OETCS/WP5/M5.X

openETCS

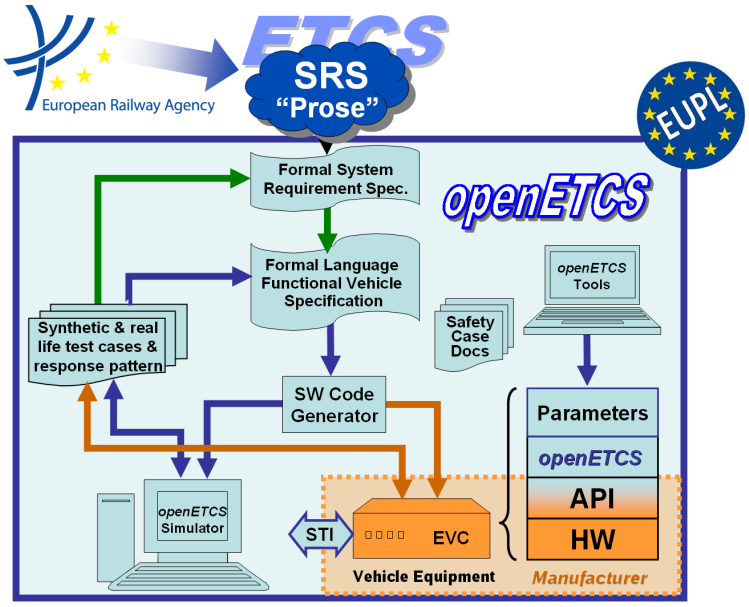
ITEA2 Project
Call 6 11025
2012 – 2015

Work-Package 5: “Demonstrator”

ERSA Demonstrator 1st release API

Nicolas Van Landeghem, Didier Weckmann, Alexis Julin,
Stéphane Chenevoy

August 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 5: “Demonstrator”

OETCS/WP5/M5.X
August 2014

ERSA Demonstrator 1st release API

Document approbation

Lead author:	Technical assessor:	Quality assessor:	Project lead:
location / date	location / date	location / date	location / date
signature Nicolas Van Landeghem (ERSA)	signature Nicolas Van Landeghem (ERSA)	signature Ainhoa Gracia (SQS)	signature Klaus-Rüdiger Hase (DB Netz)

Nicolas Van Landeghem, Didier Weckmann, Alexis Julin, Stéphane Chenevoy

ERSA
5 Rue Maurice Blin
67500 Haguenau, France

Description of work

Prepared for openETCS@ITEA2 Project

Modification History

Version	Section	Modification / Description	Author
1.0	All Parts	Document creation	Nicolas Van Landeghem

Table of Contents

1 Introduction

The purpose of this document is to describe ERSA's API of its EVC simulator: its breakdown into modules, the exchanged data and the interfaces with external devices. The EVC simulator behaves according to the SRS Class 1 version 3.3.0.

2 API content

The API definition is split between its public types, structures, methods and members. Methods are divided into families, e.g. check consistency and basic configuration, control EVC simulation, set EVC configuration data, store data, handle train data, handle internal events and retrieve internal data.

2.1 Function prototypes

Initialise the EVC simulator: creation of EVC data structure, initialisation of interfaces
[in] ulLogId, key used as prefix for log files
return 0 on success
int32_t Init(uint32_t ulLogId);

Start the EVC simulator modules
return 0 on success
int32_t Start_processes(void);

Start the EVC simulation
return 0 on success
int32_t Run(void);

Stop the EVC simulation
return 0 on success
int32_t Stop(void);

Get port (socket) used for STM communication (between STM module and STM controller). If SetSTMCommunicationPort() has not been called previously, the server will look for an available socket in a certain range. May stall if the stm server is being initialized.
return Port (socket) number to be used for STM communication
int32_t WaitSTMCommunicationPort(void);

Modify the EVC configuration. It should be call just after constructor
[in] bSet, indicate if the config should be set or reset
[in] Config, flag to identify the configuration
void Modify_EVC_Configuration(bool bSet, eConfigData Config);

Get current train speed (km/h)
return train speed in km/h
double GetTrainSpeed(void);

Get EOA location
return EOA location or 0 if not applicable
t_distance GetEstimatedFrontLoc(void);

Get current on-board ETCS mode
return train mode as M_MODE
eTrainMode GetCurrentMode(void);

Get current on-board ETCS level
return train level

eLevel GetCurrentLevel(void);

Get EOA location
return EOA location or 0 if not applicable
t_distance GetEoaLocation(void);

Get EOA speed (km/h)
return EOA speed (km/h) or 0 if not applicable
double GetEoaSpeed(void);

Get messages from UI
[out] pointer to UI message
[out] pointer to type of message (balise, loop, radio, other)
return true if message is valid, 0 otherwise
bool GetIhmMessages(char * szMessage, eMSG_TYPE * type);

Get driver action produced on DMI
[out] location where action was sent
[out] timestamp when action was sent
[out] action requested by driver
[out] list of data if relevant (driver id, etc.)
return true if message is valid, 0 otherwise
bool GetDriverAction(t_distance & rdLocation, t_time & rdTimeStamp, eDriverActionInfo & rDriverAction, SVarDataList & rVarDataList);

Get track condition profile
[out] list of track conditions information
[in/out] counter used to test changes
return true if list is available and has changed (according to counter value)
bool GetTCProfile(STCInfoList & rTCInfoList, int32_t & rlCounter);

2.2 Types

Table 1. Basic types

Data type	Variable name	Comment
Double	t_time	Time in s
Double	t_distance	Distance in m

2.3 Enumerations

Table 2. eConfigData

Enum member	Enum value	Comment
CFG_RADIO_INTERNAL_TIME_STAMP	0	Internal time stamping (not use T_TRAIN)
CFG_USE_JRU	1	Use JRU (generates JRU data file)
CFG_RECORD_TO_CSV_FILE	2	Record supervision data to CSV file
CFG_LOOP_WITH_SSCODE	3	Loop are received with spread spectrum code
CFG_BAL_WITH_ODO_STAMP	4	Balise are received with odo stamp
CFG_LOCAL_TIME_STAMP	5	Request local time stamp otherwise GMT time in log & JRU record
CFG_RECORDER_LOG_ADD_FULL _TIME_STAMP	6	add time stamp in EuroCabLog.dat like 2009-05-29/08:15:21.29
CONFIG_SIZE	7	Maximum number of config value

Table 3. eTrainMode, ETCS train mode

Enum member	Enum value	Comment
MODE_FULL	0	Full supervision
MODE_ON_SIGHT	1	On sight
MODE_STAFF_RESP	2	Staff responsible
MODE_SHUNTING	3	Shunting
MODE_UNFITTED	4	Unfitted
MODE_SLEEPING	5	Sleeping
MODE_STAND_BY	6	Stand by
MODE_TRIP	7	Trip
MODE_POST_TRIP	8	Post trip
MODE_SYST_FAILURE	9	System failure
MODE_ISOLATION	10	Isolation
MODE_NON_LEADING	11	Non leading
MODE_LIMITED_SUP	12	Limited supervision
MODE_STM_NATIONAL	13	STM national
MODE_REVERSING	14	Reversing
MODE_PASSIVE_SH	15	Passive shunting
MODE_UNKNOWN	16	Unknown
MODE_NO_POWER	17	No power
MODE_MAX_NB	18	Maximum number of mode value

Table 4. eLevel, ETCS level

Enum member	Enum value	Comment
LEVEL_0	0	Level 0
LEVEL_STM	1	Level STM
LEVEL_1	2	Level 1
LEVEL_2	3	Level 0
LEVEL_3	4	Level 3
LEVEL_UNKNOWN	5	Level unknown
NB_LEVEL	6	Number of different level

Table 5. eDriverActionInfo, DMI actions from driver

Enum member	Enum value	Comment
DMI_ACTION_DRIVER_ID	0	
DMI_ACTION_TRAIN_RUNNING_NUMBER	1	
DMI_ACTION_TRAIN_DATA	2	
DMI_ACTION_SR_DATA	3	
DMI_ACTION_LEVEL_0	4	
DMI_ACTION_LEVEL_1	5	
DMI_ACTION_LEVEL_2	6	
DMI_ACTION_LEVEL_3	7	
DMI_ACTION_LEVEL_STM	8	
DMI_ACTION_START_OF_MISSION	9	
DMI_ACTION_SLIPPERY_TRACK	10	
DMI_ACTION_NON_SLIPPERY_TRACK	11	
DMI_ACTION_NON_LEADING	12	
DMI_ACTION_NON_LEADING_EXIT	13	
DMI_ACTION_OVERRIDE_EOA	14	
DMI_ACTION_OVERRIDE_RS	15	
DMI_ACTION_SHUNTING	16	
DMI_ACTION_SHUNTING_EXIT	17	
DMI_ACTION_TAF_ACK	18	
DMI_ACTION_ACK_ICON	19	
DMI_ACTION_ACK_TEXT	20	
DMI_ACTION_ACK_BRAKE	21	

Table 6. eMSG_TYPE, type of data message transmitted to MMI

Enum member	Enum value	Comment
MSG_BALISE	0	Balise message
MSG_LOOP	1	Loop message
MSG_RADIO	2	Radio message
MSG_MMI	3	DMI message
MSG_INFO2	4	Informative message

2.4 Structures

Table 7. SVarDataList, contains information about a track condition

Member type	Member name	Comment
t_distance	dStartLocation	Start location of TC (m)
t_distance	dEndLocation	End location of TC (m)
eTCInfo	TCInfo	Type of track condition
SVarDataList	VarList	Data associated to the track condition

Table 8. SVarData, contains variable and its value (used to indicate driver data entry on DMI)

Member type	Member name	Comment
string	VarName	Name of data
string	VarValue	Value of data