp shall be applied on all events requiring such computations. FIXME: Make a list of such events? FIXME

This time-stamp shall fulfill following constraints:

- Time-stamp clock shall have a precision of 1 μs, with a deviation compared to real time of less than 0.1%;
- Time-stamp clock shall be the same on all units of the system.

4.4.4 Execution time constraints

The following real-time constrains shall be ensured:

- The maximum execution time taken by Application software for initialization step is 100 ms;
- The maximum execution time of the Basic and Application software in a cycle shall be FIXME: 300 + XX ms; FIXME
- The maximum execution time of the Application software in a cycle shall be 100 ms.

4.5 Event burst

In some situation, a burst of events can occur. The following dimensioning shall be ensured:

- The maximum number of events sent by a unit shall be at most FIXME: XX events; FIXME
- The maximum number of events received by a unit shall be at most FIXME: XX events. FIXME FIXME

FIXME: Do we need to define the maximum total number of events in fly? FIXME

FIXME: Do we need to define what to do when those maximums are reached?

5 Abstract information flows

6 Concrete interface with openETCS model

References

Alstom Transport API Appendix Functional Data Dictionary: 2014, 'Appendix Functional Data Dictionary'. Alstom Transport, v1.0 edition. https://github.com/openETCS/requirements/blob/master/D2.7-Technical_Appendix/OETCS_API%20Requirements_appendix_functional_data_dictionary_v1.0.pdf.

Alstom Transport API application layer: 2014, 'Appendix application layer v1.0'. Alstom Transport, v1.0 edition. https://github.com/openETCS/requirements/blob/master/D2.7-Technical_Appendix/OETCS_API%20Requirements_appendix_application_layer_v1.0.pdf.

Alstom Transport API comments: 2014, 'Comments on Alstom Transport API proposal'. Alstom Transport. https://github.com/openETCS/requirements/blob/master/D2.7-Technical_Appendix/2014-05-13-Munich-Meeting/OETCS_API_review_2014_05_12.xlsx.

Baro, S. and J. Welte: 2013, 'WP2/D2.6-9 Requirements for openETCS'. 2.0.0 edition. https://github.com/openETCS/requirements/blob/master/D2.6-9/D2_6-9.pdf.

Boverie, N.: 2014, 'API Requirements for OpenETCS'. Alstom Transport, v1.2 edition. https://github.com/openETCS/requirements/blob/master/D2.7-Technical_Appendix/OETCS_API%20Requirements_v1.2.pdf.

openETCS SCADE model: 2014, 'openETCS SCADE model'. openETCS. https://github.com/openETCS/modeling/tree/master/model/Scade/System.

openETCS SysML model: 2014, 'openETCS SysML model'. openETCS. https://github.com/openETCS/modeling/tree/SysML_modeling/model/sysml/WP3-Initial-Architecture.

Steinke, U.: 2014, 'openETCS SCADE Modelling Guide'. https://github.com/openETCS/modeling/blob/master/ModelingRules/SCADE_Modelling_Guide.pdf.

SUBSET-023: 2012, 'Glossary of Terms and Abbreviations, SUBSET-023'. ERA, v3.0.0 edition.

SUBSET-026: 2012, 'System Requirements Specification, SUBSET-026'. ERA, v3.3.0 edition.

SUBSET-041: 2012, 'Performance Requirements for Interoperability, SUBSET-041'. ERA, v3.1.0 edition.

Welvaarts, J. and B. Jacob: 2014, 'Requirements on openETCS API'. 11.05.2014 edition. https://github.com/openETCS/requirements/blob/master/D2.7-Technical_Appendix/2014-05-13-Munich-Meeting/Bullit%20point%20openETCS%20requirements_20140511.pdf.

Index