OETCS TK-01-01

openETCS

ITEA2 Project
Call 6 11025
2012 – 2015

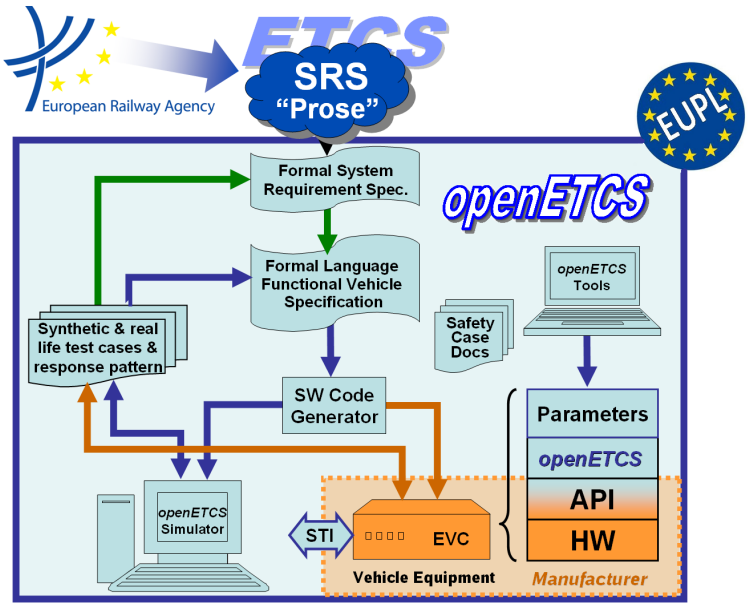
Work-Package 3: “Modeling”

# openETCS System Architecture and Design Specification

**Third iteration: Scope of openETCS ITEA2 Functions**

Baseliyos Jacob, Bernd Hekele, Peyman Farhangi, Stefan Karg, Valerio D´Angelo, Uwe Steinke, Christian Stahl, Jakob Gärtner, Jos Holtzer, Jan Welvaarts, Vincent Nuhaan and Jacob Gärtner

November 2014




**Funded by:**

Federal Ministry of Education and Research
Région de Bruxelles-Capitale
GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, ENERGÍA 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<br><br>Baseliyos Jacob<br>(DB Netz AG) | signature<br><br>[assessor name]<br>([affiliation]) | signature<br><br>Izaskun de la Torre<br>(SQS) | signature<br><br>Klaus-Rüdiger Hase<br>(DB Netz) |

Baseliyos Jacob, Bernd Hekele, Peyman Farhangi, Stefan Karg, Valerio D´Angelo

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

Uwe Steinke

Siemens AG

Christian Stahl

TWT-GmbH

Jakob Gärtner

LEA Engineering

Jos Holtzer, Jan Welvaarts, Vincent Nuhaan

Nederlands Spoorwegen

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 objecrive of the ITEA2 project. The goal is to demonstrate the proof of concept on the ETCS Level 2 Utrecht Amsterdam track with real scenarios. It has to be read as an add-on to the models in SysML, Scade and to additional reading referenced from the document.

# 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

# Figures and Tables

## Figures



## Tables

# 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 2015, it is of paramount importance that a first design deliverable 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.

## 1.2 Objectives

The prime objective of WP3 is to produce a fully formal 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
- Compartment 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

## 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:**

- **Responsibilities** 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 ot 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 Assumptions and preconditions

- All future contributions shall be fully aligned an compliant the finalized and approved document
- Alls 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 openETCS history and iterations

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

**Third Iteration Functional Scope: The OBU functions for Szenarios defined in chapter 3**

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

The appropriate functionality has been divided into a list of functions of different complexity to make it more managable by the different designer.

All these functions are object of the openETCS project and have to be analysed from their requirements and subsequently modelled and implemented. With the given manpower in WP 3, 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.

**The first objective of this third iteration are:**

- “Make the train run as soon as possible, on the scenarios defined in third iteration (see chapter 3), 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.

# 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 topological 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. |

# 2 Input documents

**This paragraphs gives an overview about the input doducments used for the:**

- Analysis of the OBU Functions
- Functional decomposition and allocation of macro- and microfunctions
- Design of the OBU Functions
- Determination of "Use Cases" and Szenarios for the different iteration of the Architecture and Design Document

## 2.1 Document x

## 2.2 Document x

## 2.3 Document x

## 2.4 Document x

## 2.5 Document x

# 3 Use case description - proof of concept Utrecht - Amsterdam

## 3.1 User Stories 1 - 4

# 4 Architecture Description

## 4.1 System Architecture

## 4.2 Interfaces

# 5 Runtime API

## 5.1 Interfaces

## 5.2 Header - service functions

# 6 Design Description

## 6.1 F1: Receive information from Trackside

## 6.2 F2: ETCS Kernel

### 6.2.1 Manage_TrackSideInformation_Integration

The block “Manage_TrackSideInformation_Integration” 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.

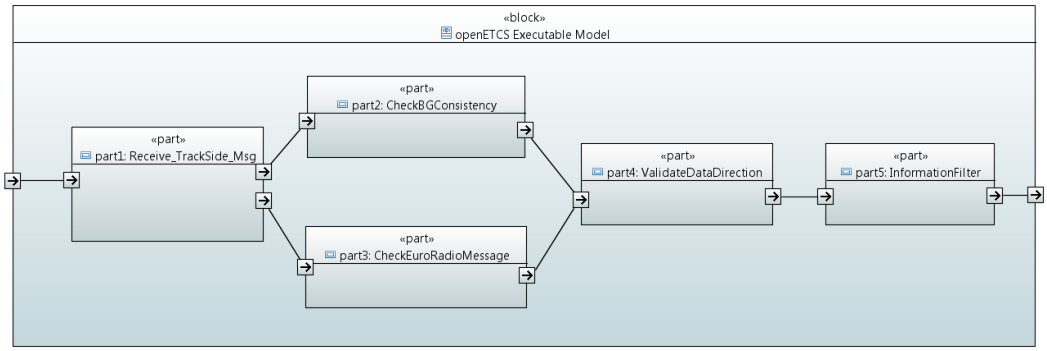



**Figure 1. Structure of the Receive message and check consistency module**

#### 6.2.1.1 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 | fullChecks | bool | Configuration |
| 1 | API_trackSide_Message | API_Msg_Pkg::API_TrackSideInput_T | API |
| 2 | ActualOdometry | Obu_BasicTypes_Pkg::odometry_T | Odometer |
| 3 | reset | bool | Environment |
| 4 | trainPosition | TrainPosition_Types_Pck::trainPosition_T | Calculate Train Position |
| 5 | modeAndLevel | BG_Types_Pkg::ModeAndLevelStatus_T | Mode and Level |
| 6 | tNvContact | Obu_BasicTypes_Pkg::T_internal_Type | Database |
| 7 | lastRelevantEventTimestamp | Obu_BasicTypes_Pkg::T_internal_Type | Database |
| 8 | connectionStatus | Radio_Types_Pkg::sessionStatus_Type | Manage Radio Communication |
| 9 | inSupervisingRbcId | int | Database |
| 10 | inAnnouncedBGs | TrainPosition_Types_Pck::positionedBGs_T | Calculate Train Position |
| 11 | q_nvlocacc | Q_NVLOCACC | Database |

**Table 2. Overview over input**

### Input 0: fullChecks

The boolean indicates, if all checks on the message should be performed. The possible values are given in table 3.

| Value | Interpretation |
|---|---|
| true | All checks are performed. |
| false | The module Information Filter is deactivated. |

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

### Input 1: API_trackSide_Message

The API_trackSide_Message is the message received from the API. The API performs pre-processing of RTM and BTM messages and deliveres a maximum of a single message per cycle to the SCADE model.

### Input 2: ActualOdometry

The input ActualOdometry is provided by the external odometry module of the train. It contains location information with inaccuracies.

### Input 3: 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**

#### Input 4: trainPosition

The input trainPosition is generated by the “Calculate Train Position” module and contains the current position of the train.

#### Input 5: modeAndLevel

The input is generated by the “Mode and level management” module. It provides the current level and mode of the EVC.

#### Input 6: tNvContact

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

#### Input 7: 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 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 8: 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 the timing 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 5. Possible values for the input connectionStatus**

#### Input 9: inSupervisingRbcId

For the submodule “Information Filter”, the information is needed, which radio messages are sent by the supervising RBC. To recognize these messages, the identifier of the supervising RBC is needed.

#### Input 10: inAnnouncedBGs

This input provides information about balise groups which will be passed by the train soon. This information is generated by “Calculate Train Position” based on the linking information received from trackside.

#### Input 11: q_nvlocacc

The national value determines the location accuracy and is delivered by the database.

### 6.2.1.2 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.

An overview over the output dataflows is provided in table 6.

| Index | Output name | Output type |
|---|---|---|
| 0 | outputMessage | Common_Types_Pkg::ReceivedMessage_T |
| 1 | ApplyServiceBrake | bool |
| 2 | BadBAliseMessageToDMI | bool |
| 3 | errorLinkedBG | bool |
| 4 | errorUnlinkedBG | bool |
| 5 | passedBG | BG_Types_Pkg::passedBG_T |
| 6 | outPositionParams | Common_Types_Pkg::PositionReportParameter_T |
| 7 | outRadioManagement | Common_Types_Pkg::radioManagementMessage_T |
| 8 | radioSequenceError | bool |
| 9 | radioMessageConsistencyError | bool |

**Table 6. Dataflow at output**

**Output 0: outputMessage** The element outputMessage 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 |
|---|---|---|
| valid | bool | true, if no consistency errors were detected. |
| source | Common_Types_Pkg::MsgSource_T | Defines, if this is a Euroradio or Eurobalise message. |
| packetMetadata | Common_Types_Pkg::Metadata_T | contains the metadata of the packets |
| radioMetadata | Common_Types_Pkg::RadioMetadata_T | contains the metadata of the radio specific header variables |
| BG_Common_Header | BG_Types_Pkg::BG_Header_T | Header of Eurobalise message |
| Radio_Common_Header | Radio_Types_Pkg::Radio_TrackTrain_Header_T | Header of Euroradio message |
| packets | Common_Types_Pkg::Packets_T | Structure of packets in messages |

**Table 7. 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. Which values are valid is defined in the field radioMetadata.

**Output 1: 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 2: BadBaliseMessageToDMI** Information to be passed to the DMI to indicate the reception of a “bad balise” to the driver.

| Value | Interpretation |
|---|---|
| true | A error in a linked balise group was detected. |
| false | No error in a linked balise group was detected. |

**Table 8. Possible values for the input errorLinkedBG**

**Output 3: errorLinkedBG**

| Value | Interpretation |
|---|---|
| true | A error in an unlinked balise group was detected. |
| false | No error in an unlinked balise group was detected. |

**Table 9. Possible values for the input errorUnlinkedBG**

**Output 4: errorUnlinkedBG**

**Output 5: passedBG** The output passedBG provides the received balise group message in a special format needed by the module “Calculate train position”.

**Output 6: outPositionParams** The output outPositionParams provides the parameters for the position report in a special format needed by the module “Provide Position Report”.

**Output 7: outRadioManagement** The output outRadioManagement provides the messages for radio session management in a special format needed by the module “Management of Radio Communication”.

| Value | Interpretation |
|---|---|
| true | A sequence error or a timeout has been detected in the radio message. |
| false | No error in the radio message sequence was detected. |

**Table 10. Possible values for the input `radioSequenceError`**

### Output 8: `radioSequenceError`

| Value | Interpretation |
|---|---|
| true | A consistency error has been detected in the radio message. |
| false | No consistency error in the radio message was detected. |

**Table 11. Possible values for the input `radioMessageConsistencyError`**

### Output 9: `radioMessageConsistencyError` 6.2.1.3 Receive_TrackSide_Msg in Manage_TrackSideInformation_Integration

#### Reference to the SRS (or other requirements)

- 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
- Rules for Euroradio messages: Subset 26, chapter 8.4.4

#### Short description of the functionality

This function defines the interface of the OBU model to the openETCS generic API for Eurobalise and Euroradio messages. On the interface, either a valid telegram/message is provided or a telegram/message is indicated which could not be received correct when passing the balise or receiving the radio message. The function passes a balise 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. In case of a radio message, the message is converted to an internal format for further processing and passed without changing the information contained.

#### Interface

### Functional Design Description

#### 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.

3. Treatment of 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.
4. 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.
5. This function does not process information from the packets. The information is passed to the check without further processing of the values.

### Reference to the Scade Model

The SCADE model can be found on github under the following path:

https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/ManageLocationRelatedInformation/BaliseGroup/Receive_TrackSide_Msg

#### 6.2.1.4 CheckBGConsistency in Manage_TrackSideInformation_Integration

### Reference to the SRS or other Requirements (or other requirements)

- Definition of the Balise Telegram: subset 26 section 7 and 8
- 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
- Active Functions Table: 4.5.2

### Short description of the functionality

This function has the task to verify the completeness and correctness of the received messages from balise 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.

#### Interface

#### Functional Design Description

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" .

#### Reference to the Scade Model

The SCADE model can be found on github under the following path:

https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/ManageLocationRelatedInformation/BaliseGroup/CheckBGConsistency

### 6.2.1.5 CheckEuroradioMessage in Manage_TrackSideInformation_Integration

#### Reference to the SRS or other Requirements (or other requirements)

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

#### Short description of the functionality

The function “CheckEuroradioMessage” has to perform several checks on the received radio message. These checks include checking of the message sequence, completeness of messages. Invalid messages are marked as invalid in the header.

The bitwalker is responsible for checking the validity of the values in fields. If a consistency error is detected by the bitwalker, it is signalled to the model. If the bitwalker marks a packet as valid, all variables are expected to contain a valid value.

#### Interface

#### Functional Design Description

- Content checks
  - The whole message must be complete and contains all necessary fields. (SRS 3.16.1.1)
  - The message must respect the ETCS language. (SRS 3.16.1.1)
  - The variables of the message does not contain invalid values. (SRS 3.16.1.1)
  - 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 $n$: $T\_TRAIN_n <= T\_TRAIN_{n-1}+T\_NVCONTACT$ (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 necessary information for creating an error report is provided as an output.
- This module will not trigger the reaction for an interrupted radio connection to the RBC. The reaction sepcified by `M_NVCONTACT` will be triggered by the RBC session management module.

The check by the Euroradio-protocol (SRS 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.

#### Reference to the Scade Model

The SCADE model can be found on github under the following path:

`https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/ManageLocationRelatedInformation/BaliseGroup/CheckEuroRadioMessage`

### 6.2.1.6 ValidateDataDirection in Manage_TrackSideInformation_Integration

#### Reference to the SRS or other Requirements (or other requirements)

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

#### Short description of the functionality

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

#### Interface

#### Functional Design Description

- The module contains two processing paths for either a radio message or for a balise message.

#### Reference to the Scade Model

The SCADE model can be found on github under the following path:

https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/ManageLocationRelatedInformation/BaliseGroup/ValidateDataDirection

### 6.2.1.7 InformationFilter in Manage_TrackSideInformation_Integration

#### Reference to the SRS or other Requirements (or other requirements)

- The functionality of Select Usable Info is described in Chapter 4.8 of subset-026 [**?** ]. The following list gives an overview of the most important sections for each of the blocks in the model.
- § 4.8.2, § 4.8.2, § 4.8.3, § 4.8.4

#### Short description of the 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.

#### Interface

**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

### 6.2.1.8 SysML Model

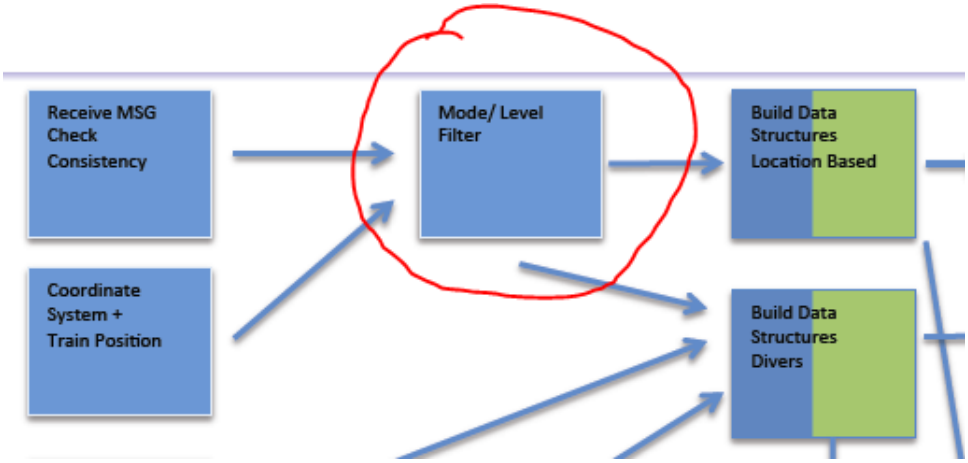



**Figure 2. Filter In and out**

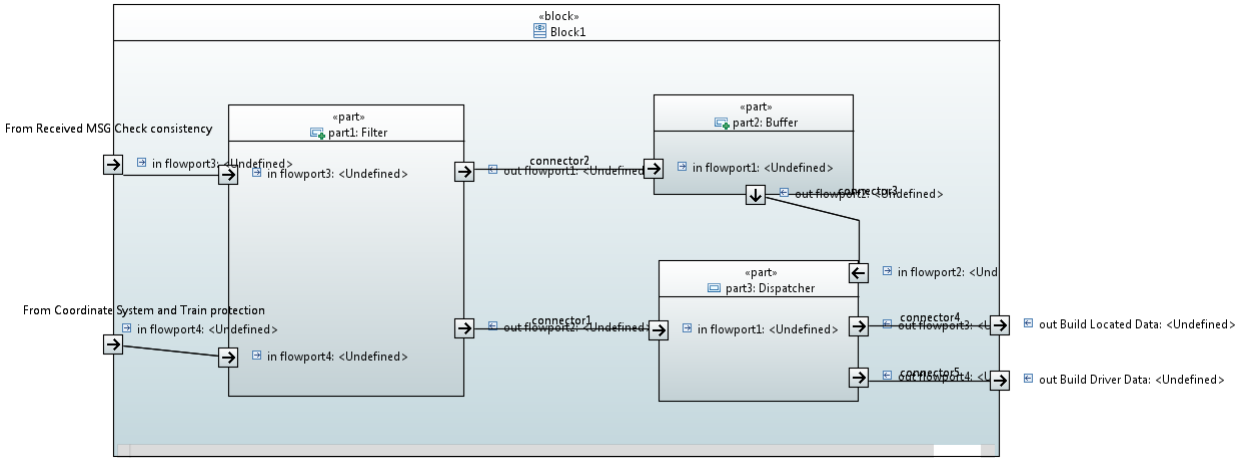



**Figure 3. SysML Filter**

### Functional Design Description

The fillter 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 strcuture.

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

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

**Third filter** The third filter, i.e. the filter on the modes, is described in [**?** , 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 [**?** , Chapter 4.8.5].

**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. Therefole 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 thel mode

| Package/Variables | Information | From RBC | Onboard operating level | | | | |
|---|---|---|---|---|---|---|---|
| | | | 0 | NTC | 1 | 2 | 3 |
| Packet 3 | National Values | No | A | A | A | A | A |
| | | Yes | R [2] | R [2] | R [2] | A | A |
| Packet 5 | Linking | No | R [1] | R [1] | A | R [1] | R [1] |
| | | Yes | R [2] | R [2] | R [2] | A [3] | A [3] |
| V_Main Packet 12 | Signalling Related Speed Restriction | No | R [1] | R [1] | A | R [1] | R [1] |
| | | Yes | | | | | |
| Packet 12, 15<br>Packet 80<br>Packet 49 | Movement Authority<br>+ (optional) Mode Profile<br>+ (optional) List of Balises for SH area | No | R [1] | R [1] | A [4] | R [1] | R [1] |
| | | Yes | R [2] | R [2] | R [2] | A [3] [4] [5] | A [3] [4] [5] |
| Packet 16 | Repositioning Information | No | R | R | A | R | R |
| | | Yes | | | | | |
| Packet 21 | Gradient Profile | No | R [1] | R [1] | A | R [1] | R [1] |
| | | Yes | R [2] | R [2] | R [2] | A [3] | A [3] |
| Packet 27 | International SSP | No | R [1] | R [1] | A | R [1] | R [1] |
| | | Yes | R [2] | R [2] | R [2] | A [3] | A [3] |
| Packet 51 | Axle Load speed profile | No | R [1] | R [1] | A | R [1] | R [1] |
| | | Yes | R [2] | R [2] | R [2] | A [3] | A [3] |
| Packet 41 | Level Transition Order | No | A | A | A | A | A |
| | | Yes | A | A | A | A | A |
| Packet 46 | Conditional Level Transition Order | No | A [11] | A [11] | A [11] | A [11] | A [11] |
| | | Yes | | | | | |
| Packet 42 | Session Management | No | A | A | A | A | A |
| | | Yes | A | A | A | A | A |
| Packet 45 | Radio Network registration | No | A | A | A | A | A |
| | | Yes | A | A | A | A | A |
| Packet 57 | MA Request Parameters | No | | | | | |
| | | Yes | A | A | A | A | A |
| Packet 58 | Position Report parameters | No | | | | | |
| | | Yes | A | A | A | A | A |
| Package 63 + Message Radio 2 + (optional) Packet 49 | SR Authorisation + (optional) List of Balises in SR mode | No | | | | | |
| | | Yes | R | R | R | A [3] | A [3] |
| Packet 137 | Stop if in SR mode | No | R | R | A | A | A |
| | | Yes | | | | | |
| D_SR in Packet 13 | SR distance information from loop | No | R | R | A | R | R |
| | | 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 | | | | | |
| Messager 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<br>Packet 80<br>Packet 49 | Co-operative shortening of MA<br>+ (optional) Mode Profile<br>+ (optional) List of Balises for SH area | No | | | | | |
| | | Yes | R | R | R | A [3] [4] [5] | A [3] [4] [5] |
| Message Radio 16 | Unconditional Emergency Stop | No | | | | | |
| | | Yes | R [2] | R [2] | R [2] | A | A |
| Message Radio 15 | Conditional Emergency Stop | No | | | | | |
| | | Yes | R [2] | R [2] | R [2] | A | A |
| Message Radio 18 | Revocation of Emergency Stop (Conditional or Unconditional) | No | | | | | |
| | | Yes | R | R | R | A | A |
| Message Radio 27 | SH refused | No | | | | | |
| | | Yes | R | R | R | A [3] | A [3] |
| Message Radio 28 + (optional) Packet 49 | SH authorised + (optional) List of Balises for SH area | No | | | | | |
| | | Yes | R | R | R | A [3] | A [3] |
| ?? | Trackside constituent System Version | No | A | A | A | A | A |
| | | Yes | A | A | A | A | A |
| Packet 2 | System Version order | No | A | A | A | A | A |
| | | Yes | | | | | |
| Message Radio 34 | Track Ahead Free Request | No | | | | | |
| | | Yes | R | R | R | A [3] | A [3] |
| Packet 140 Track to train, Packet 40 Train to track | Train Running Number | No | | | | | |
| | | Yes | R | R | R | A | A |
| Message Radio 38 | Initiation of session | No | | | | | |
| | | Yes | R | R | R | A | A |
| Message 39 | Acknowledgement of session termination | 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 |
| | Information from National System X through STM interface | Onboard operating level | | | | | |
| | | 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 4. Level Filter**

### Filter on Level

### Filter on Modes

| 40 | Packet 39, 68 | Track conditions sound horn, non stopping areas, tunnel stopping areas | NR | A[2][4] | R | R | A | A | A | A | A | R | R | A | A | R | A[1] | NR | NR | A | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 41 | Packet 67 | Track condition big metal masses | NR | A[2][4] | A | A | A | A | A | A | A | A | A | A | A | A | A[1] | NR | NR | A | R |
| 42 | Header Balise | Location Identity (NID_C + NID_BG) | NR | A[2] | A | A | A | A | A | A | A | A | A | A | 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 | R | R | A | NR | NR | R | R |
| 44 | Message Radio 8 | Acknowledgement of Train Data | NR | A[2] | R | R | A | A | A | A | A | R | R | A | A | A | A | NR | NR | A | A |
| 45 | Message 9 | Co-operative shortening of MA | NR | R | R | R | A | A | R | A | R | R | R | A | R | R | R | NR | NR | R | R |
| 46 | Packet 80 | + (optional) Mode Profile | | | | | | | | | | | | | | | | | | | |
| 47 | Packet 49 | + (optional) List of Balises for SH area | | | | | | | | | | | | | | | | | | | |
| 48 | Message Radio 16 | Unconditional Emergency Stop | NR | A[2] | R | R | A | A | A | A | A | R | R | A | R | R | A[1] | NR | NR | A | R |
| 49 | Message Radio 15 | Conditional Emergency Stop | NR | R | R | R | A | A | R | A | R | R | R | A | R | R | A[1] | NR | NR | A | R |
| 50 | Message Radio 18 | Revocation of Emergency Stop | NR | R | R | R | A | A | R | A | R | R | R | A | R | R | A[1] | NR | NR | R | R |
| 51 | | (Conditional or Unconditional) | | | | | | | | | | | | | | | | | | | |
| 52 | Message Radio 27 | SH refused | NR | A[2] | R | R | A | A | A | A | A | R | R | A | R | R | A[1] | NR | NR | R | R |
| 53 | Message Radio 28 + (optional) Packet 49 | SH authorised + (optional) List of Balises for SH area | NR | A[2] | R | R | A | A | A | A | A | R | R | A | R | R | A[1] | NR | NR | R | R |
| 54 | ?? | Trackside constituent System Version | NR | A | A | 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 | A | A | A | NR | NR | A | A |
| 56 | Message Radio 34 | Track Ahead Free Request | NR | A[2] | R | R | R | A | A | A | A | R | R | R | R | R | A[1] | NR | NR | R | R |
| 57 | Packet 140 Track to train, Packet 40 Train to track | Train Running Number | NR | A[2] | R | R | A | A | A | A | A | R | A | R | A | A | A | NR | NR | A | A |
| 58 | Message Radio 38 | Initiation of session | NR | A | R | R | A | A | A | A | A | A | A | R | A | A | A | NR | NR | R | A |
| 59 | Message 39 | Acknowledgement of session termination | NR | A | A | A | A | A | A | A | A | A | A | A | A | A | A | NR | NR | A | A |
| 60 | Message 40 | Train Rejected | NR | A[2] | R | R | R | 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 | R | R | R | NR | NR | R | R |
| 62 | Message Radio 43 | SoM Position Report Confirmed by RBC | NR | A[2] | R | R | R | 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 | A | A | R | R | A | R | R | A[1] | NR | NR | A | A |
| 64 | Packet 139 | Reversing Supervision Information | NR | A[2][4] | R | R | A | A | A | A | A | R | R | A | R | R | A[1] | NR | NR | A | A |
| 65 | Packet 254 | Default Balise/Loop/RIU Information | NR | A[2] | A | A | A | A | A | A | A | A | A | A | A | A | A | NR | NR | A | A |
| 66 | Packet 90 | Track Ahead Free up to level 2/3 transition location | NR | A[2] | R | R | A | A | A | A | A | R | R | A | A | A | A | NR | NR | A | R |
| 67 | Package 52 | Permitted Braking Distance Information | NR | A[2][4] | R | R | A | A | A | A | A | R | R | A | R | R | A[1] | NR | NR | A | R |
| 68 | Package 88 | Level Crossing information | NR | A[2][4] | R | R | A | A | A | A | A | R | R | A | R | R | A[1] | NR | NR | A | R |
| 69 | Package 60) | Virtual Balise Cover order | NR | A | A | A | A | A | A | A | A | A | A | A | A | A | A | NR | NR | A | A |
| 70 | Package 44 | Data to be used by applications outside ERTMS/ETCS | NR | A | A | A | A | A | A | A | A | A | A | A | A | A | A | NR | NR | A | A |

| | Packet, Message | Information | Modes | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | NP | SB | PS | SH | FS | LS | SR | OS | SL | NL | UN | TR | PT | SF | IS | SN | | RV |
| 1 | Packet 3 | National Values | NR | A[2] | A | A | A | A | A | A | A | A | A | A | A[1] | NR | NR | A | | A |
| 2 | Packet 5 | Linking | NR | AD[4] | R | R | A | A | A | A | R | A | A | R | A[1] | NR | NR | A | | R |
| 3 | V_Main in Packet 12 | Signalling Related Speed Restriction | NR | AD[4] | R | R | A | A | A | A | R | R | A | R | A[1] | NR | NR | A | | R |
| 4 | Packet 12, 15 | Movement Authority | NR | AD[4] | R | R | A | A | A | A | R | R | A | R | A[1] | NR | NR | A | | R |
| 5 | (optional) Packet 80 | + (optional) Mode Profile | | | | | | | | | | | | | | | | | | |
| 6 | (optional) Packet 49 | + (optional) List of Balises for SH area | | | | | | | | | | | | | | | | | | |
| 7 | Packet 16 | Repositioning Information | NR | R | R | R | A | A | R | A | R | R | R | R | R | NR | NR | R | | R |
| 8 | Packet 21 | Gradient Profile | NR | AD[4] | R | R | A | A | A | A | R | R | A | R | A[1] | NR | NR | A | | R |
| 9 | Packet 27 | International SSP | NR | AD[4] | R | R | A | A | A | A | R | R | A | R | A[1] | NR | NR | A | | R |
| 10 | Packet 51 | Axle load speed profile | NR | AD[4] | R | R | A | A | A | A | R | R | A | R | A[1] | NR | NR | A | | R |
| 11 | ?? | STM max speed | NR | A[2] | R | R | A | A | A | A | R | R | A | A | A[1] | NR | NR | A | | R |
| 12 | ?? | STM system speed/distance | NR | A[2] | R | R | A | A | A | A | R | R | A | A | A[1] | NR | NR | R | | R |
| 13 | Packet 41 | Level Transition Order and Conditional Level Transition Order | NR | A[2] | A[7] | A[7] | A | A | A | A | A | A | A | A | A[1][5] | NR | NR | A | | R |
| 14 | Packet 42 | Session Management | NR | A | A[3] | A[3] | A | A | A | A | A | A | A | A | A[1] | NR | NR | A | | A |
| 15 | Packet 49 | Radio Network registration | NR | A[2] | A | A | A | A | A | A | A | A | A | A | A[1] | NR | NR | A | | A |
| 16 | Packet 57 | MA Request Parameters | NR | A[2] | R | R | A | A | A | A | R | R | A | R | A[1] | NR | NR | A | | R |
| 17 | Packet 58 | Position Report parameters | NR | A[2] | R | R | A | A | A | A | R | A | A | R | A[1] | NR | NR | A | | A |
| 18 | Package Radio 63 + Message Radio 2 + (optional) Packet 49 | SR Authorisation+ (optional) List of Balises in SR mode | NR | AD[4] | R | R | R | R | A | R | R | R | R | R | A[1] | NR | NR | R | | R |
| 20 | Packet 137 | Stop if in SR mode | NR | R | R | R | R | R | A | R | R | R | R | R | R | NR | NR | R | | R |
| 21 | D_SR in Packet 13 | SR distance information from loop | NR | R | R | R | R | R | A[6] | R | R | R | R | R | R | NR | NR | R | | R |
| 22 | Packet 65 | Temporary Speed Restriction | NR | AD[4] | R | R | A | A | A | A | R | R | A | A | A[1] | NR | NR | A | | R |
| 23 | Packet 66 | Temporary Speed Restriction Revocation | NR | AD[4] | R | R | A | A | A | A | R | R | A | A | A[1] | NR | NR | A | | R |
| 24 | Package 64 | Inhibition of revocable TSRs from balises in L2/3 | NR | A[2] | R | R | A | A | A | A | R | R | A | A | A[1] | NR | NR | A | | R |
| 25 | Packet 141 | Default Gradient for TSR | NR | AD[4] | R | R | A | A | A | A | R | R | A | A | A[1] | NR | NR | A | | R |
| 26 | Packet 70 | Route Suitability Data | NR | AD[4] | R | R | A | A | A | A | R | R | A | R | A[1] | NR | NR | A | | R |
| 27 | Packet 71 | Adhesion Factor | NR | AD[4] | R | R | A | A | A | A | R | R | A | R | A[1] | NR | NR | A | | R |
| 28 | Packet 72 | Plain Text Information | NR | A[2] | R | R | A | A | A | A | R | A | A | A | A[1] | NR | NR | A | | A |
| 29 | Packet 76 | Fixed Text Information | NR | A[2] | R | R | A | A | A | A | R | A | A | A | A[1] | NR | NR | A | | A |
| 30 | Packet 78 | Geographical Position | NR | A[2] | R | R | A | A | A | A | R | A | A | A | A[1] | NR | NR | A | | R |
| 31 | Packet 131 | RBC Transition Order | NR | AD[4] | A[8] | A[8] | A | A | A | A | A | A | R | A | A[1] | NR | NR | R | | R |
| 32 | Packet 132 | Danger for SH information | NR | R | R | A | R | R | R | R | R | R | R | R | R | NR | NR | R | | R |
| 33 | Package 135 | Stop Shunting on desk opening | NR | R | A | R | R | R | R | A | R | R | R | R | R | NR | NR | R | | R |
| 34 | Package 133 | Radio Infill Area information | NR | R | R | R | A | A | A | A | R | R | R | R | R | NR | NR | R | | R |
| 35 | Package 42 | Session Management with (non-)equipped RBC | NR | R | R | R | A | A | A | A | R | R | R | R | R | NR | NR | R | | R |
| 36 | Package 134 | EOLM information | NR | R | R | A | A | A | A | A | A | A | A | A | R | NR | NR | A | | A |
| 37 | Message Radio 45 | Assignment of Co-ordinate system | NR | A[2] | R | R | R | R | A | R | R | A | A | R | A[1] | NR | NR | A | | R |
| 38 | Package 136 | Infill Location Reference | NR | R | R | R | A | A | R | R | R | R | R | R | R | NR | NR | R | | R |
| 39 | Packet 39, 68 | Track Conditions excluding sound horn, non stopping areas, tunnel stopping areas and big metal masses | NR | AD[4] | R | R | A | A | A | A | R | A | A | A | A[1] | NR | NR | A | | R |

**Figure 5. 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 system requirement specification (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 balise group (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)

#### Reference to the Scade Model

The SCADE model can be found on github under the following path:

`https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/ManageLocationRelatedInformation/BaliseGroup/InformationFilter`

### 6.2.2 Train Supervision

The task of block "Train Supervision" is to monitor the speed of the train and the train location and as such to ensure that the speed remains within the given speed and distance limits. This block is mainly based on [**?** , Chapt. 3.13].

The block "Train Supervision" takes as input (1) movement related information such as train speed, train position and acceleration, (2) train related information such as brake information and train length, and (3) track related information such as speed and distance limits and national values.

Based on this information a list of targets is calculated and, for each such target, the speed the train should have. Braking curves calculate at which location at the track the train has to accelerate and to perform the brake, respectively. These calculations lead to commands being sent to the driver and the brake system.

The functionality is modeled using four operators, as shown in Figure **??**, which are explained below.

For a more detailed analysis of "Train Supervision" and a functional breakdown, we refer to the appendix.

#### 6.2.2.1 Input

For providing the output, the module needs different input data flows. Table 12 gives an overview.

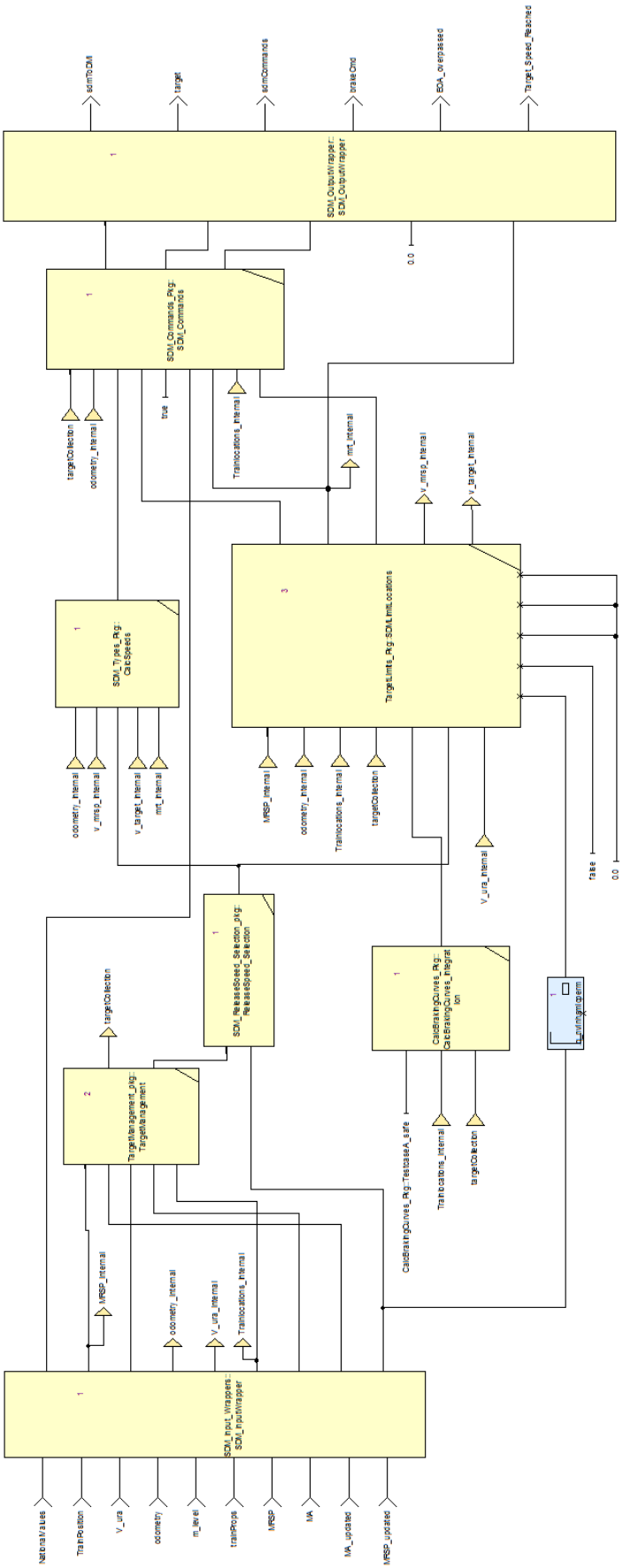



**Figure 6. Structure of component ProvidePositionReport**

| Index | Input name | Input type | Source |
|---|---|---|---|
| 0 | MRSP | MRSP_Profile_t | ?? |
| 1 | MA | MAs_t | ?? |
| 2 | NationalValues | P3_NationalValues_T | ??? |
| 3 | TrainPosition | trainPosition_T | Manage Track Data |
| 4 | V_ura | V_internal_Type | ?? |
| 5 | odometry | odometry_T | Odometry |
| 6 | m_level | M_LEVEL | Mode and Level |
| 7 | trainProps | trainProperties_T | Database |
| 8 | MA_updated | bool | internal |
| 9 | MRSP_updated | bool | internal |

**Table 12. Overview of input**

#### Input 0: MRSP

This input is the most restrictive speed profile.

#### Input 1: MA

This input is a movement authority.

#### Input 2: NationalValues

This input is packet 3 of [**?** , Chapt. 8], describing the national values.

#### Input 3: TrainPosition

This input is the current train position.

#### Input 4: V_ura

This input is the speed under reading amount.

#### Input 5: odometry

This input is the odometry data.

#### Input 6: m_level

This input is the current level of the train.

#### Input 7: trainProps

This input is a set of train related properties.

#### Input 8: MA_updated

This flag is true if the movement authority has been updated in this clock cycle and false otherwise.

#### Input 9: MRSP_updated

This flag is true if the most restrictive speed profile has been updated in this clock cycle and false otherwise.

#### 6.2.2.2 Output

Based on the input the block produces the following output. Table 13 gives an overview.

| Index | Output name | Output type |
|---|---|---|
| 0 | sdmToDMI | speedSupervisionForDMI_T |
| 1 | target | Target_T |
| 2 | sdmCommands | SDM_Commands_T |
| 3 | brakeCmd | Brake_command_T |
| 4 | EOA_overpassed | bool |
| 5 | Target_Speed_Reached | bool |

**Table 13. Overview of output**

#### Output 0: sdmToDMI

This output contains information about different speeds and positions, on the one hand and the current supervision status, on the other hand. This information shall be displayed to the driver.

#### Output 1: target

This output is the most restrictive displayed target (MRDT).

#### Output 2: sdmCommands

This output gives some intermediate results of operator SDM_Commands. It is currently used for test purposes only.

#### Output 3: brakeCmd

This output is the brake command, indicating whether performing the service brake or the emergency brake have been commanded.

#### Output 4: EOA_overpassed

This output is true if the end of authority has been overpassed and false otherwise.

#### Output 5: Target_Speed_Reached

This output is true if the current speed is greater than or equal the target speed and false otherwise.

#### 6.2.2.3 SDM_InputWrapper in Train Supervision

#### Reference to the SRS or other Requirements (or other requirements)

- [**?** , Chapt. 3.13]: Speed and distance monitoring

#### Short description of the functionality

The motivation for this operator is to convert all inputs of block “Speed Supervision” that contain information about length, speed, distance, and acceleration defined as integer into real to allow for highest precision in the calculations. In addition, to ease the modeling, inside block “Speed Supervision” only units meters, seconds, and meters per square second are used.

#### Interface

#### Functional Design Description

This operator forwards input messages, takes data from complex data types or transforms inputs messages into an internal type thereby converting int to real.

#### Reference to the Scade Model

**only in special case or link to the Scade model**

### 6.2.2.4 TargetManagement in Train Supervision Ben

#### Reference to the SRS or other Requirements (or other requirements)

- [**?** , Chapt. 3.13.8.2]: Determination of the supervised targets

#### Short description of the functionality

This operator calculates/updates the list of targets to be supervised by the block “Train Supervision”. Taking the current movement authority and the most restrictive speed profile as an input, the operator outputs a list of locations corresponding to the most restrictive speed profile, a list of locations corresponding to a limit of authority, the location of an end of authority, or the location of supervised location.

#### Interface

#### Functional Design Description

#### Reference to the Scade Model

**only in special case or link to the Scade model**

### 6.2.2.5 CalcBrakingCurves_Integration in Train Supervision Ben

#### Reference to the SRS or other Requirements (or other requirements)

- [**?** , Chapt. 3.13.8.3]: Emergency Brake Deceleration curves (EBD)

- [**?** , Chapt. 3.13.8.4]: Service Brake Deceleration curves (SBD)
- [**?** , Chapt. 3.13.8.5]: Guidance curves (GUI)

#### Short description of the functionality

For each type of target a certain braking curve has to be calculated. This curve enables us, given a target speed, to calculate the destination when this speed will be reached. In addition, given a target location, also the speed of the train at this location can be calculated.

#### Interface

#### Functional Design Description

Some details about how the curves are calculated ...

Currently, the model supports the calculation of the following braking curves:

- the Emergency Brake Deceleration curve for the most restrictive speed profile,
- the Emergency Brake Deceleration curve for the limit of authority,
- the Emergency Brake Deceleration curve for the end of authority, and
- the Service Brake Deceleration curve for the end of authority

#### Reference to the Scade Model

**only in special case or link to the Scade model**

### 6.2.2.6 SDMLimitLocations in Train Supervision

#### Reference to the SRS or other Requirements (or other requirements)

- [**?** , Chapt. 3.13.9]: Supervision Limits

#### Short description of the functionality

This operator calculates the various locations needed to determine the speed and distance monitoring commands.

#### Interface

#### Functional Design Description

#### Reference to the Scade Model

**only in special case or link to the Scade model**

### 6.2.2.7 CalcSpeeds in Train Supervision

#### Reference to the SRS or other Requirements (or other requirements)

- [**?** , Chapt. 3.13.9]: Supervision Limits

#### Short description of the functionality

This operator calculates the various speeds needed to determine the speed and distance monitoring commands.

#### Interface

#### Functional Design Description

This operator will be integrated into other operators in the next iteration.

#### Reference to the Scade Model

**only in special case or link to the Scade model**

### 6.2.2.8 SDM_Commands in Train Supervision

#### Reference to the SRS or other Requirements (or other requirements)

- [**?** , Chapt. 3.13.10]: Speed and distance monitoring commands

#### Short description of the functionality

This operator models the speed and distance monitoring commands. More precisely, it triggers the service or emergency brake and outputs the current supervision status of the OBU together with information on speeds and locations to the driver.

#### Interface

#### Functional Design Description

The OBU can be in any of three types of speed and distance monitoring modes: ceiling speed monitoring, release speed monitoring and target speed monitoring. We use a state machine to model the switching between the three modes: each state models a mode and a transition between to states is enabled if the condition two switch between the two corresponding modes is evaluated to true. In each mode, the OBU can be in up to five different supervision stati. The behavior of changing from one status to another is also modeled as a state machine. As a result, the model is a hierarchical state machine.

#### Reference to the Scade Model

**only in special case or link to the Scade model**

### 6.2.2.9 SDM_OutputWrapper in Train Supervision

#### Reference to the SRS or other Requirements (or other requirements)

- [**?** , Chapt. 3.13]: Speed and distance monitoring

#### Short description of the functionality

This operator is the counterpart to operator SDM_OutputWrapper—that is, it converts all internal outputs of block “Speed Supervision” that contain information about length, speed, distance, and acceleration defined as real into int, such that all other blocks can stick to their types and also performs the calculation into units used by the environment.

#### Interface

#### Functional Design Description

This operator forwards input messages and transforms inputs messages into an internal type thereby converting real to int.

#### Reference to the Scade Model

**only in special case or link to the Scade model**

### 6.2.3 Manage ETCS Procedures

#### 6.2.3.1 Macrofunction x in Manage ETCS Procedures

#### Reference to the SRS or other Requirements (or other requirements)

#### Short descriptoiin of the functionality

#### Interface

#### Functional Design Description

#### Refernce to the Scade Model

**only in special case or link to the Scade model**

#### 6.2.3.2 Macrofunction x in Manage ETCS Procedures

#### Reference to the SRS or other Requirements (or other requirements)

#### Short descriptoiin of the functionality

#### Interface

#### Functional Design Description

#### Refernce to the Scade Model

**only in special case or link to the Scade model**

#### 6.2.3.3 Macrofunction x in Manage ETCS Procedures

**Reference to the SRS or other Requirements (or other requirements)**

**Short descriptoiin of the functionality**

**Interface**

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

#### 6.2.3.4 Macrofunction x in Manage ETCS Procedures

**Reference to the SRS or other Requirements (or other requirements)**

**Short descriptoiin of the functionality**

**Interface**

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

### 6.2.4 Manage Track Data

#### 6.2.4.1 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.

In detail, the calculateTrainPosition function provides a couple of essential subfunctions for the onboard unit. These are mainly

- creating and maintaining an obu internal coordinate system for all types of location based data
- storing all linked and unlinked balise groups resulting from over passing or from announcements (linking information) from the track
- calculating and maintaining the locations of all stored balise groups during the train trip, based on odometry and linking information

- permanently calculating the current train position based on odometry and passed balise group information
- providing the last recently passed linked balise group as the LRBG
- providing additional position attribute information
- deleting stored balise groups, when appropriate
- detecting linking consistency errors
- determining, if linking is used on board

The calculation algorithms for locations and positions are implemented as specified in `https://github.com/openETCS/SRS-Analysis/blob/master/System%20Analysis/WorkingRepository/Group4/SUBSET_26_3-6/DetermineTrainLocationProcedures.pdf` .

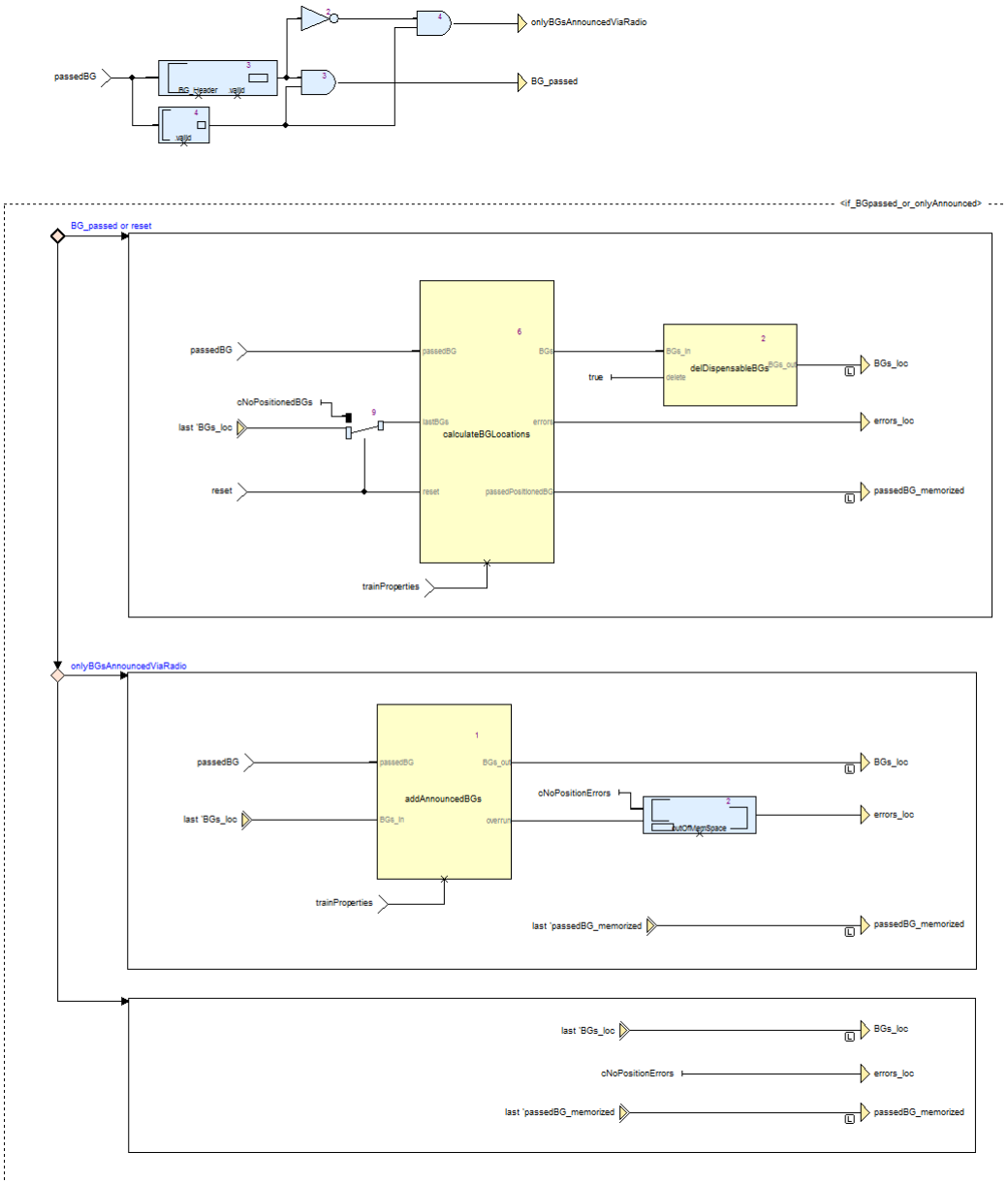



**Figure 7. Calculating the balise group locations**

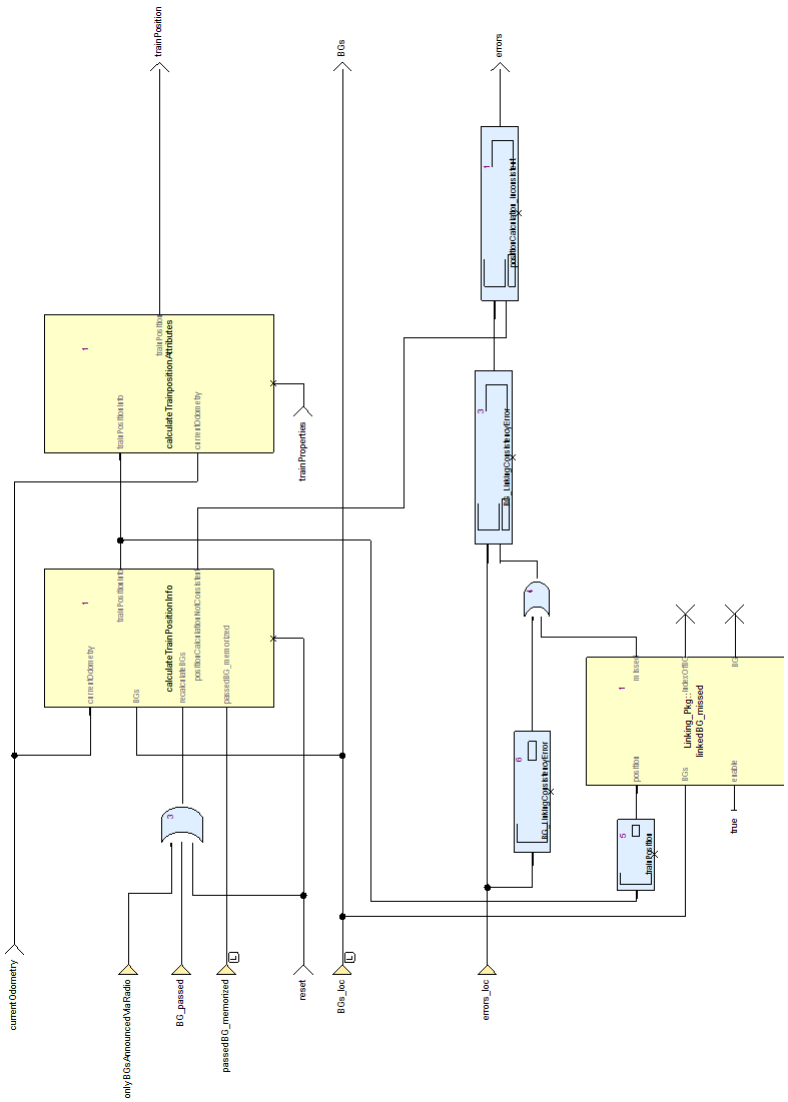



**Figure 8. Calculating the current train position and attributes**

### Functional Structure in Stages

calculateTrainPositions receives its input information via its passedBG entry as an event. The first decicion to be made is, if an input event is available and if it origins from a balise group just over passed or from the RBC via radio.

If the passedBG input events is caused by over passing a balise group, the balise group gets an OBU coordinate system location assigned ("calculateBGLocations") and is stored internally. If the just passed balise group announces more balise groups ahead via linking, they are stored with their locations calculated.

If the passedBG input event origins from RBC data received ("addAnnouncedBGs"), it only announces balise groups ahead. These announced balise groups get their locations calculated with reference to the LRBG and are stored as well.

"calculateBGLocations" and "addAnnouncedBGs" produce a list of known balise groups ("BGs").

The following stages "calculateTrainPositionInfo" and "calculateTrainpositionAttributes" all the time calculate the current train position by using the list of known balise groups (including the LRBG) and the current odometry information and determine additional position attributes.

In parallel "linkedBG_missed" is part of linking consistency supervision. It detects, when an announced balise group is not found within its expectation window.

In more detail, "calculateTrainPosition" is divided into a data flow of different stages, which are being performed sequentially:

1. ***calculateBGLocations***: Calculate the balise group locations
   The 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 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 expectation 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 (LRBG), if there is one.
   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* in the order they are passed and by their announced 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 inaccuries around the current train position (= location of the passed BG).

2. ***delDispensableBGs***: Delete dispensable balise groups
   The function 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 as specified in [**?** , Chapter 3.6.2.2.2 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. ***addAnnouncedBGs***: This function is executed once each time balise groups ahead are announced by the RBC. The locations of the announced *BGs* are calculated with reference to the LRBG reported by the RBC.

4. ***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. Additionally, it watches the list of announced *BGs* and provides the "Linking information is used" information as specified in chapt. 3.4.4.2.1.1 of the SRS.

5. ***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 in addition provides the current LRBG and the previous LRBG from the list *BGs*.

6. ***linkedBG_missed***: This function observes the list of *BGs* and the current train position. If an announced balise group is not found within its expectation window, an error flag will be raised.

#### 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 [**?** , Chapter 3.6].

#### Design Constraints and Choices

The following constraints and prerequisites apply:

1. The input data received from the balises groups or via radio must have been checked and filtered for validity, consistency and the appropriate train orientation before delivering them to calculateTrainPosition.

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 or if an announced balise group is missed when the end of its expectation window is reached. It does not yet implement all conditions of linking consistency.

4. calculateTrainPosition is not yet prepared for train movement direction changes.

5. calculateTrainPosition does not yet consider repositioning information.

### 6.2.4.2 Provide Position Report

#### 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,

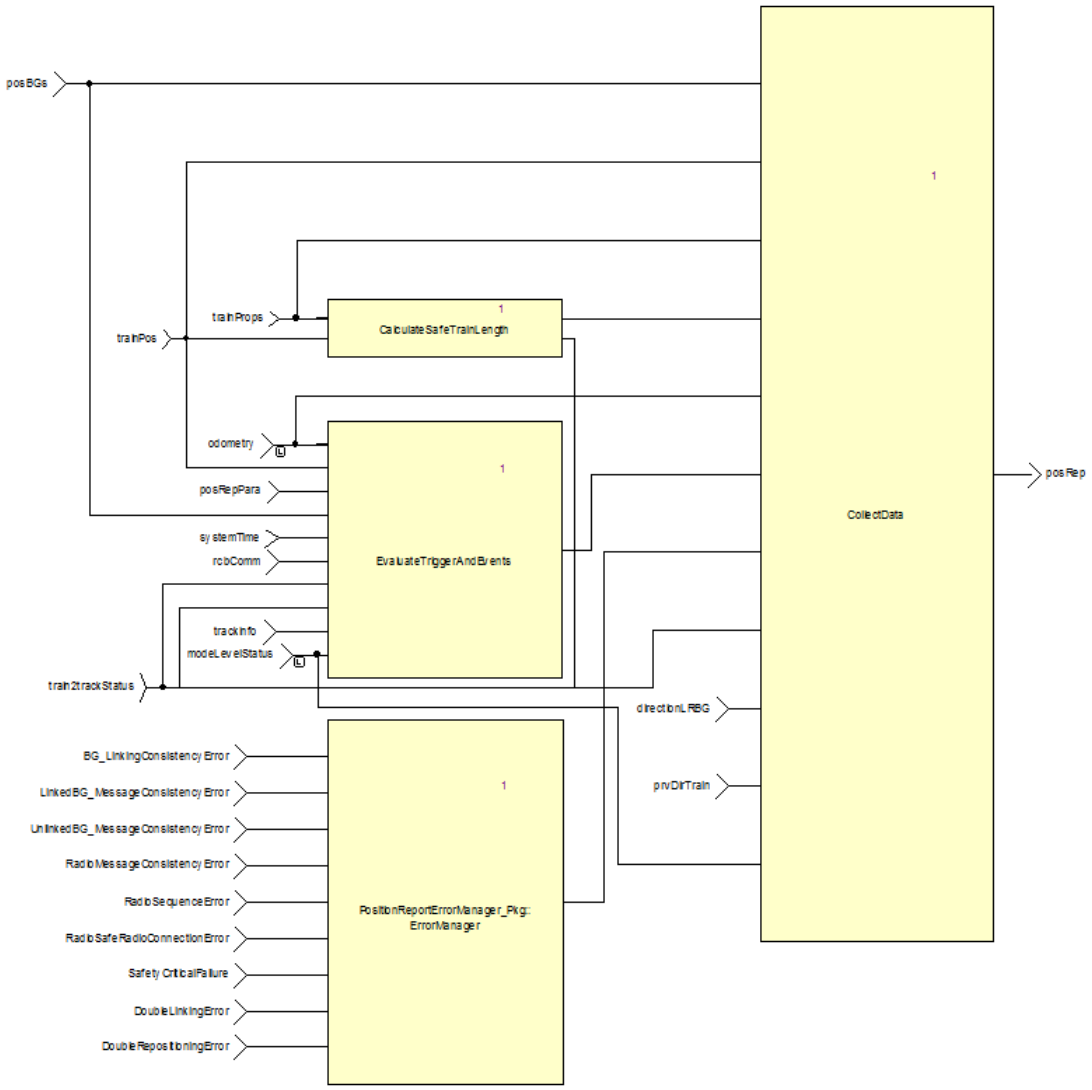



**Figure 9. Structure of component ProvidePositionReport**

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 four operators, as shown in Figure 9, which are explained below.

**CalculateSafeTrainLength** Calculates the the safeTrainLength and the MinSafeRearEnd according to [**?** , Chapter 3.6.5.2.4/5].
`safeTrainLength = absolute(EstimatedFrontEndPosition - MinSafeRearEnd)`, where `MinSafeRearEnd = 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 [**?** , Chapter 3.6.5.1.4]).

**ErrorManager** Takes a boolean flag for each possible error that has been occurred and outputs the respective error using type `M_ERROR`

**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 [**?** , 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 sent 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.
7. The *ErrorHandler* is currently restricted to deal with a single error. As a consequence, as for each error reported a report has to be sent to the RBC, the number of reports is limited to one.

#### Open Issues

1. The specification requires to store the last eight balise groups for which a position report has been sent (see [**?** , Chapter 3.6.2.2.2.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 [**?** , Chapter 3.4.2.3.3.6]. Moreover, such a coordination system can be invalid and then has to be rejected (see [**?** , 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.

### 6.2.4.3 Macrofunction x in Manage Track Data

#### Reference to the SRS or other Requirements (or other requirements)

#### Short descriptoiin of the functionality

#### Interface

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

#### 6.2.4.4 Macrofunction x in Manage Track Data

**Reference to the SRS or other Requirements (or other requirements)**

**Short descriptoiin of the functionality**

**Interface**

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

#### 6.2.4.5 Macrofunction x in Manage Track Data

**Reference to the SRS or other Requirements (or other requirements)**

**Short descriptoiin of the functionality**

**Interface**

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

#### 6.2.4.6 Macrofunction x in Manage Track Data

**Reference to the SRS or other Requirements (or other requirements)**

**Short descriptoiin of the functionality**

**Interface**

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

### 6.2.5 Manage Data

#### 6.2.5.1 Macrofunction x in Manage Data

**Reference to the SRS or other Requirements (or other requirements)**

**Short descriptoiin of the functionality**

**Interface**

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

#### 6.2.5.2 Macrofunction x in Manage Data

**Reference to the SRS or other Requirements (or other requirements)**

**Short descriptoiin of the functionality**

**Interface**

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

#### 6.2.5.3 Macrofunction x in Manage Data

**Reference to the SRS or other Requirements (or other requirements)**

**Short descriptoiin of the functionality**

**Interface**

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

#### 6.2.5.4 Macrofunction x in Manage Data

**Reference to the SRS or other Requirements (or other requirements)**

**Short descriptoiin of the functionality**

**Interface**

**Functional Design Description**

**Refernce to the Scade Model**

only in special case or link to the Scade model

#### 6.2.6 Manage Outputs

##### 6.2.6.1 Macrofunction x in Manage Outputs

Reference to the SRS or other Requirements (or other requirements)

Short descriptoiin of the functionality

Interface

Functional Design Description

Refernce to the Scade Model

only in special case or link to the Scade model

##### 6.2.6.2 Macrofunction x in Manage Outputs

Reference to the SRS or other Requirements (or other requirements)

Short descriptoiin of the functionality

Interface

Functional Design Description

Refernce to the Scade Model

only in special case or link to the Scade model

##### 6.2.6.3 Macrofunction x in Manage Outputs

Reference to the SRS or other Requirements (or other requirements)

Short descriptoiin of the functionality

Interface

Functional Design Description

Refernce to the Scade Model

only in special case or link to the Scade model

##### 6.2.6.4 Macrofunction x in Manage Outputs

Reference to the SRS or other Requirements (or other requirements)

Short descriptoiin of the functionality

#### Interface

#### Functional Design Description

#### Refernce to the Scade Model

**only in special case or link to the Scade model**

### 6.2.7 Mode and Level

The "Management of Modes and Levels" function is mainly described in chapter 4 and 5 of [**?** ]. Modes and levels define the status of the ETCS regarding on-board functional status and track infrastructure.

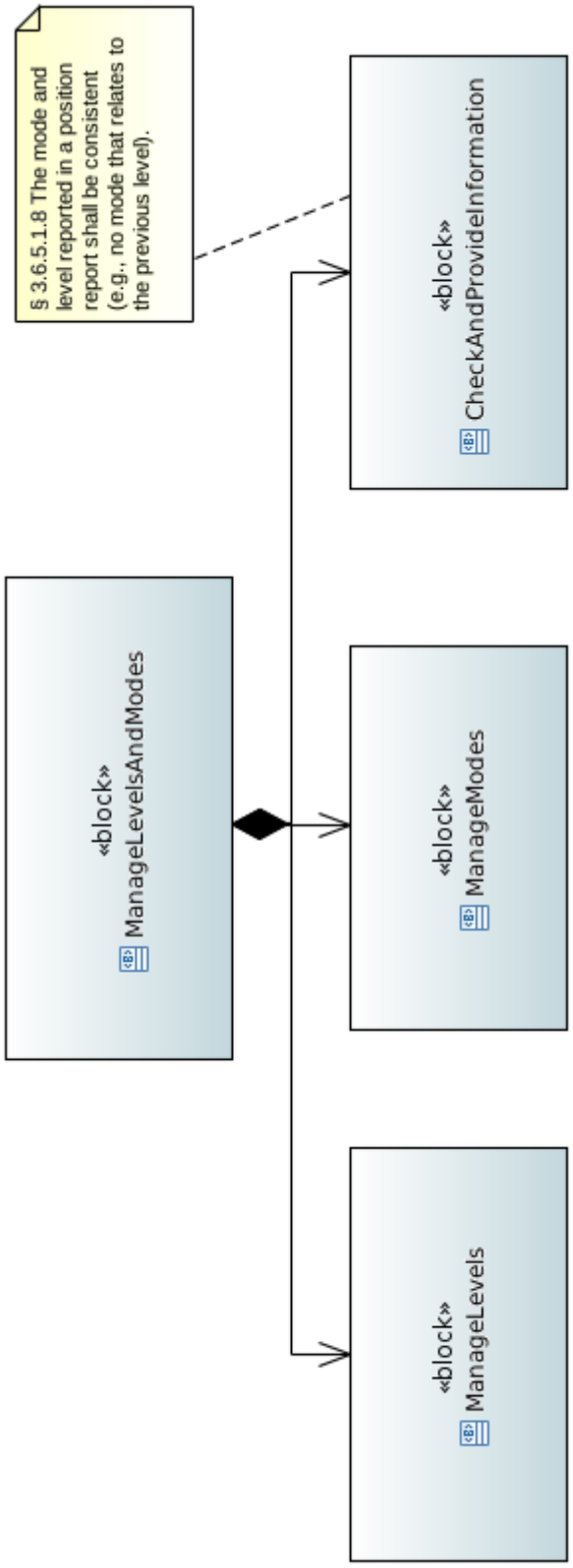



**Figure 10. High level Architecture**

#### 6.2.7.1 Function Level Management

**Reference to the SRS or other Requirements**

see [**?** ] section 5.10

**Short description of the functionality**

The level management subsystem receives level transition order tables and selects the order with the highest probability. It stores the information about the selected transition order and transits to the requested level once the train passes the location of the level transition.

If required, the driver is asked to acknowledge the transition, in case of no acknowledge or if conditions for the level transition are not fulfilled, the train gets tripped.

**Interface**

The interface consists of the following inputs:

- *conditional transitions:* a priority table containing the conditional level transition orders (from paquet 46)
- *level transition priority table:* a priority table containing the (non-conditional) level transition orders (from paquet 41)
- *train standstill:* a Boolean value indicating whether the train is at standstill (from odometry)
- *driver level transition:* a level transition order selected by the driver (from DMI)
- *ERTMS capabilities:* the ERTMS capabilities of the track
- *getAck:* Boolean input that signals the acknowledgment of the driver (from DMI)
- *resetIdle:* Boolean input to reset without acknowledge
- *currentDistance:* the current position of the train given with the same reference as the position of the level transition order (train position , from localisation)
- *ackDistance:* the maximal distance for driver acknowledge after the level transition (from paquet 41)
- *immediateAck:* a Boolean that signals that an immediate acknowledge is required
- *received L2 L3 MA:* a Boolean that indicates that a level 2 or level 3 movement authority for the track behind the level transition has been received (from paquet 15)
- *received L1 MA:* a Boolean that indicates that a level 1 movement authority for the track behind the level transition has been received (from paquet 12)
- *received target speed:* a Boolean indicating that a target speed for the track behind the level transition has been received (from paquet 27) ?

and the following outputs:

- *next level:* the next level after this computation cycle
- *Trip train:* a Boolean indicating whether the train should be tripped
- *previous level:* the previous level before this computation cycle
- *needsAckFromDriver:* a Boolean that indicates whether an acknowledgment from the driver is necessary

#### Functional Design Description

On the most abstract level the design consists of the *manage_priorities* function which takes the level transition order priority tables as inputs and computes the highest priority transition.

This transition order is the fed to the *computeLevelTransitions* operator. This operator consists of three main parts. The *ComputeTransitionConditions* operator that emits the fulfilled conditions to change from a given level to a new level, the *LevelStateMachine* that stores the current level and takes the computed change conditions as input for possible level transitions and finally the *driverAck* operator which contains a state machine that stores the information whether the system is currently waiting for a driver acknowledge and emits the train trip information if necessary.

#### Reference to the Scade Model

The Scade model is available on github: `https://github.com/openETCS/modeling/tree/master/openETCSArchitectureAndDesign/WorkGroups/Group3/SCADE/LevelManagement/`

### 6.2.7.2 Function Mode Management

#### Reference to the SRS or other Requirements

see [**?** ] sections 4.4, 4.6, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.11, 5.12, 5.13, 5.19

#### Short description of the functionality

This function is in charge of the computation of new mode to apply according to conditions from inputs (track information, driver interactions, train data,...) and other functions.

#### Interface

The inputs are the following:

- *Cab* identification of the current cabin (A or B)
- *Continue_shunting_Function_Active*: boolean to describe the activation state of the shunting function
- *Current_Level*: outputs of the Level management function
- *Data_From_DMI*: set of data received from the driver via the DMI interface, indeed:
  - *Ack_LS : bool* Driver acknoledges LS mode
  - *Ack_OS : bool*

  - *Ack_RV : bool*
  - *Ack_SH : bool*
  - *Ack_SN : bool*
  - *Ack_SR : bool*
  - *Ack_TR : bool*
  - *Ack_UN : bool*
  - *Req_Exit_SH : bool* driver selects exit of shunting
  - *Req_NL : bool* Driver requests NL mode
  - *Req_Override : bool* Driver requests override function
  - *Req_SH : bool* driver requests SH mode
  - *Req_Start : bool* Driver requests start of mission
  - *ETCS_Isolated: bool*: isolation status of the ETCS

- *Data_From_Localisstion*: set of data received from the function in charge of localistion of the train, indeed:

  - *BG_In_List_Expected_BG_In_SR : bool*: the identity of the overpass balise group is in the list of expected balises related to SR mode (from SR to trip mode condition 36)
  - *BG_In_List_Expected_BG_In_SH : bool*: the identity of the overpass balise group is in the list of expected balises related to SH mode (from SH to trip mode condition 52)
  - *Linked_BG_In_Wrong_Direction : bool* balise group contained in the linking information is passed in the unexpected direction (from FS, LS, OS to trip mode condition 66) *Localisaion function ?*
  - *Train_Position*: output provided by function in charge of computation of train possition (type TrainPosition_Types_Pck::trainPosition_T)
  - *Train_Speed : Obu_BasicTypes_Pkg::Speed_T* provided by odometry function
  - *Train_Standstill : bool* provided by odometry function

- *Data_From_Speed_and_Supervision*: set of data received from the function in charge of speed and supervision management, indeed:

  - *Estim_front_End_overpass_SR_Dist : bool*: the train overpass the SR distance with its estimated front end (from SR to trip mode condition 42)
  - *Estim_Front_End_Rear_SSP : bool*: estimated front end is rear of the start location of either SSP or gradient profile stored on-board (from FS, LS, OS to trip mode condition 69)
  - *Override_Function_Active*: boolean to indicate the state of the activation function
  - *EOA_Antenna_Overpass : bool*: the train overpasses the EOA with min safe antenna position Level 1 (from FS, LS, OS to trip mode condition 12)
  - *EOA_Front_End : bool* the train overpasses the EOA with min safe front end, Level 2 or 3 (from FS, LS, OS to trip mode condition 16)
  - *Train_Speed_Under_Overide_Limit : bool* supervision when override function is active (to SR mode condition 37)

- *Data_From_TIU : TIU_Types_Pkg::Message_Train_Interface_to_EVC_T*:message provided by TIU interface

- *Data_From_Track*: set of data received from track side (via RBC or Balises telegram), indeed:

  - *MA_SSP_Gradiant_Available : bool* MA, SSP and gradient have been received, checked and stored on-board from paquet 12, 15, 21 and 27 or message 3 or 33
  - *Mode_Profile_On_Board : Level_And_Mode_Types_Pkg::T_Mode_Profile* from packet 80
  - *Shunting_granted_By_RBC : bool* from message 27 and 28
  - *Trip_Order_Given_By_Balise : bool*
  - *List_Bg_Related_To_SR_Empty : bool* from packet 63
  - *Stop_If_In_shunting : bool* from packet 135
  - *Stop_If_In_SR : bool* from packet 137
  - *Error_BG_System_Version : bool*
  - *Linking_Reaction_To_Trip : bool*
  - *RBC_Ack_TR_EB_Revocked : bool* from message 6
  - *RBC_Authorized_SR : bool* from message 2
  - *Reversing_Data : Level_And_Mode_Types_Pkg::T_Reversing_Data* from packet 138/ 139
  - *T_NVCONTACT_Overpass : bool* Maximal time without new safe message overpass
  - *Emergency_Stop_Message_Received*: boolean to describe the reception of Emergency Stop message from message 15 or 16

- *Failure_Occured*: boolean to indicate safety failure occurence
- *Interface_To_National_System*: boolean to indicate existance of an interface to a national system
- *National_Trip_Order*: boolean to indicate reception of a trip order from a national system
- *OnBoard_Powered*: boolean to indicate the poxering state of the system
- *Stop_Shunting_Stored*: boolean to store the information in regards of shunting function
- *Valid_Train_Data_Stored*: boolean to indication train data are available and valid.

The outputs are the following:

- *currentMode* the new computed mode (typeis Level_And_Mode_Types_Pkg::T_Mode, default value is Level_And_Mode_Types_Pkg::SB )
- *EB_Requested* boolean to request triggering of emergency brake
- *Service_Brake_Command* boolean to request command of service brake
- *Data_To_DMI*: set of data provided to the DMI Level_And_Mode_Types_Pkg::T_Data_To_DMI :
  - *Ack_LS : bool* Driver acknoledges LS mode
  - *Ack_OS : bool*
  - *Ack_RV : bool*
  - *Ack_SH : bool*
  - *Ack_SN : bool*

  - *Ack_SR : bool*
  - *Ack_TR : bool*
  - *Ack_UN : bool*
  - *Req_Exit_SH : bool* driver selects exit of shunting
  - *Req_NL : bool* Driver requests NL mode
  - *Req_Override : bool* Driver requests override function
  - *Req_SH : bool* driver requests SH mode
  - *Req_Start : bool* Driver requests start of mission
  - *ETCS_Isolated: bool*: isolation status of the ETCS

- *Data_To_BG_Management*: set of date to trackside Level_And_Mode_Types_Pkg::T_Data_To_BG_Management :

  - *EoM_Procedure_req : bool* request of end of mission procedure indeed end of the communication session for message 150
  - *Clean_BG_List_SH_Area : bool* request to clean the BG list when entering an SH area §5.6.2
  - *MA_Req : bool* for message 132
  - *Req_for_SH_from_driver : bool* for message 130

### Functional Design Description

Three subfunctions are defined:

**Inputs** proceeds to inputs check and preparation.

**ComputeModesCondition** performs all specific procedure linked to mode management and defined in [**?** ] sections 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.11, 5.12, 5.13, 5.19 and specifies the conditions to define a mode transition according condition table of section 4.6.3 of [**?** ]

**SwitchModes** performs the mode selection according the conditions and priorities defined in transition table section 4.6.2 of [**?** ]

**Outputs** prepares paquet of outputs.

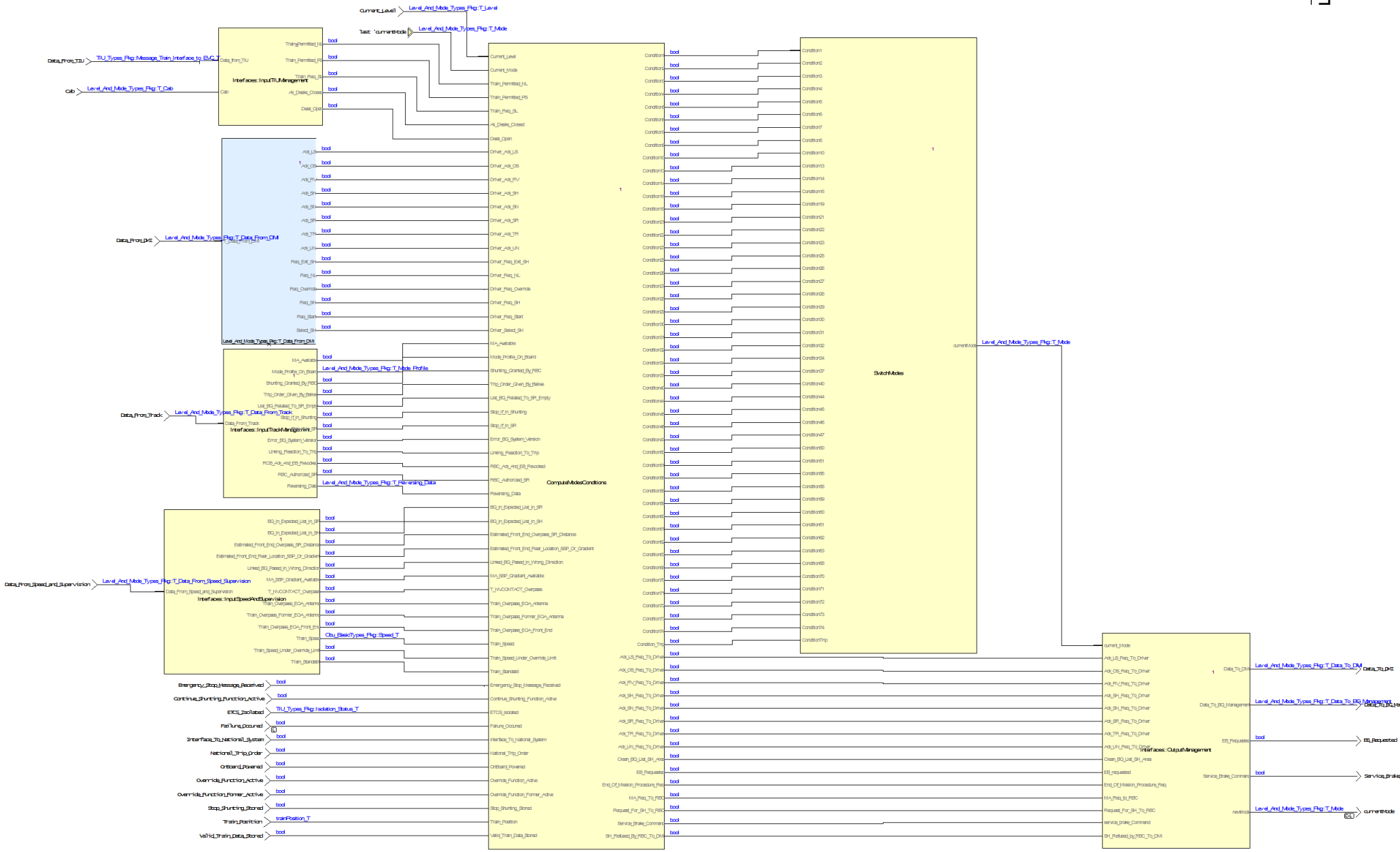



**Figure 11. Modes subfubction architecture**

#### Reference to the Scade Model

The Scade model is available on github: `https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/ManageLevelsAndModes/Modes`

#### 6.2.7.3 Function Check and Provide Level and Mode

#### Reference to the SRS or other Requirements

see [**?** ] section 3.6.5

#### Short description of the functionality

checks compatibility between mode and level and provides outputs

#### Interface

*To design*

#### Functional Design Description

*To design*

#### Reference to the Scade Model

*To design*

### 6.2.8 Manage RBC Procedure

#### 6.2.8.1 Macrofunction x in Manage RBC Procedure

#### Reference to the SRS or other Requirements (or other requirements)

#### Short descriptoiin of the functionality

#### Interface

#### Functional Design Description

#### Refernce to the Scade Model

**only in special case or link to the Scade model**

#### 6.2.8.2 Macrofunction x in Manage RBC Procedure

#### Reference to the SRS or other Requirements (or other requirements)

#### Short descriptoiin of the functionality

#### Interface

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

#### 6.2.8.3 Macrofunction x in Manage RBC Procedure

**Reference to the SRS or other Requirements (or other requirements)**

**Short descriptoiin of the functionality**

**Interface**

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

#### 6.2.8.4 Macrofunction x in Manage RBC Procedure

**Reference to the SRS or other Requirements (or other requirements)**

**Short descriptoiin of the functionality**

**Interface**

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

### 6.2.9 Manage DMI Procedure

#### 6.2.9.1 Macrofunction x in Manage DMI Procedure

**Reference to the SRS or other Requirements (or other requirements)**

**Short descriptoiin of the functionality**

**Interface**

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

#### 6.2.9.2 Macrofunction x in Manage DMI Procedure

**Reference to the SRS or other Requirements (or other requirements)**

**Short descriptoiin of the functionality**

**Interface**

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

#### 6.2.9.3 Macrofunction x in Manage DMI Procedure

**Reference to the SRS or other Requirements (or other requirements)**

**Short descriptoiin of the functionality**

**Interface**

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

#### 6.2.9.4 Macrofunction x in Manage DMI Procedure

**Reference to the SRS or other Requirements (or other requirements)**

**Short descriptoiin of the functionality**

**Interface**

**Functional Design Description**

**Refernce to the Scade Model**

**only in special case or link to the Scade model**

## 6.3 F3: Measure Train Movement

## 6.4 F4: Manage Radio Communication

### 6.4.1 Manage Radio Communication

#### 6.4.1.1 Management of Radio Communication (*MoRC*)

**Reference to the SRS**

The management of radio communication is specified in Subset-026, chap. 3.5.

### Short description of the functionality

The management of radio communication *MoRC* implements the on board management part of a single communication session with the track, i.e. a single RBC. It controls the establishing, maintaining and termination process of a radio communication session and steers the underlying communication safety layer and the mobile device. Those and the data transfer itself are not part of the function.

### Interface

**Inputs** The MoRC function takes as inputs datagrams received from track, OBU internal phases and status information and configuration data:

- Datagrams received from track (*inMessage*):
  - Packet 42 (session management) received from balise group or RBC
  - Packet 45 (radio network registration) received from balise group or RBC
  - Message 32 (RBC/RIU System Version) received from RBC: *MoRC* only needs to know if the system version received from track side is supported by the OBU.
  - Message 38 (initiation of a communication session) received from RBC
  - Message 39 (acknowledgement of termination of a communication session)
- *obuEventsAndPhases*: information about OBU internal events and OBU internal phases:
  - *atPowerDown*
  - *atPowerUp*
  - *atStartOfMission*
  - *startOfMissionProcedureIsGoingOn*
  - *startOfMissionProcedureCompleted*
  - *trainIsRejectedByRBC_duringStartOfMission*
  - *endOfMissionIsExecuted*
  - *driverClosesTheDeskduringStartOfMission*
  - *driverHasManuallyChangedLevel*
  - *afterDriverEntryOfANewRadioNetworkID*
  - *triggerDecisionThatNoRadioNetworkIDAvailable*
  - *isPartOfAnOngoingStartOfMissionProcedure*
  - *trainPassesALevelTransitionBorder*
  - *trainPassesA_RBC_RBC_border_WithItsFrontEnd*
  - *trainExitedFromAnRBCArea*
  - *modeChangeHasToBeReportedToRBC*
  - *trainFrontInsideInAnAnnouncedRadioHole*
  - *trainFrontReachesEndOfAnnouncedRadioHole*
  - *OBU_hasToEstablishANewSession*
  - *isInCommunicationSessionWithAnRIU*

  - *errorConditionRequiringTerminationDetected*

- Current OBU internal states:
  - *currentTime*: current OBU system time
  - *t_train*: current trainborne clock (T_TRAIN) as specified in Subset-026, chap. 7
  - *mode*: current OBU mode
  - *level*: current OBU level
- *statusOfMobile*: status of the associated mobile device
- configuration parameters:
  - *onboardPhoneNumbers* (NID_RADIO)
  - *radioNetworkIDs*: Identities of radio networks (NID_MN): default, memorized or from driver
  - *nid_engine*: Onboard ETCS identity (NID_ENGINE)
  - *connectionStatusTimerInterval*: Connection status timer period

**Outputs** MoRC generates a couple of outputs:

- *MessageToRBC*: messages to be sent to the RBC:
  - Message 155 (initiation of a communication session)
  - Message 156 (termination of a communication session)
  - Message 159 (session established)
  - Message 154 (no compatible version supported)
- Action triggers:
  - *sendAPositionReport*: triggers a position report to be sent to the RBC
  - *memorizeTheLastRadioNetworkID*: triggers to store the last radio network ID for later use
  - *orderTheRegistrationOfItsConnectedMobiles*
  - *rejectOrderToContactRBC_or_RIU*
  - *InformTheDriverThatNoConnectionWasSetup*
  - *requestTheSetupOfASafeRadioConnection*: initiate the setup of a safe radio connection
  - *requestReleaseOfSafeRadioConnectionWithTrackside*: initiate the release of a safe radio connection
  - *ignoreMessagesFromRBC_except_m39_AckOfTerminationOfCommunicationSession*
  - *sessionSuccessfullyEstablished*
- *cmdsToMobile*: control commands to the mobile device
- Status information:
  - *sessionStatus*: current session status
  - *mobileSWStatus*: connection status
  - *currentRadioNetworkID*: current radio network ID

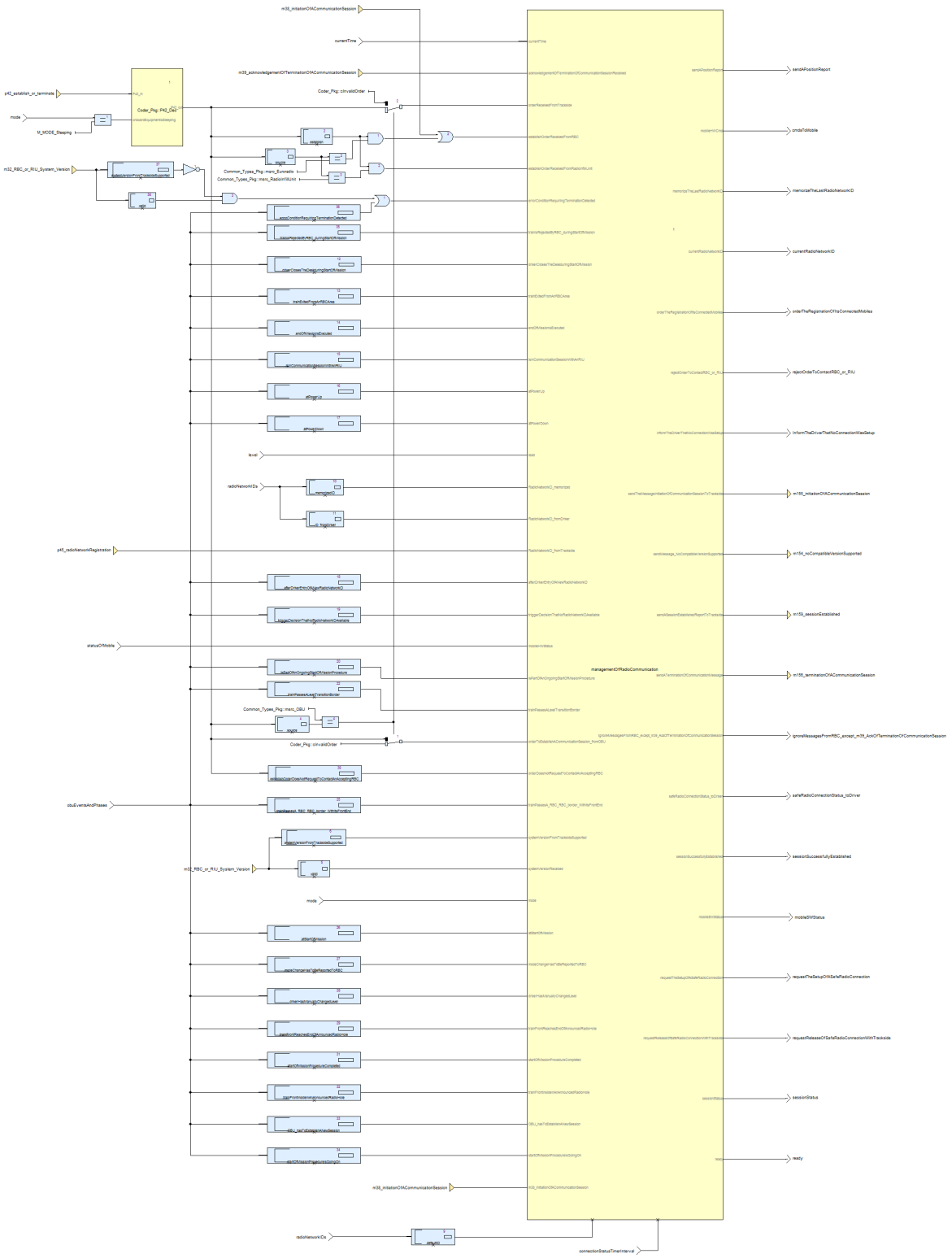


**Figure 12. Main function of *MoRC***

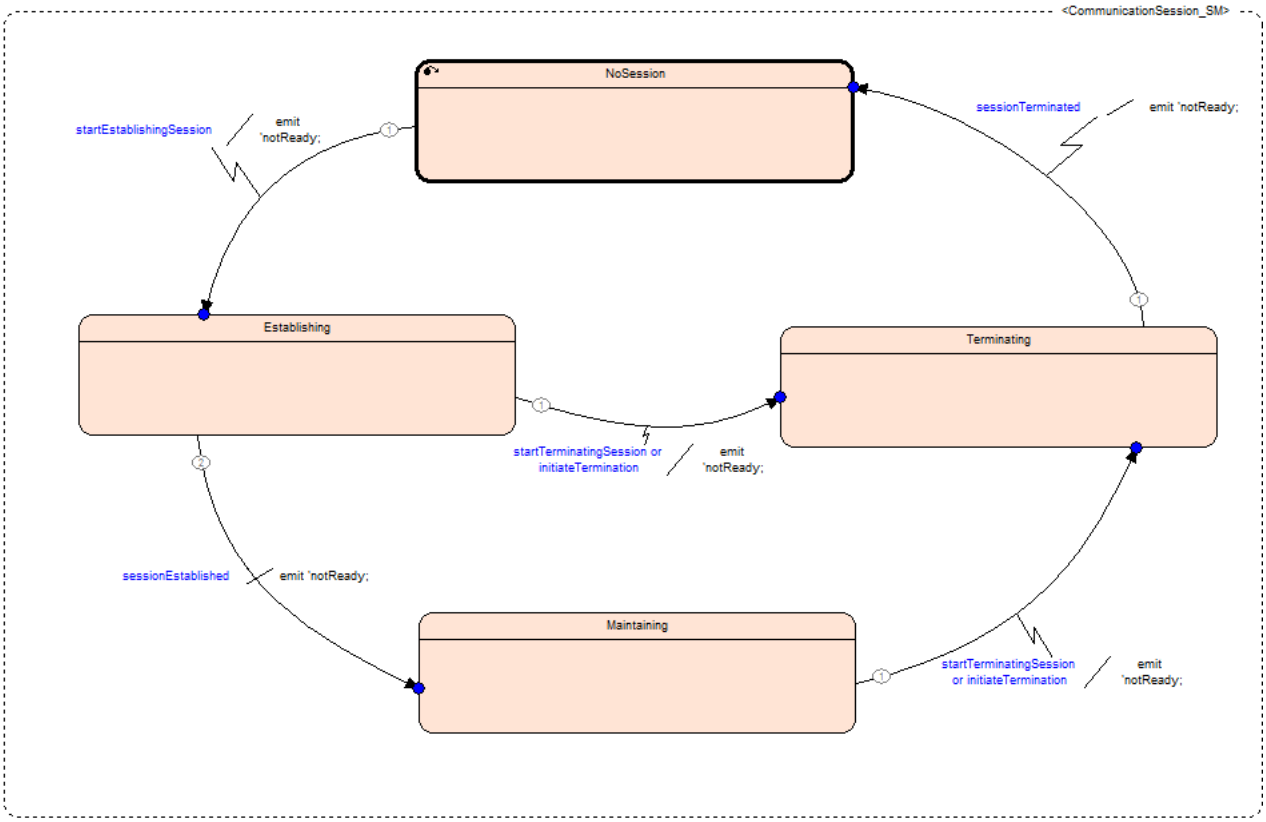



**Figure 13. Implementation of session states**

#### Functional Design Description

The kernel function of the *MoRC* component is *managementOfRadioCommunication* (figure ???). The implementation is kept close to the prose of Subset-026, chap. 3.5. Since chap. 3.5 rarely refers to terms, variable types, packets and messages of the ETCS language as specified in Subset-026, chap. 7 and 8, *managementOfRadioCommunication* does neither.

To be capable of being integrated with other OBU software components, *MoRC* had to be wrapped with a transformer between the ETCS and the "chap. 3.5" language. This is the purpose of the main function of *MoRC*, *MoRC_Main*.

The function *managementOfRadioCommunication* implements the session states establishing, maintaining and termination as described in Subset-026, chap. 3.5. A SCADE state machine reflects this state model (figure ???) accurately. Within each of the states, the activities needed as long as the state is active, are performed. When there is no communication session (state *NoSession*) currently, the state machine waits for events that initiate a session (subfunction *initiate_a_Session*). When the appropriate conditions are fulfilled, the state machine moves to the *Establishing* state. Here in, it runs through the sequence required fore establishing a session (subfunction *establish_a_Session*. Dependent on the results, the state machine changes over to the *Maintaining* or *Terminating* state. While in *Maintaining*, the communication connection is monitored. When an event triggering the session termination occurs, the state machine switches to the state *Terminating* with the subfunction *terminating_a_CommunicationSession* and performs the session termination sequence.

In parallel to the main state machine, *managementOfRadioCommunication* monitors all the time whether the session has to be terminated (subfunction *initiateTerminatingASession*) or if the session has the be terminated and subsequently established (subfunction *terminateAndEstablishSession*). *registeringToTheRadioNetwork* is responsible for connection to the radio network. *safeRadioConnectionIndication* controls the radio connection indication for the driver.

#### Reference to the Scade Model

The MoRC SCADE model resides at `https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/Radio/MoRC` .

**only in special case or link to the Scade model**

## 6.5 F5: Manage JRU

## 6.6 F6: DMI Controller

### 6.6.1 DMI Controller

#### Reference to the SRS or other Requirements (or other requirements)

§ 4.7, § 5.4, § 3.12.3

**§ F I1: Packet 72:** this packet will inclsude the information received from the track for the entered section with the display and display conditions depending from:
- train status Level ETCS

- train status modus ETCS
- distance
- timeperiod
- acknowledgement from the driver on the DMI display

The displaying of this information can be ordinary or multiple (combination of several conditions).

**§ F I2: Missing acknowledgement:** for the case that a acknowledgement from the driver in combination with another condition is expected at the end, and the acknowledge is missing, the ETCS on-board unit will trigger the service brake.(missing acknowledgement).

**§ F I3: Packet 76:** this package contains information for sending fixed messages.

### Short descriptoiin of the functionality

The DMI controller interact with the DMI display and is responsible for alls procedures between the DMI display and Driver. Furthermore the DMI controller will interact with the DMI Management to compute the receive information (e.g. driver number request, ...) and send, if necessary, data or reports ot the DMI Management (acknowledge, text messages, ..).

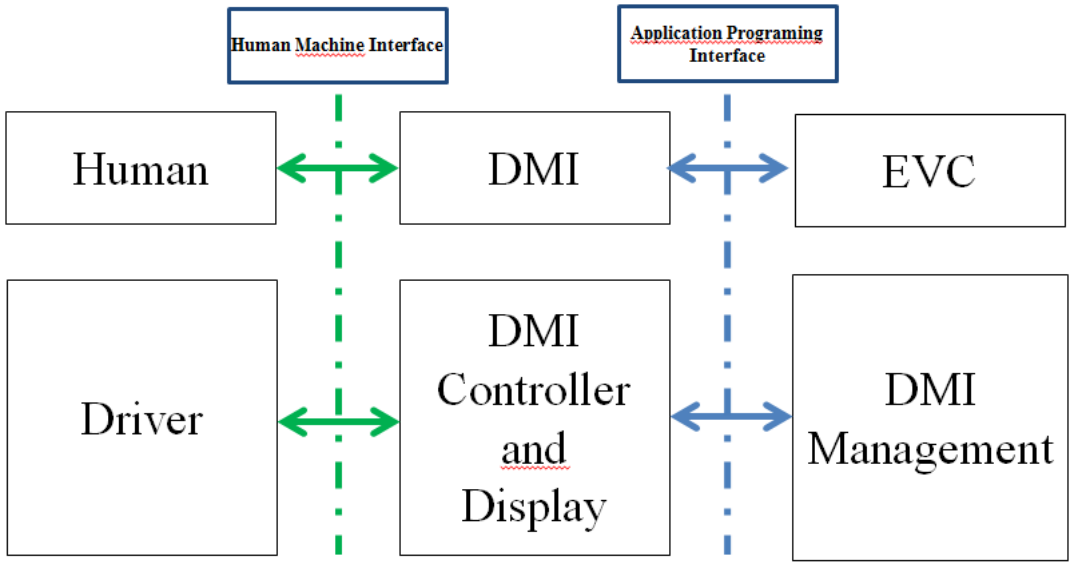



**Figure 14. DMI Interfaces**

### Interface

### Functional Design Description

### Refernce to the Scade Model

**only in special case or link to the Scade model**