JRC Scientific and Technical Reports

TECHNICAL SPECIFICATIONS FOR NAVIGATION SYSTEMS IN LONG JOURNEY ANIMAL TRANSPORTS

J Hofherr, G Fiore, F Natale, J Bishop, S Mainetti, E Ruotolo

EUR 23238 EN - 2008

JRC EUROPEAN COMMISSION

ipsc Institute for the Protection and Security of the Citizen

The Institute for the Protection and Security of the Citizen provides research-based, systems-oriented support to EU policies so as to protect the citizen against economic and technological risk. The Institute maintains and develops its expertise and networks in information, communication, space and engineering technologies in support of its mission. The strong cross-fertilisation between its nuclear and non-nuclear activities strengthens the expertise it can bring to the benefit of customers in both domains.

European Commission
Joint Research Centre
Institute for the Protection and Security of the Citizen

Contact information
Address: Via E Fermi 2749, 21027 Ispra (Varese)
E-mail: gianluca.fiore@jrc.it
Tel.: +39 0332 786710
Fax: +39 0332 786280

http://ipsc.jrc.ec.europa.eu/
http://www.jrc.ec.europa.eu/

JRC 36616

EUR 23238 EN
ISBN 978-92-79-08310-5
ISSN 1018-5593
DOI 10.2788/679

Luxembourg: Office for Official Publications of the European Communities

Printed in Italy

DOCUMENT CONTROL

Revision history

Version	Date	Author	Summary of changes
0.01	29/11/05	M. Ferretti	System definition
0.02	28/12/05	M. Ferretti	Contents revision
0.03	09/01/06	S. Cubeddu	paging
0.04	18/01/06	R. Sanchez	FSM section added
0.05	19/01/06	M. Ferretti	System and MU specs added
0.06	25/01/06	M. Ferretti	CAN specs
0.07	26/01/06	R. Sanchez	XML structure
0.08	27/01/06	R. Sanchez	XML structure
0.09	30/01/06	S. Cubeddu	Paging chapter 9
0.10	30/01/06	P. Ferretti	System behaviour & communication defs.
0.11	15/02/06	J. Hofherr	Restructuring of report
0.12	28/02/06	S. Cubeddu	Paging
0.13	04/03/06	M. Ferretti	CUI, CBC sections, warning parameters
0.14	06/03/06	R. Ventrella	Inspection, setup, buses
0.15	14/03/06	M.Ferretti	Shock & vibration specifications
0.16	15/03/06	S. Cubeddu	Paging
0.17	19/03/06	R. Ventrella	Restructuring of inspection protocol and Cabin UI
0.18	11/04/06	G Fiore, J Bishop, J. Hofherr	Review on content and layout
0.19	21/05/06	J. Bishop, H. Hofherr	Rearrange order
0.20	29/05/06	S. Cubeddu	Paging and corrections
0.21	25/09/06	M., P. Ferretti, J. Bishop, J. Hofherr	Amendments: transmission intervals, data security, Annex VIII
0.22	16/07/07	G. Fiore, J. Hofherr, P. Ferretti, S. Cubeddu, F. Natale	Separation of functional requirements, second OBU, XML scheme, Annex V
0.22	18/12/07	J. Hofherr	Amendments to text after internal and external review

Approvals this document requires the following approvals

Version	Date	Name	Function	Signature
0.22		A Poucet	Head of Unit	
0.22		G Fiore	Task leader	

There may be a need to regularly update the technical specifications to allow adoption to further technical experience and developments in the fields of animal welfare requirements, navigation and communication technology. A regular review and change procedure is proposed.

The first review has been carried out by JRC after an international workshop with all relevant stakeholders in June 2006 and in May/June 2007 regarding a second on-board-unit.

Table of Contents

ABBREVIATIONS AND GLOSSARY

Animal compartment(s)	part(s) of means of road transport (trailer or semi-trailer) where livestock is kept during transportation
CAN	Controlled Area Network – network connecting and empowering different hardware
CBC	CAN-Bus Connector
CUI	Cabin user interface - hardware which allows driver of transport vehicle to enter a predefined set of information into the navigation system
DTD	Data Type Description
FSM	Finite state machine
GALILEO	Independent European global civilian controlled satellite positioning system, inter-operable with GPS and GLONASS, the two other global satellite positioning systems
GPS	Global positioning system used for civilian and military purpose
GPRS	General Packet Radio Service
GNSS	Global navigation satellite systems
GSM	Global System Mobile
Journey	The entire transport operation from the place of departure to the place of destination, including any unloading, accommodation and loading occurring at intermediate points of the journey
Journey log	Document to be filled in by the transporter/organizer for the planning and execution of a long journey as laid down in Annex II to Regulation (EC) 1/2005 and verified by the Competent Authorities
Long journey	A journey that exceeds 8 hours, starting form when the first animal of the consignment is moved.(Article 2 of Regulation (EC) 1/2005)
OEM	Original Equipment Manufacturer
Remote receiver, remote site	remote data processing facility (to be defined) which can receive data from the satellite positioning system by a predefined data link
Navigation system	Satellite-based infrastructures providing global, continuous, accurate and guaranteed timing and positioning services or any technology allowing for recording information equivalent to those required in the journey log as referred to in Annex II, Section 4 to Regulation (EC) 1/2005 and information concerning opening/closing of the loading flap

Transporter	Any natural of legal person transporting animals on his own account, of for the account of a third party
Truck	A heavy motor vehicle designed for carrying or pulling loads (in road transport of animals with a single fixed loading compartment with one or more tiers) and able to haul a trailer
Trailer	A large transport vehicle designed to be hauled by a truck or tractor (in road transport of animals with a single loading compartment with one or more tiers)
Tractor	A truck having a cab and no body, used for pulling a semi-trailer
Semi-trailer	A trailer having wheels at the rear only, with the forward portion being supported by the truck tractor or towing vehicle (in road transport of animals with a single loading compartment with one or more tiers)
TRACES certificate	Certificate relating to intra-Community trade in animals or common veterinary entry document as described in Commission Decision 2004/292/EC of 30.03.2004 on the introduction of the Traces system and amending Decision 92/486/EECCertificate which has to be entered into the TRACES – OJ L94, 31.3.2004, p. 63
RS232	Full Duplex Serial Communication Port
RS485	Half Duplex Serial Communication Port
USB	Universal Serial Bus
VIN	Vehicle identification number as defined in ISO 3779
VRN	Vehicle registration number (number plate)
XML	Extended Markup Language

INTRODUCTION

Background

In the wake of an opinion of the Scientific Committee on Animal Health and Animal Welfare on animal welfare during transport of 11 March 2002, Community legislation was amended to give priority to the need for the enforceability of animal welfare requirements. So far, only few of the animal welfare requirements can be verified, mainly after the transportation (retrospective). Since long journeys are likely to have more detrimental effects on the welfare of animals than shorter ones, specific procedures should be designed to ensure better enforcement of the standards, in particular by increasing the traceability of such transport operations.

In the framework of the administrative arrangement No 30042-2005-12 A1CO between Directorate General for Consumer Protection (thereafter: DG SANCO) and the Joint Research Centre (thereafter: JRC), JRC carries out a **project "animal welfare in transportation".** This project provides technical support to DG SANCO in establishing an effective navigation system in accordance with Regulation (EC) 1/2005 on the protection of animals during transport[1].

To achieve better enforcement of the standards, the **objectives** of the project are to develop a navigation system for long road journeys as referred to in Regulation (EC) 1/2005 which allows demonstrating and verifying at any given moment during the journey that animal requirements are respected. By recording and transmitting certain animal welfare parameters during the journey in real time, the system will also provide the transporters with a tool to facilitate compliance with animal welfare requirements in transportation, to decrease the administrative burden for transporters and competent authorities and will contribute to the prevention of fraud. The system could enable the competent authorities to perform more targeted, effective and efficient controls on such transports and to ensure the uniform enforcement of the above regulation within the Community.

The project comprises two consecutive phases. In the actual, **first phase** the technical specification for a navigation system for road vehicles as referred to in Regulation (EC) 1/2005 will be drawn up taking into account the input from Member States and all relevant stakeholders. In a **second phase** a prototype of a navigation system should be tested in-field.

[1] Council Regulation (EC) No 1/2005 of 22 December 2004 on the protection of animals during transport and related operations and amending Directives 64/432/EEC and 93/119/EC and Regulation (EC) No 1255/97 – OJ L3, 05.01.2005, p.1

Scope of the document

This document defines the technical specifications for navigation systems as referred to in Article 6(9) of Regulation (EC) 1/2005 to the aim

- To provide users, developers and manufacturers of navigation and communication systems a description of necessary components and to which requirements such systems shall conform;
- To ensure that devices of different manufacturers are compatible and interoperable;
- To allow users, such as transport companies and competent authorities to plan for the integration of such systems in the daily routines.

The technical specifications are drawn up in conformance with the minimum requirements set up by the Commission and comprise four parts:

1. Functional Requirements	describing the functional requirements conforming navigation systems have to achieve;
2. Technical Specification File	describing the hardware components of conforming navigation systems;
3. Technical Architecture	describing the technical architecture of conforming navigation systems;
4. Functional Design	describing the functional requirements of conforming navigation systems regarding hardware and software.

1. FUNCTIONAL REQUIREMENTS

This chapter refers to the functional requirements of conforming navigation systems.

Regulation (EC) 1/2005, in particular Article 6(9) thereof sets the requirements for navigation systems for which the technical specifications are described in this document:

> *"9. Transporters of domestic Equide, except registered Equide, and domestic animals of the bovine, ovine, caprine and porcine species over long road journeys shall use a navigation system as referred to in Annex I, Chapter VI, paragraph 4.2, as from 1 January 2007 for means of transport by road for the first time in service and as from 1 January 2009 for all means of transport by road. ....."*

Annex I, Chapter VI, paragraph 4 to Regulation (EC) 1/2005 requires an appropriate navigation system allowing for recording and providing information equivalent to those required in the journey log and information concerning opening/closing of the loading flap.

The above Regulation also requires a temperature monitoring and recording system which alerts the driver of the vehicle when the temperature in the animal compartment(s) reaches the maximum of 30°C or the minimum of 5°C. Although this temperature range applies irrespectively the species/categories of animals transported, a differentiation of temperature requirements for different species and categories of animals is under discussion. For the actual and possible future specific temperature requirements see Annex I to these Technical Specifications.

Regulation (EC) 1/2005 does not necessarily require a connection of the temperature monitoring and recording system with the navigation system. However, the technique available enables the integration of the temperature monitoring and recording into the navigation system, avoiding duplication of devices and information and allowing shared use and simplification of systems.

With regard to Article 6(9) of Regulation (EC) 1/2005, the Commission formulated minimum functional requirements a navigation system shall comply with[2]. These functional requirements list the system components, the component requirements, their mechanical and thermal specifications as well as the system functionalities. The system should permit expansion for further sensors and data interfaces and protocols for other permitted applications, such as digital tachograph, eCall or TRACES should be foreseen.

This document describes the technical specifications which are in line with the above legal requirements for satellite navigation systems in long journey transportation.

[2] Draft Commission Regulation (SANCO C(2007)10119 final) setting out specifications for the implementation of a satellite navigation system pursuant to Council Regulation (EC) 1/2005

2. TECHNICAL SPECIFICATION FILE

This chapter describes the hardware components of navigation systems referred to in Regulation (EC) 1/2005 and Draft Commission Regulation (SANCO C(2007)10119 final)

- Hardware components of a navigation system
- Onboard Unit (OBU)
- Temperature sensor(s)
- Loading door sensor(s)
- Coupling sensor
- Cabin User Interface (CUI)
- System Bus specifications

2.1 HARDWARE COMPONENTS OF A NAVIGATION SYSTEM

A navigation system conforming to the requirements set out in Chapter 1 of this document shall consist of at least the following hardware components for the required functions, such as collecting, recording and transmission of defined sets of data:

- One or two **Onboard Unit(s) (OBU(s))**
- one or more **temperature sensor(s)**
- one or more **loading door sensor(s)**
- one **coupling sensor**
- a **Cabin User Interface (CUI)**

All parts shall be interconnected. The hardware shall permit expansion for further sensors, e.g. for measurement of humidity.

For the mechanical and thermal specifications Chapter III of Draft Commission Regulation (SANCO C(2007)10119 final) shall apply.

2.2 ONBOARD UNIT (OBU)

The system has to provide a platform for recording of environmental parameters and events like temperatures, door status and trailer coupling and associating them to the current time and position of the vehicle. Environmental parameters have to be compared with legislative requirements, e.g. 5-30°C as currently defined in Annex I, Chapter VI, point 3.1 to Regulation (EC) 1/2005. However, the system shall be able to accommodate different temperature ranges and different species and categories of animals transported as legislation develops. For possible future specific temperature requirements see Annex I to these Technical Specifications..

The OBU has to store the data locally and to forward it to a remote receiver.
The OBU has to be set up to allow the driver of the animal transport to enter predefined information and to receive warnings as the animal welfare parameters reach predefined thresholds.

2.2.1 OVERVIEW ON THE LAYOUT

Fig. 2.1 illustrates a possible layout design of an OBU, here reported just as an example.

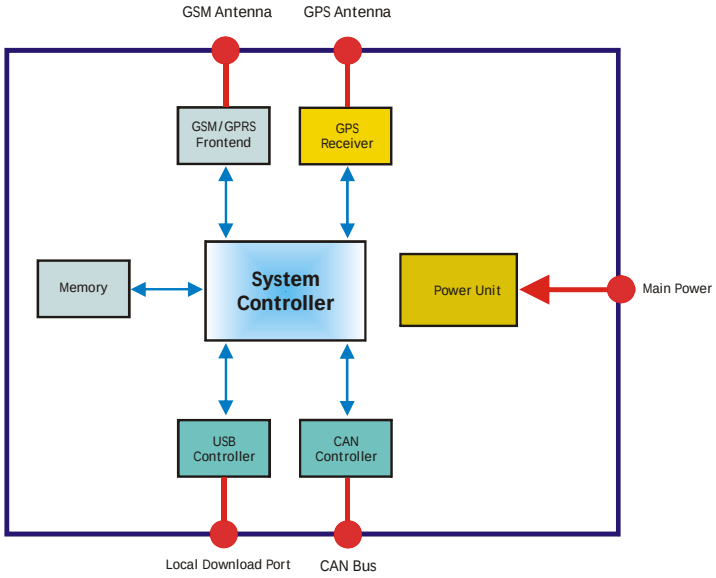


Fig. 2.1: OBU Block Diagram

Independent of the design and layout, the OBU shall foresee the following plugs/ports:

1. GPS Antenna plug.
2. GSM/GPRS Antenna plug.
3. local data interface port
4. CAN-bus port
5. Power Input

2.2.2 GPRS Communication Module

Although operating mainly in GPRS mode, the minimum specifications which also assure future expansion of system capabilities are listed in Annex II to this document.

2.2.3 GNSS (GPS/GALILEO) Module

Annex III to this document shows typical performance specifications for a GPS receiver.

2.2.4 Memory

Reference is made to Chapter 2.1.4 of the Draft Commission Regulation (SANCO C(2007)10119 final)

2.2.5 Local data interface

For the configuration of the navigation system and for other possible connections (e.g. built in tests, system diagnostics, on-the-spot animal welfare inspections) a standardized plug-in connection with the OBU is required.

Depending on the position of the OBU, at least an appropriate USB-connector will be required (e.g. waterproof, dustproof, etc). Additional, other standardized communication buses, e.g. CAN, RS 232, or Bluetooth can be foreseen.

2.2.6 Power Supply

The power supply of the OBU has to be designed to ensure the following:

- Operation in automotive environment,
- Self power in case of main power loss,
- Power output to feed peripheral devices through CAN-Bus.

2.2.6.A Power Input

The power input shall accept DC voltage from the vehicle/trailer power line and feeds the OBU internal electronics and, through a dedicated output, powers the peripheral devices.

The power input section defined specifications are listed in Table T-2.1

Description	Value
Minimum Input Voltage	Vin_{min} = 10,00 Vdc
Maximum Input Voltage	Vin_{max} = 40,00 Vdc
Transients Handling	Nominal Breakdown Voltage, $V_{(BR)}$ = 45 V Minimum Peak Pulse Current, $I_{(PPM)\ min}$ = 7,5 A (1) Minimum Peak Power Dissipation, $Pd_{p\ min}$ = 600 W(2)
Maximum Absorbed Power	Average, P_{avg} = 15,00 W Peak, P_{pk} = 85,00 W (3)

Tab. T-2.1: The power input section

The power input section shall guarantee protection against voltage inversions, i.e. positive and negative leads swapped on OBU power input.

2.2.6.B Backup Power

The backup battery charge is maintained from the vehicle power source by means of an internal charger.

2.2.6.C Power Output

The power output section feeds the peripheral devices through the CAN-Bus.

2.3 TEMPERATURE SENSOR(S)

Temperature sensors shall be calibrated and maintain their function and measurement accuracy even when exposed to unfavourable conditions as mentioned in Chapter 3 of Draft Commission Regulation (SANCO C(2007)10119 final). Apart from the robustness, the temperature sensors shall meet the following specifications:

- Input supply voltage range: 9-20V;
- Maximum power consumption: 1,2W;
- Temperature data shall be read via the CAN-Bus.

(1) Non repetitive 10µs < tp < 1ms pulse

(2) Repetitive 10µs < tp < 1ms pulse, 0,01% duty-cycle

(3) Repetitive 100µs < tp < 10ms pulse, 0,1% duty-cycle

2.4 LOADING DOOR SENSOR(S)

Loading door sensors shall maintain their function even when exposed to unfavourable conditions as mentioned in Chapter 3 of Draft Commission Regulation (SANCO C(2007)10119 final). Apart from the robustness, the sensors shall meet the following specifications:

- Input supply voltage range: 9-20V;
- Maximum power consumption: 1,2W;
- Loading door state shall be read via the CAN-Bus;.

2.5 COUPLING SENSOR

If a coupling sensor is a device it shall maintain its function even when exposed to unfavourable conditions as mentioned in Chapter 3 of Draft Commission Regulation (SANCO C(2007)10119 final). Apart from the robustness, the sensors shall meet the following specifications:

- Input supply voltage range: 9-20V;
- Maximum power consumption: 1,2W;
- Coupling state shall be read via the CAN-Bus.

2.6 CABIN USER INTERFACE (CUI)

Reference is made to Chapter 2.5 of Draft Commission Regulation (SANCO C(2007)10119 final)

2.7 SYSTEM BUS SPECIFICATIONS

The OBU shall control the network of sensors and the CUI in a master-slave fashion (the OBU acts as the master, while all other devices act as slaves). Networked systems are defined according to the ISO/OSI seven layer stack protocol specification, a.k.a. ISO standard 7498. The network layer adopted by the automotive industry is the CAN-Bus (Controller Area Network/ ISO 11898).

The CAN-Bus together with the CANopen protocol provides the most advantages, namely:

- Adequate data transmission rates of up to 1Mbps;
- Powerful redundant error checking procedures;
- Immunity to electrically noisy environments;
- Bus cabling integrates the power supply to all devices;
- Cost-effective and easy to install because of a single cable.

For this reasons, it is mandatory to fit the OBU with a CAN-Bus for the connection with the peripheral devices. The technical specifications will describe the required standardisation for the CAN-Bus only.

2.7.1 CAN-Bus

The CAN-Bus used by the navigation system can be either integrated to the vehicle or independent / stand-alone. If the navigation system is connected to the vehicle's CAN-Bus the OBU could use data and services provided by the vehicle manufacturer (e.g. vehicle speed, signals, instrument panel lighting commands). However, integration with the vehicle CAN-Bus will require that the OBU receives the necessary certifications to assure vehicle manufacturers that the system will not affect vehicle safety.

The ISO11898 standard defines the OSI reference model illustrated in Figure 2.2. For details regarding the specifications reference is made to ISO11898 standard.

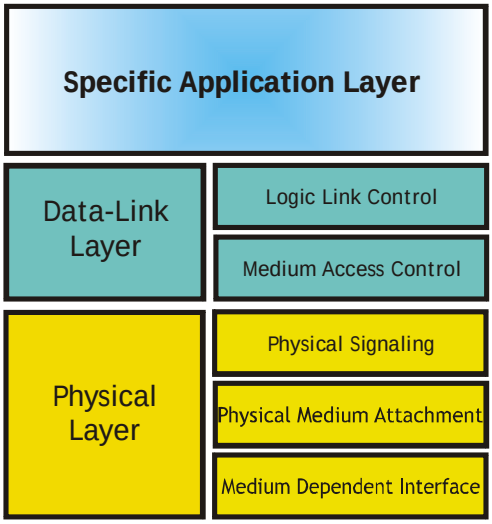


Fig. 2.2: ISO11898 CAN OSI model

The physical and data-link layers of the ISO11898 CAN specification are fully adopted. The "Specific Application Layer" for the navigation system is defined in Annex VII.

2.7.2 CAN-Bus Electrical Specifications

The CAN-Bus consists of a four wire cable which physically connects all navigation system devices and provides them with electrical power. The connection scheme is illustrated in Fig. 2.3

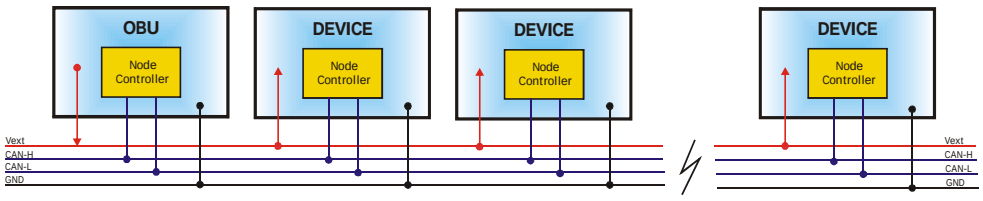


Fig. 2.3: OBU Connection Scheme

3. SYSTEM ARCHITECTURE

This chapter describes the technical architecture of navigation systems

- Configuration overview
- Semi-trailer configuration
- Truck and trailer configuration
- Flexibility for further developments

3.1 CONFIGURATION OVERVIEW

The system has to be compatible with different types of long journey animal transport vehicles. In order to monitor the welfare situation of the animals transported also in case of uncoupling, the OBU should preferably be installed at the vehicle with the animal compartments. The following chapters show the differences in the layout for the different types of vehicles.

To ensure interoperability between vehicles of different manufacturers and owners mechanical and electrical connections are defined in Annexes V, VI and VII.

3.1.1 Vehicle Typology and System layout

Vehicles for long journey animal transportation may be divided into two main categories:

- Semi-trailer – vehicle with wheels on the rear part, for road transport of animals with a single loading compartment with one or more tiers and which has to be articulated to a tractor;
- Truck with or without trailer – motorised vehicle for road transport of animals with a loading compartment with one or more tiers fixed on the truck and with the possibility to articulate a trailer.

Fig. 3.1 and 3.2 display the two truck categories.

Fig. 3.1: Semi-trailer

Fig. 3.2: Truck and trailer

3.1.2 Semi-trailer

The OBU should be installed on the semi-trailer. The coupling sensor would indicate if uncoupled from the tractor. Depending on the layout of the semi-trailer it may have a single or several loading doors/flaps. Each of them shall be monitored by a loading door sensor. Door sensors are not necessary for openings foreseen only for feeding/watering/inspection. Depending for what species and categories of animals used, semi-trailers may have a single floor or several tiers. At least one temperature sensor should be foreseen for each tier. Sensor location(s) will depend on the semi-trailers design characteristics, but shall in general be positioned where the extremes of temperature may be experienced. The CUI installed in the cabin of the tractor would provide the OBU with the information entered by the driver, such as number and category of animals, start, breaks and end of journey and exceptional events, such as injuries, death of animals. The OBU would send to the CUI information and warnings for display (e.g. temperature(s) inside the loading compartment). All devices are connected by a single CAN-Bus. This choice simplifies installation since only one cable is wired throughout the vehicle. Fig. 3.3 and 3.4 show a simplified overview for a system in a semi-trailer articulated to a tractor.

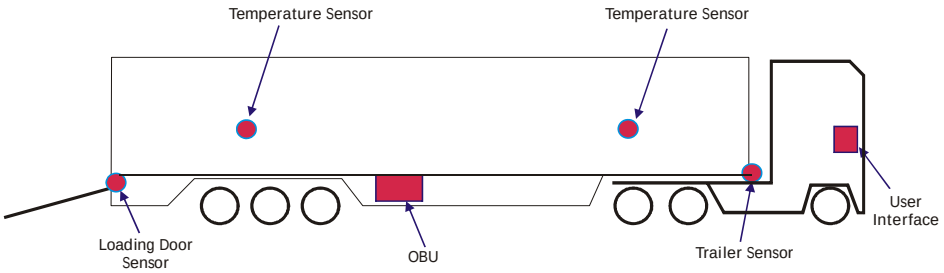


Fig. 3.3: OBU installation on a semi-trailer

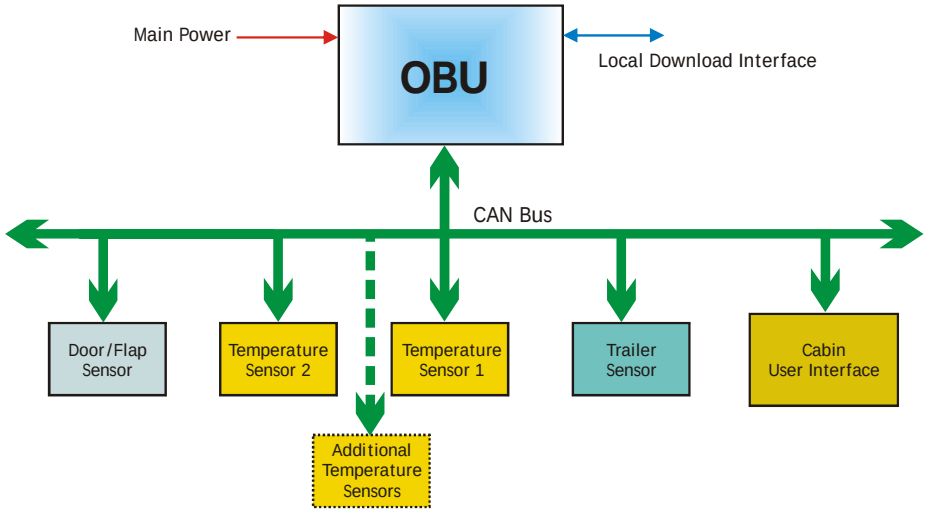


Fig. 3.4: OBU installation on a semi-trailer

3.1.3 Truck and trailer

In a truck and trailer configuration, one OBU (primary OBU) should be installed on the truck and a second OBU (secondary OBU) on the trailer.

The coupling sensor would indicate if a trailer is coupled to the truck.

When the trailer is de-coupled from the truck, the secondary OBU should act autonomously, to provide the position of the trailer and time, logging the data recorded on the trailer and transferring data to the remote receiver according to the prescribed XML message structure.

When the trailer is coupled to a truck with an OBU (primary OBU), the secondary OBU should be linked to the rest of the system by CAN-Bus, according to the CANopen protocol. The secondary OBU should have a different Node ID (see table T-A-VII-2) and behave as a slave CAN device of the primary OBU.
In the truck and trailer configuration the task of the primary OBU is to log all data including those related to the secondary OBU and transmit all data to the remote receiver according to the XML message structure described in chapter 4.3. The secondary OBU logs only data related to the trailer and a copy of the data related to the transport entered via the CUI, without the need of transmitting this data to the remote receiver.

For the peripheral devices Chapter 3.1.2 should apply. Fig. 3.5 and 3.6 show a simplified overview of a truck and trailer system.

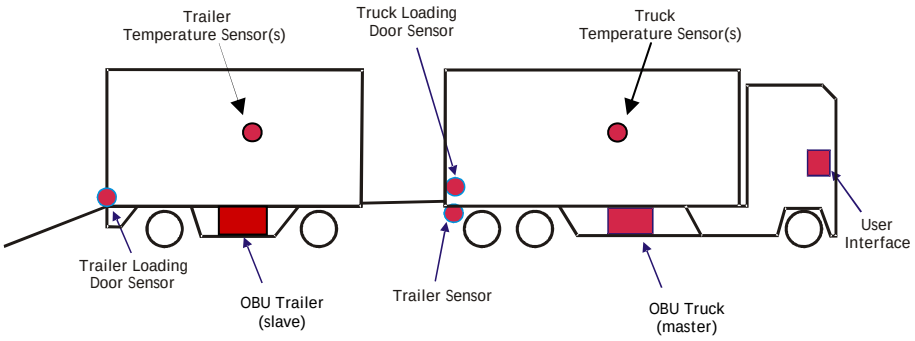


Fig. 3.5: OBU installation on a truck and trailer

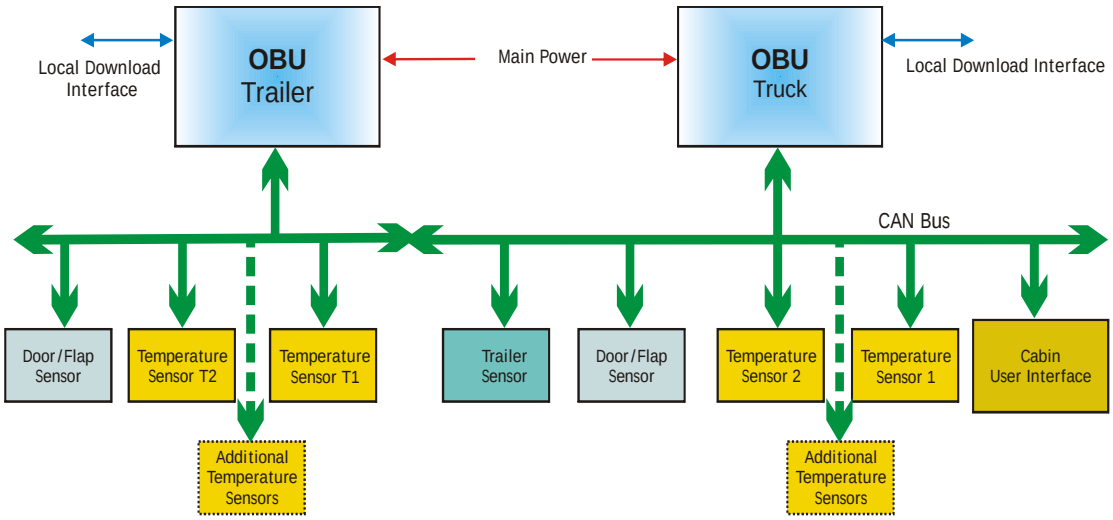


Fig. 3.6: OBU installation on a truck with trailer

3.2 Flexibility for Future Developments

Additional sensors and devices could be added to the system without affecting backward compatibility with an installed base. Definition of the message interface in XML provides for additional measurements, events, operations.

The CAN-Bus based system is flexible enough to be used for additional implementations not foreseen and/or specified in this document. Designers may add CAN devices according to functional or marketing needs. In case devices, other that the one described herein are added, these are not allowed to interfere with the specified working principles and mechanisms – i.e. additional devices may never corrupt or modify data on the CAN-Bus and may never interfere in the communication between the OBU and the other devices described herein.

The OBU and the devices described in this document always have an higher priority level in accessing the bus than additional features or devicers.

Additional devices and permitted applications (e.g. digital tachograph, eCall) may use some of the information shared by the devices described herein for their own purposes – i.e. temperature sensors, GPS position, etc… – according to the CANopen protocol.

4. FUNCTIONAL DESIGN

This chapter describes the functional design of navigation systems regarding hardware and software management.

- OBU operational behaviour
- Logging events
- XML message structure
- Remote receiver communication
- Local communication (Inspection)

4.1 OBU OPERATIONAL BEHAVIOUR

This chapter describes requirements in terms of processes, local and remote communication and hardware management. Open Software shall be used for running the navigation system.

4.1.1 Overview

OBU software may be described as a complex finite state machine (FSM) that manages and controls activities of the hardware present in the system. The software requires a large amount of flexibility and robustness to constantly manage external sensors and to run communication protocols, scheduled events and processes.

The OBU software shall be able to handle external protocols, e.g. communication protocols of a remote database or possible inspection software according to precise layer-based specifications. Most of the communications protocols operated by OBU software have XML as their highest layer; lower layer specifications depend on the kind of communication which takes place.

The subsequent sections describe the OBU software specifications and its macro-states using a functional approach.

The OBU software has to be adapted to the different hardware configuration of trailers and semi-trailers.

4.1.2 FSM DESCRIPTION

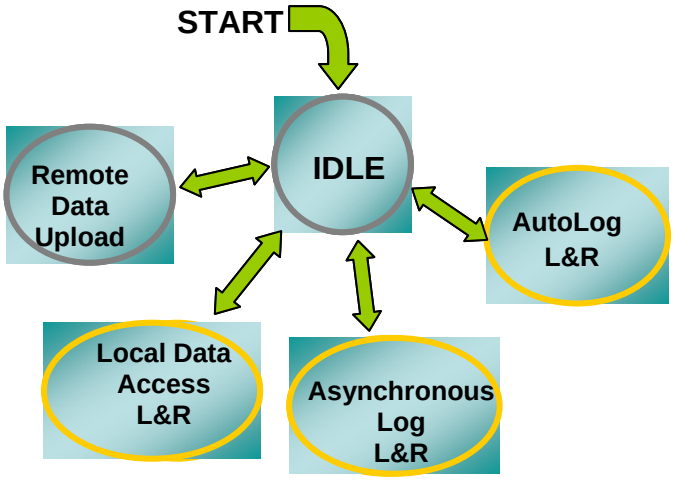


Fig. 4.1: OBU Software FSM – Semi-trailer Configuration

Figure 4.1 shows macro-states for the software. In the figure some states are marked with "L&R" (Log and Record), such as "AutoLog", "Local Data Access" or "Asynchronous Log" states. This means that the system when within those states shall log a record in the internal memory.

4.1.2.A Idle state

The OBU software's initial state shall be what is known as an idle state. In this state the software polls and supervises internal timers or external events that may produce a state change. Changes may be towards "AutoLog", "Remote Data Upload", "Local Data Access" or "Asynchronous Log" state.

4.1.2.B AutoLog state

AutoLog state is in charge of logging single records into the system memory. When logging automatically, the system acquires temperature readings, compares readings with tolerable ranges (defined in Annex I, Chapter VI, point 3.1 to Regulation (EC) 1/2005) and displays warnings on the CUI when out-of-range condition occur. AutoLog time is 5 minutes predefined value (one Log is acquired and stored every 5 minutes). Transmission time is predefined 60 minutes (12 logs are stored, 1 is transmitted)

4.1.2.C Local Data Access state

Local Data Access state occurs when the system is interacting with an external device through the local data port. In this state, in case of inspection, the OBU software and the external device shall respect the application protocol to access the data stored in the internal memory (i.e. without modifying). The OBU software logs an inspection record.

4.1.2.D Remote Data Upload state

The Remote Data Upload state occurs when a record has to be transmitted to a remote receiver.

When in Remote Data Upload state, the OBU software has to open a valid communication towards a remote receiver, send the records to be transmitted and verify correct transmission. If for any reason the communication link is not available (poor cellular network coverage), the OBU software shall return more frequently to the Remote Data Upload state until all records are transmitted, without compromising other scheduled tasks.

4.1.2.E Asynchronous Log state

Events which are not initiated by the internal timers, such as loading/unloading, coupling/uncoupling, or data entry by the driver trigger the creation of appropriate records in the system log and subsequent transmission to the remote receiver.

4.2 LOGGING EVENTS

This section describes the events which shall generate a log record.

States / Record type	Priority level	Auto Log state	Asynchronous state	Upload state	Download state
Position/time	Mid	X	X	X	X
Temperature	Mid	X			
Plug/Unplug second OBU	Mid		X		X
Trailer presence	Low		X		
Loading door(s)	High		X	X	
Warnings					
- Temperature	High		X	X	
- Sensors failure	High		X	X	
- OBU tamper	High		X	X	
- OBU battery state	Low		X		
- OBU main power	Low		X		
- OBU temperature	Low		X		
Transmission	Mid			X	
Login	High				X

Tab. T 4.1: Relation between record types and states of the system as well as their level of priority

As shown in Tab 4.1, the system shall assign 3 different priority levels according to the importance of the given events. In the case that two or more events occur at the same time and/or an event occurs while another event is processed, the system shall recognise their priority levels and process the event(s) of higher priority first. After a high- or mid-level priority event is logged, the OBU software is in charge of resuming normal operation starting a new set of automatic logs, i.e. under normal conditions the next automatic record shall be logged in the normal intervals after the event occurred and the next record shall be sent to the remote receiver in the predefined interval.

4.2.1 Automatic Log

Periodic automatic logs provide track of the vehicle, time and temperature(s). Single records shall be logged automatically by the system at approximately 5 minute intervals; Asynchronous event and Data Upload processing may affect logging intervals.

4.2.1.A Positioning and time

Positioning and time has always to be logged and all other data logged are referred to the position and time.

4.2.1.B Temperature

Temperature related records are generated by the automatic log procedure. The OBU compares the temperature reading with the appropriate temperature range according to Annex I, Chapter V, point 3.1 to Regulation (EC) 1/2005, adding an out-of-range label to the record if necessary. For possible further temperature ranges see Tab T 1.1 to this document.

When a temperature event, understood as an out-of-range temperature event or back-to-range temperature is verified, a single record containing the relevant information shall be sent to the CUI, logged to the memory and uploaded with a high-level priority.

4.2.2 Asynchronous events

4.2.2.A Plug/Unplug second OBU

In a truck the event of connecting or disconnecting a trailer with an own OBU shall be logged and uploaded with a medium-level priority. The second OBU is identified by its vehicle identification number.

4.2.2.B Trailer Presence

Events regarding the connection to a tractor (in a semi-trailer) shall be logged and uploaded with a low-level priority.

4.2.2.C Loading Door(s)

Events regarding the loading door(s), and hence the loading door sensor(s), are considered as external events. When the loading door is opened or closed the system shall log a high priority record to be sent as soon as possible to the remote receiver. Afterwards, the system continues its normal operation, starting with a new set of automatic logs.

4.2.2.D Warnings

Warning events indicate events (malfunctioning, tamper, absence of device) that represent a potential risk to the system, its behaviour or its data Some of these events have the highest priority for the OBU software; each time a warning is generated or certain tamper events are identified, a single record shall be logged immediately and transmitted to the remote receiver as soon as possible. Tamper events priority overrides loading door or coupling priority; this means that any event which is considered a manumission event shall be logged instantly even if the loading door is in its open state.

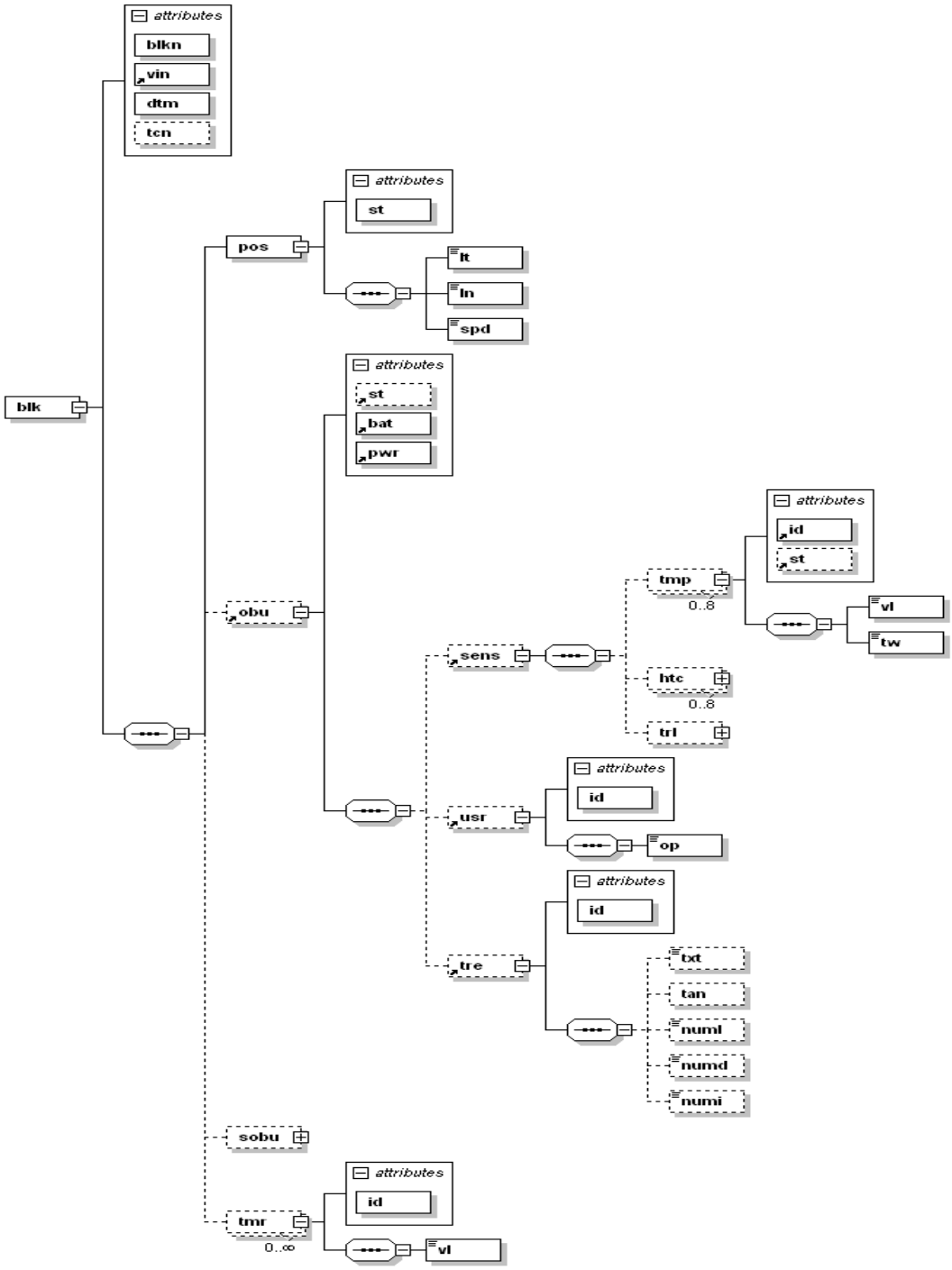
Manumission events include:

- Opening OBU box;
- Removing OBU;
- Unplugging sensors or communication wire(s);
- Unplugging power wire;
- Battery state low;
- OBU over-temperature.

After a warning event has been logged, the system shall resume normal operation starting with a new set of automatic logs.

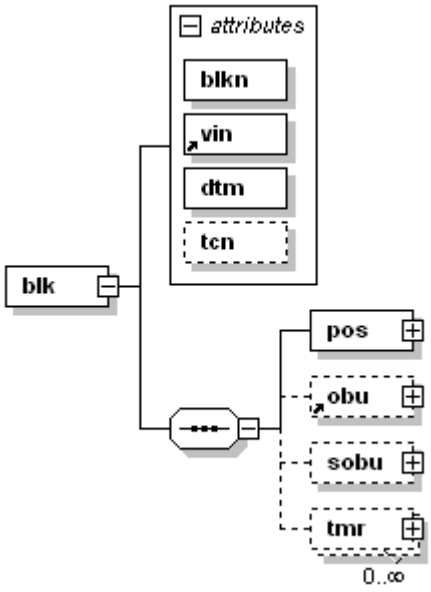
4.3 XML MESSAGE STRUCTURE

To ensure proper communication of the navigation system with a remote receiver or in download, a standardised message structure shall be defined. XML messages shall be used for the communication. Data transmitted shall be grouped in data blocks containing the data elements and attributes shown in the following XML schema diagram.



The element and attributes in the diagram are described below. A reference XML schema definition (XSD) is given in Annex V.

element blk

Description	The higher level of the XML tree structure
Diagram	
Type	Complex
Use	Required

attribute vin

Description	The vehicle registration number as defined in ISO 3779. It is used to identify the OBU. In a truck and trailer configuration it identifies the primary OBU transmitting the message.
Type	Xs:string
Use	Required
Format	maximum length: 20 characters

attribute blkn

Description	Identification number of the block acquired. This attribute is cyclic and incremental, i.e each data block is identified by a distinguishable number when stored in the memory.
Type	xs:integer
Use	Required

attribute dtm

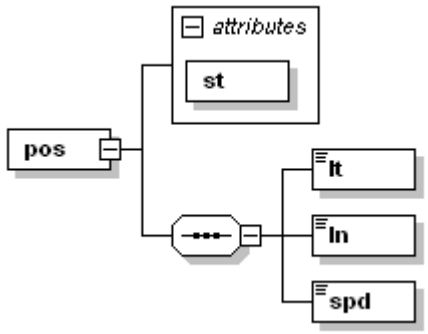
Description	The time and date of block acquisition in UTC and ISO 8601 format (year/month/day/hours/minutes/seconds). The time refers to GMT time.
Type	xs:dateTime

Use	Required

attribute tcn

Description	This attribute is compulsory when a new journey starts and refers to TRACES certificate number or in its absence to the journey log number. The tcn attribute is enabled for messages entered by the driver.
Type	xs:string
Use	Optional
Format	Length: 30 characters

element pos

Description	Information on position, speed and GNSS status
Diagram	
Type	Complex
Use	Required

attribute st

Description	Status of the GNSS receiver at the moment of the acquisition
Type	xs:string
Use	Required
Format	A: valid V: non valid

element ln

Description	The longitude where the acquisition has occurred.
Type	xs:string
Use	Required
Format	+/-ddd.mmmmmm where d=degree, m=decimal degree, + = East and - = West of the prime meridian. A point on the prime meridian is Eastern Hemisphere; the 180th meridian is Western maximum length : 11 characters (length according to the GALILEO/GPS receiver accuracy) mandatory “.” (point)

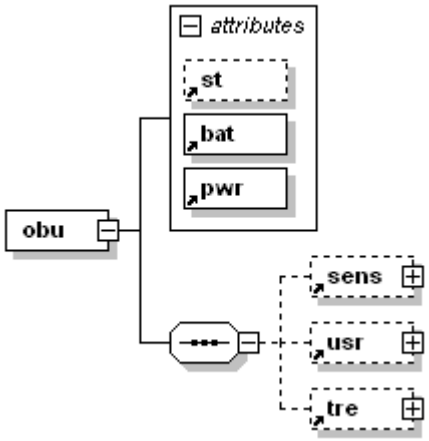
element lt

Description	The latitude where the acquisition has occurred.
Type	xs:string
Use	Required
Format	+/-dd.mmmmmm where d=degrees, m=decimal degrees, + = North and - = South of the equator. Points on the equator are assigned to the Northern Hemisphere maximum length : 11 characters (length according to the GALILEO/GPS receiver accuracy) mandatory "." (point)

element spd

Description	The speed of the vehicle in km/hour
Type	xs:decimal
Use	Required
Format	Maximum length : 6 characters At least one decimal number, separated by a mandatory "." (point) Valid range: 0 to 250

element obu

Description	Information related to the primary obu (identified by the attribute vin of blk)
Diagram	
Type	Complex
Use	Optional

attribute st

Description	Status of the obu. Is required only in case of alarm.
Type	xs:string
Use	Optional
Format	alarm: absent, malfunctioning, or subject to unauthorised action (unplugging, opening, removing)

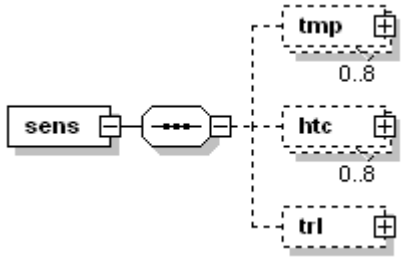
attribute bat

Description	Battery charge indicator
Type	xs:integer
Use	Required
Format	Range: 000 to 100

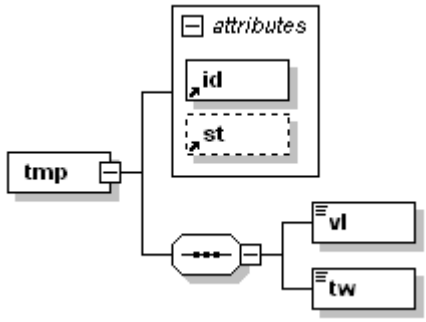
attribute pwr

Description	Power presence
Type	xs:string
Use	Required
Format	Single character P: powered U: un-powered

element sens

Description	Information related to the sensors connected to the obu. Appears when at least one sensor is active.
Diagram	
Type	Complex
Use	Optional

element tmp

Description	Temperature sensors. The OBU system shall be able to manage up to eight temperature sensors.
Diagram	
Type	Complex
Use	Optional

attribute id

Description	Identification number of the sensor. Each sensor shall be

	identified by unique numbers in increasing order.
Type	xs: integer
Use	Required

attribute st

Description	Status of the sensor. Is required only in case of alarm.
Type	xs:string
Use	Optional
Format	alarm: absent, malfunctioning, or subject to unauthorised action (unplugging, opening, removing)

element vl

Description	Value acquired value by the sensor.
Type	xs:string for htc and trl sensors xs:decimal for tmp sensors
Use	Required
Format	For tmp sensors: acquired temperature expressed in Degrees Celsius; range: -99.9 to 99.9 (°C). For loading doors sensors (htc), single character: O: open C: closed. For coupling sensors (trl), single character: C: coupled, U: uncoupled.

element tw

Description	Warning for temperature sensors. The tw element assumes an alarm status if the acquisition is out of the allowed temperature range. Changes in the tw (from ok to alarm or from alarm to ok) trigger a block record and an immediate transmission procedure. For the temperature thresholds see part 1 of this document. The temperature thresholds apply only to temperature sensors placed in the animal compartment(s). Possible external thermometers shall not generate alarm warnings.
Type	xs:string
Use	Optional
Format	Alarm Ok

element htc

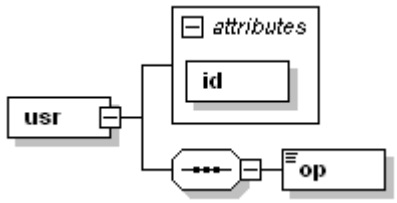
Description	Loading door sensors. (see above for the description of attributes and children elements). The OBU system shall be able to manage up to eight loading door sensors.
Diagram	
Type	Complex

Use	Optional

element trl

Description	Coupling sensors. (see above for the description of attributes and children elements). The OBU system shall be able to manage at least one coupling sensor.
Diagram	
Type	Complex
Use	Optional

element usr

Description	Information on an external authorised interventions on the OBU, such as set up, updates carried through the external access port. Possible official animal welfare inspection logging onto the navigation system would also be indicated.
Diagram	
Type	Complex
Use	Optional

attribute id

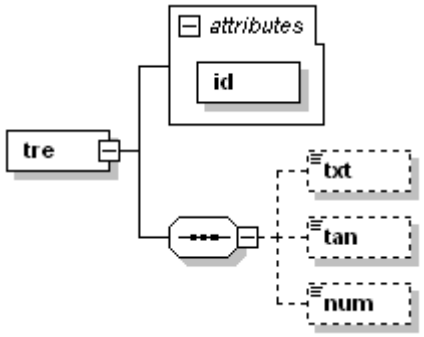
Description	Identification number of the intervention. Each intervention shall be identified by numbers in increasing order.
Type	xs: integer
Use	Required

element op

Description	Indicates the kind of operation that was accomplished in the OBU system. The operation is detected by the OBU software according to the communication protocol established for the setup or inspection software.
Type	xs:string
Use	Required
Format	setup: declares that an authorized setup operation has been carried out. ctrl: declares that an authorized control/inspection operation has been carried out.

element tre

Description	Information on regular or extraordinary events that may

	occur during a journey. Within this element regular events such as start of journey are saved and communicated to the remote receiver. Other more unusual events such as number of injured/dead animals may be also communicated. The tre element captures information entered by the driver of the vehicle using the CUI, logs it internally in a single record and then transmits it to the remote receiver. This element may not be present in each data block. The element is referred to a specific OBU. In a truck and trailer configuration the CUI should give the possibility to enter information related to the tre element independently for the primary and secondary obu.
Diagram	
Type	Complex
Use	Optional

attribute id

Description	Type of journey event.
Type	xs:string
Use	Required
Format	str: start. Indicates the start of a journey; rst: rest. Indicates the beginning of a break of the journey; cnt: resume. Indicates that the journey was resumed after a break; stp: stop. Indicates the end of a journey; msg: message. Indicates that an extraordinary event has occurred and the driver has successfully communicated it to the system trough the CUI.

element txt

Description	Messages entered by the driver regarding extraordinary event notifications. The txt element will indicate the nature of the event according to predefined messages available in the CUI.
Type	xs:string
Use	Optional
Format	Maximum length: 1000 characters

element tan

Description	This element appears when a new journey starts and refers to the species/categories of animals loaded. The tan element is only enabled for messages entered by the driver regarding type of animal loaded.
Type	xs:string
Use	Optional
Format	

Species/categories of animals		tan
pigs up to 30kg live weight	unweaned	pu3u
	weaned	pu3w
pigs more than 30kg live weight	compartment(s) not equipped with misting devices	pm3e
	compartment(s) equipped with misting devices	pm3n
cattle	unweaned	ctlu
	weaned	ctlw
sheep with long fleece	unweaned	slfu
	weaned	slfw
sheep with short fleece/goats	unweaned	slsu
	weaned	slsw
domestic equidae	unbroken	dequ
	other than registered	deqr

element numl

Description	The num element is only enabled for messages entered by the driver at start of the journey regarding the number of animals loaded.
Type	xs:integer
Use	Optional
Format	Range: 0 – 9999

element numi

Description	The num element is only enabled for messages entered by the driver at end of the journey regarding the number of animals injured.
Type	xs:integer
Use	Optional
Format	Range: 0 – 9999

element numd

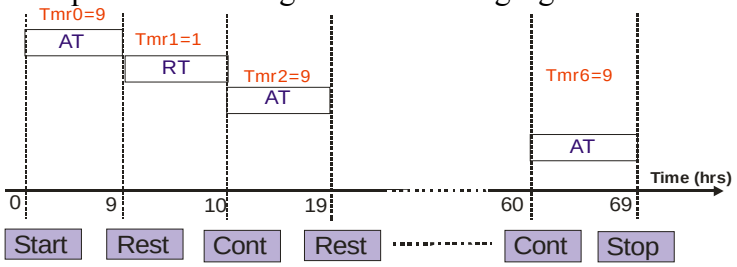
Description	The num element is only enabled for messages entered by the driver at end of the journey regarding the number of animals dead.
Type	xs:integer
Use	Optional

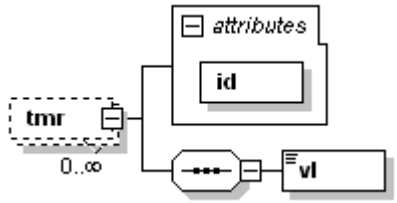
Format	Range: 0 – 9999

element sobu

Description	In a truck and trailer configuration a secondary OBU may present. This element acquired information related to the secondary OBU. All the attributes and children elements are the same as for the obu with the addition of a vin attribute which identifies the sobu.
Type	Complex
Use	Optional

element tmr

Description	This element indicates internal timers as regards travelling times and rest times during a journey. The timers shall respect the intervals foreseen in Table T 1.1. Even timers shall accumulate travelling times. Odd timers shall show rest period(s), that is the time during a journey when the vehicle stops to allow the animals to rest, either on the truck or unloaded. The accepted intervals vary depending the species and categories loaded. Timers shall operate during a journey as shown in an example of a time diagram in following figure.  The start of the journey is entered by the driver in the CUI. From that moment on, the first timer (**tmr0**) will start counting the travelling time. When stopping for a rest period, the driver shall enter the event through the CUI; this shall cause **tmr0** to stop, and **tmr1** to start counting. The information regarding the first part of the journey (travelling time) will be always available on each successive block until the journey is completed. The second timer (**tmr1**) will count until the driver enters that the journey is resumed. At that moment, **tmr1** stops counting and **tmr2** starts counting the travelling time. The end of the journey is entered by the driver through the CUI; at this point a block is transmitted with all timer information captured throughout the journey. Summing up the values recorded by the timers would show the duration of the journey. The system shall be able to accommodate in one journey at

	least travelling with 4 interruptions (rests).
Diagram	
Type	Complex
Use	Optional

attribute id

Description	Identification number for each time interval
Type	xs:integer
Use	Required

element vl

Description	Duration of each time interval
Type	xs:duration
Use	Required

4.4 COMMUNICATION WITH A REMOTE RECEIVER

Communication between the OBU and a remote receiver takes place by means of several protocol layers (see figure 4.4).

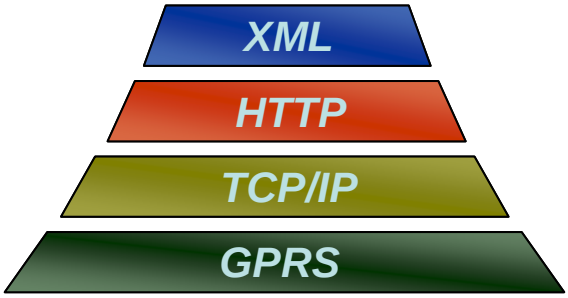


Fig. 4.4: Example for a Central Database Communication Layers

By using GPRS as the lower protocol layer, information may be shared among the OBU and the fixed network regulating transmission costs and guaranteeing optimal performance.
Communication should be short and concise, therefore an appropriate bandwidth is required, as the OBU is not stationary and the GPRS/GSM signal may fluctuate.
The next layer used is the TCP/IP protocol which is supported by the GPRS protocol.
Above the TCP/IP protocol layer, depending of the type of remote receiver, e.g. HTTP 1.1 or HTTPS could be employed.
Within the HTTP protocol, the PUT method is used to send information from the OBU to the remote receiver. An example is depicted in figure 4.5.

```
POST /[redacted]/Put HTTP/1.1
Host: [redacted]
Content-Type: application/x-www-form-urlencoded
Content-Length: length

inputXML=string
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<string xmlns="http://tempuri.org/">string</string>
```

Fig. 4.5: PUT/POST method example

The remote receiver has to provide the proper service which is able to support the required method.
The highest layer, where the recorded data are sent, is implemented using the XML message structure as described above.
After a communication attempt is accomplished, the remote receiver shall acknowledge receipt with a HTML page.

4.5 INSPECTION

The system shall allow downloading of data from the OBU memory on site, e.g. by the competent authorities for welfare during an inspection. Inspection events force the system to log a single record exhibiting the inspection event.
An inspection event is verified after a correct communication with the inspection software has been established and records were downloaded from the OBU. More details regarding the inspection software and protocols may be found in Annex VII.

4.6 INFORMATION SECURITY OBJECTIVES

Information security measures are required for protection against malicious and non-malicious threats, for example:

- Switching off/opening/removing of the OBU, disconnecting from power supply or disconnecting/removing sensors,
- Incorrect or incomplete data transmission from the OBU to the remote receiver,
- Interception of data transmitted over the GSM network (hooking),
- Unauthorised source masquerading as an authorised OBU, transmitting false data to the remote receiver which is not able to discriminate real from false data,
- Interception of data from the OBU to the remote receiver and substitution with false data (e.g. regarding travelling time, temperature).
- Unauthorised access to the remote receiver's database intercepting or modifying correctly received and stored data,
- Manipulation of correctly received data by an authorised user (e.g. deletions of data, changing vehicle ID, travelling time, temperature) at the remote receiver's database.

Elements of an information security system are means and techniques for assuring integrity, authenticity and confidentiality of the data generated by the navigation system.

4.6.1 Data Integrity

Data integrity is the property whereby data has not been altered in an unauthorised manner since the time it was created, transmitted, or stored by an authorised source. Operations which invalidate integrity include:

- Insertion of bits, including entirely new data items from fraudulent sources,
- Deletion of bits (short of deleting entire data items),
- Re-ordering of bits or groups of bits,
- Inversion or substitution of bits,
- Any combination of the above (e.g. message splicing, re-use of proper substrings to construct new or altered data items).

Data integrity includes the notion that data items are complete.

4.6.2 Message Authentication

Message authentication corroborates specific data with its original source (proof of origin). Message authentication techniques include

- Use of shared secret keys (message authentication code - MAC),
- Use of digital signature schemes (public key, private key).

Authentication schemes require a key management system.

4.6.3 Confidentiality

Confidentiality assures that the content of a message is only available to authorised parties. Confidentiality is achieved by means of encryption using

- shared secret keys (same key is used to encrypt and to decrypt the message),
- public key encryption (only the holder of the private key can decrypt the message).

Confidentiality schemes require a key management system.

An information security system shall be set up to ensure data integrity/security, however, security measures result in additional costs and the appropriate trade-offs between security and costs have to be found. For the OBU information safety requirements are defined in Chapter 2.2. Regarding data transmission and remote receiver's database solutions depend on the type of remote receiver. Therefore, no single, simple solution can be offered. Possible requirements are listed in Annex VIII.

ANNEX I TEMPERATURE, TRAVELING AND RESTING TIME REQUIREMENTS

Actual and possible future temperature, traveling and resting time requirements:

Regarding temperatures, traveling and resting times the thresholds in the table below show the requirements as set out in Regulation 1/2005 (rose marked fields) and possible further differentiations in species and categories of livestock which should be taken into account for the design of satellite navigation systems (light yellow marked fields).

Species/category of animals		Min temperature °C		Max temperature °C		Traveling times (hours)	Rest time (hours)
		Reg. 1/2005	Draft COM Reg.	Reg. 1/2005	Draft COM Reg.		
pigs up to 30kg live weight	unweaned	+ 5	+ 14	+ 30	+ 29	9 + 9	≥ 1
	weaned					24	0
pigs more than 30kg live weight	compartment(s) not equipped with misting devices		+ 10		+ 25		
	compartment(s) equipped with misting devices		+ 10		+ 30		
cattle	unweaned		+ 5		+ 27	9 + 9	≥ 1
	weaned					14 + 14	
sheep with long fleece	unweaned		0		+ 25	9 + 9	
	weaned					14 + 14	
sheep with short fleece/goats	unweaned		+ 10		+ 29	9 + 9	
	weaned					14 + 14	
domestic equidae	unbroken		+ 5		+ 25	9 + 9	
	other than registered					24	0
Other animals/goods		--	--	--	--	--	--

Tab. T-A-I-1: Temperature, traveling and resting thresholds rose: requirements of Regulation 1/2005, light yellow: possible future differentiations regarding categories and temperatures)

ANNEX II GRPS/GMS SPECIFICATIONS

Functional Specifications for the GPRS/GSM module

Feature	Description
Controls	Complete AT-Command set for GSM/GPRS (V. 07.07 and 07.05 included) ;
GSM/DCS	Transmit and Receive Frequencies: ▪ RX (E-GSM 900): da 925 a 960 MHz ▪ RX (DCS 1800) : da 1805 a 1880 MHz ▪ TX (E-GSM 900): da 880 a 915 MHz Class and Transmission power: ▪ Classe 4 (2W) per E-GSM ▪ Classe 1 (1W) per DCS
GPRS	GPRS Multislot Class 8 Multislot Class 2 support PBCCH support CS1 to CS4 Coding
DATA/FAX	Asynchronous data circuit transparent and not transparent up to 14.400bits/s Fax Group 3 compatible
GSM RF Performance	*Receiver:* E-GSM900 Ref. Sensitivity = -104dBm static & TU-High DCS1800 Ref. Sensitivity = -104dBm static & TU-High Selectivity @ 200KHz > +9dBc Selectivity @ 400KHz > +41dBc Linear dynamic range: 63dB Co-Channel rejection ≥ 9dBc *Transmitter:* Minimum output power E-GSM900: 35dBm ± 5dBm @ 25°C Minimum output power DCS1800: 0dBm ± 5dBm @ 25°C
SMS	SMS MT, MO e SMS CB SMS storage into SIM-Card
SIM	3V rated SIM-Card Interface

ANNEX III GPS SPECIFICATIONS

Functional Specifications for the GPS receiver

Feature	Description
Receiver Type	L1 – frequency 1.575,42 Mhz C/A Code 12 Channels
Accuracy	Position: 2.5m CEP (Circular Error Polarity) 5m SEP (Spherical Error Polarity) Position DGPS or SBAS: 2m CEP (Circular Error Polarity) 3m SEP (Spherical Error Polarity) Velocity: 0,1m/s Time: 1 μs synchronized to GPS time
Sensitivity	-138dBm
Time to First Operation	FIX Time – Open sky or stationary requirements @ -125dBm signal strength Max Cold Start Time: 38s Max Warm Start Time: 35s Max Hot Start Time: <4.0s
Dynamic Behaviour	Full Scale Altitude: 5.000m minimum Full Scale Speed: 100m/s minimum Full Scale Acceleration: 4G minimum
Supported Protocols	Protocol Message: NMEA-0183 ASCII DGPS Protocol: RTCM SC-104, WAAS (GSW 2.3 s/w only) GALILEO Protocol: (to be specified)
Antenna Supply Voltage	3V or 5V – min 40mA rated

ANNEX IV LIST OF STANDARDS REFERRED TO

- **IEC 60068-2-64**
  Revision: 93 chg: crgd date: 10/00/93 environmental testing - Part 2: Test Methods Test Fh: Vibration Broad-Band Random (Digital Control) And Guidance - (edition 1 * bilingual * also see iec 60068 set (same as IEC 60068-2-36)
- **IEC 60068-2-56**
- **JIS C 60068-2-56** -Revision: 96 Chg: W/ REAF Date: 02/20/01 ENVIRONMENTAL TESTING PART 2: TESTS. TEST CB: DAMP HEAT, STEADY STATE, PRIMARILY FOR EQUIPMENT (ENGLISH * SAME AS IEC 60068-2-56)
- **IEC 60068-2-30**
- Environmental testing – Part 2-30: Tests – Test Db:Damp heat, cyclic (12 h + 12 h cycle)
- **IEC 60068-2-14**
- Basic environmental testing procedures : Part 2 – Tests – Test N : Change of temperature
- **ISO/DIS 11898-1:** Road vehicles -- Controller area network (CAN) -- Part 1: Data link layer and physical signaling
- **ISO/DIS 11898-2:**
  Road vehicles -- Controller area network (CAN) -- Part 2: High-speed medium access unit
- **ISO/CD 11898-3:** Road vehicles -- Controller area network (CAN) -- Part 3: Low-speed fault tolerant medium dependent interface
- **ISO/CD 11898-4:** Road vehicles -- Controller area network (CAN) -- Part 4: Time triggered communication
- **IEC 60529 Ed. 2.1 b:2001**
  Degrees of protection provided by enclosures (IP Code) CONSOLIDATED EDITION
  Edition: 2.1
  International Electrotechnical Commission
  27-Feb-2001
- **DIN EN 60529 (German – equal to IEC 60529)**
  Degrees of protection provided by enclosures (IP code) (IEC 60529:1989 + A1:1999); German version EN 60529:1991 + A1:2000
  DIN-adopted European Standard
  01-Sep-2000
- **DIN/VDE 0470-1**
  Degrees of Protection Provided by Enclosures (IP Code) - GERMAN ONLY - SAME as IEC 60529, EN 60529
  Deutsches Institut fuer Normung/ Verband der Elektrotechnik, Elektronik, Informationstechnik
  01-Nov-1992
- **ISO 8601**
  date and time format
- **ISO 17799** : Information Technology – Security Techniques – Code Of Practice For Information Security Management

ANNEX V XML SCHEMA DEFINITION (XSD)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<xs:element name="blk">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="pos">
					<xs:complexType>
						<xs:sequence>
							<xs:element name="lt">
								<xs:simpleType>
									<xs:restriction base="xs:string">
										<xs:length value="14"/>
									</xs:restriction>
								</xs:simpleType>
							</xs:element>
							<xs:element name="ln">
								<xs:simpleType>
									<xs:restriction base="xs:string">
										<xs:length value="14"/>
									</xs:restriction>
								</xs:simpleType>
							</xs:element>
							<xs:element name="spd">
								<xs:simpleType>
									<xs:restriction base="xs:decimal">
										<xs:minInclusive
value="0"/>
										<xs:maxInclusive
value="250"/>
										<xs:fractionDigits
value="1"/>
									</xs:restriction>
								</xs:simpleType>
							</xs:element>
						</xs:sequence>
						<xs:attribute name="st" use="required">
							<xs:simpleType>
								<xs:restriction base="xs:string">
									<xs:length value="1"/>
									<xs:enumeration value="V"/>
									<xs:enumeration value="A"/>
								</xs:restriction>
							</xs:simpleType>
						</xs:attribute>
					</xs:complexType>
				</xs:element>
				<xs:element ref="obu" minOccurs="0"/>
				<xs:element name="sobu" minOccurs="0">
```

```
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element ref="sens" minOccurs="0"/>
                                                        <xs:element ref="tre" minOccurs="0"/>
                                                        <xs:element ref="usr" minOccurs="0"/>
                                                </xs:sequence>
                                                <xs:attribute ref="vin" use="required"/>
                                                <xs:attribute ref="st"/>
                                                <xs:attribute ref="bat" use="required"/>
                                                <xs:attribute ref="pwr" use="required"/>
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="tmr" minOccurs="0" maxOccurs="unbounded">
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element name="vl" type="xs:time"/>
                                                </xs:sequence>
                                                <xs:attribute name="id" type="xs:integer"
use="required"/>
                                        </xs:complexType>
                                </xs:element>
                        </xs:sequence>
                        <xs:attribute name="blkn" use="required">
                                <xs:simpleType>
                                        <xs:restriction base="xs:integer"/>
                                </xs:simpleType>
                        </xs:attribute>
                        <xs:attribute ref="vin" use="required"/>
                        <xs:attribute name="dtm" type="xs:dateTime" use="required"/>
                        <xs:attribute name="tcn">
                                <xs:simpleType>
                                        <xs:restriction base="xs:string">
                                                <xs:length value="30"/>
                                        </xs:restriction>
                                </xs:simpleType>
                        </xs:attribute>
                </xs:complexType>
        </xs:element>
        <xs:element name="obu">
                <xs:complexType>
                        <xs:sequence>
                                <xs:element ref="sens" minOccurs="0"/>
                                <xs:element ref="usr" minOccurs="0"/>
                                <xs:element ref="tre" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attribute ref="st"/>
                        <xs:attribute ref="bat" use="required"/>
                        <xs:attribute ref="pwr" use="required"/>
                </xs:complexType>
        </xs:element>
        <xs:attribute name="st">
                <xs:simpleType>
```

```
                <xs:restriction base="xs:string">
                    <xs:enumeration value="alarm"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="id">
            <xs:simpleType>
                <xs:restriction base="xs:integer">
                    <xs:minInclusive value="1"/>
                    <xs:maxInclusive value="7"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="vin">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:maxLength value="20"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:element name="usr">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="op">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:enumeration value="ctrl"/>
                                <xs:enumeration value="setup"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="id" type="xs:integer" use="required"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="tre">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="txt" minOccurs="0">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:length value="1000"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="tan" minOccurs="0"/>
                    <xs:element name="numl" minOccurs="0">
                        <xs:simpleType>
                            <xs:restriction base="xs:integer">
                                <xs:minInclusive value="0"/>
                                <xs:maxInclusive value="9999"/>
                            </xs:restriction>
```

```
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="numd" minOccurs="0">
                        <xs:simpleType>
                            <xs:restriction base="xs:integer">
                                <xs:minInclusive value="0"/>
                                <xs:maxInclusive value="9999"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="numi" minOccurs="0">
                        <xs:simpleType>
                            <xs:restriction base="xs:integer">
                                <xs:minInclusive value="0"/>
                                <xs:maxInclusive value="9999"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="id" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="msg"/>
                            <xs:enumeration value="stp"/>
                            <xs:enumeration value="cnt"/>
                            <xs:enumeration value="rst"/>
                            <xs:enumeration value="str"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:complexType>
        </xs:element>
        <xs:element name="sens">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="tmp" minOccurs="0" maxOccurs="8">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="vl">
                                    <xs:simpleType>
                                        <xs:restriction base="xs:decimal">
                                            <xs:minInclusive value="-
99.9"/>
                                            <xs:maxInclusive
value="99.9"/>
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:element>
                                <xs:element name="tw">
                                    <xs:simpleType>
                                        <xs:restriction base="xs:string">
```

```
                                                    <xs:enumeration
value="ok"/>
                                                    <xs:enumeration
value="alarm"/>
                                                </xs:restriction>
                                            </xs:simpleType>
                                        </xs:element>
                                    </xs:sequence>
                                    <xs:attribute ref="id" use="required"/>
                                    <xs:attribute ref="st"/>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="htc" minOccurs="0" maxOccurs="8">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="vl">
                                            <xs:simpleType>
                                                <xs:restriction base="xs:string">
                                                    <xs:enumeration
value="O"/>
                                                    <xs:enumeration
value="C"/>
                                                </xs:restriction>
                                            </xs:simpleType>
                                        </xs:element>
                                    </xs:sequence>
                                    <xs:attribute ref="id" use="required"/>
                                    <xs:attribute ref="st"/>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="trl" minOccurs="0">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="vl">
                                            <xs:simpleType>
                                                <xs:restriction base="xs:string">
                                                    <xs:enumeration
value="U"/>
                                                    <xs:enumeration
value="C"/>
                                                </xs:restriction>
                                            </xs:simpleType>
                                        </xs:element>
                                    </xs:sequence>
                                    <xs:attribute ref="id" use="required"/>
                                    <xs:attribute ref="st"/>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:attribute name="bat">
```

```
                <xs:simpleType>
                        <xs:restriction base="xs:integer">
                                <xs:minInclusive value="0"/>
                                <xs:maxInclusive value="100"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="pwr">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:length value="1"/>
                                <xs:enumeration value="U"/>
                                <xs:enumeration value="P"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:attribute>
</xs:schema>
```

ANNEX VI – CONNECTORS AND I/O DEFINITIONS

A-VI-1 CAN-Bus Connector

The CAN-Bus Connector (CBC) has to be in a place of the truck/tractor/trailer/semi-trailer next to the mechanical coupling device to the other part of the vehicle where typically electrical and brakes pneumatic connections are achieved.
The connector shall be a "Harting" type, made by several manufacturers.
In the following figures and tables, complete description, reference part numbers and pin assignments are given.
All connectors, both cable and vehicle mating units, are HAN 6XX series by Harting or equivalent manufacturers.

Vehicle mating unit

The vehicle mating unit has predefined female contacts and is housed in a waterproof metal enclosure with protective cap. The part numbers reported under the figures are only indicative. A female insert has to match and be fully equivalent to the part number indicated.

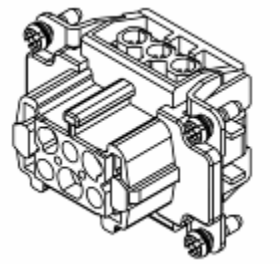


Fig. A-VI-1: Female insert

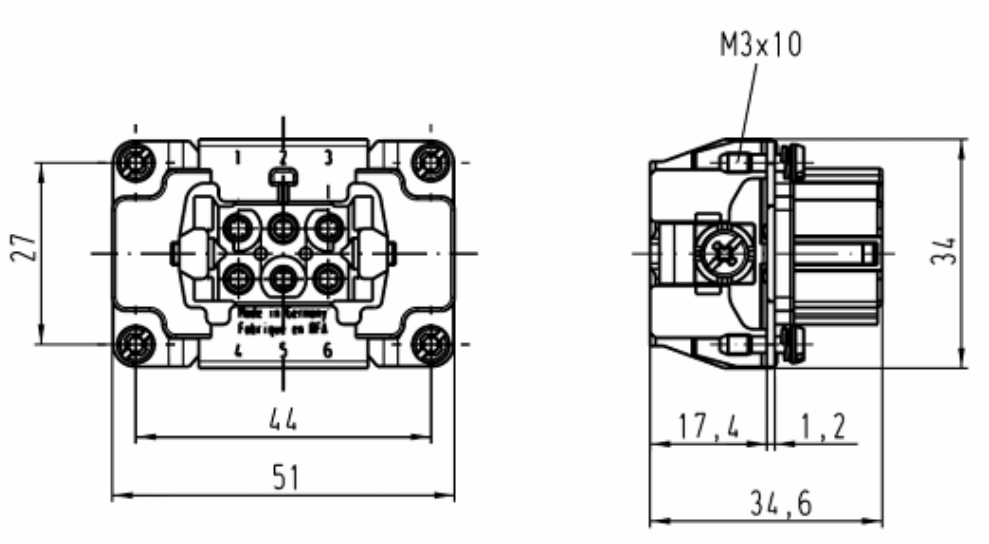


Fig. A-VI-2: Female insert dimensions (ref. Harting part nr. 09330062701)

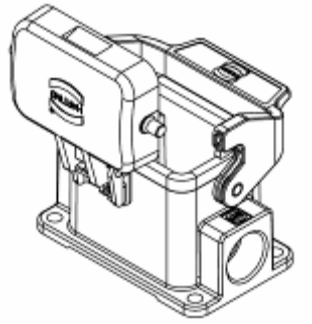


Fig. A-VI-3: Surface mounting housing

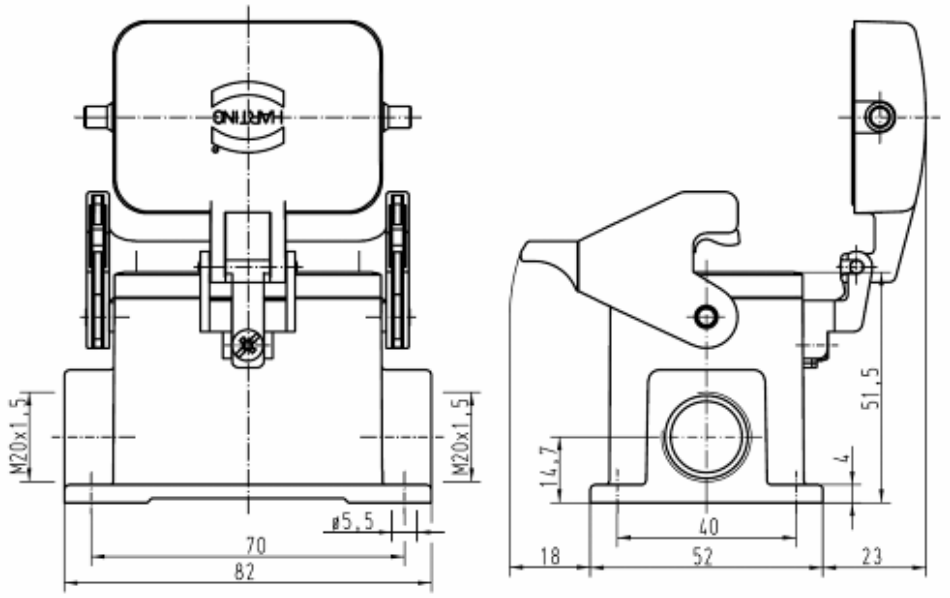


Fig. A-VII-4: Surface mounting housing drawing (ref. Harting part nr. 19300062295)

Cable assembly

Both ends of the cable between the parts of the system installed on the truck/tractor and trailer/semi-trailer shall be fitted with the male connector shown in the figures below.

The part numbers reported under the figures are only indicative. The male insert, shall be fully equivalent to the part number indicated.

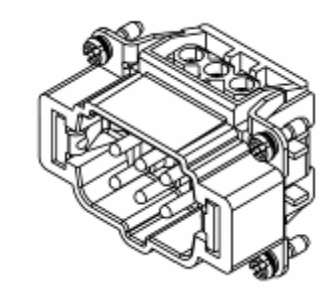


Fig. A-VI-5: Male insert

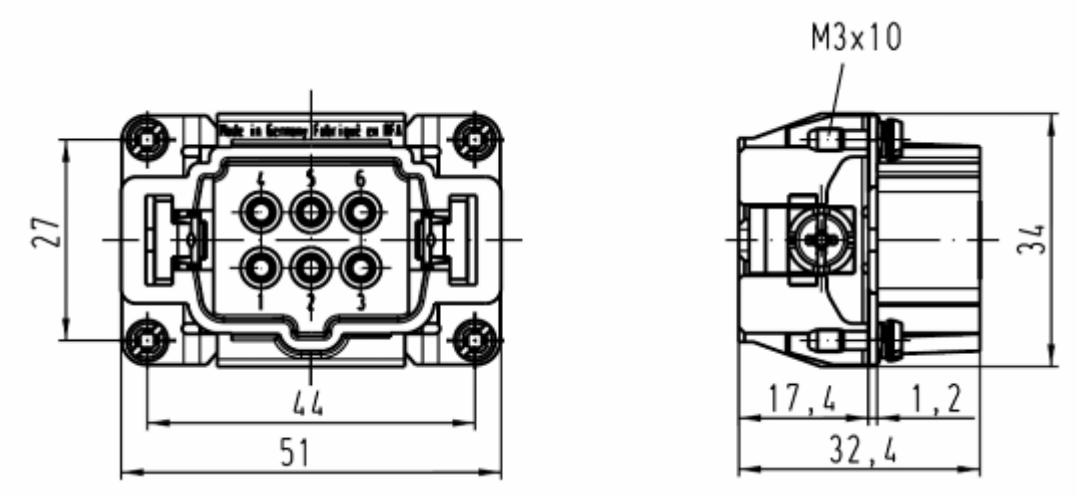


Fig. A-VI-6: Male insert dimensions (ref. Harting part nr. 09330062601)

Fig. A-VI-7: cable hood

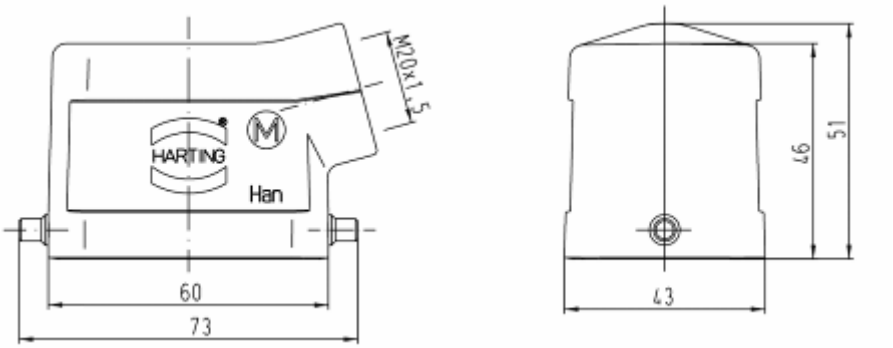


Fig. A-VI-8: Cable hood (ref. Harting part nr. 19370061540)

CAN-Bus connector pin assignment

In order to guarantee full compatibility between systems made by different manufacturers and installed on different vehicle parts that may be coupled together, the following pin assignment for the CBC shall be respected:

Pin nr	Node Name	Description
1	**+Vext**	12V Devices power line.
2	**CAN-H**	CAN high line
3	**CAN-L**	CAN low line
4	**GND**	Bus Ground
5	**Rfu**	Unconnected - reserved for future use

6	**Rfu**	Unconnected - reserved for future use

Tab. T-A-VI-1: CBC Pin assignment

- CAN-H and CAN-L are the two CAN-Bus connections. A twisted pair line will wire each device of the system. Please refer to the ISO11898 standard for a detailed description of CAN-L and CAN-H electrical specifications.
- GND is the main system ground provided by the OBU.
- Vext is the power line from the OBU, i.e. on the OBU, Vext is an output. All peripherals will be powered from this power line, i.e. on the peripherals/sensors, Vext is an input.

Vext provided by the OBU is defined as:

- Minimum Vext Voltage: 9,00 Vdc
- Maximun Vext Voltage: 14,00 Vdc
- Peak Output current: 3,00 Adc

A-VI-2 Antennas Connectors

No specific model is defined. The connectors to be used shall have appropriate protection and robustness according to the operating environment and installation location on the truck/trailer. All considerations and requirements that apply to the antennas connectors are listed in section 2.8.

A-VI-3 Local Data Access Connector

The local data access connection is implemented through an USB port for inspection and other purposes. A local user may plug a laptop PC or a pocket PC (or whatever host device) provided of a USB port and download data by means of an inspection software. The communication protocol between the OBU and the remote PC is defined in Annex VII.

The OBU shall thus be provided of a standard USB-B receptacle.

For all information regarding the USB bus, please refer to the Universal Serial Bus 1.1 and 2.0 specifications.

Depending on the location of the OBU on the vehicle, the USB shall be appropriately protected by a cap or cover that complies with the specifications in section 2.8 of this document.

A-VI-4 Other connectors

No specific models are defined

ANNEX VII – COMMUNICATION PROTOCOLS

A-VII-1 THE INSPECTION COMMUNICATION PROTOCOL

The inspection application shall be a software to download the logged blocks stored in the OBU internal memory onto a notebook PC, Pocket PC, etc. The connection is established using a USB port shared by both the OBU and the desktop or notebook, or pocket-PC USB port.

The application thus downloads the data formatted by the OBU according to the navigation system predefined XML data-format. It should show these blocks in a suitable way and in a friendly format, hiding the XML hierarchical structure as much as possible. Through the inspection software, it shall be possible to store the downloaded data on storage devices, such as hard disks, USB keys, external flash memories, etc. The software shall enable the user to run queries on the download data, such as journey times, temperatures. In this sense the application shall be able to work both in connected or stand alone mode.

The inspection software shall be a client application running on whatever computer platform. The communication between the client and the OBU is established lying on a suitable protocol made by data packet commands. The USB ports shall be configured with the same baud rate from both the client application and the OBU board. The default value may vary depending of the OBU hardware capabilities, a suitable value would be 230400 baud. A USB time out parameter shall be foreseen in order to avoid the USB infinitively waiting for the client commands, resulting in an unwanted pending idle state of the OBU. The connection for the download shall be interrupted (unplugged) by the OBU firmware if in another state than the Local Data Access state, in which it will be kept for a download. However, a time out shall also apply for this situation.

A-VII-1.1 Protocol data packet commands

The protocol data packet commands shall be divided in:

1. A **Login** command, logging onto the OBU and opening a communication session;
2. A **Download** command, downloading records from the OBU to a local device;
3. A **Logoff** command, by which the client application will notify the end of a communication session to the OBU and closing the communication.

A-VII-1.2 The Login command

The login command is sent from the client application to the OBU, having the USB cable plugged and active, i.e. not unplugged by the OBU firmware. With this command, the client will open a communication session with the OBU. After the login is accepted, the OBU will notify the client application to start the communication session.

The protocol for the Login command packet data shall be the following:

1. In order to start the handshaking, the client sends a BELL ASCII character to the OBU, i.e. **0x07** hexadecimal (0x in the following reference) value.
2. The OBU answers to this wakeup character with an ACK ASCII one, i.e. **0x06**.
3. The client sends back to the OBU the **Inspection Login Token** character, i.e. **0x05**.
4. The OBU board answers to this notification with an ACK ASCII character, i.e. **0x06**.
5. If the login is not successful for some reasons, the OBU will send a **0x00** token to the client, meaning an incorrect login was sent by the client. At this point, the client receives this “login failed” flag and is thus able to manage the situation properly, i.e. telling the user “Incorrect login”. The OBU has then to stay in listen mode, without unplugging the USB, in order to let the client retry to login. If the client does not retry to login, the communication will stop, and the OBU exits the “Local Data Access” state.
6. The OBU answers to the client with a **0x01** if login was successful.

A-VII-1.3 The Download command

The download command is sent from the client application to the OBU. If the OBU is not in the "Local Data Access" state, it will refuse a download request and will unplug the USB cable by the firmware.

The protocol for the Login command packet data will be the following:

1. In order to start the handshaking, the client will send a BELL ASCII character to the OBU, i.e. **0x07**.
2. If the OBU board is in the "Local Data Access", it will answer to this wakeup character with an ACK ASCII character, i.e. **0x06**. Otherwise it will answer with a **0x00** character, telling the client that download could not be executed, and the OBU will exit the communication session, unplugging the USB by firmware.
3. When the client receives the ACK character from the OBU, it will send to the OBU the **Inspection Download Token** character, i.e. **0x06**.
4. The OBU will start to download logged blocks to the client. The logged blocks are formatted according to the OBU XML file. The XML rows shall be organized in ASCII null terminating strings, without any number of character limitations. Each string shall end with the **0x00** null tail token character. The client will know the transmission ends when it reaches the last XML element node, i.e. the "</data>". After having sent the data, the OBU remains in the "Local Data Access" state and unplugs by firmware the USB from its port. The user shall manually re-plug to allow a new download session. Closing and restarting the client application has the same effect. The software shall inform the user about the need of re-plugging.

A-VII-1.4 The Logoff command

The Logoff command is sent from the client application to the OBU. With this command the client notifies to the OBU the closing of the communication session. The OBU than exits the "Local Data Access" state. After this command, any other download command will be ignored, until another successful inspection login is performed.

The protocol for the Logoff command packet data is as follows:

1. In order to start the handshaking, the client sends a BELL ASCII character to the OBU board, i.e. **0x07**.
2. The OBU board answers to this wakeup character with an ACK ASCII character, i.e. **0x06**.
3. The client sends to the OBU the **Inspection Logoff Token** character, i.e. **0x07**. This command resets the OBU Login state, leaving it ready for another login. The OBU unplugs the USB by firmware after this command.

A-VII-2 THE SET UP COMUNICATION PROTOCOL

The setup application is a software used by a installer for reading and writing the configuration parameters to the OBU. The connection is established using a USB cable shared by both the OBU board and the desktop or notebook USB port.

The setup software shall work in connected mode only. Communication shall be password protected by the software manufacturer. The setup application will talk with the OBU using a setup application protocol. The parameters exchanged with this protocol could be totally or partially displayed in the setup application, according to needs.

The setup software is a client application running on whatever computer platform may be installed both on a desktop and on a notebook computer. This software enables the installer to read the setup parameters from the OBU, change them locally on his machine and to write them back for a re-set of the OBU unit. The data exchange between the client and the OBU is established by data packet commands. The USB ports shall be configured with the same baud rate from both the client application and the OBU board. The default value may vary depending of the OBU hardware capabilities: a suitable value would be 230400 baud.

The client establishes the communication after the installer has plugged the USB cable into the OBU board related port. A USB time-out parameter shall be taken into account in order to avoid the USB to infinitively wait for the client commands, resulting in an unwanted pending idle state. An automatic read after a successful login may be adopted.

A-VII-2.1 Protocol data packet commands

The protocol data packet command is divided in:

1. A **Login** command, by which the installer logs onto the OBU, opening a communication session.
2. A **Read** command, by which the installer can read the currently stored OBU configuration parameters.
3. A **Write** command, by which the installer can write back to the OBU board the updated parameters.
4. A **Logoff** command, by which the client application notifies the session termination to the OBU, closing the communication.

With the login command, after a suitable handshaking, the client asks the OBU board to open a communication session. The client application sends to the OBU the user installer name together with the password related to the setup software. Both are ASCII null terminating strings, with a twenty characters maximum length.
The password is stored internally in the OBU firmware and is modified by the setup software and treated like the other configuration parameters. If the login is successful, the OBU notifies it to the client application letting the user start the communication session. The protocol for the Login command packet data is defined as:

1. In order to start the handshaking, the client sends a BELL ASCII character to the OBU, i.e. the **0x07** hexadecimal (0x in the following reference) value.
2. The OBU answers to this wakeup character with an ACK ASCII character, i.e. **0x06**.
3. The client sends back to the OBU the **Setup Login Token** character, i.e. **0x01**.
4. The OBU answers to this notification with an ACK ASCII character, i.e. **0x06**.
5. The client sends to the OBU the ASCII null terminating string password, made 20 ASCII characters maximum and with a **0x00** as tail token. The password string shall never be empty.
6. The OBU checks the received password with the one stored in the OBU firmware and related to the setup software. The check shall be done inside the OBU firmware in order to keep the password almost hidden within the OBU. If the password matches, the OBU sends to the client a **0x01** token and the login was successful.
7. If the login was not successful, i.e. the sent password did not match, the OBU sends to the client a **0x00** token, meaning an incorrect password was sent by the client. At this point, the client receives this “login failed” flag, i.e. telling the user “Incorrect password”. The OBU shall stay in the listening mode, without unplugging the USB, in order to let the client retry to login. If the client does not retry to login the communication stops without with the OBU remaining in the “Setup login mode”.
8. If the login was successful, the client sends the user name to the OBU. This is an ASCII null terminating string, made by 20 characters maximum ending with a **0x00** null character token. The string shall never be empty. The OBU will log the user name in a block and will enter in the “Setup login mode”, accepting some other commands from the client.

A-VII-2.2 The read command

The read command is sent from the setup application to the OBU. This OBU shall be in the “Setup login mode”. If not in this mode, the OBU refuses the read request and unplugs the USB cable by the firmware.
The protocol for the read command packet data is defined as:

1. To start the handshake, the client sends a BELL ASCII character to the OBU, i.e. **0x07**.

2. If the OBU is in the “Setup login mode”, it answers to this wakeup character with an ACK ASCII character, i.e. **0x06**. Otherwise, i.e. if the OBU is not in the “Setup login mode”, it answers with a **0x00** character and exits the communication session, unplugging the USB by firmware.
3. If successful, the client sends to the OBU the **Setup Read Token** character, i.e. **0x02**.
4. After having recognized the requested command, the OBU starts sending to the setup client all the parameters. The following items are sent in the order specified and are all intended as sent from the OBU to the setup client. The OBU just listens to the incoming group of characters and manages them in the proper way (ref. setup software).
5. The **serial number** string, representing the OBU serial number. The string is a sequence of ASCII characters terminated by NULL.
6. The **company name** string, i.e. the company owning the truck. The string is a sequence of ASCII characters terminated by NULL.
7. The **plate number**, i.e. the truck/trailer plate number. The string is a sequence of ASCII characters terminated by NULL.
8. The **chassis number**, i.e. the truck/trailer chassis number. The string is a sequence of a ASCII characters terminated by NULL.
9. The **autolog time**, i.e. the time between two subsequent logs to be stored and sent to the remote receiver. The value shall be set to minutes. The value shall be sent as a sequence of 4 bytes, each one defined as a single precision real number according to the IEEE floating point number convention.
10. The **internal autolog time**, i.e. the cycle in which the OBU logs automatically a data block that has not to be sent to the remote receiver, but could be downloaded to the inspector software. It is intended and useful for debugging purposes. The value shall be set to minutes. The value shall be sent as a sequence of 4 bytes, as a single precision real number according to the IEEE floating point number convention.
11. The **setup password**, i.e. the password used by the installer to login OBU from the setup client application. The setup password is defined as a string with twenty characters maximum length ending with a NULL character. The string shall never be empty.
12. The **reset counter**, i.e. the number of times the OBU has been manually reset and hence restarts from the beginning, e.g. due to a power failure. The value shall be sent as a sequence of 4 bytes, as a single precision real number according to the IEEE floating point number convention.
13. The **PIN**, i.e. the phone SIM PIN. The string is a sequence of an ASCII characters terminated by NULL.
14. The **PUK**, i.e. the phone SIM PUK. The string is a sequence of an ASCII characters terminated by NULL.
15. The **APN server**, i.e. the Access Point Name server. The string is a sequence of ASCII characters terminated by NULL.
16. The **APN user name**, i.e. the Access Point Name user name. The string is a sequence of an ASCII characters terminated by NULL.
17. The **APN password**, i.e. the Access Point Name password. The string is a sequence of an ASCII characters terminated by NULL.
18. The **Send Data Type**, i.e. the protocol used by the OBU for sending periodically the logged block to the remote receiver. It shall be a byte and the allowed values are:
    - **0x00**, no format adopted. The logged blocks will not be sent to the remote receiver.
    - **0x01**, SMTP protocol, i.e. email format.
    - **0x02**, FTP protocol, the remote receiver shall expose a FTP server.
    - **0x03**, HTTP protocol, the remote receiver shall expose a HTTP server managing web services, for instance.
19. The **SMTP server**, i.e. the server used for sending the emails using the SMTP protocol. The string is a sequence of ASCII characters terminated by NULL.

20. The **SMTP sender name**. The string is a sequence of ASCII characters terminated by NULL.
21. The **SMTP sender address**. The string is a sequence of ASCII characters terminated by NULL.
22. The **SMTP sender domain**. The string is a sequence of ASCII characters terminated by NULL.
23. The **email receiver address**. The string is a sequence of ASCII characters terminated by NULL.
24. The **email cc receiver address**. The string is a sequence of ASCII characters terminated by NULL.
25. The **email subject**. The string is a sequence of ASCII characters terminated by NULL.
26. The **FTP server**, i.e. the address of the central FTP server. The string is a sequence of a ASCII characters terminated by NULL.
27. The **FTP user name**. The string is a sequence of ASCII characters terminated by NULL.
28. The **FTP password**. The string is a sequence of ASCII characters terminated by NULL.
29. The **FTP put path**, i.e. the path in which the file is put. The string is a sequence of ASCII characters terminated by NULL.
30. The **FTP file name**, i.e. the file name of sent file. The string is a sequence of ASCII characters terminated by NULL.
31. The **TCP server**. The string is a sequence of ASCII characters terminated by NULL.
32. The **TCP port**. The string is a sequence of ASCII characters terminated by NULL.
33. The **TCP service**, i.e. the string related to the web service the remote receiver will expose. The typical use is a HTTP command. The string is a sequence of ASCII characters terminated by NULL.
34. The **EOT byte**, i.e. the End of Transmission ASCII character **0x04**.

A-VII-2.3 The write command

The write command is sent from the setup application to the OBU. The OBU shall be in the "Setup login mode", i.e. the user shall have done a successful login before trying to write data to the OBU. If the OBU board is not in the above mentioned state, it refuses the read request and unplugs the USB cable by firmware.

The meaning and the length of each exchanged bunch of bytes is the same as described in the chapter above for the Read command. The parameters the client uploads to the OBU shall be stored in the OBU memory. For this purpose the OBU sends, for each parameter, a synchronization character (synchro char) back to the client, signalling that the next parameter can be sent. The protocol for the Write command packet data is defined as:

1. In order to start the handshake, the client sends a BELL ASCII character to the OBU, i.e. **0x07**.
2. If the OBU is in the "Setup login mode", it answers to this wakeup character with an ACK ASCII character, i.e. **0x06**. Otherwise it answers with a **0x00** character, telling the client the write operation could not be performed and the OBU exits the communication session unplugging the USB by firmware. If successful, the client sends to the OBU the **Setup Write Token** character, i.e. **0x03**.
3. The OBU recognizes the Setup Write command and sends back a synchro char **0x00**, telling the client it can start to upload the parameters.
4. The setup receives the synchro char and sends the **Serial Number** string.
5. The OBU receives it and sends back the **0x00** synchro character.
6. The setup receives the synchro char and sends the **Company name** string.
7. The OBU receives it and sends back the **0x00** synchro char.
8. The setup receives the synchro char and sends the **Plate Number** string.
9. The OBU receives it and sends back the **0x00** synchro char.
10. The setup receives the synchro char and sends the **Chassis Number** string.
11. The OBU receives it and sends back the **0x00** synchro char.

12. The setup receives the synchro char and sends the **Autolog time** single precision float number.
13. The OBU receives it and sends back the **0x00** synchro char.
14. The setup receives the synchro char and sends the **Internal Autolog time** single precision float umber.
15. The OBU receives it and sends back the **0x00** synchro char.
16. The OBU receives it and sends back the **0x00** synchro char.
17. The setup receives the synchro char and sends the **Setup password** string.
18. The OBU receives it and sends back the **0x00** synchro char.
19. The setup receives the synchro char and sends the **Phone PIN** string.
20. The OBU receives it and sends back the **0x00** synchro char.The setup receives the synchro char and sends the **Phone PUK** string. The OBU receives it and send back the **0x00** synchro char.
21. The setup receives the synchro char and sends the **Access Point Node server** string.
22. The OBU receives it and sends back the **0x00** synchro char.
23. The setup receives the synchro char and sends the **Access Point Node user name** string.
24. The OBU receives it and sends back the **0x00** synchro char.
25. The setup receives the synchro char and sends the **Access Point Node password** string.
26. The OBU receives it and sends back the **0x00** synchro char.
27. The setup receives the synchro char and sends the **Sent data type** byte.
28. The OBU receives it and sends back the **0x00** synchro char.
29. The setup receives the synchro char and sends the **SMTP server** string.
30. The OBU receives it and sends back the **0x00** synchro char.
31. The setup receives the synchro char and sends the **SMTP sender name** string.
32. The OBU receives it and sends back the **0x00** synchro char.
33. The setup receives the synchro char and sends the **SMTP sender address** string.
34. The OBU receives it and sends back the **0x00** synchro char.
35. The setup receives the synchro char and sends the **SMTP sender domain** string.
36. The OBU receives it and sends back the **0x00** synchro char.
37. The setup receives the synchro char and sends the **email receiver address** string.
38. The OBU receives it and sends back the **0x00** synchro char.
39. The setup receives the synchro char and sends the **email cc receiver address** string.
40. The OBU receives it and sends back the **0x00** synchro char.
41. The setup receives the synchro char and sends the **email subject** string.
42. The OBU receives it and sends back the **0x00** synchro char.
43. The setup receives the synchro char and sends the **FTP server** string.
44. The OBU receives it and sends back the **0x00** synchro char.
45. The setup receives the synchro char and sends the **FTP user name** string.
46. The OBU receives it and sends back the **0x00** synchro char.
47. The setup receives the synchro char and sends the **FTP password** string.
48. The OBU receives it and sends back the **0x00** synchro char.
49. The setup receives the synchro char and sends the **FTP put path** string.
50. The OBU receives it and sends back the **0x00** synchro char.
51. The setup receives the synchro char and sends the **FTP put file name** string.
52. The OBU receives it and sends back the **0x00** synchro char.
53. The setup receives the synchro char and sends the **TCP server** string.
54. The OBU receives it and sends back the **0x00** synchro char.
55. The setup receives the synchro char and sends the **TCP port** string.
56. The OBU receives it and sends back the **0x00** synchro char.
57. The setup receives the synchro char and sends the **TCP service** string.
58. The OBU receives it and sends back the **0x00** synchro char.
59. The setup sends the **End Of Transmission** ASCII character **0x04**.

60. The OBU board waits for this character and than exits closing the transmission.

A-VII-2.4 The logoff command

The Logoff command is sent from the setup application to the OBU. With this command, after a suitable handshaking, the client notifies to the OBU closing the transmission session and the OBU exits the "Setup login mode". After this command, any other Read or Write command is ignored, unless successfully logged in the setup login mode again.

The protocol for the Logoff command packet data is defined as:

1. In order to start the handshaking the client sends a BELL ASCII character to the OBU, i.e. **0x07**.
2. The OBU answers to this wakeup character with an ACK ASCII character, i.e. **0x06**.
3. The client sends to the OBU the **Setup Logoff Token** character, i.e. **0x04**. This command resets the OBU Login state, leaving it ready for another login. The OBU unplugs the USB by firmware after this command. No other Read or Write commands are accepted and processed after the OBU board has been logged off.

A-VII-3 CAN APPLICATION PROTOCOL

A-VII-3.1 Introduction

The OBU shall be connected to the sensors and to the CUI according to a master-slave net fashion, in which the OBU acts as the master, while the other satellites act as slaves. A suitable bus net shall be adopted in order to let the packets to be exchanged throughout the NET. The exchanged messages will not be for data purposes only, but will be used for service purposes too, like system bootstrap, in order to let the OBU to configure the sensors and the CUI according to some factory stored parameters. Keeping in mind these purposes, a suitable bus shall be physically wired and hence a related protocol stack shall be implemented working on this bus, according to the ISO/OSI seven layer stack protocol specification, a.k.a. ISO standard 7498.

The CAN-Bus and the CANopen application protocol

CAN is a communication protocol based on the ISO 11898 specification, concerning the physical layer and the data link layer of this protocol. The messages running on this bus, a.k.a. CAN frames, can be found in the related specification (ref. ISO 11898).

The reasons for choosing the CAN and the CANopen protocol for the navigation system can be summarized as follows:

1. The CAN protocol has been specifically designed for implementing bus net in automotive environments. The related controllers are built upon the characteristics that shall be fulfilled in this kind of systems.
2. The CAN controller chips natively implements the physical layer and data link layers. This means that the basic frames on which the CAN is based upon, are recognized and exchanged as native items.
3. The controller chips can manage the two lowest ISO / OSI layers, the remaining layers shall be defined using the CANopen application protocol.
4. The CANopen is an application layer protocol, based on open specifications made by the CAN In Automation organization. The open specifications are public domain. This is by far the most used CAN application protocol in the automotive world

A-VII-3.2 Overview

The OBU system CAN application protocol shall be based on CANopen. The high level OBU application protocol shall be supported by the CAN-Bus in its lower layers (see *CAN-Bus Specifications* section). The CAN application protocol is used within the OBU system as a robust communication protocol among the system peripherals. Figure A-VII-1 shows typical system

peripherals which are to be connected together with a CAN-Bus and use the CAN application protocol.

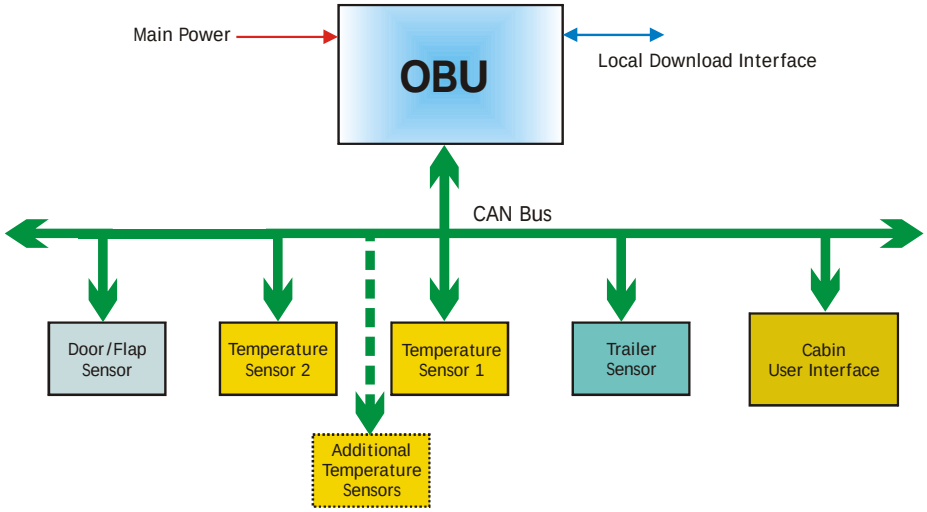


Fig. A-VII-1: OBU CAN peripherals

The OBU application protocol shall be a complex master/slave protocol that allows peripheral configuration, real-time data sharing, error control and a high degree of expandability. Analogue and digital sensors shall be supported. In addition, more elaborated messages may be shared among devices connected to the CAN-Bus, which guarantees a proper setting for managing the CUI.

CAN application protocol assures correct communication among different-constructors devices with minimal internal configuration. This could be done by means of an Object Dictionary which serves as an interface between the device application and the CAN application protocol. Figure 2.2 shows the ISO11891 CAN OSI model. The CAN application protocol is part of the *Specific Application Layer*, where it communicates with the device application using the Object Dictionary as illustrated in figure A-VII-2.

The Object Dictionary contains all possible information regarding the device application. It is possible for a master device to access a slave Object Dictionary gaining access to its application data, communication and configuration parameters.

The OBU is also in charge of the network management. Peripherals communication state may be changed and controlled by the OBU using network management messages. This is useful to guarantee correct communication among network nodes.

Another important issue of the CAN application protocol is its ability to monitor nodes communication capacity by means of a heartbeat message. When the OBU recognizes a faulty node, it asks for the node current state; the node answers with a heartbeat message. In order to implement with success the CAN application protocol, different kinds of rules shall be followed related to devices and messages configurations. In the following, the term “master/network-master” will refer to the OBU, and the term “slave/network-slave” will be used for system peripherals (CUI and sensors).

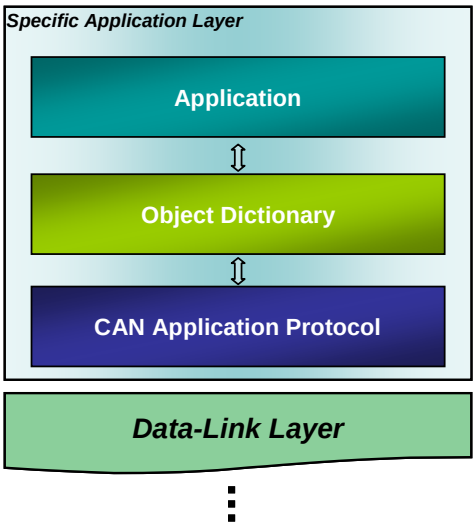


Fig. A-VII-2: Application Layer

A-VII-3.3 Protocol Messages

CAN application protocol messages may be of different kind related to different purposes:

- Process Data Objects (PDO)
- Service Data Objects (SDO)
- Network Management (NMT)
- Special Function Objects: Boot-up and Heartbeat.

A-VII-3.3.1 Process Data Objects (PDO)

Process data objects are used to transmit information from one node to (an)other node(s) in the CAN network. A single PDO is mapped to a single CAN frame and therefore is able to carry up to 8 bytes of application information.

Within the OBU system, PDOs are used widely to transmit data from sensors to the OBU, and to transmit some important information from the OBU to the CUI and vice versa.

PDOs shall be seen as fast data carriers that most of the time are requested by the network master (OBU) to the network slaves (sensors) by means of a remote transmit request message (see figure A-VII-3). Even though both mechanisms are very similar, there are slight differences that make each of them appropriate to specific situations according to the implementation desired.

Remote frames are typically a simple method to request information because it does not presume a configuration procedure and gives complete control to the OBU. The master shall poll slaves frequently in order to know their state.

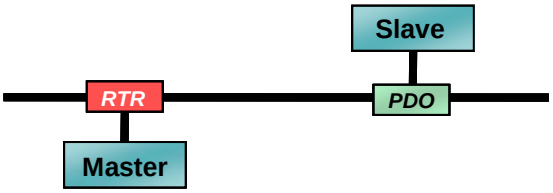


Fig. A-VII-3: Requested PDO

Requested PDOs are used among all sensors in the OBU system. This is the fundamental protocol object that the OBU uses to poll its peripherals.

A-VII-3.3.2 Service Data Objects (SDO)

Service data objects are used within the CAN application protocol to configure peripherals and gain access to device application data. SDOs are used to write or read Object Dictionary entries of a peripheral.

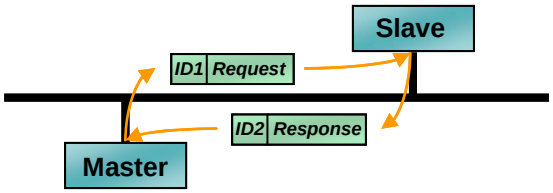


Fig. A-VII-4: SDO Communication

SDOs transport protocol uses peer-to-peer communication, behaving more as a client/server protocol than a master/slave one. SDO-client in the OBU system shall always be the OBU which request access to the SDO-server (peripherals) Object Dictionary. SDOs have lower priority than PDOs because their use is intended for not critical communications. Information reliability is sustained by required confirmation after each message. Figure A-VII-4 illustrates the way a typical SDO communication takes place.

SDOs are useful to configure sensors and CUI communications and internal parameters. SDOs are also used to get and control information related to the CUI, such as the user commands and the information messages.

A-VII-3.3.3 Network Management (NMT)

Network management objects are used by the OBU (master) to control and verify communication status of the different nodes in the network. This kind of message is useful to guarantee the correct functioning of nodes that are communicating with each other along the network. NMTs provide a powerful mechanism to inform slave nodes when to start operating, using a suitable configuration procedure.

The most important concept behind NMTs messages is the fact that each one of the slaves shall have a predefined communication state machine which may be controlled by the network master. The state machine is presented in figure A-VII-5.

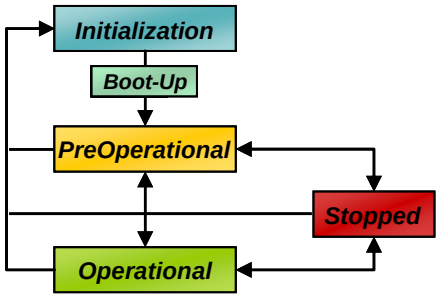


Fig. A-VII-5: Slave node state machine

Each slave node shall start in its “initialization” state; in this state the node executes its internal configuration and boot-up procedures. After that, the node shall go automatically to its “PreOperational” state. While changing state, the slave-node shall send a Boot-up message to the network, indicating in this way to the network master that the device is available to receive configuration commands by means of SDOs. “PreOperational” state is used by the master to configure the slave if necessary. Configuration may take place in terms of communication, that is indicating protocols to be used, and in terms of device functions, such as CUI welcoming message. After this is done, the master shall send to the slave a start command indicating that normal functioning shall start, i.e. imposing the “Operational” state to the network slave(s).

The network master (OBU) has the possibility to change a node’s state whenever it is needed; this is done by means of proper NMT messages. If, for instance, a particular node configuration needs to be changed, the master may configure its run-time by means of SDOs and then force the node to go to its regular initialization state. It is also possible for the master to stop the execution of a node if for any reason its behaviour is erroneous; this option reduces unwanted network collisions that may be produced by inoperative slaves. NMT messages consent a great deal of

flexibility in terms of network management. One of the most important features of NMTs is the possibility to send broadcast messages to all network nodes, allowing the network master to start slaves operation at the same time.

A-VII-3.3.4 Special Function Objects (SFO)

Objects other than those that are described before are defined as Special Function Objects. SFOs are used by the network master or network slaves to provide solutions to specific situations that may be found in a CAN network. SFOs offer appropriate protocols that allow error control and node monitoring features.

Boot-up messages are used by node slaves to indicate that their "Initialization" state has finished. Boot-up messages are useful to tell the master that a node is ready to start configuration procedure or regular operation. This kind of message is also valuable to let the network master know how many slaves are operational in the CAN network. The boot-up message is also useful to tell the network master that a node has been restarted or a new node has been installed. Heartbeat objects are generated by slave nodes. It indicates to the master that the sending node is still working properly. The heartbeat object is intended for notifying node communication errors. The Heartbeat message shall indicate the node current state (PreOperational, Operational or Stopped). In fact, a boot-up message is just a Heartbeat object with a special state value. In the OBU system, the communication control mechanism towards network slaves is provided by means of the repeated polling procedure; if, for any reason, a node fails answering the master poll, a Heartbeat message is requested and a control procedure is executed.

A-VII-4 CANOPEN BASED PROTOCOL FUNCTIONAL DESCRIPTION

A-VII-4.1 Messages Identifiers

Message identifiers are a fundamental part in any application protocol based on the CAN-Bus. CAN data-link and physical layer implementations support their operation on network identifiers associated to the type of message to be sent. This is done as an alternative to other network protocols where identifiers are associated to the network node. This section deals with CAN frame (low-layer) message identifiers that shall be employed according to the application object (high-layer).

CAN frames are also known as Communication Objects (COB); protocol messages are sent across a CAN network inside a COB. COBs are identified by their COB-ID, i.e. the CAN frame identifier, that determines the priority of the COB for the MAC sub-layer.

As described previously, CAN application protocol messages may be of different types. Protocol messages vary according to the kind of operation that is intended; there are different types of protocol messages: Process Data objects (PDO), Service Data Objects (SDO), etc. All of these messages or objects shall be mapped in one or several COBs.

According to CAN specifications (see *CAN-Bus Specifications* section) COBs may have 29-bit identifiers (extended frame); in the OBU system implementation, 11-bit identifiers (standard frame) are enough to map application protocol messages.

Application protocol messages are mapped inside COBs by selecting accurately the COB-IDs to be used. Some of the application messages use the COB-ID not only as an identifier of the type of message, but also as an identifier of the destination node. In the OBU application protocol it is possible to differentiate between a maximum of 127 nodes (7 bits); node IDs usually share the COB-ID 11-bit identifier. The COB-ID space has been divided in order to map protocol messages and node IDs according to their priority. The COB identifier space is presented in figure A-VII-6.

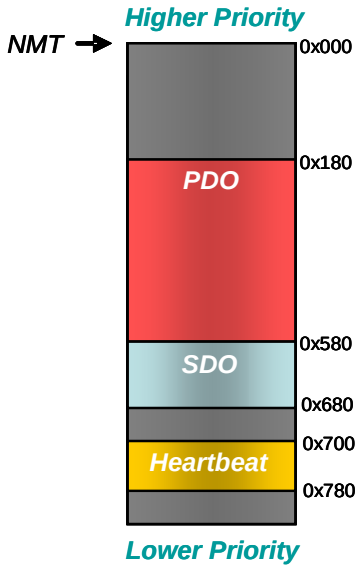


Fig. A-VII-6: COB-ID space

The following table shows the COB identifier allocation:

Application Protocol Object	COB-ID (hex)	Slave node behaviour
NMT	0x000	Receive only
PDO	0x180 + NodeID 0x200 + NodeID 0x280 + NodeID 0x300 + NodeID 0x380 + NodeID 0x400 + NodeID 0x480 + NodeID 0x500 + NodeID	Transmit PDO 1 Receive PDO 1 Transmit PDO 2 Receive PDO 2 Transmit PDO 3 Receive PDO 3 Transmit PDO 4 Receive PDO 4
SDO	0x580 + NodeID 0x600 + NodeID	Transmit Receive
Heartbeat	0x700 + NodeID	Transmit

Tab. T-A-VII-1:– COB-ID Allocation

Some protocol messages required the Node identifier in the COB-ID. Following table T-A-VII-1 it is clear that the Node ID is expected when transmitting PDO, SDO or Heartbeat messages. It is possible for each slave to have up to four possible PDOs and one SDO configurations. Most of the time, however, sensors shall use one PDO type. Node IDs are also mapped in order to guarantee all possible combinations according to the OBU system sensors possibilities. The following table shows the Node ID space:

Node – Sensor type	Node-ID (hex)
ALL (BROADCAST)	0x00
OBU	0x01
DOOR/HATCH	0x08 – 0x0F
TRAILER	0x10 – 0x17
CABIN USER INTERFACE	0x18 – 0x1F
TEMPERATURE SENSOR	0x20 – 0x27
TAMPER	0x68 – 0x6F
BATTERY	0x70 – 0x77
INTERNAL TIMER	0x78 – 0x7F

Tab. T-A-VII-2 – Node-ID Space

Tampers, batteries and internal timers are not real CAN network nodes. They are part of the XML sensors description, and therefore they are included in the node ID space for coherence. Protocol messages are defined in the CAN network by combining the values in the COB-ID space and in the Node-ID space.

A-VII-4.2 Node State Machine and Network Management Messages

The node state machine describes the procedure that a slave node shall guarantee in order to enable network management.

Figure A-VII-7 shows the node state machine. A node shall enter its "initialization" state after a HW-reset or Power-On. In this state, the node shall execute its basic node initialization procedures such as CAN controller initialization, application software initialization, etc.

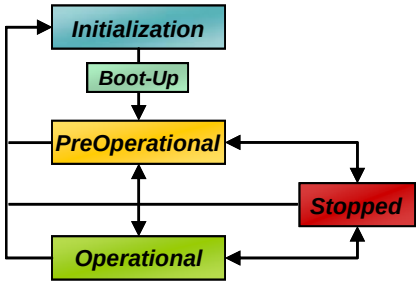


Fig. A-VII-7: Slave node state machine

After following initialization procedures, the node shall go to its "PreOperational" state and inform the network-master that it is ready to start normal operation. When changing from "Initialization" state to "PreOperational" state, the node shall send a boot-up message.

When a node is in its "PreOperational" state, the network-master may configure it. After that, the network master shall send a *network management message* changing the node state from PreOperational to Operational.

At any moment while the network is operative, the master may change any node state as described by the node state machine. The network master may choose to change a node state if there is something wrong with the node.

In the OBU system, the OBU (network master) may decide to change a node state if its communication protocol seems to be not working. The process to determine whether a node is working or not is as follows:

- OBU polls a slave to ask for its measured value (sensor) or state (Cabin User Interface).
- If the node does not answer after a suitable amount of time, the OBU asks for its state by means of a Node State Request message.
- If the node does not answer, the OBU changes the node state to "Stop" and sets it as not present. On the other hand, if the node answers with a Heartbeat/Node-guarding message, the OBU changes the node state to "Initialization" and waits for its Boot-up message.

Therefore, four important types of messages that are used to guarantee network communication and management: boot-up, Heartbeat/Node-guarding, Node State Request and Node State Change. Boot-up, Heartbeat and Node State Request are mapped using Heartbeat Messages COB-IDs. Node State Change is mapped using NMT messages COB-IDs.

A-VII-4.2.1 Boot-Up Mapping

Boot-up messages are mapped within a single CAN frame using the COB-ID designated for Heartbeat messages (see section *"Messages Identifiers"*). The following figure shows the format of the Boot-up frame.

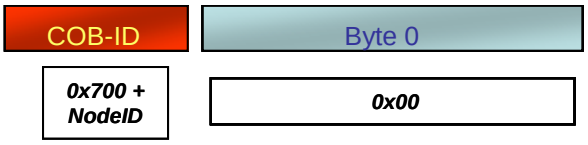


Fig. A-VII-8: Boot-up message

A-VII-4.2.2 Heartbeat/Node-Guarding Mapping

Heartbeat/Node-Guarding messages are used by slave nodes to tell the master in which state they are. This message is sent after an explicit request from the master. The following figure shows the typical Heartbeat/Node-Guarding frame.

Possible node states are described in the following table.

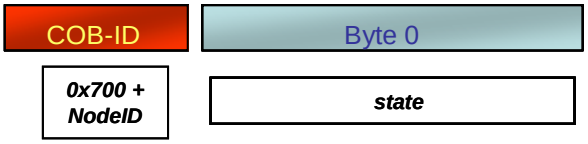


Fig. A-VII-9: Heartbeat message

State Coding	
Code	**State**
0x04	*Stopped*
0x05	*Operational*
0x7F	*PreOperational*

Tab. T-A-VII-3: State Coding

A-VII-4.2.3 Node State Request Mapping

Node State Request messages are simply Heartbeat messages mapped in a RTR, that is, without data bytes. The following figure shows the Node State Request frame.

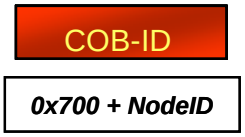


Fig. A-VII-10: Node State Request message

A-VII-4.2.4 Node State Change Mapping

Node State Change messages are mapped within NMT objects. This ensures them to have the highest priority in the CAN network. A NMT uses two data bytes to indicate the node identification and the command specifier. The following figure shows the typical Node State Change frame.

Command Specifiers are described in the following table.

CS	Meaning
0x01	Start Node (go to Operational state)
0x02	Stop node(go to Stop state)
0x80	Enter PreOperational State

0x81	Reset Node(go to Initialization state)

Tab.T-A-VII-4: NMT Command Specifiers

A-VII-4.3 Object Dictionary

Object Dictionaries (ODs) are part of the CAN Application protocol by offering an exhaustive and configurable description of every node in the CAN network. A node OD provides accurate information regarding:

- Data types used in the node.
- Node communication parameters.
- Application data and configuration parameters.

By accessing a slave OD with a SDO, the network master (or any other authorized node) may be able to configure slave communication parameters, configure application parameters or have access to information related to the slave function.

When sending requested PDOs, nodes are sending entries that are found in their own OD. Therefore, PDOs are closely related to ODs. Moreover, nodes ODs are the interface between CAN Application protocol and Node application sub-layer. ODs are organized according to an *index* number. Each index represents an object (or set of objects) in the OD. Each object in the OD has a precise meaning and function; for instance, an object in the OBU temperature sensor's OD shall be the temperature value. An object may have also a *sub-index* (depending on the type of object), a *data type,* an *access attribute* and a *default value*.

Indexes are 16-bit values. Sub-indexes are 8-bit values. It is possible to address more than 16 millions of objects in an OD.

In the OBU system, the following set of data types are used for OD's entries:

- Unsigned 8 bits.
- Unsigned 32 bits.
- Real 32 bits.
- Record : collection of objects (sub-index required).
- VisibleString*N* : *N*-character string.

Access attributes represent the accessibility to objects in the OD from the CAN application protocol, and hence, from external nodes. Possible values are:

- rw : read and write access;
- wo : write only access;
- ro : read only access.

The OD index space is divided as shown in table T-A-VII-5. Nodes in the CAN network does not have to implement all objects in the OD index space, only objects that are applicable to them

Index	Object
0x0001 – 0x001F	Static Data Types
0x0020 – 0x003F	Complex Data Types
0x0040 – 0x005F	Manufacturer Specific Complex Data Types
0x0060 – 0x007F	Device Profile Specific Data Types
0x0080 – 0x009F	Device Profile Specific Complex Data Types
0x1000 – 0x1FFF	Communication Profile Area
0x2000 – 0x5FFF	Manufacturer Specific Profile Area
0x6000 – 0x9FFF	Standardized Device Profile Area

Tab. T-A-VII-5: OD Index Space

Within the OBU system, every node has to have clearly defined ODs in order to allow an efficient communication among them. Probably the most important part of the Object Dictionary implemented in the OBU system has to do with the communication profile area (0x1000 – 0x1FFFF), where all attributes and behaviours of the node in terms of communication through the CAN network are described, i.e. PDO configuration, SDO configuration, etc.

The manufacturer profile area (0x2000 – 0x5FFF) presents useful information about the specific device to be connected in the CAN network. In the OBU system this area is used to describe

internal functions and variables that are related to the node specific operation. For the temperature sensor, for instance, this area shows the actual temperature value measured.

A-VII-4.3.1 Analogue Sensors OD

Analogue sensors in the OBU system are temperature sensors. Object Dictionaries, and hence internal configurations of these devices shall be as presented in table T-A-VII-6.

A-VII-4.3.2 Digital Sensors OD

Digital sensors in the OBU system are used for the door sensors and for the coupling sensor. Digital sensor OD is presented in table T-A-VII-6 and T-A-VII-7. The identification of the sensor is guaranteed by the transmission protocol used to communicate with the master. The transmission type used is asynchronous after a RTR; this kind of transmission means that the sensor responds only after a straight query from the master.

Index	Sub-Index	Name	Data Type	Attr.	Default value	Comments
Communication Profile Area						
0x1800		***PDO1 TX Parameters***	Record			
	0	N. Entries	U8	ro	2	
	1	COB-ID used by PDO	U32	ro	0x200 + NodeID	0x200 is associated with OBU PDO1-RX NodeID = OBU ID (0x01)
	2	Transmission Type	U8	ro	0xFD	0xFD represents an asynchronous transmission after a master RTR.
0x1A00		***PDO1 TX Mapping***	Record			
	0	N. Entries	U8	ro	2	
	1	Temperature	U32	ro	0x200F0120	Default value points to the following object: Index = 0x200F Sub-Index = 0x01 Number of bits = 0x20
Manufacturer Specific Profile Area						
0x2000	***-***	***Sensor Type***	U8	ro	0x00	2-bit sensor type code
0x200F		***Sensor Values***	Record			
	0	N. Entries	U8	ro	2	
	1	Temperature(°C)	R32	ro	0xFFFFFFFF	Measured Temperature. If temperature is not available, the value shall be the default value.

Tab. T-A-VII-6: Analogue Sensor Object Dictionary

Index	Sub-Index	Name	Data Type	Attr.	Default value	Comments
Communication Profile Area						
0x1801		***PDO2 TX Parameters***	Record			
	0	N. Entries	U8	ro	2	
	1	COB-ID used by PDO	U32	ro	0x300 + NodeID	0x300 is associated with OBU PDO2-RX NodeID = OBU ID (0x01)
	2	Transmission Type	U8	ro	0xFD	0xFD represents an asynchronous transmission after a master RTR.
0x1A01		***PDO2 TX Mapping***	Record			
	0	N. Entries	U8	ro	1	
	1	Digital	U32	ro	0x200F0008	Default value points to the following object: Index = 0x200F Sub-Index = 0x00 Number of bits = 0x08
Manufacturer Specific Profile Area						
0x200F		***Digital Value***	U8	ro	0x00	0x00: Low0xFF: High

Tab. T-A-VII-7: Digital Sensor Object Dictionary

A-VII-4.3.3 Cabin User Interface OD

The CUI's Object Dictionary describes communication and application parameters involved in the OBU system Application Protocol profile. In the same way as analogue and digital sensors described previously, CUI communication capabilities are closely related to the OBU. Communication procedures take place only towards and from the OBU. Table T-A-VII-8, T-A-VII-9 and T-A-VII-10 show the CUI's OD.

Index	Sub-Index	Name	Data Type	Attr.	Default value	Comments
Communication Profile Area						
0x1200		***SDO Server Parameters***	Record			
	0	N. Entries	U8	ro	2	
	1	COB-IDClient->Server(rx)	U32	ro	0x600 + NodeID	NodeID = Cabin user Interface
	2	COB-IDServer->Client (tx)	U32	ro	0x580 + NodeID	NodeID = Cabin user Interface
0x1400		***PDO1 RX Parameters***	Record			PDO sent by OBU.
	1	COB-ID used by PDO	U32	ro	0x180 + NodeID	0x180 is associated with OBU PDO1-TX NodeID = Cabin User Interface
	2	Transmission Type	U8	ro	0xFE	0xFE represents an asynchronous transmission.
0x1401		***PDO2 RX Parameters***	Record			PDO sent by OBU..
	0	N. Entries	U8	ro	2	
	1	COB-ID used by DO	U32	ro	0x280 + NodeID	0x280 is associated with OBU PDO2-TX NodeID = Cabin User Interface
	2	Transmission Type	U8	ro	0xFE	0xFE represents an asynchronous transmission.
0x1402		***PDO3 RX Parameters***	Record			PDO sent by OBU.
	0	N. Entries	U8	ro	2	
	1	COB-ID used by PDO	U32	ro	0x380 + NodeID	0x380 is associated with OBU PDO3-TX NodeID = Cabin User Interface
	2	Transmission Type	U8	ro	0xFE	0xFE represents an asynchronous transmission.

Tab. T-A-VII-8: Cabin User Interface Object Dictionary

Index	Sub-Index	Name	Data Type	Attr.	Default value	Comments
Communication Profile Area						
0x1600		***PDO1 RX Mapping***	Record			
	0	N. Entries	U8	ro	2	
	1	Temperature	U32	ro	0x20100120	Default value points to the following object:Index = 0x2010Sub-Index = 0x01Number of bits = 0x20
0x1601		***PDO2 RX Mapping***	Record			
	0	N. Entries	U8	ro	2	
	1	Temperature	U32	ro	0x20100220	Default value points to the following object:Index = 0x2010 Sub-Index = 0x02 Number of bits = 0x20
0x1602		***PDO3 RX Mapping***	Record			
	0	N. Entries	U8	ro	2	
	1	CurrState	U32	ro	0x20010108	Default value points to the following object:Index = 0x2001Sub-Index = 0x01Number of bits = 0x08
	2	Time	U32	ro	0x20020020	Default value points to the following object:Index = 0x2002Sub-Index = 0x00Number of bits = 0x20
0x1802		***PDO3 TX Parameters***	Record			
	0	N. Entries	U8	ro	2	
	1	COB-ID used by PDO	U32	ro	0x400 + NodeID	0x400 is associated with OBU PDO3-RX NodeID = OBU ID (0x01)
	2	Transmission Type	U8	ro	0xFD	0xFD represents an asynchronous transmission after a master RTR.
0x1A02		***PDO3 TX Mapping***	Record			
	0	N. Entries	U8	ro	1	
	1	NextState	U32	ro	0x20010208	Default value points to the following object:Index = 0x2001Sub-Index = 0x02 Number of bits = 0x08
Manufacturer Specific Profile Area						
0x2001		***State***	Record			
	0	N. Entries	U8	ro	2	
	1	Current State	U8	wo	0x00	Cabin User Interface current state. Information received from the OBU.
	2	Next State	U8	ro	0x00	Cabin User Interface next state. This field is updated when there is a precise request from the user. Otherwise it should have the same value shown by the Current State field.

Tab. T A-VII-9: Cabin User Interface Object Dictionary

Index	Sub-Index	Name	Data Type	Attr.	Default value	Comments
0x2002		***Time***	VisibleString4	wo	"HHMM"	Time received from the OBU. Format: 4-byte "H1 H2 : M1 M2"
0x2010		***Temperature***	Record			
	0	N. Entries	U8	ro	2	
	1	Temperature1	R32	wo	0xFFFFFFFF	Temperature 1 to be displayed.
	2	Temperature2	R32	wo	0xFFFFFFFF	Temperature 2 to be displayed.
0x2040		***User Message***	Record			
	0	N. Entries	U8	ro	100	Length of the user message string
	1	Mess char 1	U8	ro	0	Message string 1st character
	…	Mess char n	U8	ro	0	Message string nth character
	100	Mess char 100	U8	ro	0	Message string 100th character

Tab. T-A-VII-10: Cabin User Interface Object Dictionary (4/4)

A-VII-4.3.4 D OBU OD

OBU's Object Dictionary presents a description of the OBU communication and configuration parameters. OBU OD is not exhaustive in terms of its internal configuration parameters because, in the OBU system, other nodes are not supposed to change (not even read) its inner variables. OBU OD fixes its communication capabilities according to the system description in order to allow perfect interaction among it and its network-slaves. OBU OD is shown in table T-A-VII-11, T-A-VII-12 and T-A-VII-13.

Configurable Indexes and sub-indexes are changed according to the RTR receiver node. RTRs are used by the OBU to poll network slaves.

Index	Sub-Index	Name	Data Type	Attr.	Default value	Comments
Communication Profile Area						
0x1280		***SDO ClientParameters***	Record			
	0	N. Entries	U8	ro	3	
	1	COB-ID Client->Server(tx)	U32	ro	0x600 + NodeID	NodeID = Cabin user Interface
	2	COB-IDServer->Client(rx)	U32	ro	0x580 + NodeID	NodeID = Cabin user Interface
	3	SDO serverNodeID	U8	ro	NodeID	NodeID = Cabin User Interface
0x1400		***PDO1 RX Parameters***	Record			PDO sent by analogue sensors.
	0	N. Entries	U8	ro	2	
	1	COB-ID used by PDO	U32	ro	0x200 + NodeID	NodeID = OBU ID (0x01)
	2	Transmission Type	U8	ro	0xFD	0xFD represents an asynchronous transmission after a master RTR.
0x1401		***PDO2 RX Parameters***	Record			PDO sent by digital sensors.
	0	N. Entries	U8	ro	2	
	1	COB-ID used by PDO	U32	ro	0x300 + NodeID	NodeID = OBU ID (0x01)
	2	Transmission Type	U8	ro	0xFD	0xFD represents an asynchronous transmission after a master RTR.
0x1402		***PDO3 RX Parameters***	Record			PDO sent by Cabin User Interface.
	0	N. Entries	U8	ro	2	
	1	COB-ID used by PDO	U32	ro	0x400 + NodeID	NodeID = OBU ID (0x01)
	2	Transmission Type	U8	ro	0xFD	0xFD represents an asynchronous transmission after a master RTR.
0x1800		***PDO1 TX Parameters***	Record			PDO sent to Cabin User Interface.
	0	N. Entries	U8	ro	2	
	1	COB-ID used by PDO	U32	ro	0x180 + NodeID	NodeID = Cabin User Interface
	2	Transmission Type	U8	ro	0xFE	0xFE represents an asynchronous transmission. In this case it is triggered by an internal counter.
0x1801		***PDO2 TX Parameters***	Record			PDO sent to Cabin User Interface.
	0	N. Entries	U8	ro	2	
	1	COB-ID used by PDO	U32	ro	0x280 + NodeID	NodeID = Cabin User Interface
	2	Transmission Type	U8	ro	0xFE	0xFE represents an asynchronous transmission. In this case it is triggered by an internal counter.
0x1802		***PDO3 TX Parameters***	Record			PDO sent to Cabin User Interface.
	0	N. Entries	U8	ro	2	
	1	COB-ID used by PDO	U32	ro	0x380 + NodeID	NodeID = Cabin User Interface
	2	Transmission Type	U8	ro	0xFE	0xFE represents an asynchronous transmission. In this case it is triggered by an internal counter.

Tab. T- A-VII-11: OBU Object Dictionary – A

Index	Sub-Index	Name	Data Type	Attr.	Default value	Comments
Communication Profile Area						
0x1600		***PDO1 RX Mapping***	Record			
	0	N. Entries	U8	ro	2	
	1	Temperature	U32	ro	0x20A0- -20	Default value points to the following object:Index = 0x20A0Sub-Index = - - *(configurable*)*Number of bits = 0x20
0x1601		***PDO2 RX Mapping***	Record			
	0	N. Entries	U8	ro	1	
	1	Digital	U32	ro	0x20D- - -08	Default value points to the following object:Index = 0x20D - *(configurable*)*Sub-Index = - - *(configurable*)*Number of bits = 0x08
0x1602		***PDO3 RX Mapping***	Record			
	0	N. Entries	U8	ro	1	
	1	CabinNextState	U32	ro	0x20E00208	Default value points to the following object:Index = 0x20E0 Sub-Index = 0x02 Number of bits = 0x08
0x1A00		***PDO1 TX Mapping***	Record			
	0	N. Entries	U8	ro	2	
	1	Temperature	U32	ro	0x20A00120	Default value points to the following object:Index = 0x20A0Sub-Index = 0x01 *(1st temp. sensor)*Number of bits = 0x20
0x1A01		***PDO2 TX Mapping***	Record			
	0	N. Entries	U8	ro	2	
	1	Temperature	U32	ro	0x20A00220	Default value points to the following object:Index = 0x20A0 Sub-Index = 0x02 *(2nd temp. sensor)*Number of bits = 0x20
0x1A02		***PDO3 TX Mapping***	Record			
	0	N. Entries	U8	ro	2	
	1	CabinCurrState	U32	ro	0x20E00108	Default value points to the following object:Index = 0x20E0Sub-Index = 0x01 Number of bits = 0x08
	2	RelatedTime	U32	ro	0x20E00320	Default value points to the following object:Index = 0x20E0 Sub-Index = 0x03Number of bits = 0x20

Tab. T-A-VII-12: OBU Object Dictionary - B

Index	Sub-Index	Name	Data Type	Attr.	Default value	Comments
Manufacturer Specific Profile Area						
0x20A0		***Temperature Sensors***	Record			
	0	N. Entries	U8	ro	8	
	1	Temp1	R32	rw	0xFFFFFFFF	Temperature from Temperature sensor 1. (NodeID = 0x20)
	2	Temp2	R32	rw	0xFFFFFFFF	Temperature from Temperature sensor 2. (NodeID = 0x21)
	8	Temp8	R32	rw	0xFFFFFFFF	Temperature from Temperature sensor 8. (NodeID = 0x27)
0x20D0		***Door/Hatch Sensors***	Record			
	0	N. Entries	U8	ro	8	
	1	Hatch1	U8	rw	0xFF	Hatch sensor 1 state.(NodeID = 0x08)
	2	Hatch2	U8	rw	0xFF	Hatch sensor 2 state.(NodeID = 0x09)
	8	Hatch8	U8	rw	0xFF	Hatch sensor 8 state.(NodeID = 0x0F)
0x20D1		***Coupling sensors***	Record			
	0	N. Entries	U8	ro	8	
	1	Trail1	U8	rw	0x00	Coupling sensor 1 state.(NodeID = 0x10)
	2	Trail2	U8	rw	0x00	Trailer sensor 2 state.(NodeID = 0x11)
	8	Trail8	U8	rw	0x00	Coupling sensor 8 state.(NodeID = 0x17)
0x20E0		***Cabin Interface***	Record			
	0	N. Entries	U8	ro	3	
	1	Cabin Interface State	U8	ro	0x00	Cabin user Interface State.
	2	Cabin Interface Requested ext State	U8	rw	0x00	Cabin User Interface State requested by Cabin
	3	Current State Cabin Time	VisibleString4	ro	"HHMM"	Time related to the current cabin state. Format: 4-byte "H1 H2 : M1 M2"

Tab. T-A-VII-13: OBU Object Dictionary – C

A-VII-4.4 Protocol Message Mapping

Protocol messages, such as PDOs and SDOs, are mapped in one or more CAN frame(s). The method used to map protocol messages in CAN frames may be found in the Object Dictionaries related to each node. Nevertheless, for clearness and completeness, this section illustrates in detail how this mapping procedure is carried out.

A-VII-4.4.1 Analogue sensors Protocol Message Mapping

As described by its Object Dictionary, analogue sensor most used protocol messages are PDOs. Analogue sensors support one type of transmit PDO (TPDO). In the object addressed by index 0x1800 there may be found the transmission parameters of the Analogue Sensor TPDO. Parameters include the COB-ID used by the CAN frame and the transmission type.

The object indexed by 0x1A00 shows the objects that are actually mapped inside the TPDO. These objects are the measured temperatures. They are described by means of a pointer to the mapped objects. The pointer shows the index, sub-index and length of the pointed object.

Figure A-VII-11 shows a typical CAN frame used by an analogue sensor when sending a PDO. The figure shows the CAN frame in terms of its ID and data bytes. ID is always 0x201 due to the fact that analogue sensors PDO are always sent to the OBU (0x200 + OBU ID (0x01)). Multi byte objects are always sent LSB first (little Indian).

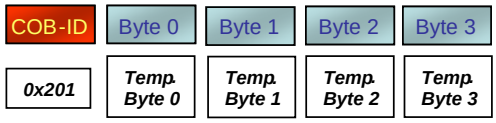


Fig. A-VII-11: Analogue Sensor TPDO

A-VII-4.4.2 Digital sensors Protocol Message Mapping

Digital sensor's PDOs are used to notify to the OBU the status of the sensor. Digital sensors Object Dictionary describes PDO parameters and mapping. PDO transmitted by digital sensors is related to the second PDO received by the OBU. As described by the object indexed by 0x1801, the COB-ID is always 0x301 and the transmission type is always asynchronous after a RTR. Object indexed by 0x1A01 defines that this PDO has one mapped object, which is related to the state of the sensor. The mapped object corresponds to the object indexed by 0x200F and has a length of 8 bits.

Figure A-VII-12 shows a typical PDO transmitted by the digital sensor.

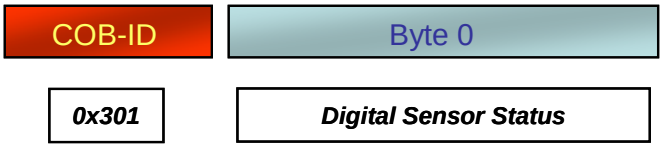


Fig. A-VII-12: Digital Sensor TPDO

A-VII-4.4.3 Cabin User Interface Protocol Message Mapping

The CUI's Object Dictionary provides information about the transmission of the PDO (TPDO) and receives PDO (RPDO) that are supported by the CUI. The CUI's RPDOs are correlated with PDOs transmitted by the OBU. The CUI's TPDO is related to one of the PDOs received by the OBU. The information sent by the CUI to the OBU using its PDO regards a request for changing its state. The CAN frame used to map CUI's TPDO presents a similar format as the CAN frame used in Digital Sensors. Figure A-VII-13 shows the PDO transmitted by the CUI.

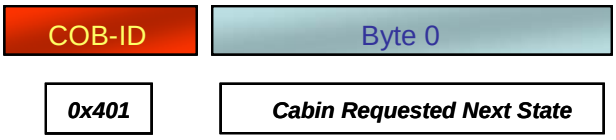


Fig. A-VII-13: Cabin User Interface TPDO

A-VII-4.4.4 OBU Protocol Message Mapping

PDOs transmitted (TPDO) by the OBU, as described by its Object Dictionary, are three. All of them are received by the CUI. OBU's OD clearly describe variables and data sent using its TPDOs: temperature values, journey-related time and Cabin Status. The first two TPDOs used to send information to the CUI from the OBU, contain the measured temperature values from analogue sensors. In the case that two temperature sensors are installed, the OBU shall send information regarding two temperature sensors; if more sensors are installed, the OBU shall sent by default the first two. In figure A-VII-14 is presented the layout of the CAN frame containing any of these TPDOs. The third TPDO is used to send the CUI state and the time related to the state. This PDO is useful to show frequently to the cabin its own state and the time to be displayed to the user. The time is sent as a 4-character string (HH:MM). Figure A-VII-15 shows the format of the CAN frame used to map this TPDO; since the less significant byte shall be sent first, the character indicating the last *minute* digit is sent firstly and the character indicating the first *hour* digit is sent lastly.

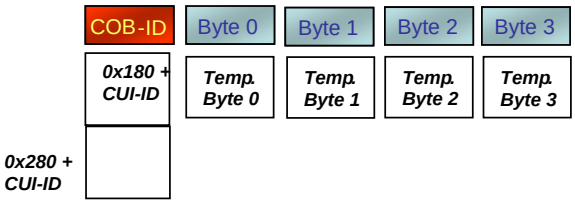


Fig. A-VII-14: OBU TPDOs (Sensors)

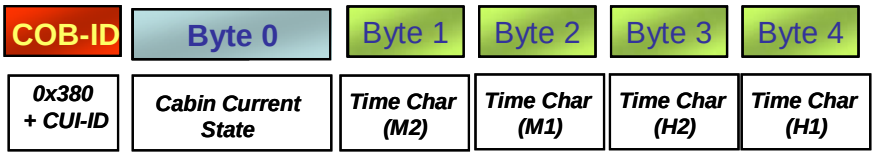


Fig. A-VII-15: OBU TPDO (Cabin State - Time)

A-VII-4.4.5 The Cabin Unit Interface finite state machine and its SDO Message Mapping

The OBU periodically sends a remote request to the CUI to check whether the user has changed the CUI state putting in a command. The CUI sends back a TPDO telling the OBU its next state or simply telling its current state. This is represented in the TPDO as enumerate fitted in a single byte, having the following values:

- **0x00**: No trip
- **0x01**: Trip Start
- **0x02**: Trip Stop
- **0x03**: Message
- **0x04**: Trip Rest
- **0x05**: Trip Continue.

The state of the CUI is kept at any single step by the OBU, and hence this last one behaves as finite state machine, in which the input variables are collected by the CUI. The state switching is fired when the OBU receives from the CUI a state different from the one current lying in the OBU. In this case this last one updates the current state and eventually notifies it to the CUI by means of a PDO, as explained below. If the track is at rest and the user notifies a journey start to the OBU, a FSM context switching is done.

The OBU hence waits for the journey start message coming from the CUI and after having got it, it starts to send to the CUI three TPDO, as explained in the OBU message protocol. These contain the temperatures collected values, the journey elapsed time and the CUI state flag, kept by the OBU. These three TPDOs are sent from the OBU to the CUI during each journey, i.e. after a journey start or resume.

Usually, a SDO can bring up several bytes that shall be sent using several CAN frames, while the PDO can transport at most eight bytes and can be mapped within a single CAN frame. The SDO are used to read or write directly some values from and to the object dictionary. The SDO are exchanged between a server node and a client node, in a sort of peer to peer fashion. The client can download the data to the server at the OD index and sub index sent together with each SDO.

For doing this the SDO protocol is called **Initiate domain download** consisting of a single SDO frame by which the client tells the server it's going to download data to it. After this start handshaking message, the client starts to pump several SDO frames called **segments** to the server. In this case, the CUI stores in its dictionary the user message and shall send it to the OBU after the CUI has notified to the OBU the incoming changing state with the above explained TPDO in response to the remote request sent by the OBU. The OBU notes that the user has input a command and eventually asks to the CUI to **upload** the read user message if any, i.e. from this point of view the CUI becomes the server while the OBU becomes the client. This is called **upload domain SDO protocol**, happening every time the client asks the server some data stored

somewhere in the server object dictionary. The SDO could be divided in two main types: the ones sent with **expedition** and the ones sent with **segments**. The first case happens when the data exchanged in the SDO are at most 4 bytes wide, and could be mapped within a single SDO CAN frame. In this case the SDO behaves more or less like a SDO message, and the only difference is that it needs to be confirmed by the server in case of upload, or by the client in case of download. The user message is a 100 characters null terminating string and could not be mapped within a single SDO frame, resulting in **segmented** SDO communication.
The messages involved in the upload domain protocol between the client and the server are mainly split in two types: the **Initiate domain upload**, by which the client asks the server to start the communication and the **Segment domain upload**, by which the server sends each segment, i.e. up to 7 bytes of information keeping the first one as **SDO command specifier**.
The **Initiate domain upload message** sent by the client can be represented as follow:

- Byte 0: O command specifier
- Bytes 1, 2: Server Object Dictionary index, 2 bytes wide.
- Byte 3: Server Object Dictionary sub index. In this entry usually it's stored the number of entries starting from sub index + 1 that shall be uploaded to the client.
- Bytes 4-7: In case of expedited transfer in these four bytes is stored the data. In case of segmented transfer, they shall be kept empty.

The **Command Specifier** byte bits have the following meaning:

1. Bit 7-5 : kept to 010b, i.e. 2h, representing the Initiate domain upload command ID.
2. Bit 4-0, n.c. The server answers to this message with a SDO having the following command specifier byte:
3. Bit 7-5 : kept to 010b, i.e. 2h, representing the Initiate domain upload command ID.
4. Bit 4: n.c.
5. Bit 3-2: if bits 1 and 0 are both 1, standing for expedited transfer, they represent n, i.e. the number of bytes in the last 4 SDO bytes not carrying data, i.e. 8-n up to 7th byte.
6. Bit 1 called **e** bit, 1 if the transfer is expedited or 0 if segmented.
7. Bit 0 called **s** bit, 1 if the data size is indicated or zero if not.

If, for example, the client asks to the server to upload the data stored in the server OD at 0x1801 sub index 3 with expedited transfer. The resulting SDO protocol shall be (each row represents the eight data bytes lying in a SDO frame):

- Client: 40 01 18 03 00 00 00 00 (little indian convention, i.e. LSB sent first)
- Server: 40 01 18 03 FE 03 00 00, where the upload word is 0x03FE.

In this case, the data were expedited, the server noticed it was just one word wide and could be fitted in a single SDO CAN frame. If these data are more than 4 bytes as in the OBU vs. CUI case, a set of **Segment domain segment** shall be send from the CUI to the main client OBU in order to encompass the whole at most 100 characters wide user message string.

The **Segment domain upload** SDO message is represented as follow:

1. Byte 0: SDO Command specifier
2. Byte 1-7: transmit data.

The SDO command specifier byte bits take this meaning:

- Bit 7-5: 011b, i.e. 3h, standing for the segment domain upload command ID.
- Bit 4: toggle bit. This bit is reset to 0 on the very first segment, and shall be toggled on each subsequent segment.
- Bit 3-1: number n of bytes that do not contain data in the following seven ones (8-n up to 7th one).
- Bit 0: **c** bit, 0 if the segment is not the last one and there comes some other segments to be uploaded, 1 for the end closing segment.

Using the **c** bit, the server does not need to send a closing SDO upload domain segment, and has not been designed in the CANopen specification. Taking into account this upload protocol, the SDO mapping resulting in the OBU vs. CUI communication consists of:

1. The client OBU sends a **Initiate domain upload** to the CUI server. The transfer is not expedited but segmented, because it has to exchange up to 100 bytes representing the user written message. This data is mapped in the CUI to the address: 0x2040 sub index 0, in which there will be the written message visible string. The resulting SDO is 40 40 20 00 00 00 00.

2. The server CUI recognises the request from the OBU and sends up to **15 Segment domain download** SDO messages, from which the client takes the written message data. The string could also be shorter than 100 characters, so 15 is the worst case in which the whole characters are sent. In this case the first 14 SDOs messages contain 14x7= 98 bytes while the remaining SDO contain the last 2 characters. In all other cases few than 15 SDOs are sent.

ANNEX VIII - INFORMATION SECURITY REQUIREMENTS

A-VIII-1 Data integrity, authenticity and confidentiality during transmission

The integrity, authenticity and confidentiality of messages exchanged between the OBU and the remote receiver are all assured on establishment of an HTTPS connection. An entity has to be appointed to manage the cryptographic keys and their related certificates. All of these functions are implemented in a variety of commercial-off-the-shelf software packages and hardware devices which are available with various degrees of security certification.

A-VIII-2 Data integrity, authenticity and confidentiality at the remote receiver

The operator of the remote receiver could be the transport company for their own fleet, a private service provider or a centralised service. The operator of the remote receiver has to implement an appropriate information security management system – ISMS to assure integrity, authenticity and availability of the data. Best practices in ISMS are laid down in the ISO 17799 standard.

European Commission

EUR 23238 EN – Joint Research Centre – Institute for Protection and Safety of the Citizen
Title: Technical Specifications for Navigation System in Long Journeys Animal Transport
Author(s): J Hofherr, G Fiore, F Natale, J Bishop, S Mainetti, E Ruotolo
Luxembourg: Office for Official Publications of the European Communities
2008 – 99 pp.
EUR – Scientific and Technical Research series – ISSN 1018-5593
ISBN 978-92-79-08310-5
DOI 10.2788/679

How to obtain EU publications

Our priced publications are available from EU Bookshop (http://bookshop.europa.eu), where you can place an order with the sales agent of your choice.

The Publications Office has a worldwide network of sales agents. You can obtain their contact details by sending a fax to (352) 29 29-42758.

The mission of the JRC is to provide customer-driven scientific and technical support for the conception, development, implementation and monitoring of EU policies. As a service of the European Commission, the JRC functions as a reference centre of science and technology for the Union. Close to the policy-making process, it serves the common interest of the Member States, while being independent of special interests, whether private or national.

JRC
EUROPEAN COMMISSION

LB-NA-23238-EN-C

ISBN 978-92-79-08310-5