ITEA 2

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

openETCS

openETCS - WP5 Demonstrator
Test WorkShop 7 & 8 Oct. 2014

supported by:

Federal Ministry of Education and Research

Liberté • Égalité • Fraternité
République Française
Ministère de l'Enseignement Supérieur et de la Recherche

Investissements d'avenir

Région de Bruxelles-Capitale

Gobierno de España
Ministerio de Ciencia e Innovación

openETCS – WP5 demonstrator

E.R.S.A.

Strasbourg, 7th & 8th Oct. 2014

Agenda

1. **Welcome to the WP5 TWS**
   - **Team presentation**
   - **Organisational topics**
2. **WP5 purpose**
   - **Reference implementation of an OBU**
   - **Architecture of OBU & Test Environment**
   - **Ways to use an OBU**
3. **Installation of the WP5 tools**
   - **Virtual Machines distribution**
   - **Redhat Packet Manager deployment**
   - **Licensing distribution and installation**

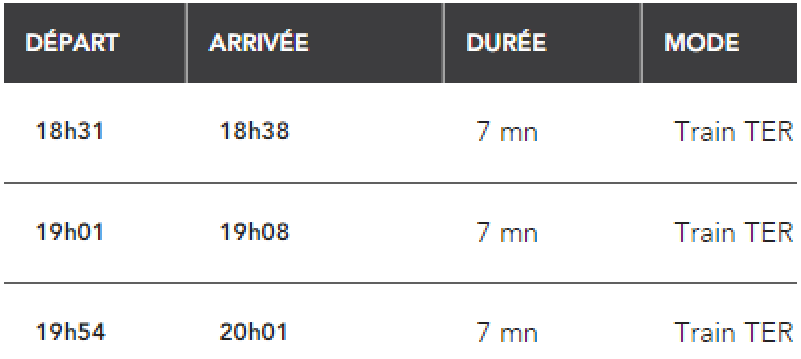
Agenda

4. M5.1 OBU simulator
   - Where it is
   - How it works
5. M5.3 Test Environment
   - Where it is
   - How it works
   - Scenarios: purpose and content
6. Writing a scenario file
7. Interface with M5.1: API to EVC
8. Feedback round

1. Welcome to the WP5 TWS

Team presentation:

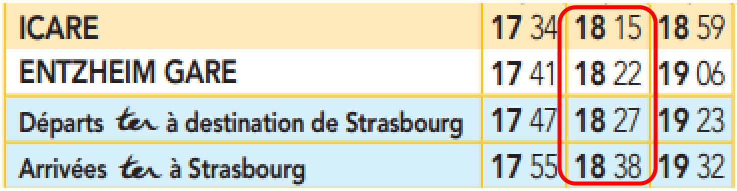
- Patrick Deutsch, ERSA Director
- Didier Weckmann, Software developer
- Alexis Julin, Software developer
- Nicolas Van Landeghem, ERSA project leader
- Who are you?



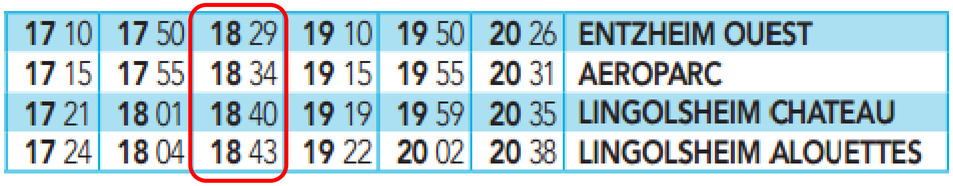
DÉPART	ARRIVÉE	DURÉE	MODE
18h31	18h38	7 mn	Train TER
19h01	19h08	7 mn	Train TER
19h54	20h01	7 mn	Train TER

Organisational topics:

- Lunch at 1:00 p.m.
- Restart at 1:45 p.m.
- End at 6:00 p.m.



ICARE	17 34	18 15	18 59
ENTZHEIM GARE	17 41	18 22	19 06
Départs ter à destination de Strasbourg	17 47	18 27	19 23
Arrivées ter à Strasbourg	17 55	18 38	19 32

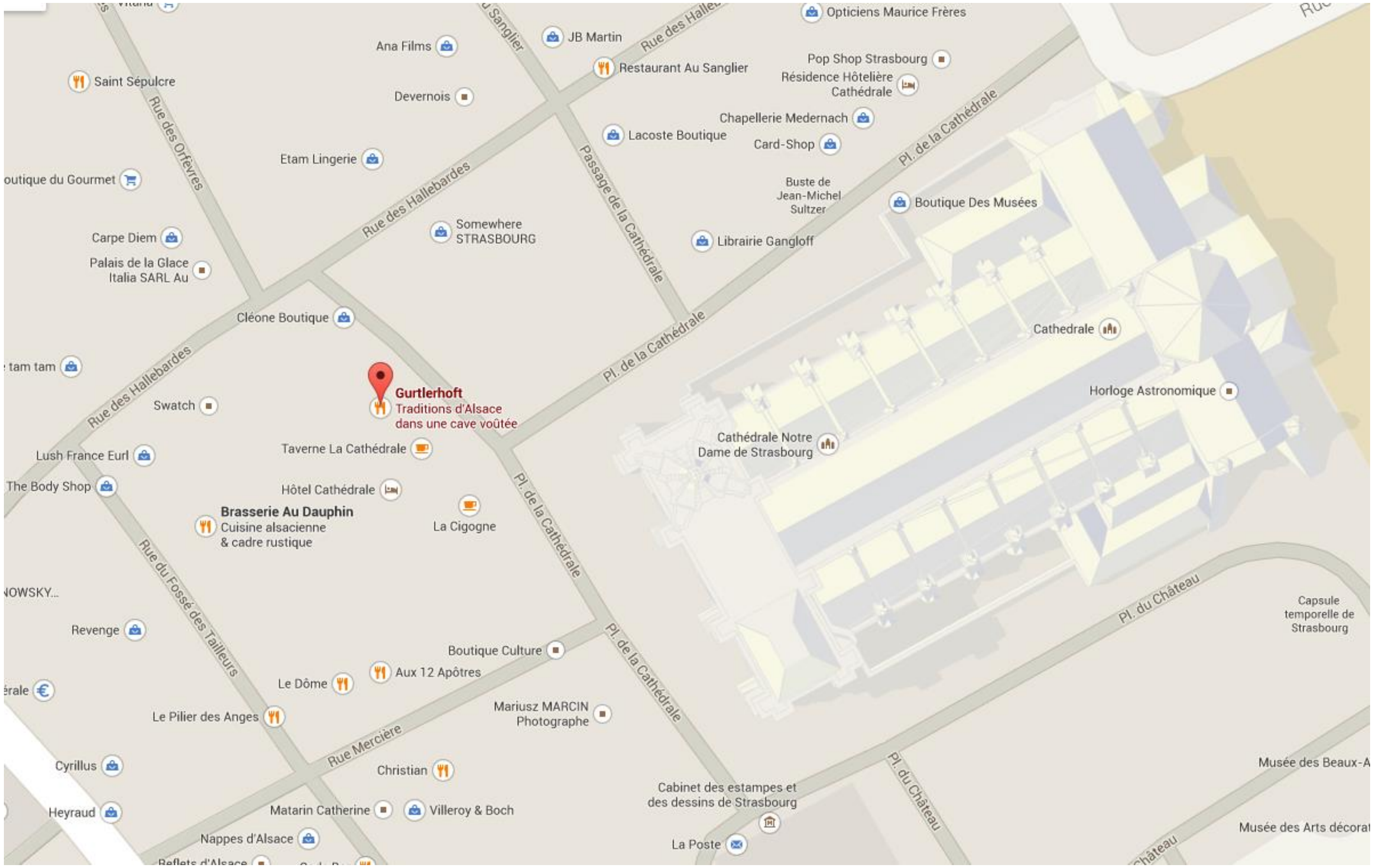


17 10	17 50	18 29	19 10	19 50	20 26	ENTZHEIM OUEST
17 15	17 55	18 34	19 15	19 55	20 31	AEROPARC
17 21	18 01	18 40	19 19	19 59	20 35	LINGOLSHEIM CHATEAU
17 24	18 04	18 43	19 22	20 02	20 38	LINGOLSHEIM ALOUETTES

1. Welcome to the WP5 TWS

Organisational topics:

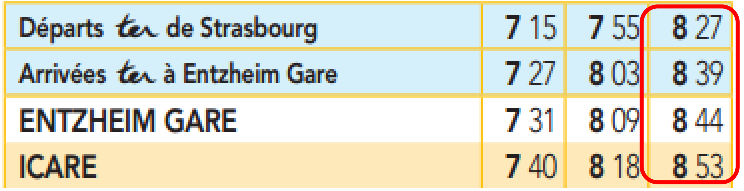
- Dinner at 7:45 p.m.
- Gurtlerhoft
  13 place de la
  Cathédrale
  Strasbourg



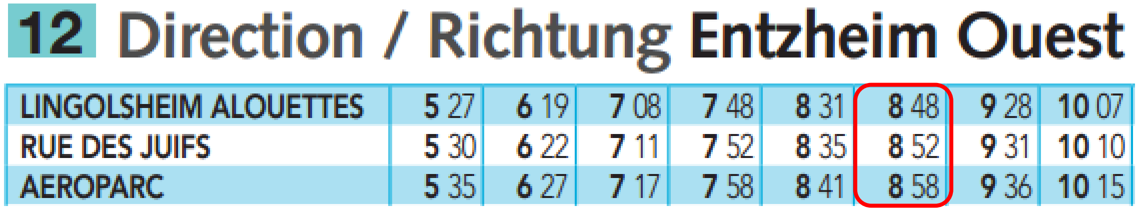
1. Welcome to the WP5 TWS

Organisational topics:

- Start on Wed. at 9:00 a.m.
- Lunch at 12:30 p.m.
- Restart at 1:15 p.m.
- Feedback round at 2:30 p.m.
- Stop at 3:00 p.m.

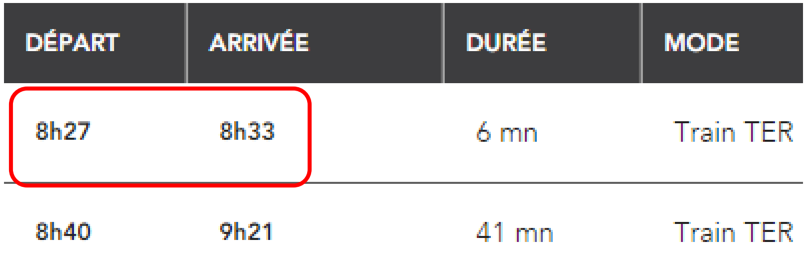


Départs ter de Strasbourg	7 15	7 55	8 27
Arrivées ter à Entzheim Gare	7 27	8 03	8 39
ENTZHEIM GARE	7 31	8 09	8 44
ICARE	7 40	8 18	8 53

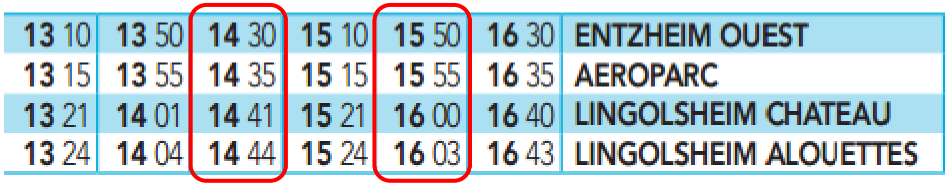


12 Direction / Richtung Entzheim Ouest

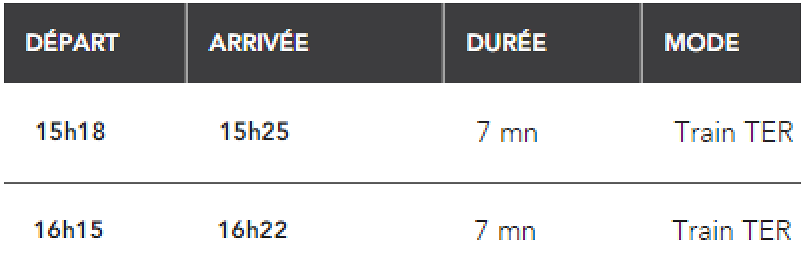
LINGOLSHEIM ALOUETTES	5 27	6 19	7 08	7 48	8 31	8 48	9 28	10 07
RUE DES JUIFS	5 30	6 22	7 11	7 52	8 35	8 52	9 31	10 10
AEROPARC	5 35	6 27	7 17	7 58	8 41	8 58	9 36	10 15



DÉPART	ARRIVÉE	DURÉE	MODE
8h27	8h33	6 mn	Train TER
8h40	9h21	41 mn	Train TER



13 10	13 50	14 30	15 10	15 50	16 30	ENTZHEIM OUEST
13 15	13 55	14 35	15 15	15 55	16 35	AEROPARC
13 21	14 01	14 41	15 21	16 00	16 40	LINGOLSHEIM CHATEAU
13 24	14 04	14 44	15 24	16 03	16 43	LINGOLSHEIM ALOUETTES



DÉPART	ARRIVÉE	DURÉE	MODE
15h18	15h25	7 mn	Train TER
16h15	16h22	7 mn	Train TER

2. WP5 purpose

- **Reference implementation of an OBU**
  - **Show Baseline 3.3.0 implementation**
  - **Help partners to understand some SRS requirements**
  - **Supply partners with a working API to the OBU**
  - **Provide partners the ability to compare their OBU behaviour → WP3**
  - **Provide partners the ability to define test cases → WP4**

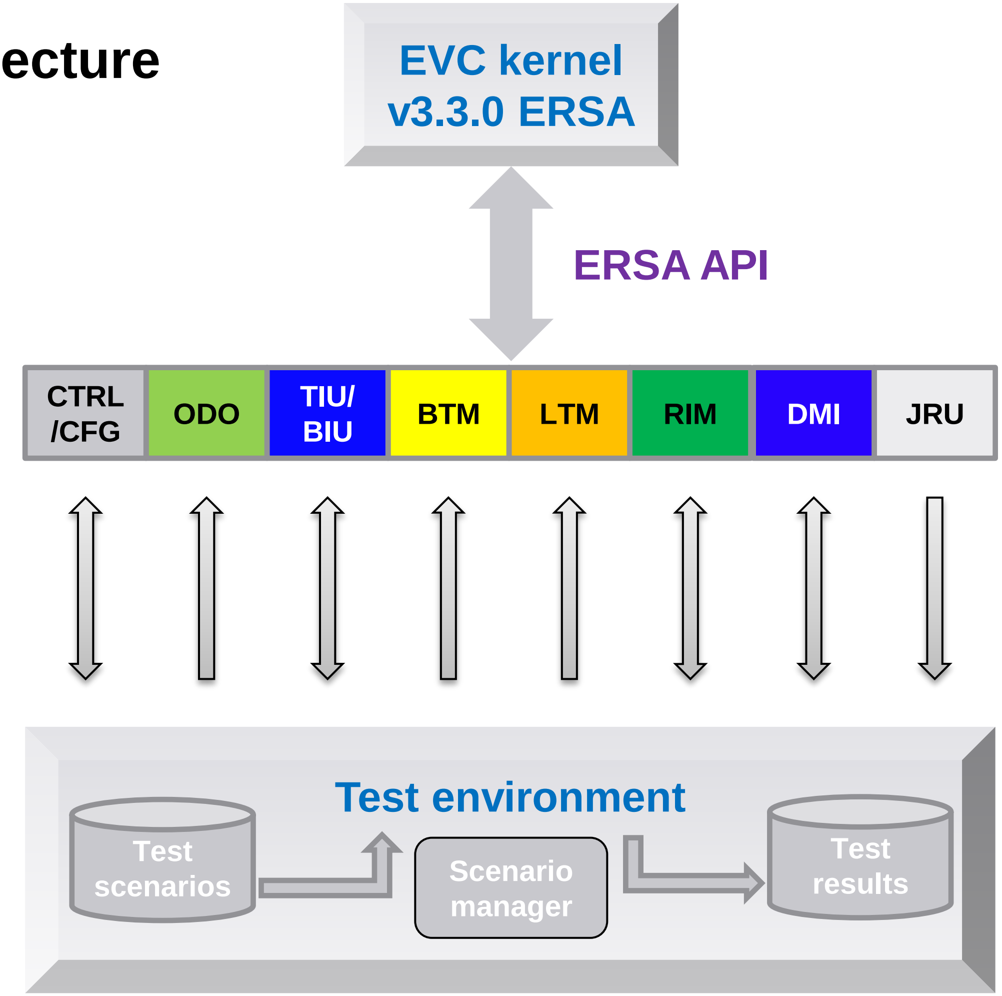
➔ OBU is a software component : it is a library

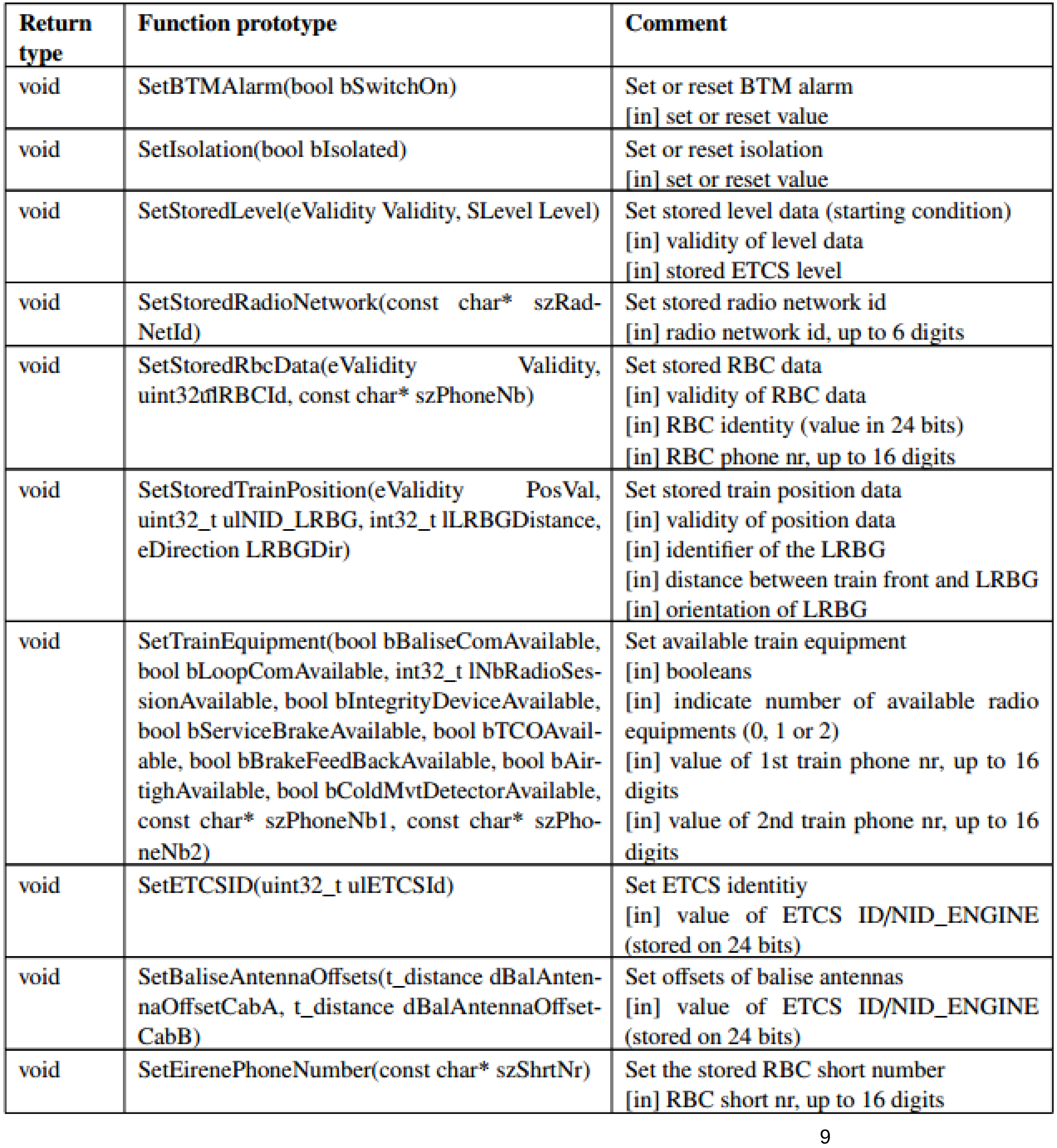
- **Architecture of OBU and Test Environment**
  - **The OBU is composed by one kernel and simulated peripherals**
  - **Test Environment stimulates simulated peripherals**
  - **Test Environment is a separated executable tool**
  - **In a 1st iteration, Test Environment and OBU shall run on the same machine**

➔ Test Environment is a software component : it is an exe

2. WP5 purpose

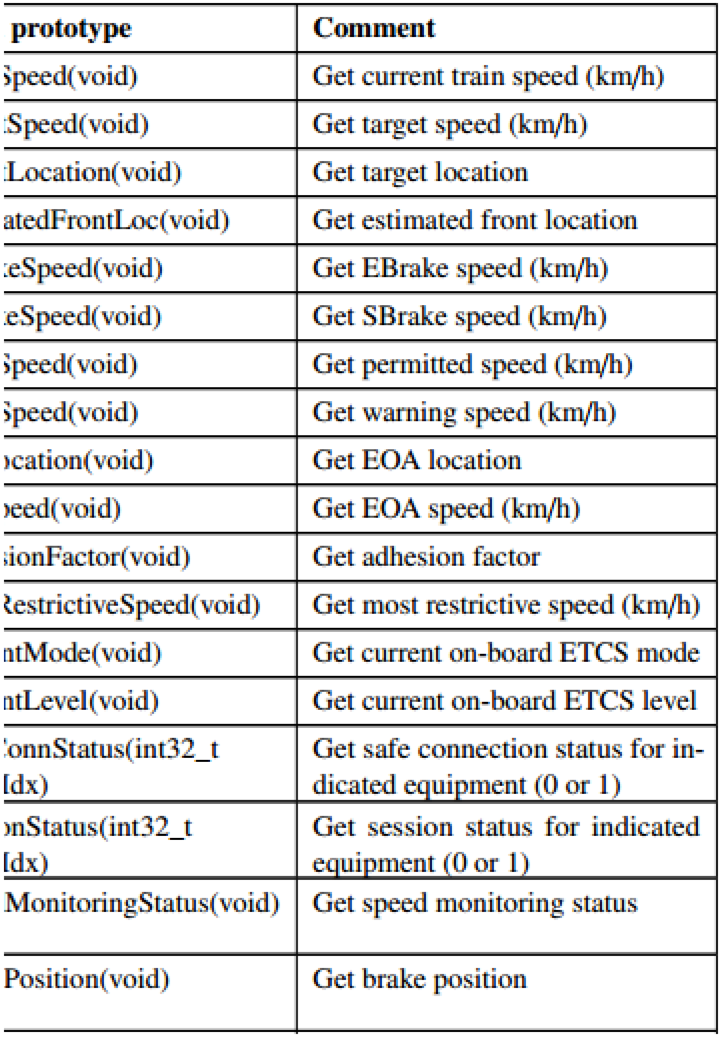
Current OBU architecture





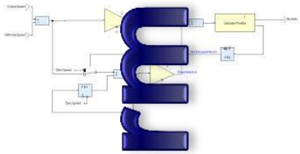
Return type	Function prototype	Comment
void	SetBTMAlarm(bool bSwitchOn)	Set or reset BTM alarm [in] set or reset value
void	SetIsolation(bool bIsolated)	Set or reset isolation [in] set or reset value
void	SetStoredLevel(eValidity Validity, SLevel Level)	Set stored level data (starting condition) [in] validity of level data [in] stored ETCS level
void	SetStoredRadioNetwork(const char* szRadNetId)	Set stored radio network id [in] radio network id, up to 6 digits
void	SetStoredRbcData(eValidity Validity, uint32_t ulRBCId, const char* szPhoneNb)	Set stored RBC data [in] validity of RBC data [in] RBC identity (value in 24 bits) [in] RBC phone nr, up to 16 digits
void	SetStoredTrainPosition(eValidity PosVal, uint32_t ulNID_LRBG, int32_t lLRBGDistance, eDirection LRBGDir)	Set stored train position data [in] validity of position data [in] identifier of the LRBG [in] distance between train front and LRBG [in] orientation of LRBG
void	SetTrainEquipment(bool bBaliseComAvailable, bool bLoopComAvailable, int32_t lNbRadioSessionAvailable, bool bIntegrityDeviceAvailable, bool bServiceBrakeAvailable, bool bTCOAvailable, bool bBrakeFeedBackAvailable, bool bAirtighAvailable, bool bColdMvtDetectorAvailable, const char* szPhoneNb1, const char* szPhoneNb2)	Set available train equipment [in] booleans [in] indicate number of available radio equipments (0, 1 or 2) [in] value of 1st train phone nr, up to 16 digits [in] value of 2nd train phone nr, up to 16 digits
void	SetETCSID(uint32_t ulETCSId)	Set ETCS identitiy [in] value of ETCS ID/NID_ENGINE (stored on 24 bits)
void	SetBaliseAntennaOffsets(t_distance dBalAntennaOffsetCabA, t_distance dBalAntennaOffsetCabB)	Set offsets of balise antennas [in] value of ETCS ID/NID_ENGINE (stored on 24 bits)
void	SetEirenePhoneNumber(const char* szShrtNr)	Set the stored RBC short number [in] RBC short nr, up to 16 digits

9

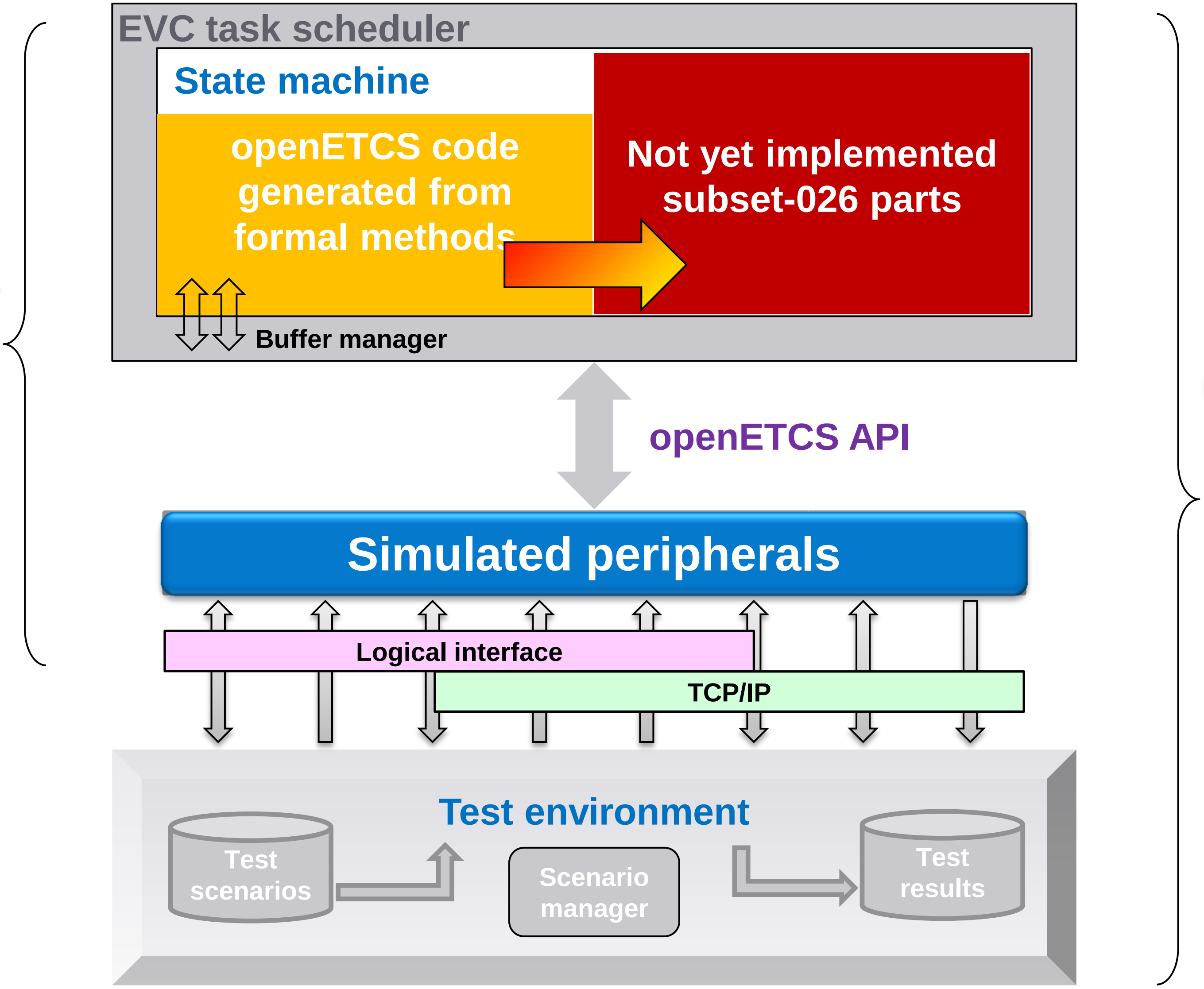


prototype	Comment
Speed(void)	Get current train speed (km/h)
Speed(void)	Get target speed (km/h)
Location(void)	Get target location
atedFrontLoc(void)	Get estimated front location
eSpeed(void)	Get EBrake speed (km/h)
eSpeed(void)	Get SBrake speed (km/h)
Speed(void)	Get permitted speed (km/h)
Speed(void)	Get warning speed (km/h)
cation(void)	Get EOA location
eed(void)	Get EOA speed (km/h)
ionFactor(void)	Get adhesion factor
RestrictiveSpeed(void)	Get most restrictive speed (km/h)
ntMode(void)	Get current on-board ETCS mode
ntLevel(void)	Get current on-board ETCS level
onnStatus(int32_t Idx)	Get safe connection status for indicated equipment (0 or 1)
nStatus(int32_t Idx)	Get session status for indicated equipment (0 or 1)
MonitoringStatus(void)	Get speed monitoring status
Position(void)	Get brake position

2. WP5 purpose



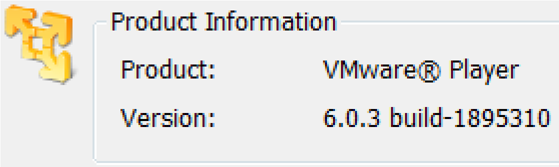
Hw platform



PC platform

3. Installation of WP5 tools

- **Virtual Machines distribution**



Product Information

Product: VMware® Player

Version: 6.0.3 build-1895310

  - **Install VMware Player version 6.0.3 at least**
  - **From the HDD cut 1 out of 12 Cent_OS VM**
  - **From the HDD cut the corresponding out of 12 Cent_OS license file**
  - **On 1st start of the VM, select it was “Moved…” not “Copied…”**
  - **Log in with user *TWS* and password *password***
  - **Make your Wi-Fi available within the VM**

- **Redhat Packet Manager deployment**
  - **Go to https://extranet.ersa-france.com/openetcs**
  - **Log in with user *openetcs* and password *iH4UBYTC***
  - **Download the file named *openETCS_WP5_1_0_2.tar.bz2***
  - **Unpack it in the user directory of your VM, either by the right click command or typing the** >tar xvfj openETCS_WP5_1_0_2.tar.bz2 **command from a terminal**
  - **Move the unpacked files to your own directory**

3. Installation of WP5 tools

- **Ensure you are connected to the internet and open a terminal**

- **Connect as root with the** >su **command and the corresponding password** ***password***
- **Install mysql-server package with the** >yum install mysql-server **command**
- **Start mysql service with the** >service mysqld start **command**
- **Install qt-mysql driver package with the** >yum install qt-mysql **command**
- **Change current path to the path you unpacked the archive file**
- **Install SRS baseline 230d database with** >rpm –Uvh srs230d_db-1.0.3-220.i386.rpm
- **Install SRS 330 v1.1 database with** >rpm –Uvh srs330_class1_v11_db-1.0.0-282.i386.rpm
- **Install SRS 330 v2.0 database with** >rpm –Uvh srs330_class1_v20_db-1.0.0-282.i386.rpm
- **Install openETCS package with** >yum localinstall openETCS-1.0.12-1.i386.rpm

- **Prefer not launching Test Runner (aka Test Environment), until the license file has been installed**

3. Installation of WP5 tools

- **Licensing distribution and installation**
  - **Each VM has its own unique license file (fingerprint based)**
  - **Open a terminal on the /usr/local/openETCS/Licensing path**
  - **Log in as root with the** >su **command and the corresponding password *password***
  - **Launch the license installer script by calling the following command**

    ```
    >./licenseinstaller –i –l ‘xxxx’ /opt/ERSA/license/license.rc
    ```

    The xxxx characters shall be replaced by the key found in the licensing file: it starts with the first character and includes the # **character.**

  - **Launch the Test Environment to ensure the license file has been properly installed**
    - **From a terminal, in the openETCS path, type >./test_runner (useless without scenario specified)**
    - **From a terminal, in the openETCS path, type >./test_runner oETCS_scenarios/Name_of_scenario.sce**

→ The Test Environment starts or returns an error message if the license is not working properly

4. M5.1 OBU simulator

- **Where it is**
  - **/usr/local/openETCS/lib contains the EVC baseline 3.3.0 kernel library: libevc_com.so**
    It is a dynamically linked shared object library
  - **The log files produced by the EVC kernel can be found in the /usr/local/openETCS/test_runner/data folder in text format**

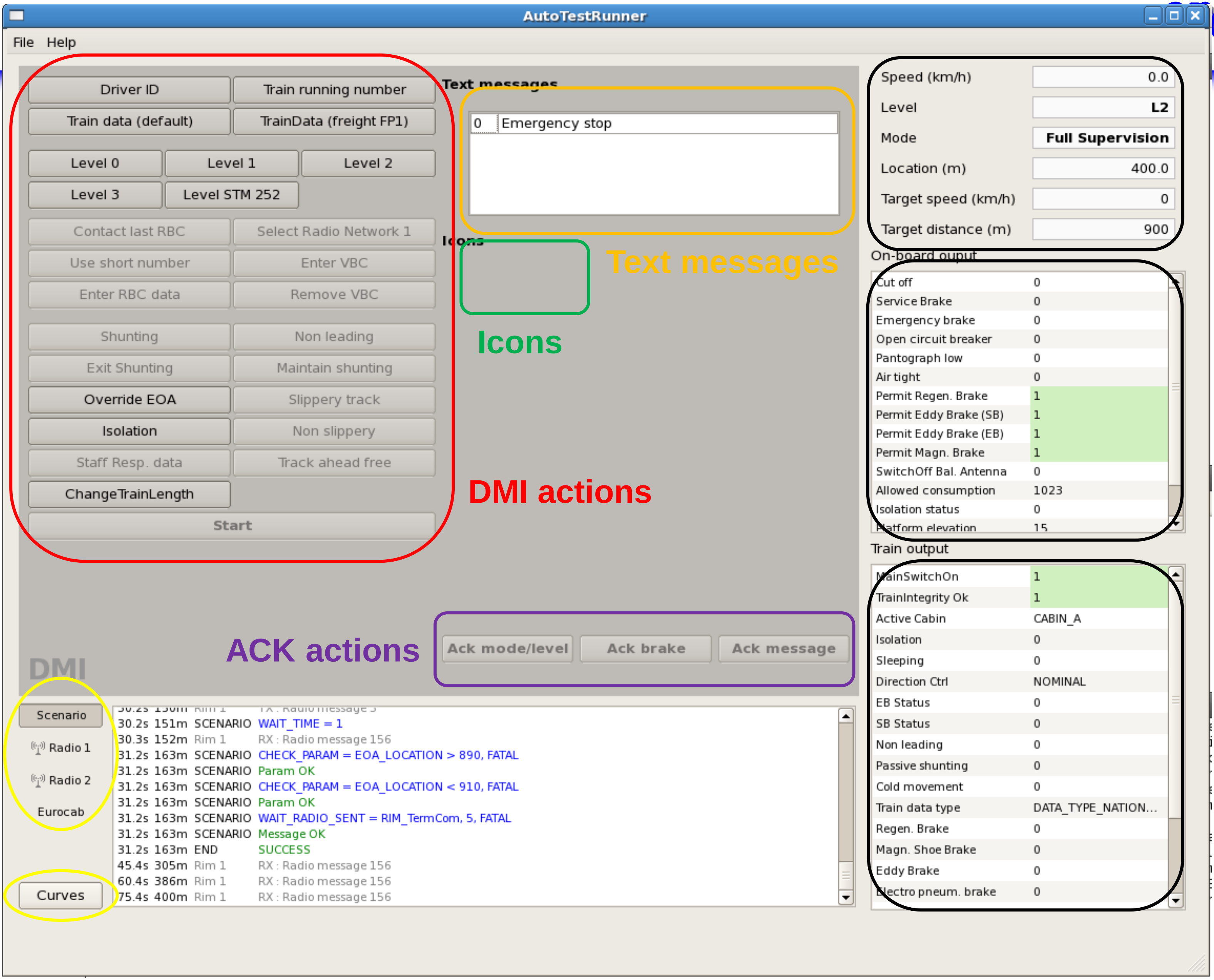
- **How it works**
  - **The EVC is started by the Test Runner. The EVC provides access to its kernel functions and also to its peripherals to exchange data. These are called JRU_Com, DMI_Com, Bal_Com, Odo_Com, etc.**
  - **Each simulated peripheral runs its own thread. Each peripheral is handled by a thread within the EVC**
  - **EVCSim, the EVC simulator, is the entry point of the simulation. It controls the simulation and checks the system state. The kernel gathers data from all other threads and deals with: perform mode and level transitions, validate data entered by the driver, calculate train supervision curves, calculate the MRS limits for the current train location, supervise the train movements, trigger warnings &interventions, record events occurring in the system.**

5. M5.3 Test Environment

- **Where it is**
  - /usr/local/openETCS/test_runner contains the Test Environment application: test_runner
  - The log files produced by the Test Environment can be found in the /usr/local/openETCS/test_runner/data folder in text format
  - Scenarios are stored in the /usr/local/openETCS/test_runner/oETCS_scenarios folder in text format. These 7 files are provided as example and can be extended according to test cases

- **How it works**
  - The Test Environment starts and it starts its linked EVC. A simplified view of a DMI is displayed and a panel of information is visible
  - The specified scenario starts and some virtual driver actions are replayed
  - The scenario status is displayed in the lower part of the window: test conditions, results, as well as events are logged

5. M5.3 Test Environment



0	Emergency stop

Speed (km/h)	0.0
Level	**L2**
Mode	**Full Supervision**
Location (m)	400.0
Target speed (km/h)	0
Target distance (m)	900

On-board ouput

Cut off	0
Service Brake	0
Emergency brake	0
Open circuit breaker	0
Pantograph low	0
Air tight	0
Permit Regen. Brake	1
Permit Eddy Brake (SB)	1
Permit Eddy Brake (EB)	1
Permit Magn. Brake	1
SwitchOff Bal. Antenna	0
Allowed consumption	1023
Isolation status	0
Platform elevation	15

Train output

MainSwitchOn	1
TrainIntegrity Ok	1
Active Cabin	CABIN_A
Isolation	0
Sleeping	0
Direction Ctrl	NOMINAL
EB Status	0
SB Status	0
Non leading	0
Passive shunting	0
Cold movement	0
Train data type	DATA_TYPE_NATION...
Regen. Brake	0
Magn. Shoe Brake	0
Eddy Brake	0
Electro pneum. brake	0

```
30.2s 150m Rim 1     TX : Radio message 5
30.2s 151m SCENARIO  WAIT_TIME = 1
30.3s 152m Rim 1     RX : Radio message 156
31.2s 163m SCENARIO  CHECK_PARAM = EOA_LOCATION > 890, FATAL
31.2s 163m SCENARIO  Param OK
31.2s 163m SCENARIO  CHECK_PARAM = EOA_LOCATION < 910, FATAL
31.2s 163m SCENARIO  Param OK
31.2s 163m SCENARIO  WAIT_RADIO_SENT = RIM_TermCom, 5, FATAL
31.2s 163m SCENARIO  Message OK
31.2s 163m END       SUCCESS
45.4s 305m Rim 1     RX : Radio message 156
60.4s 386m Rim 1     RX : Radio message 156
75.4s 400m Rim 1     RX : Radio message 156
```

5. M5.3 Test Environment

- **Scenario purpose and content**
  - **Main goal is to describe test case by creating initial conditions, train data and national default values.**
  - **It shall be possible to automate the driver action(s) from the DMI side**
  - **Balises and radio content shall be simulable**

- **Some scenarios possibilities**
  - **Driver actions: some keywords are used to simulate driver actions. Expected time before playing the driver action can be specified after the action itself**
  - **Wait actions: there are many wait triggers e.g. wait radio sent, wait location, wait speed, wait standstill. A timeout can be specified after the wait condition and an error type is declared as 3rd parameter**
  - **Move actions acts on the train movement**
  - **Checking parameters is also an option. E.g. checking the permitted speed to be > 50 km/h can be performed**
  - **Many others are also available…**

6. Writing a scenario file

- **Scenario structure uses the following sections**
  - **SCENARIO:** Main section for the scenario execution
  - **BaliseTrackside:** Description of balise contents to be sent to on-board
  - **LoopTrackside:** Description of loop messages to be sent to on-board
  - **"User defined name" radio message:** Description of **ONE** radio message
  - **SpeedProfile:** Description of the speed profile used for train movement simulation
  - **Config_TrainData:** Description of train data used by on-board
  - **Config_EVCInit:** Description of starting conditions used by on-board
  - **Config_SRSNationalDefaults:** Description of the default national values used by on-board
  - **Config_RBCData1:** Description of the first RBC parameters used for testing
  - **Config_RBCData2:** Description of the second RBC parameters used for testing
  - **Config_Scenario:** Specific options for the Test Environment about the current scenario

SCENARIO and SpeedProfile are mandatory to run a scenario. Other parts are optional.

The M5.3 Test Environment document can be used as user manual.

6. Writing a scenario file

- Additional information

  - Add SRS reference to simplify scenario comprehension
  - Add comments (use symbol #) to simplify scenario comprehension
  - Initial EVC configuration, initial national values, … use default data specified in Test Environment documentation (M5.3 on gitHub)
  - SRS messages (from radio, balise or loop) use default values specified in SRS-Subset 26 (or 0 if there is no default value)
  - Simply modify values to test in a balise message, Test Environment fills missing fields
  - Use M5.3_Test_Environment_User_guide.pdf to get more information about all controls, instructions available in Test Environment
  - To define a radio message, user has to create one section for each radio message used:
    - *[InitCommSession] for EVC request connection to RBC*
    - *[ConfigurationDetermination] for RBC answer containing system version*
    - *[SOMPositionReport] for Start Of Mission position report from EVC to RBC*

6. Writing a scenario file

- **Trigger examples:**

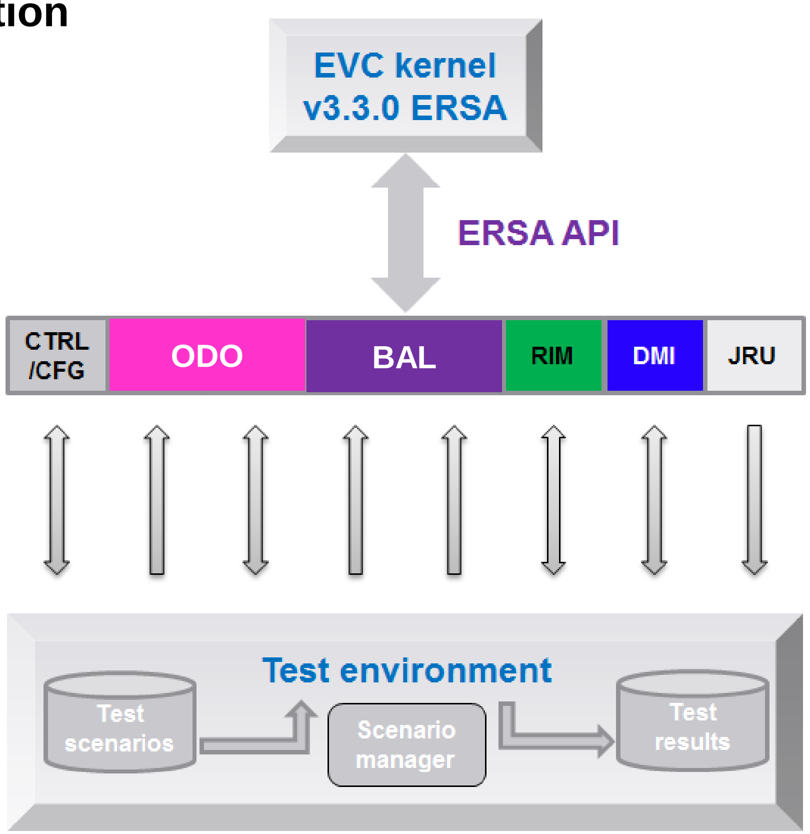
- DRIVER_ACTION
  - MainSwitchOn

- NTC_ACTION, WAIT_STATUS, RBC_RADIO, RADIO_MODULE1 RBC_RADIO2, RADIO_MODULE2, WAIT_RADIO_SENT, WAIT_RADIO_SENT2, MOVE_TRAIN, MOVE_TRAIN_BACK, WAIT_STANDSTILL, WAIT_NONE, WAIT_STANDSTILL, WAIT_LOCATION, WAIT_SPEED, CHECK_PARAM, DO_RADIO

- WAIT_TIME, CONNECT_RADIO, CONNECT_RADIO2, SET, WAIT_SYMBOL, WAIT_BUTTON, WAIT_TEXT, WAIT_DYNAMIC

7. Interface with M5.1: API to EVC

- Used in current Test Environment
  - C++, but will be converted to pure C (names and signature will be kept)
  - SIM_xxx functions for controlling the simulation
  - ODO_xxx functions for odometry and TIU
  - BAL_xxx functions for balise and loop
  - RAD_xxx functions for radio
  - DMI_xxx functions for DMI
  - No need to call any JRU function



7. Interface with M5.1: API to EVC

- To test the API in a new project:
  - Set include path to contain “evc_com.h”, “etcs_types.h” and “etcs_config.h”
  - Set lib path to link with “libevc_com.so”
  - Take a look at the “light_runner” mini project (Qt/Qmake based)

- Example: starting the EVC
  - CEvc_com evc_com;
  - evc_com.SIM_Init();
  - More details in “light_runner.cpp”

+ scenario creation

- Initial conditions: line level 2 and speed profile 50 m. = 20 km/h, 100 m. = 100 km/h and 800 m. = 0
- Start of Mission in level 1
- 1st balise @ 50 m. containing MA end of section @850 m. immediate SSP @200 km/h, gradient is null
- 2nd balise @ 100 m. with transition to level 2 after 100 m.
- Add classical checks (WAIT_STATUS, etc.)

8. Feedback round

- What you liked
- What you learned
- What was missing
- What still needs explanations
- What you expect for the future

openIT4SR WP6

Thank you very much for your attention

Have a safe trip back