

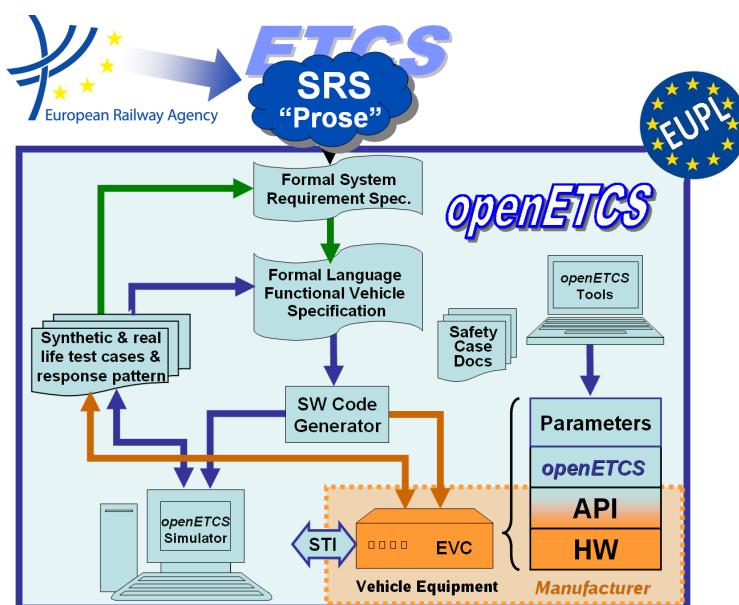
Work-Package 3: "Modeling"

openETCS System Architecture and Design Specification

Third iteration: Scope of openETCS ITEA2 Functions

Baseliyos Jacob, Bernd Hekele, Peyman Farhangi, Stefan Karg, Uwe Steinke, Christian Stahl, David Mentré, David Mentre, Jos Holtzer, Jan Welvaarts, Vincent Nuhaan and Jacob Gärtner

November 2014



Funded by:



Federal Ministry
of Education
and Research



Région de
Bruxelles-
Capitale



MINISTERIO
DE INDUSTRIA, ENERGIA
Y TURISMO

This page is intentionally left blank

Work-Package 3: “Modeling”

OETCS TK-01-01
November 2014

openETCS System Architecture and Design Specification

Third iteration: Scope of openETCS ITEA2 Functions

Document approbation

Lead author:	Technical assessor:	Quality assessor:	Project lead:
location / date	location / date	location / date	location / date
signature Jakob Gärtner (LEA Engineering / DB Netz)	signature [assessor name] ([affiliation])	signature Izaskun de la Torre (SQS)	signature Klaus-Rüdiger Hase (DB Netz)

Baseliyos Jacob, Bernd Hekele, Peyman Farhangi, Stefan Karg

DB Netz AG
Völckerstrasse 5
D-80959 München Freimann, Germany

Uwe Steinke

Siemens AG

Christian Stahl

TWT-GmbH

David Mentré

Mitsubishi Electric R&D Centre Europe

David Mentre

Mitsubishi Electric R&D Centre Europe

Jos Holtzer, Jan Welvaarts, Vincent Nuhaan

NS

Jacob Gärtner

LEA Engineering

Architecture and Functional Specification

Prepared for openETCS@ITEA2 Project

Abstract: This document gives an introduction to the architecture of openETCS. The functional scope is tailored to cover the functionality required for the openETCS demonstration as a target of the ITEA2 project: the Utrecht Amsterdam use-case. It has to be read as an add-on to the models in SysML, Scade and to additional reading referenced from the document.

Disclaimer: This work is licensed under the "openETCS Open License Terms" (oOLT) dual Licensing: European Union Public Licence (EUPL v.1.1+) AND Creative Commons Attribution-ShareAlike 3.0 – (cc by-sa 3.0)

THE WORK IS PROVIDED UNDER openETCS OPEN LICENSE TERMS (oOLT) WHICH IS A DUAL LICENSE AGREEMENT INCLUDING THE TERMS OF THE EUROPEAN UNION PUBLIC LICENSE (VERSION 1.1 OR ANY LATER VERSION) AND THE TERMS OF THE CREATIVE COMMONS PUBLIC LICENSE ("CCPL"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS OLT LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

<http://creativecommons.org/licenses/by-sa/3.0/>
<http://joinup.ec.europa.eu/software/page/eupl/licence-eupl>

Modification History

Version	Section	Modification / Description	Author
0.1	Document	Initial document providing the structure	Baseliyos Jacob
0.2	Document	Workshop Results included and some pretty-printing	Bernd Hekele

Table of Contents

Modification History	iv
Figures and Tables.....	vii
1 Introduction.....	1
1.1 Motivation.....	1
1.2 Objectives	2
1.3 Roles, responsibilities and tasks.....	2
1.4 Process	3
1.5 Assumption and Preconditions	4
1.6 Functions ERTMS/ETCS	5
1.7 openETCS Architecture: History and Iterations	5
1.7.1 First Iteration Functional Scope: The Minimum OBU Kernel Function	5
1.7.2 How to find the functions of the First Iteration in the Architecture	6
Glossary.....	7
1.8 Data dictionary	9
2 Input Documents.....	10
3 Product Backlog	11
4 Architecture description (by layers).....	12
4.1 Introduction to the Architecture.....	12
4.1.1 Abstract Hardware Architecture	12
4.1.2 Definition of the reference abstract hardware architecture.....	12
4.1.3 Reference abstract software architecture	13
4.2 Functional breakdown	14
4.2.1 F1: openETCS application programming interface (API) Runtime System and Input to the EVC).....	14
4.2.1.1 Principles for Interfaces (openETCS API).....	14
4.2.1.2 openETCS Model Runtime System	15
4.2.1.3 Input Interfaces of the openETCS API From other Units of the OBU.....	15
4.2.1.4 Message based interface (BTM, RTM)	16
4.2.1.5 Interfaces to the Time System	19
4.2.1.6 Interfaces to the Odometry System.....	19
4.2.1.7 Interfaces to the Train Interfaces (TIU).....	20
4.2.1.8 Output Interfaces of the openETCS API TO other Units of the OBU	20
4.2.2 F2: Receive messages / check consistency	21
4.2.2.1 Short Description of Functionality	21
4.2.2.2 Internal module structure.....	21
4.2.2.3 Input.....	21
4.2.2.4 Output.....	24
4.2.2.5 Data	26
4.2.2.6 Reference to the SRS (or other requirements)	26
4.2.2.7 Design Constraints and Choices of submodules	27
4.2.2.8 Part 7: ValidateDataDirection	29
4.2.3 F1.5 Select Usable Info - Mode and Level Filter	29

4.2.3.1 Interfaces.....	30
4.2.3.2 SysML Model.....	30
4.2.4 Build coordinate system and calculate train position	39
4.2.4.1 F.2.2 Calculate Train Position.....	39
4.2.4.2 Provide Position Report	41
4.2.5 Store inputs from the TIU	43
4.2.6 Store inputs from the Driver Machine Interface (DMI).....	43
4.2.7 Store data (direct orders, balise group (BG) lists, NV, track data, procedure parameters, confirmations).....	44
4.2.8 Update location based data structures.....	44
4.2.9 Manage specific location based data:.....	44
4.2.10 Build and update MRSP and list of targets at LRBG	45
4.2.11 Profile supervision, i.e. BCM and ceiling speed supervision (active for FS, OS and LS).....	45
4.2.11.1 Movement supervision	45
4.2.11.2 Area Supervision.....	45
4.2.11.3 Update procedure status (including commanding actions towards driver, radio or RBC)	46
4.2.11.4 Mode/Level management	46
4.2.11.5 DMI management.....	46
4.2.11.6 RBC communication	46
4.2.11.7 TIU communication	46
4.2.11.8 JRU and Specific Transmission Module (STM) management: not applicable for Utrecht-Amsterdam.....	47
Appendix A: Restrictions on Interfaces to openETCS OBU.....	48
A.1 Track to Train Interface	48
A.2 TIU Interfaces.....	49
A.2.1 Input to openETCS	49
A.2.2 Output from openETCS.....	50
Appendix: References	51

Figures and Tables

Figures

Figure 1. Reference abstract hardware architecture	12
Figure 2. Reference abstract software architecture	13
Figure 3. openETCS API Highlevel View.....	14
Figure 4. Structure of the Receive message and check consistency module	21
Figure 5. Filter In and out	30
Figure 6. SysML Filter	30
Figure 7. Level Filter	35
Figure 8. Mode Filter	38
Figure 9. Structure of calculateTrainPosition	40
Figure 10. Structure of component ProvidePositionReport	42

Tables

Table 2. Overview over input	22
Table 3. Possible values for the input connectionStatus	23
Table 4. Possible values for the input reset	24
Table 5. Dataflow at output	24
Table 6. Possible values for the output present	25
Table 7. Possible values for the output AcknowledgementRequired	25
Table 8. Possible values for the output radioConnectionStatusFromAPI.....	25
Table 9. Structure of ReceivedMessage_T	26

1 Introduction

1.1 Motivation

The openETCS work package 3 (WP3) aims to provide – amongst others - the software architecture for the openETCS kernel in order to eventually build the software itself. WP3 partner has put great effort in the openETCS software design, thus far without making definite choices on the software architecture itself respective of functional breakdown and data structures of the openETCS kernel. Since the project planning foresees in the production of a reference software to be used as a demonstrator by June 2014, it is of paramount importance that a design freeze of the openETCS kernel architecture be finalized shortly but no later than November 2014.

In compliance with the agreements made during the last WP 3 meeting at the 10.09.2014 in Brussels, DB has taken the initiative to design the aforesaid architecture including of functional breakdown and data structures in order to safeguard a timely delivery of these products. Furthermore, DB has ensured that these developments are focused on including end user requirements so as to develop a design in conformity with the needs and requirements of the operators. Specialists of DB and NS have cooperated together with other partners in WP3 to produce this document.

As referred to above, the architecture description has to be finalized in the month of November 2014. This version of the document is a draft version, demonstrating the general directions and philosophy of the architectural design, the functional breakdown of the software and the data structures. The design is focused on maximum efficiency in order to maximize on RAMS performance of the end product.

This document, named second iteration, is a draft document and will be developed until a complete architecture.

Since this is a work in progress, any remarks referring to the improvement of the document, including reporting errors, are more than welcome. Any additional work done thus far on the subject by other WP3 partners will be incorporated in this document as long as it is aligned with and consistent or complementary to the fundamental viewpoints advocated in this document after a review in respect to the openETCS process. At the same time, any contributions to the integration of which will demand discussion or changes of the fundamentals as proposed in this document, will be discarded with . Only in this way the ITEA2 project is able to meet its objectives as mentioned above. There will be two workshops in which there is due time and opportunities to fine-tune this document and its contents. Any comments will be addressed there.

It is urgent to definitely finalize the architecture on a short notice and therefore this document will rather prescribe than describe the openETCS architecture, functional decomposition of the system and the data structures within the limits as stated above. The document is divided in two parts, i.e.:

- A description of the general architecture of the openETCS OnBoard Kernel (software) including data structures prepared by NS....
- A description of the functional decomposition of the openETCS OBU (software) in alignment with the general architecture prepared by DB.....

Furthermore, this document describes the preconditions on which said descriptions are based on, the status and planning of upcoming activities and the main objectives of DB and NS as the End User. Wherever necessary, reference will be made to documents that underline the agreements that have been made during the openETCS architecture design process and the activities and meetings of WP3.

1.2 Objectives

The prime objective of WP3 is to produce a rapid prototype for the openETCS reference system that can function as a demonstrator in collaboration with WP 4 and WP 5 for the openETCS approach and will be used as such in the final phase of the project. That phase is the first half of 2015. This objective is defined as ...

High level Objectives of this work:

«any further general statements on the ITEA2 objectives, like...»

- Work on a model bases approach and process for effective collaborative work within an international ETCS developer team as stated above, the project needs a definite architecture design by the end of 2014. This document targets:
 - Defining the general design and conditions of the openETCS architecture, functional breakdown and data structures;
 - Providing the guidelines for discussion during the workshops that are planned in October and November 2014 that will result in the final and decisive version of this document;
 - Being the ‘platform’ for finalization i.e. whatever be the products or results of the workshops shall be integrated in this document. Apart from these general objectives, the document means to provide for the materials that will enable WP3 partners to improve the efficiency of the Work Package activities:
 - The comprehensive architecture design shall enable splitting the work load according to the building blocks defined by the architecture and allocate strictly compartmented work parcels or activities to WP3 partners.
 - Doing so will enable WP3 to avoid any double work
 - Compartmenting the work load according to the functional building blocks as defined by the architecture will enable efficient planning of activities, be it individually or the integrated WP3 planning for the coming period, aiming at a just in time delivery of all results and products;
 - Each partner that is responsible for one of the work parcels shall abide by the requirements in terms of quality and timeliness as defined by this document and prior documents and agreements made within the ITEA2 project.

1.3 Roles, responsibilities and tasks

In this section, the roles and responsibilities of the WP3 partners are confirmed, especially where they divert from what has been agreed upon at the start of WP3:

- First of all, in the last WP 3 meeting in Brussels on 10.09.2014 DB proposed to take over the lead of the architecture design and functional breakdown. At the subsequent weekly scrum

meeting on 12.09.2014, it was agreed upon by all participants that DB will take over the lead (see Appendix ...);

- **Planning:** Alstom as WP 3 leader will remain to be responsible for the planning and the allocation of the defined tasks to the different partners
- **Roles:** Alstom will also coordinate the work and safeguard that the defined results will be delivered according to the quality requirements that are agreed within the ITEA2 project and the schedule and the milestones that will be agreed upon during the coming workshops;
- All WP3 partners will deliver the results or products according to planning as will be agreed upon during the said workshops.

In the interest of a swift production of the critical documentation of which this version is a draft, specific tasks will be defined in terms of concrete results to be delivered, the timeframes in which these results must be produced and the partner who shall be responsible for that specific result and the planning. This is to safeguard the timely delivery. The process will be described in the next sections.

1.4 Process

- Alstom as WP 3 leader will be responsible for planning
- Time and quality aspects should be respected
- openETCS tools and methodology must be respected

Most of the operational requirements to WP3 in the last phases of the ITEA2 project have been described in the former paragraphs. This section will describe the process which has to lead to the final result: the reference software to be used in the demonstrator next year, more specifically the final description of the openETCS architecture including the data structures and its functional decomposition. The process will run as follows:

- DB will supervise the development of the first ‘firm’ draft of the specified products, ‘firm’ meaning that changes can only be made within the framework of these products and not to the fundamentals of these products as described in this document;
- DB will supervise the preparation of the two workshops that are proposed by Alstom and aim at defining the final and definite architecture, data structures and functional decomposition. It will make proposals for a planning of the critical tasks that remain to be done;
- Alstom will lead the two workshops following the preparations and the instructions of DB. Since all participants are intrinsically involved in the development work and tend to immerse themselves in technical discussions, for productivity purposes it is proposed to make use of a (non-technical) moderator that will be made responsible for coordinating the meeting, the discussions and the team efforts according to the agenda.
- Also for productivity reasons, introductory presentations will be restricted to the contents and setup of this document since all prior efforts have to be merged with this document and not the other way around. Following a general introduction into the work that has been so far, the other contributions will be scrutinized on their consistency with this document and any useful sections will be merged with this document.
- During the workshops, there will be ample room reserved for enhancing this document, using other documents pertaining to the same field of work that have been delivered by other partners. Only material that is aligned with the general philosophy and structures proposed by this document, will be integrated;

- In case conflicting views emerge over the benefits and value of certain contributions, at the very moment that parties conclude that they have conflicting views, these will be listed in an inventory for later discussion. The moderator shall note any such conflicts on the said inventory. Conflicting views will be treated at the end of each workshop whenever there is sufficient time or will be treated in a separate meeting that will be chaired by DB as coordinator of the ITEA2 project.
- The workshop shall be attended by a secretary provided for by DB who is responsible for making the workshop minutes. Within a week after each workshop these minutes shall be distributed among the partners that have cooperated in the workshop and be reviewed by those.
- The main objective of the workshops shall be the finalization of this document. In order to reach the specified result, the remaining tasks shall be identified and split into separate tasks or work parcels. Every task or work parcel will be allotted to one single responsible partner. Responsibility relates to the timely delivery of the defined result and according to quality requirements;
- Alstom, as WP3 leader, will be responsible for the planning, allocation of tasks or work parcels to partners and will ensure timely delivery of results;
- In case there will be tasks or work packages that cannot be finalized during the workshops or will be identified during the workshops and do not fit in the actual planning, these will be allotted in such a way that deadlines are perfectly clear and acknowledged by the party that is responsible for the results, fit within the general requirements of the project and are agreed upon in writing and executed by the responsible partners according to agreement;
- DB as partner that has integral responsibility for both the ITEA2 project and responsible as well for the architecture etc. , is entitled to interfere take over the role as leader / coordinator in case the workshops prove to be insufficiently productive;
- All output will be such, that it can be integrated in this document. It is the responsibility of DB to integrate the results and to deliver the final and definite version of this document.
- The document concept will follow the openETCS process and tools (LaTex and Git-hub).

1.5 Assumption and Preconditions

- All future contributions shall be fully aligned and compliant the finalized and approved document
- All documents produced by the partners are requested to be compliant and merge to this document; other contributions will be discarded

The workshops are all about working as swift, as efficient and as productive as possible and make full use of the potential made available for these workshops by the partners. It is expected that the partners in the workshops will have the express intention to:

- Contribute to the workshops with the intention to finalize the openETCS architecture;
- Provide resources according to the agreements made prior to the Workshops;
- Focus primarily on getting concrete results regardless of methodological issues that might arise. Where necessary or opportune, classical project management methodology will be applied;
- Provide full transparency with respect to experience, knowledge base and information touching the subjects to be treated in the workshops;
- Document on paper or electronically all output of the workshops and integrate these with the underlying document;
- Restrict discussions only to topics that have an immediate impact on the content or the quality of the end product: the improved version of this document.

1.6 Functions ERTMS/ETCS

The ERTMS / ETCS system was developed with a view to interoperability of trains on the different European rail networks. It is divided into "tracks" - and "board" finishes and shall establish a mutual message operation, by beacons or through a "radio" - The transmission system (in this case a mobile telephone network GSM-R) is performed. It defines several operating levels, and the system must also interfaces with the existing monitoring systems of the trains (using STM) have. The ERTMS / ETCS system provides the transport operator (the track) the choice of conditions concerning the use and operation. The train must therefore may go with different operating conditions on routes. Thus has the onboard equipment but must be implemented, to the interoperability of the train to ensure on the other networks. These functions must therefore correspond to one standard: the system requirement specification (SRS) (version 3.3.0).

application functions, which have two different species of origin: defined in the SRS: here one finds in particular the speed monitoring- and transfer functions; these functions must be implemented in full accordance with the SRS; they can in indeed be on any network on which the train is used; these functions are described below in Section ??;

Moreover, there are functions to adapt to the train: so, for example, the processing a "separation distance" in the airborne equipment trigger: This is dependent on the distribution of functions between the Control monitoring equipment (which the ERTMS / ETCS), and the other CCS Systems.

1.7 openETCS Architecture: History and Iterations

The openETCS Architecture and Design is implemented in iterations [?]. The current step (second iteration) is based on a step to implement the kernel functions of the ETCS system [1]. For a better understanding of the scope the Iteration is described in the following.

1.7.1 First Iteration Functional Scope: The Minimum OBU Kernel Function

The openETCS first iteration architecture and the design of the openETCS OBU software as mainly specified in [2] UNISIG Subset_026 version_3.3.0.

The appropriate functionality has been divided into a list of functions of different complexity (see the WP3 function list [3]).

All these functions are object of the openETCS project and have to be analysed from their requirements and subsequently modelled and implemented. With limited manpower, a reasonable selection and order of these functions is required for the practical work that allows the distribution of the workload, more openETCS participants to join and leads to an executable—limited—kernel function as soon as possible.

While the first version of this document focuses on the first version of the limited kernel function, it is intended to grow in parallel to the growing openETCS software.

The first objective of the first iteration was

- “Make the train run as soon as possible, with a very minimum functionality, and in the form of a rapid prototype.”

This does not contradict the openETCS goal to conform to EN50128.

- After a phase of prototyping, the openETCS software shall be implemented in compliance to EN50128 for SIL4 systems.

1.7.2 How to find the functions of the First Iteration in the Architecture

The functions will be merged with the new architecture. Wherever a function has already been in the scope it will be marked as "first iteration".

Glossary

Notation	Description
application programming interface	an abstraction that is defined by the description of an interface and the behaviour of the interface.
balise group	One or more balises which are treated as having the same reference location on the track.
balise group message	
balise telegram	A telegram contains one header and an identified and coherent set of packets. A message maybe comprised of one or several telegrams.
Balise Transmission Module	On board equipment for intermittent transmission between track and train. It shall be able to receive telegrams from a balise.
Driver Machine Interface	ERTMS train-borne device to enable communication between ETCS and/or GSM-R and the train driver.
European Vital Computer	Computer device for the onboard ETCS.
EURORADIO	The functions required of a radio network coupled with the message protocols that provide an acceptably safe communications channel between track side and train borne equipment's
Juridical Recording Unit	Device to record all actions and exchanges relating to the movement of trains sufficient for off line analysis of all events leading to an incident.
Last Relevant Balise Group	It is the first balise group met and correctly read, when the linking information is not known by the train borne equipment. It is the last linked balise group found at the expected location and correctly read when the linking information is known by the train borne equipment. The LRBG is used as a common reference between the train borne and track side equipments in levels 2 and 3
linking information	Data defining the distance between groups of balises and the action to be taken if a balise group is not detected within given limits.

Notation	Description
location	Location describes a position in terms of topographical relations.
Loop Transmission Module	Train borne equipment that reads the track mounted loop data.
odometry	The process of measuring the train's movement along the track. Used for speed measurement and distance measurement.
on-board unit	on-board equipment for ETCS and the ETCS-related GSM-R.
orientation	
radio message	The Radio Block Centre (RBC) sends electronic messages to, and receives electronic messages from, ETCS onboard equipment on trains within the area which the RBC is controlling. These messages are transmitted via GSM-R data radio
service brake	Train stopping, from a given speed, at such a deceleration that the passengers do not suffer discomfort or alarm or at an equivalent deceleration in the case of non-passenger trains.
Specific Transmission Module	The train borne equipment of the ERTMS / ETCS must be able to be interfaced with the train borne equipment of an existing train supervision system. The Specific Transmission Module shall perform a translation function between these systems and the ERTMS / ETCS.
system requirement specification	Specification describing the technical properties of a piece of equipment based on a corresponding functional requirement specification.
Systems Modeling Language	The Systems Modeling Language (SysML) is general purpose visual modeling language for systems engineering applications. SysML is defined as a dialect of the Unified Modeling Language (UML) standard, and supports the specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems. These systems may include hardware, software, information, processes, personnel, and facilities.
Train Interface Unit	The unit that provides the interface between the train borne equipment and the train. It is likely to be unique to a class of train.

Notation	Description
train position	information related to the position of a train on the railway infrastructure.

Missing Terms:

SR Staff Responsible Mode

SH Shunting Mode

RIU: Radio In-fill Unit

1.8 Data dictionary

concept for the data dictionary ...

2 Input Documents

See Wiki page on

<https://github.com/openETCS/modeling/wiki/Input-Documents-Repository>

3 Product Backlog

See on:

4 Architecture description (by layers)

4.1 Introduction to the Architecture

4.1.1 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” in 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 fulfil all the requirements and constraints of this reference abstract hardware architecture and shall not request additional constraints.

4.1.2 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* defining the on-board unit (OBU):

- European Vital Computer (EVC);
- Train Interface Unit (TIU);
- odometry (ODO);
- DMI;
- STM;
- Balise Transmission Module (BTM);
- Loop Transmission Module (LTM): Not part of this openETCS implementation;
- EURORADIO;

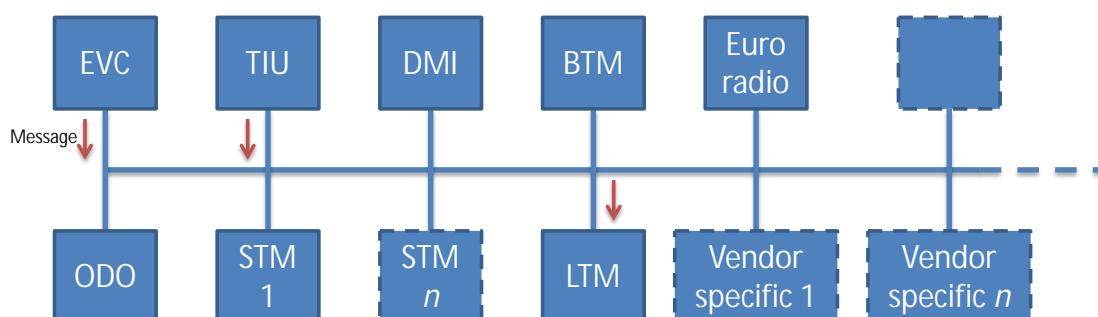


Figure 1. Reference abstract hardware architecture

- Juridical Recording Unit (JRU): Not part of this openETCS implementation;

Elements not being part of this implementation are marked.

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

4.1.3 Reference abstract software architecture

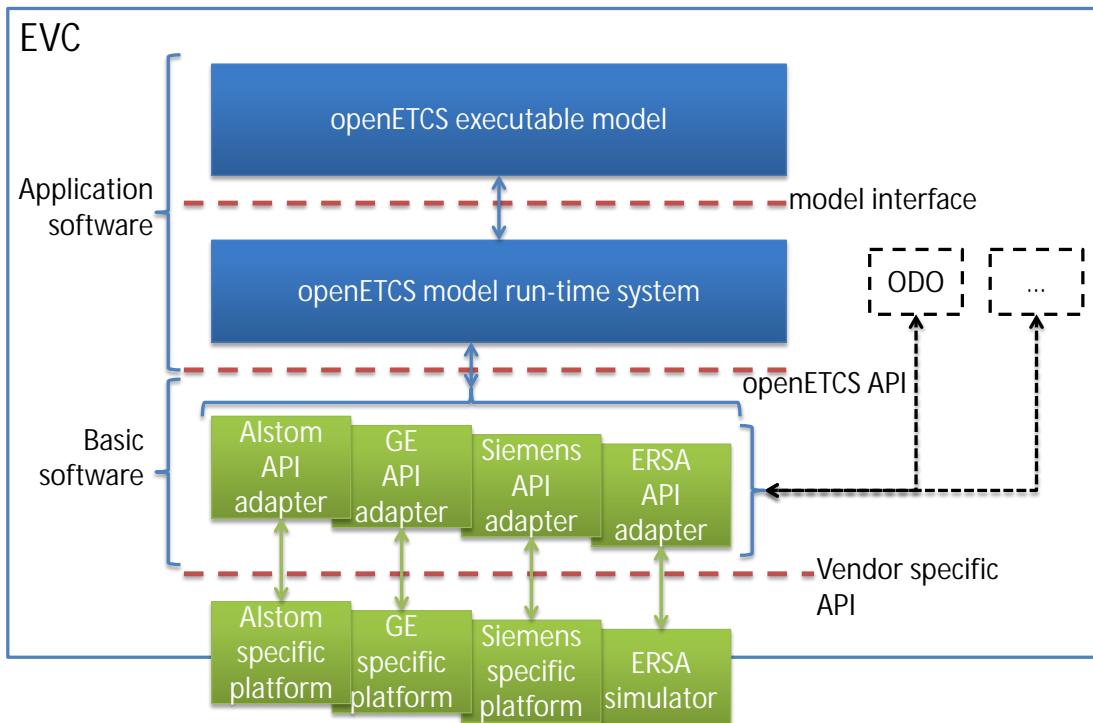


Figure 2. Reference abstract software architecture

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

- *openETCS executable model* produced by the [4] Scade Model. 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.
- *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.
- *openETCS API* is the interface between openETCS model run-time system and Vendor specific API adapter.
- *Vendor specific API* is the interface between Vendor specific API adapter and Vendor specific platform. This interface is not publicly described for all vendors. You can find the Alstom implementation as an example.

The two blocks openETCS executable model and openETCS model run-time system are making the *Application software* part. This Application software might be either openETCS reference software or vendor specific software.

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

4.2 Functional breakdown

4.2.1 F1: openETCS API Runtime System and Input to the EVC)

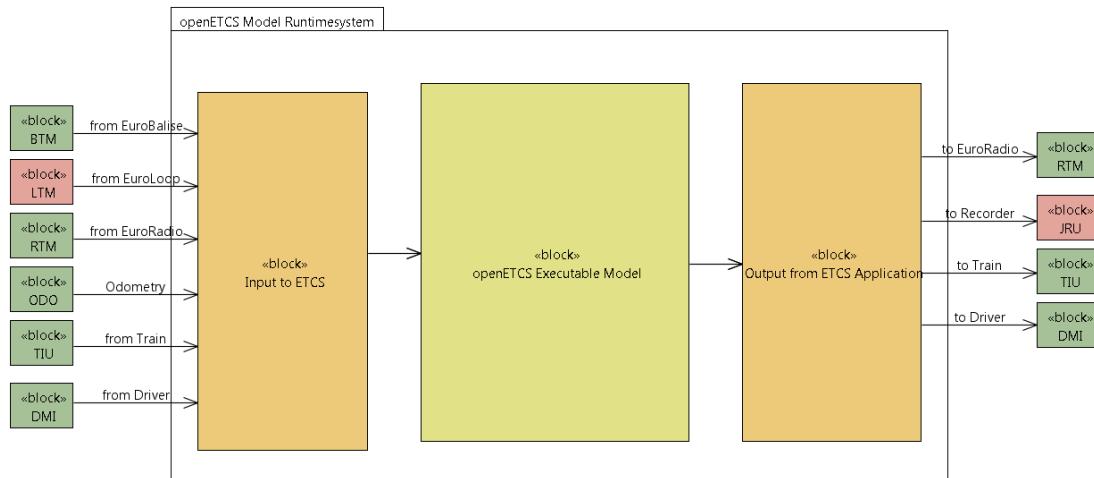


Figure 3. openETCS API Highlevel View

Figure 3 shows the structure of API with respect of the software architecture. Input boxes and output boxes not implemented in this stage are marked as red, other interfaces are marked as green. The System covers functions for processing Inputs from other Units, functions for processing Outputs to other functions and a basic runtime system. Inputs are used to feed the input to the executable model before calling it, outputs are used for collecting information provided by the executable model to be passed to the relevant interfaces after the execution cycle has finished.

4.2.1.1 Principles for Interfaces (openETCS API)

Information is exchanged *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 ??.

The information is passed to the executable model as parameters to the synchronous call of a procedure (Interface to the executable model). Since the availability of input messages to the application is not guaranteed the parts of the interfaces are defined with a "present" flag. In

addition, fields of input arrays quite often is of variable size. Implementation in the concrete interface in this use-case is the use of a "size" parameter and a "valid"-flag.

4.2.1.2 openETCS Model Runtime System

The openETCS model runtime system also provides:

- Input Functions From other Units
In this entity messages from other connected units are received.
- Output Functions to other Units
The entity writes messages to other connected units.
- Conversation Functions for Messages (Bitwalker)
The conversion function are triggered by Input and Output Functions. The main task is to convert input messages from an bit-packed format into logical ETCS messages (the ETCS language) and Output messages from Logical into a bit-packed format. The logical format of the messages is defined for all used types in the openETCS data dictionary.
Variable size elements in the Messages are converted to fixed length arrays with an used elements indicator.
Optional elements are indicated with an valid flag. The conversion routines are responsible for checking the data received is valid. If faults are detected the information is passed to the openETCS executable model for further reaction.
- Model Cycle

The version management function is part of the message handling. This implies, conversions from other physical or logical layouts of messages are mapped onto a generic format used in the EVC. Information about the origin version of the message is part of the messages.

The executable model is called in cycles. In the cycle

- First the received input messages are decoded
- The input data is passed to the executable model in a predefined order. (**Details for the interface to be defined**).
- Output is encoded according to the SRS and passed to the buffers to the units.

4.2.1.3 Input Interfaces of the openETCS API From other Units of the OBU

Interfaces are defined in the Scade project APITypes (package API_Msg_Pkg.xscade).

In the interfaces the following principles for indicating the quality of the information is used:

Indicator	Type	Purpose
present	bool	True indicates the component has been changed compared to the previous call of the routine
valid	bool	True indicates the component is valid to be used.
validated	bool	True indicates the component has been validated.

In the next table we can see the interfaces being used in the openETCS system. Details on the interfaces are defined further down.

Unit	Name	Processing Function	Description
BTM	Balise Telegram	Receive Messages	
DMI			
EURORADIO	Communication Management	Communication Management	
EURORADIO	Radio Messages	Receive Messages	
ODO	Odometer	All Parts	
TIME	Time system of the OBU	All Parts	
Startup			
TIU	Train Data	All Parts	

Information in the following sections gives a more detailed overview of the structure of the interfaces.

4.2.1.4 Message based interface (BTM, RTM)

Balise Message (Track to Train)

Message Name	Optional Packets	Restrictions in the current scope
Balise Telegram	3: National Values 41: Level Transition Order 42: Session Management 45: Radio Network registration 46: Conditional Level Transition Order 65: Temporary Speed Restriction 72: Packet for sending plain text messages 137: Stop if in Staff Responsible 255: End of Information	Used in Scenario

Balise Telegram	0, 2, 3, 5, 6, 12, 16, 21, 27, 39, 40, 41, 42, 44, 45, 46, 49, 51, 52, 65, 66, 67, 68, 69, 70, 71, 72, 76, 79, 80, 88, 90, 131, 132, 133, 134, 135, 136, 137, 138, 139, 141, 145, 180, 181, 254	Not Used in Scenario
-----------------	---	----------------------

Radio Messages (Track to Train)

The following table gives a list of relevant messages and packets in the messages. The information is taken from track descriptions and traces.

Message Name	Optional Packets	Restrictions in the current scope
3: Movement Authority	21: Gradient Profile 27: International Static Speed Profile 49: List of balises for SH Area 80: Mode profile plus common optional packets, 3: National Values 5: Linking 15: Level 2/3 Movement Authority 21: Gradient Profile 41: Level Transition Order 65: Temporary Speed Restriction	The list of packets is adjusted to our use-case.
8: Acknowledgement of Train Data		No Extra Packets
9: Request To Shorten MA	49: List of balises for SH Area 80: Mode profile	Not part of the use-case
15: Conditional Emergency Stop		No Extra Packets
16: Unconditional Emergency Stop		No Extra Packets
18: Revocation of Emergency Stop		No Extra Packets
24: General Message	From RBC: 21: Gradient Profile 27: International Static Speed Profile plus common optional packets, e.g., for our Use-Case 3: National Values 41: Level Transition Order 42: Session Management 57: Movement Authority Request Parameters 58: Position Report Parameters 72: Packet for sending plain text messages	Restricted to use-case
32: RBC/RIU System Version		No packets
33: MA with Shifted Location Reference	21: Gradient Profile 27: International Static Speed Profile 49: List of balises for SH Area 80: Mode profile plus common optional packets	

39: Acknowledgement of termination of a communication session		No Extra Packets
41: Train Accepted		No Extra Packets
Summary	List of Packets used in the scenarios 3: National Values 5: Linking 15: Level 2/3 Movement Authority 21: Gradient Profile 27: International Static Speed Profile 42: Session Management 57: Movement Authority Request Parameters 58: Position Report Parameters 72: Packet for sending plain text messages 41: Level Transition Order 49: List of balises for SH Area 65: Temporary Speed Restriction 80: Mode profile	

4.2.1.5 Interfaces to the Time System

The interface types are defined in the OBU_Basic_Types_Pkg Package. The system time is defined in the basic software.

The system TIME is provided to the executable model at the begin of the cycle. It is not refreshed during the cycle. The time provided to the application is equal to 0 at power-up of the EVC (it is not a “UTC time” nor a “Local Time”), then must increase at each cycle (unit = 1 msec), until it reaches its maximum value (i.e current EVC limitation = 24 hours)

- TIME (T_internal_Type, 32-bit INT)
Standardized system time type used for all internal time calculations: in ms. The time is defined as a cyclic counter: When the maximum is exceeded the time starts from 0 again.

4.2.1.6 Interfaces to the Odometry System

The interface types are defined in the OBU_Basic_Types_Pkg Package. The odometer gives the current information of the positing system of the train. In this section the structure of the interfaces are only highlighted. Details, including the internal definitions for distances, locations speed and time are implemented in the package.

- Odometer (odometry_T)
 - valid (bool)
valid flag, i.e., the information is provided by the ODO system and can be used.
 - timestamp (T_internal_Type)
of the system when the odometer information was collected. Please, see also general remarks on the time system.

- Coordinate (odometryLocation_T)
 - * nominal (L_internal_Type) [cm]
 - * min (L_internal_Type) [cm]
 - * max (L_internal_Type) [cm]

The type used for length values is a 32 bit integer. Min and max value give the interval where the train is to be expected. The boundaries are determined by the inaccuracy of the positioning system. All values are set to 0 when the train starts.

- speed (V_internal_Type) [km/h] General Speed of the train
- acceleration (A_internal_Type)[0.01 m/s²], Standardized acceleration type for all internal calculations : in
- motionState (Enumeration) indicates whether the train is in motion or in no motion
- motionDirection (Enumeration) indicates the direction of the train, i.e., CAB-A first, CAB-B first or unknown.

4.2.1.7 Interfaces to the Train Interfaces (TIU)

The following information is based on the implementation of the Alstom API. The interface is organised in packets. The packets of the Alstom implementation are listed in the appendix to this document.

The description of interfaces needed for the current scope will be added according to the use.

4.2.1.8 Output Interfaces of the openETCS API TO other Units of the OBU

From Function	Name	To Unit	Description
	Radio Output Message	EURORADIO	
	Communication Management	EURORADIO	
	Driver Information	DMI	
	Train Data	TIU	

Radio Messages (Train to Track)

The following table gives a list of relevant messages and packets in the messages. The information is taken from scenarios and traces.

Message Name	Optional Packets	Comment
129: Validated Train Data	0: Position Report 11: Validated train data	
132: MA Request	0: Position Report	
136: Train Position Report	0: Position Report	
146: Acknowledgement		
147: Acknowledgement of Emergency Stop	0: Position Report	

150: End of Mission	0: Position Report	
155: Initiation of a communication session		
156: Termination of a communication session		
159: Session established		

Packets: to be completed

4.2.2 F2: Receive messages / check consistency

4.2.2.1 Short Description of Functionality

The block “Receive messages / check consistency” is responsible for receiving Eurobalise-telegrams and Euroradio-messages from the API and perform several consistency checks on the input.

The block collects the telegrams of balises in order to build balise group messages. Euroradio messages are always delivered as a whole message.

On each message, a consistency check is performed, before the data is validated according to the driving direction of the train. In general, messages not designated for the current driving direction of the train are not forwarded to the further processing.

After applying consistency checks, the data direction is validated.

Information of the odometer is used to control for the train leaving the expectation window of the balises.

4.2.2.2 Internal module structure

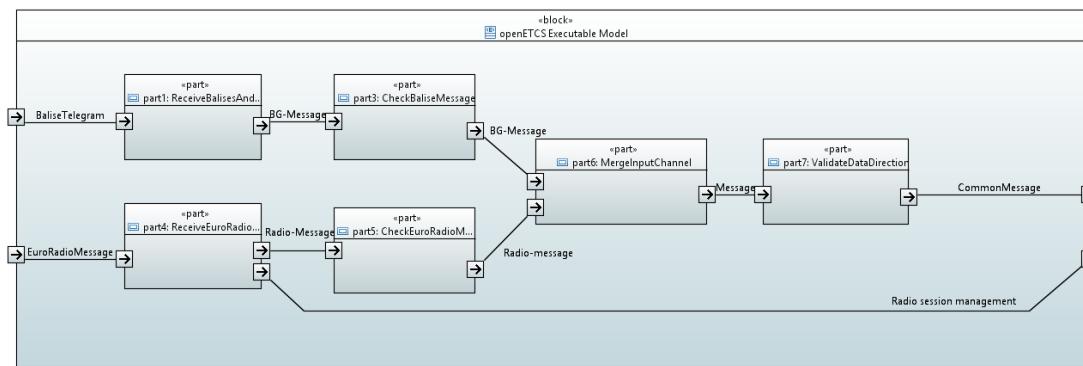


Figure 4. Structure of the Receive message and check consistency module

4.2.2.3 Input

For providing the output, the module needs different input data flows. An overview is provided in table 2

Index	Input name	Input type	Source
0	apiRtmMessage	< will be defined by API >	API
1	apiBaliseTelegram	API_Telegram_T	API
2	apiRadioDevice	int	API
3	connectionStatus	sessionStatus_Type	MoRC
5	lastRelevantEventTimestamp	T_internal_Type	MoRC or Database (?)
6	tNvContact	T_internal_Type	Database
7	currentOdometry	odometry_T	Odometry
8	lrbg	positionedBG_T	Database
9	reset	bool	internal

Table 2. Overview over input

Input 0: apiRtmMessage

The Euroradio-/Eurobalise-message is originated from the openETCS-API. The API is described in the section 4.2.1.

In the current implementation, only messages with normal priority are used in the system. Emergency messages will not be processed.

The input not only transfers the radio message but also information, if a message is present and if this message was decoded correctly and passed the lowlevel checks performed by the API.

The radio message itself consists of a header and a payload-part. The header part contains all variables of the message. The payload-part consists of all packets in the message.

Input 1: apiBaliseTelegram

The telegram is build from

- a present flag (bool)
Indicates the input decoded telegram parameter is “present”, i.e., the input has been updated by the API. Only if the telegram is present the position information (incenterOfBalise) is to be used.
- the decoded telegram including optional packets received from the balise.
- the centerOfBalisePosition parameter. This parameter is used to give the position where the BTM has recognised the center of the balise telegram.

Input 2: apiRadioDevice

The RTM-module can consist of multiple radio devices. When a handover between two RBCs is performed, messages can be received from both radio devices. The API provides information about the device, which received the message.

The values transmitted have to be defined by the API.

Input 3: connectionStatus

The input **connectionStatus** will give information about the radio connection. This input is delivered by the session management module, not from the API. The information is needed to perform a timestamp check, which is depending on the connection state.

Value	Interpretation
DISCONNECTED	The OBU is currently not connected to a RBC.
CONNECTING	The OBU is currently connecting to the RBC. Received messages belong to the process of establishing a connection.
CONNECTION_ESTABLISHED	The connection to RBC is established.

Table 3. Possible values for the input **connectionStatus**

Input 5: lastRelevantEventTimestamp

For monitoring the safe radio connection, it's necessary, that the time between two packets is less than the value of T_NVCONTACT.

In situations like level-changes or announced radioholes, not the timestamp of the last message is relevant for comparison, but the timestamp of the last relevant event. This can be e.g. the timestamp of the level change or the timestamp of the moment, when the train was passing the end of the radiohole.

For performing this check, the timestamp of the last relevant event is provided to the model as an T_internal_Type-type.

Input 6: tNvContact

For monitoring the safe radio connection, the national value T_NVCONTACT is needed as an input.

Input 7: currentOdometry

Current information giving the odometry of the train.

Input 8: lrbg

The Last Relevant Balise Group. The information has been collected before by the train position function.

Input 9: reset

To delete all data stored in the module (e.g. collected balise-telegrams, which do not yet form a complete message), a reset input can be used. If the input is set to **true**, all data kept in the module is deleted and no input is accepted.

Value	Interpretation
true	All data kept in the module is deleted and no input is accepted.
false	No action. Data at input is accepted.

Table 4. Possible values for the input reset

4.2.2.4 Output

The output of the module provides the received and processed Euroradio and Eurobalise messages. The module combines messages both from Eurobalises and from Euroradio to one common dataflow.

Additionally, status information is provided. The status information consists of the following data:

- Information, if the message has to be rejected in case of a consistency error, including further information about the error.
- Information, if an acknowledgement has to be sent to the RBC for the message
- Information about the radio connection. None or one of the following notifications:
 - Confirmation for establishing a connection or reconnection
 - Notification, that a established connection was lost, including the origin of the failure
 - Notification, that a connection could not be (re)established after 3 attempts, including the origin of the failure
 - Notification, that a connection could not be re-established after 3 attempts, including the origin of the failure

An overview over the output dataflows is provided in table 5.

Index	Output name	Output type
0	present	bool
1	rejectionReason	Boolean-Array (to be defined)
2	acknowledgementRequired	bool
3	radioConnectionStatusFromAPI	RadioConnectionStatusFromAPI_T
4	radioDeviceOut	int
5	applyServiceBreak	bool
6	badBaliseMessageToDMI	bool
7	receivedMessage	ReceivedMessage_T

Table 5. Dataflow at output

Output 0: present The present-flag specifies, if the data provided by the output `receivedMessage` is to be considered as present by the following modules or if it has to be ignored due to no new input data.

Value	Interpretation
false	The data in this element is not present and has to be ignored.
true	The data in this element is present and has to be processed by the following models.

Table 6. Possible values for the output present

Output 1: rejectedReason In case of an inconsistent message, the output `rejectedReason` is giving information to the system, which problem occurred. This information also has to be sent to the RBC as an error report.

Output 2: acknowledgementRequired The `acknowledgementRequired`-dataflow indicates, whether the reception of the message has to be acknowledged to the RBC.

Value	Interpretation
true	An acknowledgement has to be sent to the RBC for the current message delivered at output <code>receivedMessage</code>
false	No acknowledgement has to be sent for the current message delivered at output <code>receivedMessage</code>

Table 7. Possible values for the output AcknowledgementRequired

Output 3: radioConnectionStatusFromAPI The output `radioConnectionStatusFromAPI` is used, when the RTM reports problems with the radio connection. The output is derived from the Alstom-API. The output can be one of the following values:

Value	Interpretation
CONNECTION_CONFIRMATION	Confirmation for establishing a connection or reconnection
CONNECTION_LOST	Notification, that a established connection was lost
CONNECTION_FAILURE	Notification, that a connection could not be (re-)established after 3 attempts, including the origin of the failure
CONNECTION_NOT_ESTABLISHED	Notification, that a connection could not be re-established after 3 attempts, including the origin of the failure

Table 8. Possible values for the output radioConnectionStatusFromAPI

Output 4: radioDeviceOut The output radio device will give information, which device received a radio message. Trains equipped with two or more radio devices may receive messages on two interfaces in situations of a RBC handover.

Output 5: applyServiceBreak The flag indicates the balise group the train just passed could not be processed correctly. The check results in the request for a service break.

Output 6: badBaliseMessageToDMI Information to be passed to the DMI to indicate the reception of a “bad balise” to the driver.

Output 7: receivedMessage The element `receivedMessage` consists of the type `ReceivedMessage_T` combines both balise and radio messages to one common datatype. This datatype contains all variables and packets, which are possible for the given scenario.

Name	Datatype	Description
source	Enumeration	Defines, if this is a Euroradio or Eurobalise message.
valid	bool	true, if no consistency errors were detected.
BG_Common_Header	<code>BG_Header_T</code>	Header of Eurobalise message
Radio_Common_Header	<code>Radio_TrackTrain_Header_T</code>	Header of Euroradio message
Packets	structure of possible packets	-

Table 9. Structure of `ReceivedMessage_T`

The Eurobalise-common-header `BG_Header_T` consists of the fields visible in the SCADE-declaration. The structure corresponds to the structure defined in the SRS chapter 8.4.2.1. Some fields were removed since they are not needed anymore for further processing after building messages from separate telegrams.

The Euroradio-common-header `Radio_TrackTrain_Header_T` consists of the fields visible in the SCADE declaration. The structure corresponds to the structure defined in the SRS chapter 8.4.4.6.1. The structure contains all variables required by possible `NID_MESSAGE` values for the given scenario.

Note: Packet 44 not used (applications outside the ERTMS/ETCS system are not supported by this implementation).

4.2.2.5 Data

The function makes use of internal data for collecting and checking the balise telegrams.

4.2.2.6 Reference to the SRS (or other requirements)

Eurobalise

- Definition of the Balise Telegram: subset 26 section 7 and 8
- Interface to the BTM: Subset 36, section 4.2.2, 4.2.4, 4.2.9
- Handling of Balise Telegrams: Subset 26, sections 3.4.1 - 3.4.3, 3.16.2
- Check of the balise group Subset 26, section 3.16.2
- Determining the Orientation: 3.4.2
- Active Functions Table: 4.5.2

Euroradio

- SRS subset 26, chapter 8.4.4: Rules for Euroradio messages
- SRS subset 26, chapter 3.16: Data consistency

ValidateDataDirection

- The functionality is mainly described in [2, Chapter 3.6.3].

4.2.2.7 Design Constraints and Choices of submodules

Reception of messages

The first stage of the module is the reception of Euroradio-messages and Eurobalise-Telegrams from the openETCS-API. At each cycle the following conditions can occur:

1. No new Euroradio-message or Eurobalise-telegram is available.
2. A new Euroradio-message is available
3. A new Eurobalise-telegram is available
4. A new Euroradio-message and a new Eurobalise-telegram is available.

Part 1: ReceiveBaliseAndBuildBG

This function defines the interface of the OBU model to the openETCS generic API for Eurobalise Messages. On the interface, either a valid telegram is provided or a telegram is indicated which could not be received correct when passing the balise. The function passes the telegram without major changes of the information to the next entity for collecting the balise group information. This entity collects telegrams received via the interface into Balise Group Information.

Design Constraints and Choices

1. The decoding of balises is done at the API. Also, packets received via the interface are already transformed into a usable shape.
2. Only packets used inside the current model are passed via the interface:
Packet 5: Linking Information.
Linking Information is added to the linking array starting from index 0 without gaps. Used elements are marked as valid. Elements are sorted according to the order given by the telegram sequence.
3. Telegrams received as invalid are passed to the “Check-Function” to process errors in communication with the track side according to the requirements and in a single place. Telegrams are added to the telegram array starting from index 0 without gaps. Used elements are marked as valid. Elements are stored according to the order given by the telegram sequence.
4. This function does not process information from the packets. The information is passed to the check without further processing of the values.

Part 3: Check BG Consistency

Short Description of Functionality This function has the task to verify the completeness and correctness of the received messages from balis-groups.

A message consists of at least a telegram and a maximum of 8 telegrams.

- A message is still complete and correct, if a telegram is missing (or not decoded or incomplete decoded), and this telegram is duplicated within the balise group and the duplicating one is correctly read.
- By more than one telegram, the order of the telegrams must be either ascending (nominal) or Descending(reverse).
- A message is correct, if all message counters (M MCUNT) do not equal 254 (that means: The telegram never fits any message of the group).
A message counter can be equal 255 (that means: The telegram fits with all telegrams of the same balise group) and all other values must be the same.

Design Constraints and Choices This function is active in certain modes and the output and reactions are dependent on if the linking information is used.

The orientation of the BG will also be calculated in this block. The check, if the message has been received in due time and the right at the right expected location, will be performed in "Calculate Train Position".

The checks on the validity of the data in the packets and the validity with respect to the direction of motion will be performed in other modules, e.g. "Validate Data Direction" .

Part 5: CheckEuroradioMessage

"CheckEuroradioMessage" has to perform several checks on the received radio message.

- Content checks
 - The whole message must be complete and contains all necessary fields. (SRS 8.16.1.1)
 - The message must respect the ETCS language. (SRS 8.16.1.1)
 - The variables of the message does not contain invalid values. (SRS 8.16.1.1)
 - Check if the specified priority of message is equal to the priority with which the message was received. (SRS 3.16.3.1.3.1)
- Timing checks
 - Check if the timestamp of a message is greater than the timestamp of the former message (SRS 3.16.3.3.3)
 - If a message contains the timestamp “Unknown”, check if this message is part of the initiation of the communication session. (SRS 3.16.3.3.4)

- Perform the check with the current packet n : $T_TRAIN_n \leq T_TRAIN_{n-1} + T_NVCONTACT$ (SRS 3.16.1.1). This ensures, that the packet was received in due time.

For inconsistent messages, the following actions need to be performed by the module:

- If a message is not consistent, it shall be rejected (SRS 3.16.3.1.1.1). For this purpose, the message is marked as invalid.
- The RBC shall be informed, when a message was rejected (SRS 3.16.3.1.1.2). Therefore the message is marked with necessary information for creating an error report.
- If the RBC requested an ACK for a received message, message will be marked for the module to send a report to the RBC. (SRS 3.16.3.5)
- This module will not trigger the reaction for an interrupted radio connection to the RBC. The reaction specified by M_NVCONTACT will be triggered by the RBC session management module.

The check by the Euroradio-protocol (3.16.3.1.1) will not be performed by the model, but on a lower level (RTM or openETCS-API).

Safe connection supervision is not in the scope of this module. This functionality will be implemented by the “Manage Radio communication” module. The “Receive message and check consistency”-module will provide the necessary status data about the connection as an output.

4.2.2.8 Part 7: ValidateDataDirection

Short Description of Functionality

This function determines for direction information of the LRBG or an (ordinary) balise group whether this information is valid or not. The function takes as an input the LRBG and the balise groups passed and outputs the input extended with validity information.

Reference to the SRS (or other requirements)

Design Constraints and Choices

none

4.2.3 F.1.5 Select Usable Info - Mode and Level Filter

Short Description of Functionality

The function Select Usable Info filters information received from balises that have been passed, radio messages, and EUROLOOP messages. Filtering is done depending on the mode of the train, the current ETCS level, the type/content of the information, and the transition media of the information. As neither radio messages nor EUROLOOP are part of the first iteration of work, not all functionality of the filter described in the specification is currently implemented.

Reference to the SRS (or other requirements)

The functionality of Select Usable Info is described in Chapter 4.8 of subset-026 [2]. The following list gives an overview of the most important sections for each of the blocks in the model.

4.2.3.1 Interfaces

Input from: Receive MSG Check Consistency/Coordinate System - track messages and package Level and Mode Management - Mode and Level State

Output to: Build Data structure and Location Based/ Build Data Structures Drivers- forwarded packages, messages and variables

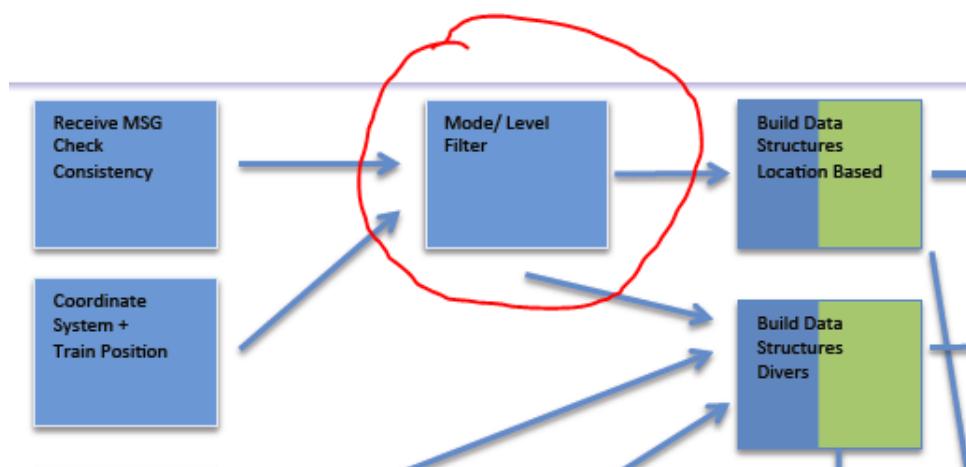


Figure 5. Filter In and out

4.2.3.2 SysML Model

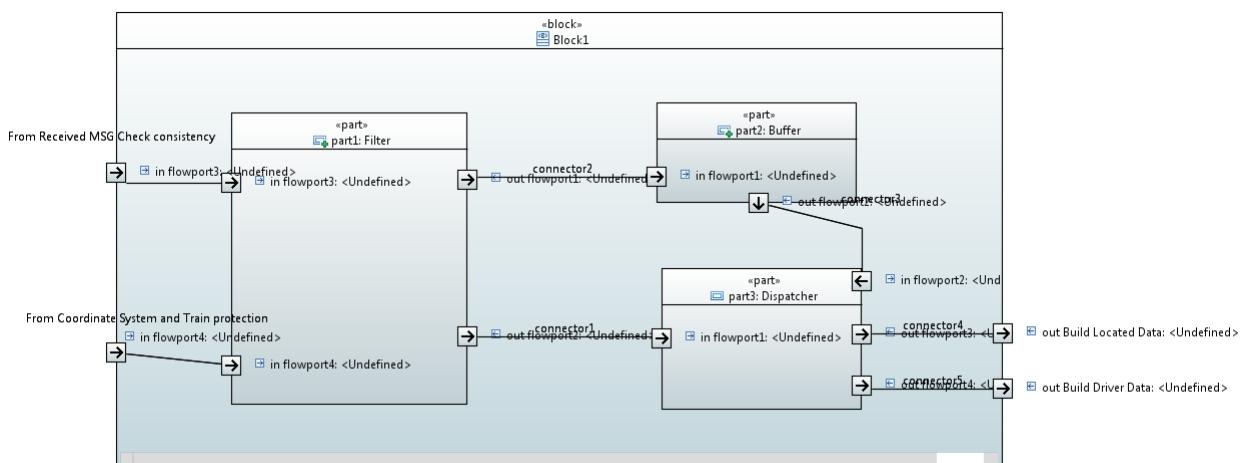


Figure 6. SysML Filter

Design Constraints and Choices

The filter receives track information (balise an radio) and will filter them in dependency of the mode and level. Therefore the filter needs the input from level and mode management. The filtered information will be forwarded to the data structure.

First filter The first filter, i.e. the filter on the level, is described in [2, Chapter 4.8.3].

Second filter The second filter, i.e. the filter on the transition media, is described in [2, Chapter 4.8.3].

Third filter The third filter, i.e. the filter on the modes, is described in [2, Chapter 4.8.4].

Transition buffers Details on the handling of the transition buffers used in the first and the second filter are described in [2, Chapter 4.8.5].

Rerference to SRS: § 4.8.2, § 4.8.2, § 4.8.3, § 4.8.4

Documentation of design

From § 4.8.1.2 The following sections have to be interpreted by applying the filters and the assigned packets/messages as shown in Figure a and 2. The first filter is detailed in section § 4.8.3 (figure 1) “Accepted information depending on the level and transmission media”, the third filter in section § 4.8.4 (figure 2) “Accepted information depending on the modes”.

From § 4.8.1.3 If a message contains level transition information, any other information in that message shall be evaluated considering the level transition information. Explanation: If a message contains level transition information, all other information in that message shall be buffered and level transition shall be read first. Then the remained balise information shall be read from the buffer in the level that was announced to the balise.

From § 4.8.1.3.1 Information received in the same message as an immediate level transition order or a conditional level transition order that causes a level transition shall be evaluated first considering the on-board currently operated level, as if a level transition order for further location had been received (i.e. conditions [1], [2] or [6] of Figure 1, if applied, shall be automatically fulfilled). Then, if relevant, it shall be immediately extracted from the buffer and re-evaluated according to the new selected level.

Explanation: As described in Explanation of § 4.8.1.3 and figure 1 – First Filter conditions [1], [2] and [6])

From § 4.8.1.4 Note: As shown in Figure 1, information stored following an announcement of a change of level, is re-checked for acceptance when the level has changed. This implies that, when the level changes, the mode is - for a short moment – still unchanged, until the stored information has been processed. The consequence for the Third Filter is that information needs to be accepted for this short period also in modes in which this information is otherwise useless. **Explanation:** when a level announced the level the mode change will be unchanged until the buffered information has been processed. The model change is the third filter (§ 4.8.3 figure 3).

table for the filter rules

Assumptions from § 4.8.2 need to be considered

Explanation: See figure 1 and 2 – announced packets/messages/variables to the filter. Exception and explanation of the meaning of R and A please read § 4.8.3.

Filter rules: Filter will filter messages, packages and variables. Therefore a rule must be defined to cover all these inputs.

Explanation figure 1: will filtering the different inputs in dependency of the level
Explanation figure 2: will filtering the different inputs in dependency of the mode

Filter on Level

Package/Variables	Information	From RBC	Onboard operating level			
			0	NTC	1	2
Packet 3	National Values	No	A	A	A	A
		Yes	R [2]	R [2]	R [2]	A
Packet 5	Linking	No	R [1]	R [1]	A	R [1]
		Yes	R [2]	R [2]	R [2]	A [3]
V_Main Packet 12	Signalling Related Speed Restriction	No	R [1]	R [1]	A	R [1]
		Yes				
Packet 12, 15	Movement Authority + (optional) Mode Profile + (optional) List of Balises for SH area	No	R [1]	R [1]	A [4]	R [1]
Packet 80						
Packet 49		Yes	R [2]	R [2]	R [2]	A [3] [4] [5]
Packet 16	Repositioning Information	No	R	R	A	R
Packet 21		Yes				
Packet 27	Gradient Profile	No	R [1]	R [1]	A	R [1]
Packet 51		Yes	R [2]	R [2]	R [2]	A [3]
Packet 21	International SSP	No	R [1]	R [1]	A	R [1]
Packet 41		Yes	R [2]	R [2]	R [2]	A [3]
Packet 46	Axle Load speed profile	No	R [1]	R [1]	A	R [1]
Packet 42		Yes	R [2]	R [2]	R [2]	A [3]
Packet 45	Level Transition Order	No	A	A	A	A
Packet 57		Yes	A	A	A	A
Packet 46	Conditional Level Transition Order	No	A [11]	A [11]	A [11]	A [11]
Packet 42		Yes				
Packet 45	Session Management	No	A	A	A	A
Packet 57		Yes	A	A	A	A
Packet 58	Radio Network registration	No	A	A	A	A
Packet 45		Yes	A	A	A	A
Packet 57	MA Request Parameters	No				
Packet 58		Yes	A	A	A	A
Packet 58	Position Report parameters	No				
Packet 45		Yes	A	A	A	A
Package 63 + Message Radio 2 + (optional) Packet 49	SR Authorisation + (optional) List of Balises in SR mode	No				
Packet 137		Yes	R	R	R	A [3]
Packet 137	Stop if in SR mode	No	R	R	A	A
Packet 137		Yes				
D_SR in Packet 13	SR distance information from loop	No	R	R	A	R
D_SR in Packet 13		Yes				

Packet 65	Temporary Speed Restriction	No	A	R [1] [2]	A	A [8]	A [8]
		Yes	R [2]	R [2]	R [2]	A [3]	A [3]
Packet 66	Temporary Speed Restriction Revocation	No	A	R [1] [2]	A	A	A
		Yes	R [2]	R [2]	R [2]	A [3]	A [3]
Package 64	Inhibition of revocable TSRs from balises in L2/3	No					
		Yes	R [2]	R [2]	R [2]	A	A
Packet 141	Default Gradient for TSR	No	A	R [1] [2]	A	A	A
		Yes					
Packet 70	Route Suitability Data	No	R [1]	R [1]	A	R [1]	R [1]
		Yes	R [2]	R [2]	R [2]	A [3]	A [3]
Packet 71	Adhesion Factor	No	R [1]	R [1]	A	R	R
		Yes	R [2]	R [2]	R [2]	A	A
Packet 72	Plain Text Information	No	A	R [1] [2]	A	A	A
		Yes	R [2]	R [2]	R [2]	A [12]	A [12]
Packet 76	Fixed Text Information	No	A	R [1] [2]	A	A	A
		Yes	R [2]	R [2]	R [2]	A [12]	A [12]
Packet 79	Geographical Position	No	A	R [1] [2]	A	A	A
		Yes	R [2]	R [2]	R [2]	A	A
Packet 131	RBC Transition Order	No	R	R	R	A	A
		Yes	R	R	R	A [3]	A [3]
Packet 132	Danger for SH information	No	A [13]	A [13]	A	A	A
		Yes					
Package 135	Stop Shunting on desk opening	No	A	A	A	A	A
		Yes					
Packet 133	Radio Infill Area information	No	R	R	A	R [1]	R [1]
		Yes					
Package 42	Session Management with neighbouring RIU	No	R	R	A	R	R
		Yes					
Packet 134	EOLM information	No	A	A	A	A	A
		Yes					
Messenger 45	Assignment of Co-ordinate system	No					
		Yes	A [10]	A [10]	A [10]	A [10]	A [10]
Packet 136	Infill Location Reference	No	R	R	A	R [1]	R [1]
		Yes					
Packet 39, Packet 68	Track Conditions excluding big metal masses	No	R [1]	R [1]	A	R [1]	R [1]
		Yes	R [2]	R [2]	R [2]	A [3]	A [3]
Packet 67	Track condition big metal masses	No	A	A	A	A	A
		Yes					

Header Balise	Location Identity (NID_C + NID_BG transmitted in the balise telegram)	No	A	A	A	A	A
		Yes					
Radio Message 6	Recognition of exit from TRIP mode	No					
		Yes	R	R	R	A	A
Message Radio 8	Acknowledgement of Train Data	No					
		Yes	A	A	A	A	A
Message 9	Co-operative shortening of MA + (optional) Mode Profile + (optional) List of Balises for SH area	No					
Packet 80		Yes	R	R	R	A [3] [4] [5]	A [3] [4] [5]
Packet 49		No					
Message Radio 16	Unconditional Emergency Stop	No					
	Conditional Emergency Stop	Yes	R [2]	R [2]	R [2]	A	A
Message Radio 15		No					
Message Radio 18	Revocation of Emergency Stop (Conditional or Unconditional)	No					
Message Radio 27		Yes	R	R	R	A	A
Message Radio 28 + (optional) Packet 49	SH authorised + (optional) List of Balises for SH area	No					
??		Yes	R	R	R	A [3]	A [3]
Packet 2		No					
Message Radio 34	Track Ahead Free Request	No					
Packet 140 Track to train, Packet 40 Train to track	Train Running Number	Yes	R	R	R	A [3]	A [3]
Message Radio 38		No					
Message 39	Initiation of session	No					
	Acknowledgement of session termination	Yes	R	R	R	A	A
		No	A	A	A	A	A
		Yes	A	A	A	A	A

Message 40	Train Rejected	No					
		Yes	R	R	R	A	A
Message 41	Train Accepted	No					
		Yes	R	R	R	A	A
Message Radio 43	SoM Position Report Confirmed by RBC	No					
		Yes	R	R	R	A	A
Packet 138	Reversing Area Information	No	R [1]	R [1]	A	R [1]	R [1]
		Yes	R [2]	R [2]	R [2]	A [3]	A [3]
Packet 139	Reversing Supervision Information	No	R [1]	R [1]	A	R [1]	R [1]
		Yes	R [2]	R [2]	R [2]	A [3]	A [3]
Packet 254	Default Balise/Loop/RIU Information	No	A	A	A	A	A
		Yes					
Packet 90	Track Ahead Free up to level 2/3 transition location	No	A [9]	A [9]	A [9]	R	R
		Yes					
Package 52	Permitted Braking Distance Information	No	R [1]	R [1]	A	R [1]	R [1]
		Yes	R [2]	R [2]	R [2]	A [3]	A [3]
Package 88	Level Crossing information	No	R [1] [2]	R [1] [2]	A	A	A
		Yes	R [2]	R [2]	R [2]	A [3]	A [3]
Package 6(0)	Virtual Balise Cover order	No	A	A	A	A	A
		Yes					
Package 44	Data to be used by applications outside ERTMS/ETCS	No	A	A	A	A	A
		Yes	A	A	A	A	A
		Onboard operating level					
	Information from National System X through STM interface	0	NTC X	NTC Y	1	2	3
??	STM max speed	A [7]	R	R [6]	A [7]	A [7]	A [7]
??	STM system speed/distance	A [7]	R	R	A [7]	A [7]	A [7]

Figure 7. Level Filter

Filter on Modes

40	Packet 39, 68	Track conditions sound horn, non stopping areas, turn left, non areas	NR	A[2][4]	R	R	A	A	A	R	R	A	R	A[1][1]	NR	NR	A	R	
41	Packet 67	Track condition long metal masses	NR	A[2][4]	A	A	A	A	A	R	R	A	A	A[1][1]	NR	NR	A	R	
42	Header Update	Location identity (NID_C + NID_BG)	NR	A[2]	A	A	A	A	A	R	R	A	A	A	NR	NR	A	A	
43	Message Radio 6	Recognition of exit from TRIP mode	NR	R	R	R	R	R	R	R	R	R	R	R	NR	NR	R	R	
44	Message Radio 8	Acknowledgement of Train Data	NR	A[2]	R	R	A	A	A	R	R	A	A	A	NR	NR	A	A	
45	Message 9	Coo-operative shortening of WA + (optional) Mode Profile	NR	R	R	R	A	R	A	R	R	R	R	R	NR	NR	A	A	
46	Packet 80	+ (optional) list of Balises for SH area																	
47	Packet 49																		
48	Message Radio 16	Unconditional Emergency Stop	NR	A[2]	R	R	A	A	A	R	R	A	A	R	R	NR	NR	A	R
49	Message Radio 15	Conditional Emergency Stop	NR	R	R	R	A	R	A	R	R	A	R	A[1][1]	NR	NR	A	R	
50	Message Radio 18	Revocation of Emergency Stop (Conditional or Unconditional)	NR	R	R	R	A	R	A	R	R	R	R	A[1][1]	NR	NR	R	R	
51																			
52	Message Radio 27	ShuttleRelease	NR	A[2]	R	R	A	A	A	R	R	R	R	R	NR	NR	R	R	
53	Message Radio 28, 1 (optional) Packet 49	ShuttleAssigned (Optional List of Balises in SH area)	NR	A[2]	R	R	A	A	A	R	R	R	R	A[1][1]	NR	NR	R	R	
54	77	TrainSafe condition System	NR	A	A	A	A	A	A	A	A	A	A	A	A	NR	NR	A	A
55	Packet 2	System Version order	NR	A	A	A	A	A	A	A	A	A	A	A	NR	NR	A	A	
56	Message Radio 34	TrackAhead Free Request	NR	A[2]	R	R	A	A	A	R	R	R	R	A[1][1]	NR	NR	R	R	
57	Packet 140 Track to Train, Packet 140 Train to Track	Train Running Number	NR	A[2]	R	R	A	A	A	R	R	A	R	A	NR	NR	R	A	
58	Message Radio 38	Initiation of session	NR	A	R	R	A	A	A	A	A	A	A	R	A	A	NR	R	
59	Message 39	Acknowledgement of session termination	NR	A	A	A	A	A	A	R	R	R	R	A	NR	NR	A	A	
60	Message 40	Train Rejected	NR	A[2]	R	R	R	R	R	R	R	R	R	R	NR	NR	R	R	
61	Message 41	Train Accepted	NR	A[2]	R	R	R	R	R	R	R	R	R	R	NR	NR	R	R	
62	Message Radio 43	Solo Position Report Confirmed by REC	NR	A[2]	R	R	R	R	R	R	R	R	R	R	NR	NR	R	R	
63	Packet 138	Reversing Area information	NR	A[2][4]	R	R	A	A	A	R	R	R	R	A	R	A[1][1]	NR	A	A
64	Packet 139	Reversing Supervision Information	NR	A[2][4]	R	R	A	A	A	R	R	A	R	A	R	A[1][1]	NR	A	A
65	Packet 254	Default BasedOnRPLU Information	NR	A[2]	A	A	A	A	A	A	A	A	A	A	NR	NR	A	A	
66	Packet 90	TrackAhead Free up to level 2/3 transition location	NR	A[2]	R	R	A	A	A	R	R	A	A	A	NR	NR	A	R	
67	Packet 52	Permit/Disallow instance	NR	A[2][4]	R	R	A	A	A	R	R	A	R	R	A[1][1]	NR	A	R	
68	Packet 88	Level Crossing information	NR	A[2][4]	R	R	A	A	A	R	R	A	R	A[1][1]	NR	NR	A	R	
69	Packet 60	Virtual Balise Cover order	NR	A	A	A	A	A	A	A	A	A	A	A	NR	NR	A	A	
70	Packet 44	Data to be used by applications outside ETHERNETS	NR	A	A	A	A	A	A	A	A	A	A	A	NR	NR	A	A	

	Packet, Message	Information	Modes
1	Packet 3	National Values	NP
2	Packet 5	Linking	NR A[2] A[2][4]
3	V_Main in Packet 12	Signalling Related Speed Restriction	NR
4	Packet 12..15	Movement Authority	R
5	(optional) Packet 80	+ (optional) Mode Profile	R
6	(optional) Packet 49	+ (optional) List of Balises for SH area	R
7	Packet 16	Reporting Information	R
8	Packet 21	Gradient Profile	R
9	Packet 27	International SSP	R
10	Packet 51	Aisle load speed profile	R
11	??	STM(max speed)	R
12	??	STM system speed/distance	R
13	Packet 41	Conditional Level Transition Order	R
14	Packet 42	Session Management	A[2] A[2][4]
15	Packet 49	Radio Network registration	A[2]
16	Packet 57	MAC Request Parameters	R
17	Packet 58	Position Report Parameters	R
18	Packet Radio 63 + Message	SS Administration+	R
19	Radio 2 + (optional) Packet 49	(optional) List of Balises in SR mode	R
20	Packet 137	Stop in SR mode	R
21	□_SR in Packet 13	SR distance information form stop	R
22	Packet 65	Temporary Speed Restriction	R
23	Packet 66	Temporary Speed Restriction Revocation	R
24	Package 64	Inhibition of several TSFs from balises in [2..3]	R
25	Packet 141	Default Gradient for TSR	R
26	Packet 70	Route Suitability Data	R
27	Packet 71	Adhesion Factor	R
28	Packet 72	Plan Test Information	R
29	Packet 76	Fixed Text Information	R
30	Packet 79	Geographical Position	R
31	Packet 131	RBC Transition Order	A[2] A[2][4] A[8]
32	Packet 132	Danger for SH information	R
33	Package 135	Stop Shunting on track opening	R
34	Package 133	Radio Infill Area information	R
35	Package 42	Session Management with neighbouring RBC	R
36	Package 134	ECU/M Information	R
37	Message Radio 45	Assignment Co-ordinate system	R
38	Package 136	Infill Location Reference	R
39	Packet 39..68	Track Conditions excluding sound insulation, noise limitation, sleepers, areas and metal masses	R

Figure 8. Mode Filter

Filtering (Mode/Level) - One packet per type

ISSUE: HOW MANY PKT 44, 65 AND 66 PER MESSAGE ARE MAXIMALLY SUPPORTED? (BH: who made this comment??)

- Check on announced and immediate level transition orders in the messages to be filtered (needed for further criteria for filtering, to decide if the data shall be stored in the transition buffer).
- Filter data stored in the transition buffer according to the current level (what to do if similar information is available in the new message??). Data can be rejected, accepted or kept in the transition buffer. (Filtering according to new level will be done directly afterwards in the next cycle)
- Filter new received messages according to the current level (new level will be done in the next cycle as according to SRS data first has to be filtered according to old level and afterwards to new level). Data can be rejected, accepted or stored in the transition buffer.
- Filter (level) accepted data according to originating RBC (supervising or other). Information from BG's, loops or RIU is not filtered with this filter.
- Filter (level and RBC) accepted data according to the current mode (only reject or accept)

4.2.4 Build coordinate system and calculate train position

- Update the coordinate system when a new BG is detected (taking into account detected “balise 1” from not completed BG’s), i.e. backward -- Calculation of position of passed BG’s.
- Management of multiple detected BG’s
- Relate the (location based) information in received messages to the reference system
- Update train position at LRBG
- Update the train position with the distance driven since last update (or reset to LRBG position)

4.2.4.1 F.2.2 Calculate Train Position

Short Description of Functionality

The main purpose of the function is to calculate the locations of linked and unlinked balise groups (BGs) and the current train position while the train is running along the track.

Functional Structure in Stages

The function calculateTrainPosition is divided into the following four functions, which are being performed sequentially:

1. **calculateBGLocations:** Calculate the balise group locations

The first stage is triggered each time the train passes a balise group (input *passedBG*). It takes the balise group header with the BG identification, the linking information (Subset 26, packet 5) and the current odometry values as inputs and calculates the location of the the passed balise group. If the passed BG has been announced via linking information previously, it takes into account the linking as well as the odometry information. If the passed BG does not meet the tolerance window announced by linking, an error flag is set. If the passed BG is an unlinked BG, its location is determined by odometry only, but related to the next previously passed linked BG, if there is one.

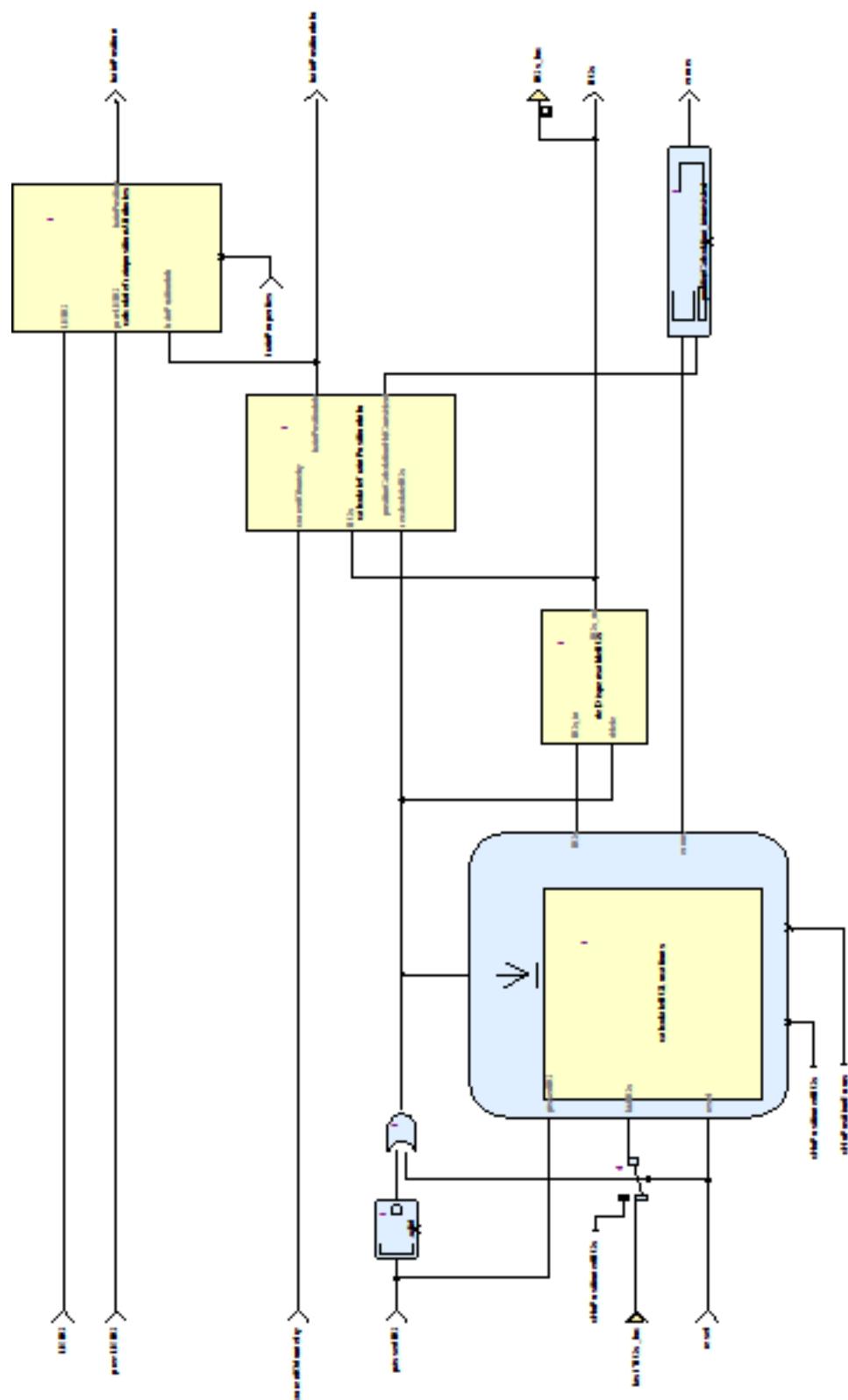


Figure 9. Structure of calculateTrainPosition

Then, if the passed BG is a linked BG comprising linking information for BGs ahead, the linking information is evaluated by creating the announced BGs and computing their locations from the linking distances.

The passed and the announced BGs are stored in a list *BGs*, ordered by their nominal location on the track.

Afterwards the locations of all BGs are further improved by re-adjusting their locations with reference to the just passed BG. This optimizes the BG location inaccuracies around the current train position (= location of the passed BG).

2. ***delDisposableBGs***: Delete dispensable balise groups

The second stage removes balise groups supposed not to be needed any longer from the list of *BGs*.

If the number of stored passed linked BGs exceeds the maximum number of eight as specified in [2, Chapter 3.6.2.2 c], all BGs astern are deleted. If only (passed) unlinked BGs are in the list and exceed the number of *cNoOfAtLeast_x_unlinkedBGs*, all passed BGs astern to those are removed from the list.

3. ***calculateTrainPositionInfo***: Calculate train position information.

This stage takes the list of stored BGs and the current odometry values as inputs and steadily provides the current train position.

4. ***calculateTrainpositionAttributes***: Calculate train position attribute information.

This stage provides several additional position related attributes that might conveniently be used by subsequent consumers in the architecture. It requires the actual LRBG and the previous LRBG to be assigned external from the list *BGs*.

Reference to the SRS (or other requirements)

The component calculateTrainPosition determines the location of linked and unlinked balise groups and the current train position during the train trip as specified mainly in [2, Chapter 3.6].

Design Constraints and Choices

The following constraints and prerequisites apply:

1. The input data received from the balises groups must have been checked and filtered for validity, consistency and the appropriate train orientation before delivering them to calculate-TrainPosition.
2. The storage capacity for balise groups is finite. calculateTrainPosition will raise an error flag when a balise group cannot be stored due to capacity limitations.
3. calculateTrainPosition will raise an error flag if a just passed balise group is not found where announced by linking information. It will not (yet) detect when an announced balise group is missing.
4. calculateTrainPosition is not yet prepared for train movement direction changes.
5. calculateTrainPosition does not yet consider repositioning information.

4.2.4.2 Provide Position Report

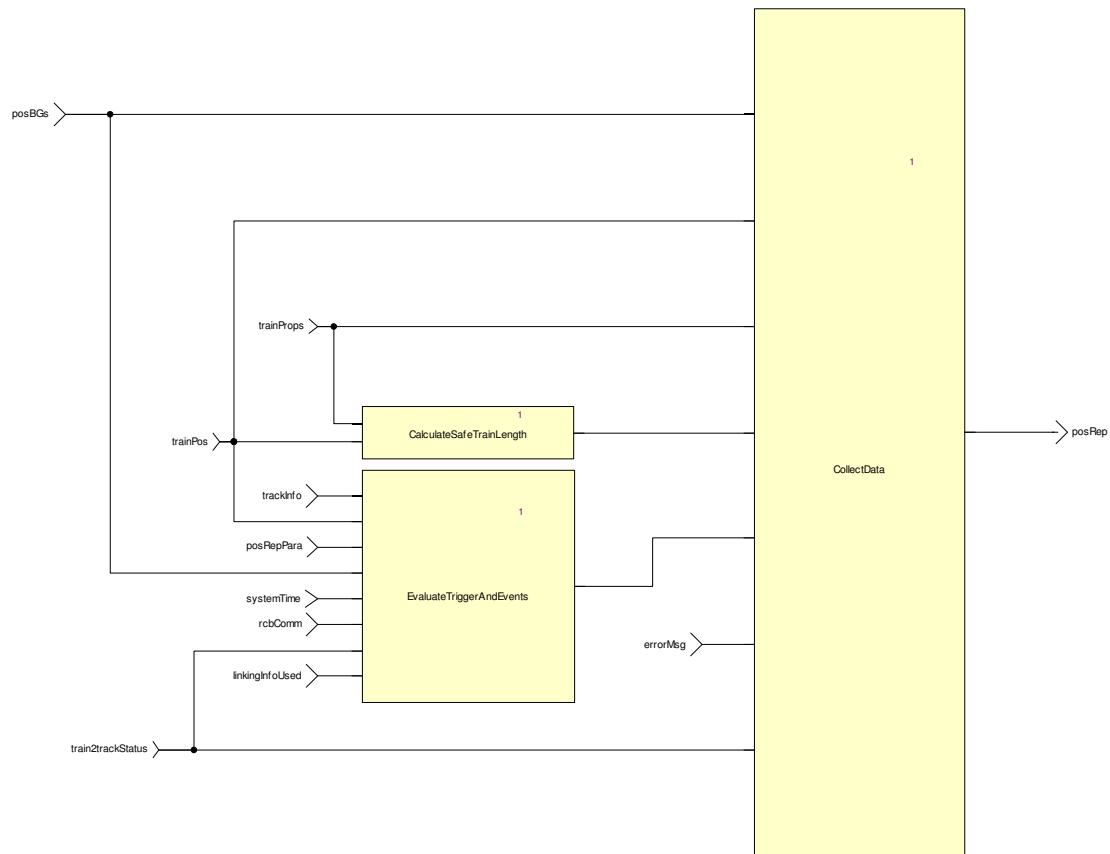


Figure 10. Structure of component ProvidePositionReport

Short Description of Functionality

This function takes the current train position and generates a position report which is sent to the RBC. The point in time when such a report is sent is determined by events, on the one hand, and position report parameters—which are basically triggers—provided by the RBC or a balise group passed, on the other hand. The functionality is modeled using three operations, as shown in Figure 10, which are explained below.

CalculateSafeTrainLength Calculates the the safeTrainLength and the MinSafeRearEnd according to [2, Chapter 3.6.5.2.4/5].

$\text{safeTrainLength} = \text{absolute}(\text{EstimatedFrontEndPosition} - \text{MinSafeRearEnd})$, where $\text{MinSafeRearEnd} = \text{minSafeFrontEndPosition} - L_TRAIN$.

EvaluateTriggerAndEvents Returns a Boolean modelling whether the sending of the next position report is triggered or not. This value is the conjunction of the evaluation of all triggers (PositionReportParameters, i.e., Packet 58) and events (see [2, Chapter 3.6.5.1.4]).

CollectData This operation aggregates data of Packet 0, ..., Packet 5 and the header to a position report.

Reference to the SRS (or other requirements)

Most of the functionality is described in [2, Chapter 3.6.5].

Design Constraints and Choices

1. The message length (i.e., attribute L_MESSAGE) is by default set to 0; the actual value will be set by the Bitwalker/API.
2. The attribute Q_SCALE is assumed to be constant; that is, all operations using this attribute do not convert between different values of that attribute.
3. *PositionReportHeader*: The time stamp (i.e., attribute T_TRAIN) is not set; this should be done once the message is being sent by the API.
4. *Packet 4*: When aggregating data for this packet, an error message might be overwritten by a succeeding error message. Because the specification allows only to send one error in one position report, errors are not being stored in a queue, for instance.
5. *Packet 44*: This packet is currently not contained in a position report as it is not part of the kernel functions.
6. The usage of attributes D_CYCLOC and T_CYCLOC as part of the triggers specified by the position report parameters (i.e., Packet 58 sent by the RBC) may lead to unexpected results if a big clock cycle together with small values for the attributes is used. The cause is that at every clock cycle the current model increments the reference value for the distance and time by at most D_CYCLOC and T_CYCLOC, respectively and not a factor of it.

Open Issues

1. The specification requires to store the last eight balise groups for which a position report has been sent (see [2, Chapter 3.6.2.2.c]).
2. For all reports that contain Packet 1 (i.e., report based on two balise groups), the RBC sends a coordinate system. It is unclear where this has to be stored (i.e., somehow the balise groups have to be stored in a database which has then to be updated), see [2, Chapter 3.4.2.3.3.6]. Moreover, such a coordination system can be invalid and then has to be rejected (see [2, Chapter 3.4.2.3.3.7-8]). On a more abstract level, we need to think about the interface between the RBC and the OBU or a proper abstraction thereof.

4.2.5 Store inputs from the TIU

- Store status inputs (sleeping indicator etc.)
- Store changed train data
- Store brake status (e.g. in the handling of brake testing).
- Store status of on-board systems (for displaying to the driver)
- Isolation
- Passive shunting

4.2.6 Store inputs from the DMI

- Store received acknowledgement and inputs in the “DMI request buffer” (includes “data-entry”)

4.2.7 Store data (direct orders, BG lists, NV, track data, procedure parameters, confirmations)

- Store direct and conditional orders
- Store BG lists for SH and SR
- Store National Values, including procedure status information
- Store new received track data (version, etc.)
- Store procedure parameters

4.2.8 Update location based data structures

- Overwrite location based data from a given location on-wards (reference location given in the message)
- Insert “Locations” in the data structure, in the order the “Locations” will be passed.

4.2.9 Manage specific location based data:

- movement authority (MA) list of sections, message 37, packet 12 (level 1), message 3, packet 15 (level 2), 16 (repositioning, i.e. extending the current section), message 33 (??), packet 70 (route suitability), message 9 (request to shorten MA), packet 90 (track ahead free leads to MA request) minimum number of elements to be stored: 6
- list of announced BG’s linking information: packet 5 minimum number of elements to be stored: 30
- adhesion factor: packet 71; only one element
- the “gradient profile” (in: pkt 21) minimum number of elements to be stored: 50
- Speed profiles: packet 27 (SSP) (the worst case can be determined at reception)
- Packet 13 minimum number of elements to be stored: 50
- Speed restrictions and non-continuous speed profiles: packet 51 (axle load profile), packet 52 (permitted braking distance), packets 65/66 (TSR), packet 88 (level crossing, incl. stop condition to be reset at standstill). minimum number of elements to be stored: TSR: 30, axle load: 30, permitted braking distance: 5 , level crossing: 10.
- Reversing area’s: packets 138, 139 minimum number of elements to be stored: 1
- Mode dependent speeds: message 2 and packet 80 minimum number of elements to be stored: 6
- Level transitions: packet 41 minimum number of elements to be stored: (see ss26, 5.10.1.6): 1
- RBC transitions: packet 131
- Radio infill area entry or exit: packet 133
- Loop announcement: packet 134
- DMI information: packets 72,76 (text messages), packet 79 (geographical position information), message 34 (track ahead free request) minimum number of elements to be stored: fixed text: 5, free text: 5, geographical position:
- Track conditions (to be passed to the TIU and displayed at the DMI): packet 39 (traction system), packet 40 (current limitation), packet 68 (diverse track conditions), packet 69 (platform conditions). Pkt 139 minimum number of elements to be stored: 20, + 1 for change power supply + 1 for platform conditions, + 1 for current limitation
- Route suitability: minimum number of elements to be stored: 3
- Big Metal Mass: Technical information (to be used for BG-filtering): packet 67 (ignore BG integrity) minimum number of elements to be stored: 5
- Virtual balise covers: minimum number of elements to be stored: 10
- list of position report locations. In: pkt. 58 minimum number of elements to be stored: 15

- Announced national values. In: pkt 3 minimum number of elements to be stored: 1 - If new national values are announced, then those can lead to more restrictive braking curves. Therefore a speed restriction has to be calculated for the location where the values become valid, based on the targets in advance of this point.

4.2.10 Build and update MRSP and list of targets at LRBG

- Overlay speed restrictions (one by one) over the resulting SSP as received from “build location based data”.
- Close the resulting profile with the “end of authority” or “limit of authority” as delivered by “MA-management” resulting in the MRSP at the LRBG. (in on-sight or limited supervision mode, those may not be available)
- Select list of “speed reductions”
- Evaluate (backwards) which targets are relevant, resulting in the list of most restrictive targets at the LRBG.
- Relocate targets (beyond this location) to the “minimum border crossing location”, i.e. the minimum safe location where National values are changed, and add the minimum resulting speed at this location to the list of targets.
- Reasoning: new national values might cause more conservative braking curves which otherwise could lead to an intervention at the border.

4.2.11 Profile supervision, i.e. BCM and ceiling speed supervision (active for FS, OS and LS)

- Update MRSP for distance driven, i.e. lower the distance to all speed decreases with the maximum distance driven since last update, lower the distance to all speed increases with the minimum distance driven since last update, reorder the distances in the list if locations to lower and to increase changed order.
- Determine the local maximum speed
- Update the list of targets, i.e. lower the distance to all targets with the maximum distance driven since last update (order will not change) and select the current most restrictive target (always the first in the list)
- braking curve monitoring; calculate the braking curves to the most restrictive target, taking into account gradients
- ceiling speed supervision; monitor against the local maximum speed.

4.2.11.1 Movement supervision

- Roll away protection

4.2.11.2 Area Supervision

In shunting, post trip, reversing and staff responsible an area (plus in some cases ceiling speed) is protected. In unfitted only a speed. The way the area is protected differs per mode therefore a function shall be available for each mode:

- area supervision in shunting (taking into account a list of BG's to be passed)
- area supervision in staff responsible (taking into account a list of BG's to be passed and/or a

- distance)
- reversing area supervision
- post trip supervision (only reverse movement).

4.2.11.3 Update procedure status (including commanding actions towards driver, radio or RBC)

- SoM:
- EoM
- Enter shunting by driver or track side command override
- Enter “on sight”
- Enter “staff responsible”
- level transitions
- Manage train trip
- Exit shunting
- Passive shunting
- Change train orientation
- Splitting/combining
- Stopping in rear of LX level crossing
- Changing train data (not by the driver)
- Handling track conditions (including indications): not applicable for Amsterdam-Utrecht
- Limited supervision entry: not applicable for Amsterdam-Utrecht
- release brakes (after trip)
- handle track ahead free request

4.2.11.4 Mode/Level management

- Manage all conditions for level changes except the direct transitions (handled in filtering): F41
- Manage all conditions for mode changes (except msg 16 if that leads to a mode change): F42

4.2.11.5 DMI management

- update the MRSP at the LRBG and construct the DMI image
- collect BCM results
- Check displaying of location dependent txt messages
- Calculate geographical position based on stored track-km references
- Display required ack-requests, T.A.F. requests etc.
- Display mode and level information

4.2.11.6 RBC communication

- Manage confirmation requests.
- Report train position
- sent train data for validation
- provide acknowledgements on data reception

4.2.11.7 TIU communication

- Communicate track conditions
- Brake test procedure.

4.2.11.8 JRU and STM management: not applicable for Utrecht-Amsterdam

Appendix A: Restrictions on Interfaces to openETCS OBU

The following summarizes restrictions in the scope of implementation of the openETCS OBU. The chosen restrictions are valid for the current scope of the modelling work. They are based on the functions needed to cover the use-case of Utrecht-Amsterdam line.

A.1 Track to Train Interface

Track to Train Interface: Packets Received and The Coverage in openETCS (Section 7.4.1):

Packet Number	Packet Name	Relevant in Scope
0	Virtual Balise Cover marker	
2	System Version order	
3	National Values	
5	Linking	
6	Virtual Balise Cover order	
12	Level 1 Movement Authority	
13	Staff Responsible distance information from loop	
15	Level 2/3 Movement Authority	
16	Repositioning Information	
21	Gradient Profile	
27	International Static Speed Profile	
39	Track Condition Change of traction system	
40	Track Condition Change of allowed current consumption	
41	Level Transition Order	
42	Session Management	
44	Data used by applications outside the ERTMS/ETCS system.	
45	Radio Network registration	
46	Conditional Level Transition Order	
49	List of balises for SH Area	
51	Axle load Speed Profile	
52	Permitted Braking Distance Information	
57	Movement Authority Request Parameters	
58	Position Report Parameters	

63	List of Balises in SR Authority
64	Inhibition of revocable TSRs from balises in L2/3
65	Temporary Speed Restriction
66	Temporary Speed Restriction Revocation
67	Track Condition Big Metal Masses
68	Track Condition
69	Track Condition Station Platforms
70	Route Suitability Data
71	Adhesion Factor
72	Packet for sending plain text messages
76	Packet for sending fixed text messages
79	Geographical Position Information
80	Mode profile
88	Level crossing information
90	Track Ahead Free up to level 2/3 transition location
131	RBC transition order
132	Danger for Shunting information
133	Radio infill area information
134	EOLM Packet
135	Stop Shunting on desk opening
136	Infill location reference
137	Stop if in Staff Responsible
138	Reversing area information
139	Reversing supervision information
140	Train running number from RBC
141	Default Gradient for Temporary Speed Restriction
143	Session Management with neighbouring Radio Infill Unit
145	Inhibition of balise group message consistency reaction
180	LSSMA display toggle order
181	Generic LS function marker
254	Default balise, loop or RIU information

A.2 TIU Interfaces

The following information is based on the structure given by the Alstom API.

A.2.1 Input to openETCS

Packet Number	Packet Name	Relevant in Scope
0	Inputs from train devices	
1	Plain text message	
2	Fixed text message	
3	brake models	
4	Not used	
5	Not used	
6	Test and failure detection	
7	STMs specific behaviour	
8	Specific from MVB (Specific to Alstom implementation)	
12	Diagnostic	
13	Inhibition Level (Specific to Alstom implementation)	

A.2.2 Output from openETCS

Packet Number	Packet Name	Relevant in Scope
0	Commands	
1	Track conditions	
2	Odometric data	
3	Other information	
4	Train type	
5	Track condition change of traction power	
6	Location reference update	
7	Sporadic commands	
8	STMs states	
9	Train information	
10	Doors control section	
11	Track description deletion information	
14	Gradients	

Appendix: References

- [1] openETCS Product Owner. *WP3 Product Backlog, Selection of First Iteration*. <https://github.com/openETCS/SRS-Analysis/labels/backlog%20item>.
- [2] ERA. *System Requirements Specification, SUBSET-026*, v3.3.0 edition, March 2012.
- [3] openETCS WP3-team. *openETCS Working Result: List of ETCS Functions*, November 2013. https://github.com/openETCS/SRS-Analysis/blob/master/System%20Analysis/List_Functions.xlsx.
- [4] openETCS. *openETCS SCADE model*, 2014. <https://github.com/openETCS/modeling/tree/master/model/Scade/System>.