IEC 61375-2-3

Edition 1.0 2015-07

INTERNATIONAL STANDARD

colour inside

Electronic railway equipment – Train communication network (TCN) – Part 2-3: TCN communication profile

IEC 61375-2-3:2015-07(en)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
info@iec.ch
www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

IEC Catalogue - webstore.iec.ch/catalogue

The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - www.iec.ch/searchpub

The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee,…). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and also once a month by email.

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 15 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary

More than 60 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IEC Customer Service Centre - webstore.iec.ch/csc

If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: csc@iec.ch.

IEC 61375-2-3

Edition 1.0 2015-07

INTERNATIONAL STANDARD

colour inside

Electronic railway equipment – Train communication network (TCN) – Part 2-3: TCN communication profile

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 45.060

ISBN 978-2-8322-2775-6

Warning! Make sure that you obtained this publication from an authorized distributor.

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

ELECTRONIC RAILWAY EQUIPMENT – TRAIN COMMUNICATION NETWORK (TCN) –

Part 2-3: TCN communication profile

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 61375-2-3 has been prepared by IEC technical committee 9: Electrical equipment and systems for railways.

The text of this standard is based on the following documents:

FDIS	Report on voting
9/2029/FDIS	9/2048/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all parts in the IEC 61375 series, published under the general title *Electronic railway equipment – Train communication network (TCN)*, can be found on the IEC website.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific publication. At this date, the publication will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

A bilingual version of this publication may be issued at a later date.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

The IEC 61375 standard series specifies a Train Communication Network for usage in railway vehicles (trains) mainly intended for the exchange of TCMS related information, but not restricted to it. The specification starts from the physical layer up to the application layer and it involves different communication technologies.

This part of IEC 61375 (IEC 61375-2-3) defines the communication profile of the Train Communication Network so as to achieve interoperability between Consists connected by Ethernet Train Backbones as defined in IEC 61375-2-5.

The reasons for prompting the preparation of this part of IEC 61375 are:

- definition of the requirements necessary for communication interoperability on Ethernet Train Backbone level
- full documentation of the requirements of all users, aligning them and setting them out in standard form
- providing guidelines for the technical solution adopted for the train backbone interoperable communication
- defining a conformance test guideline (Annex F) which gives guidance for checking the conformity of consists to the communication profile

Concrete train applications for certain functionalities are not dealt with in this part of IEC 61375. They are contained in IEC 61375-2-4.

ELECTRONIC RAILWAY EQUIPMENT – TRAIN COMMUNICATION NETWORK (TCN) –

Part 2-3: TCN communication profile

1 Scope

This part of IEC 61375 specifies rules for the data exchange between consists in trains. The aggregation of these rules defines the TCN communication profile.

The objective of the communication profile is to ensure interoperability between consists of the said trains with respect to the exchange of information. For this it defines all those items which are necessary for communication interoperability:

- an architecture with defined train directions related to different train views
- a common functional addressing concept
- common communication protocol for data exchange between functions
- a set of services for train communication control.

As a restriction, this communication profile is adhered to the Ethernet Train Backbone (ETB) technology as defined in IEC 61375-2-5. Towards the consist networks, a more abstract interface is defined which does not restrict the appliance of any consist network technology as for instance MVB (IEC 61375-3-1), CANOpen (IEC 61375-3-3) or ECN (IEC 61375-3-4).

It is not within the scope of the communication profile to define application data content and its meaning (e.g. syntax and semantics). This is within the responsibility of the application profiles. Namely two application profiles are explicitly supported as shown in Figure 1: the TCMS application profile as defined in IEC 61375-2-4, and the onboard multimedia and telematic services (OMTS) related profiles as defined in the IEC 62580 series.

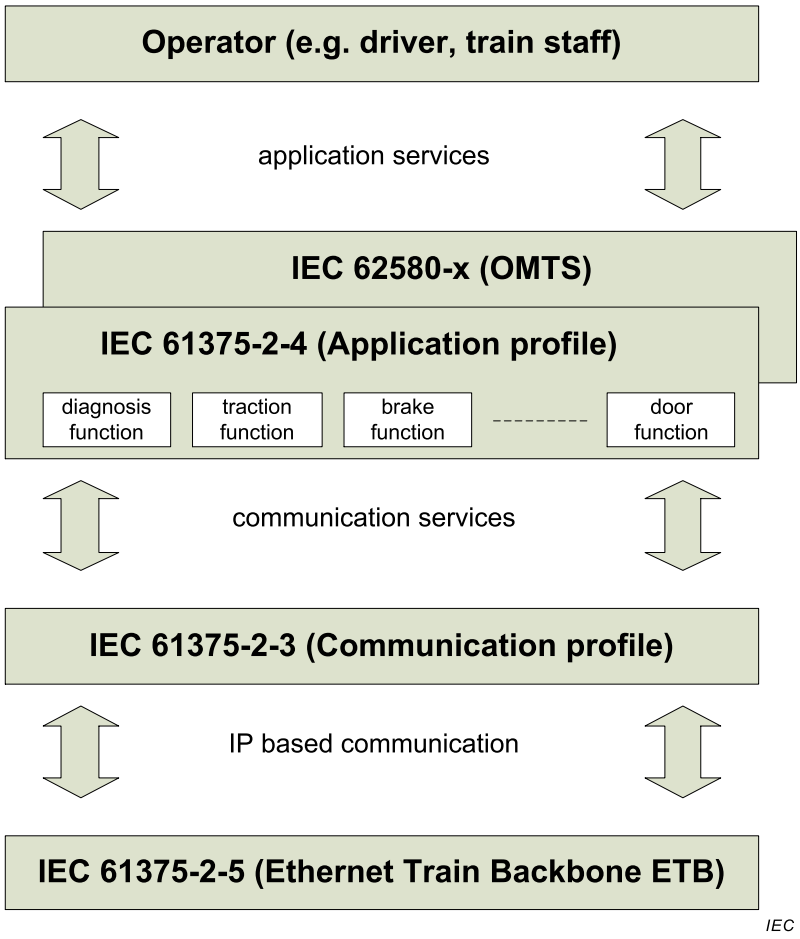


Figure 1 – IEC 61375-2-3 as connecting element between train backbone and application

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 61375-1, *Electronic railway equipment – Train communication network (TCN) – Part 1: General architecture*

IEC 61375-2-1, *Electronic railway equipment – Train communication network (TCN) – Part 2-1: Wire Train Bus (WTB)*

IEC 61375-2-4, *Electronic railway equipment – Train communication network (TCN) – Part 2-4: Application Profile* (to be published)

IEC 61375-2-5, *Electronic railway equipment – Train communication network (TCN) – Part 2-5: Ethernet train backbone*

IEC 62280*, Railway applications – Communication, signalling and processing systems – Safety related communication in transmission systems*

ISO/IEC 9646-6:1994, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 6: Protocol profile test specification*

ISO/IEC 9646-7:1995, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework – Part 7: Implementation Conformance Statements*

ISO/IEC 17011:2004, *Conformity assessment – General requirements for accreditation bodies accrediting conformity assessment bodies*

ISO/IEC 17025:2005, *General requirements for the competence of testing and calibration laboratories*

3 Terms, definitions, abbreviations, acronyms, and conventions

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1.1
application layer
upper layer in the OSI model, interfacing directly to the application

3.1.2
application layer interface
definition of the services offered by the application layer

3.1.3
application process
element within a real open system which performs the information processing for a particular application

3.1.4
big endian
definition where the most significant byte of multi byte data is stored in the lowest address memory location and sent first in the sequence of serialized transmissions

3.1.5
bridge
device which stores and forwards frames from one bus to another on the base of their Link Layer addresses

3.1.6
broadcast
nearly simultaneous transmission of the same information to several destinations

Note 1 to entry: Broadcast in the TCN is not considered reliable, i.e. some destinations may receive the information and others not.

3.1.7
closed train
train composed of one or a set of Consists, where the composition does not change during normal operation

EXAMPLE metro, sub-urban train, high speed train units.

3.1.8
closed train consist
consist within a closed train

3.1.9
communication devices
devices connected to Consist Network or Train Backbone with the ability to transport, source or sink data

3.1.10
composition
number, sequence, orientation and characteristics of the consists forming a train

3.1.11
configuration
definition of the topology of a network, the devices connected to it, their capabilities and the traffic they produce

3.1.12
consist
single vehicle or a group of vehicles which are not separated during normal operation

Note 1 to entry: A consist contains no, one or several consist networks.

3.1.13
consist network
communication network interconnecting communication devices in one Consist

3.1.14
consist network address
network address, which does not change after inauguration and which is used to address communication device in the own consist network

3.1.15
consist sequence number
sequence number of the Consist in the train as obtained during TCN inauguration

3.1.16
consumer
receiver of a message at the Transport Layer

3.1.17
destination device
receiver of a data packet

3.1.18
discovered vehicle/consist
vehicle/consist discovered during train inauguration

3.1.19
end device
unit connected to one Consist Network or to one set of Consist Networks prepared for redundancy reasons

3.1.20
Ethernet Train Backbone
train backbone based on switched Ethernet technology

Note 1 to entry: Specified in IEC 61375-2-5.

3.1.21
Ethernet Train Backbone Node
node connected to the Ethernet train backbone receiving a sequence number during inauguration and capable of forwarding user data packets between Consist Network and Train Backbone

3.1.22
function
application process which exchanges messages with another application process

3.1.23
gateway
connection between different communication technologies

3.1.24
group address
address of a multicast group to which a device belongs

3.1.25
integrity
property of a system to recognize and to reject wrong data in case of malfunction of its parts

3.1.26
leading vehicle
vehicle which controls the movements of the train

3.1.27
linear topology
topology where the nodes are connected in series, with two nodes each connected to only one other node and all others each connected to two other nodes (that is, connected in the shape of a line)

3.1.28
Local Area Network
LAN
part of a network characterized by a common medium access and address space

3.1.29
medium access control
MAC
sublayer of the Link Layer, which controls the access to the medium (arbitration, mastership transfer, polling)

3.1.30
medium
physical carrier of the signal: electrical wires, optical fibers, etc.

3.1.31
message
data item transmitted in one or several packets

3.1.32
multicast
transmission of the same message to a group of receivers, identified by their Group Address

Note 1 to entry: The word "multicast" is used even if the group includes all receivers.

3.1.33
multimedia network
network in a train which is used for data transmission of multimedia and telematic data

3.1.34
multimedia services
services which are used for multimedia

Note 1 to entry: Onboard multimedia and telematic services are defined in IEC 62580-1.

3.1.35
multiple unit
number of vehicles put together to form a unit for the conveyance of passengers and/or freight

Note 1 to entry: The driving cabs of the multiple unit can be located in special power vehicles (e.g. locomotives with only one cab) or in end vehicles fitted with driving cabs. Groups of vehicles of the multiple unit can be assembled into consists.

3.1.36
multiple unit train
train consisting of a set of closed trains, where the composition of the set may change during normal operation

3.1.37
network
set of possibly different communication systems which interchange information in a commonly agreed way

3.1.38
network address
address which identifies a communication device on network layer

3.1.39
network device
components used to set up Consist Networks and Train Networks

Note 1 to entry: These may be passive components like cables or connectors, active unmanaged components like repeaters, media converters or (unmanaged) switches, or managed active components like gateways, routers and (managed) switches.

3.1.40
network layer
layer in the OSI model responsible for routing between different busses

3.1.41
network management
operations necessary to remotely configure, monitor, diagnose and maintain the network

3.1.42
node
dvice on the Train Backbone, which may act as a gateway between Train Backbone and Consist Network

3.1.43
operational network
network in a train which is used for the transmission of train operation and safety critical data

3.1.44
operational train view
driver or train attendance view of a train as a sequence of vehicles

3.1.45
octet
byte
8-bit word stored in memory or transmitted as a unit

3.1.46
onboard multimedia and telematics subsystem
collection of train on-board services and functions for multimedia and telematics

Note 1 to entry: A definition of those services is given in IEC 62580.

3.1.47
operational services
all services which are needed to run a train safely

3.1.48
operational train direction
direction of train as seen by driver or train attendance

3.1.49
operational train directory
data structure describing the train composition as seen by driver or train attendance, including information about the leading vehicle and vehicle gaps

Note 1 to entry: The operational train directory is a part of the train topology database.

3.1.50
operator
enterprise or organization which is operating trains

3.1.51
packet
unit of a message (information, acknowledgement or control) transmitted by protocols on network or transport layer

3.1.52
PD-PDU
data unit for the transmission of process data

Note 1 to entry: There are two types of PD-PDUs. All types of PD-PDUs have the same PDU format. PD-PDU(data) is a PD-PDU containing user data and sent from a publisher to subscriber(s). This is used in push pattern and in pull pattern. PD-PDU(data) in pull pattern is also referred as PD-PDU(reply). PD-PDU(request) is a PD-PDU sent from a requester to publisher(s). This may or may not contain user data.

3.1.53
PD publisher
device which cyclically sends PD telegrams with a defined ComId to a defined address or address range

3.1.54
PD requester
device which requests a single PD telegram of a specified ComId to be sent to a specified address or address range

3.1.55
PD subscriber
device which receives PD telegrams with a defined ComId and a possible source address filter

3.1.56
producer
sender of a message at the Transport Layer

3.1.57
publisher
source of a dataset for broadcasting (see subscriber)

3.1.58
receiver
electronic device which may receive signals from the physical medium

3.1.59
redundancy group
group of devices, where only one device is active and the others are in standby

Note 1 to entry: If the active device fails, one of the standby devices takes over. Typically, a redundancy group comprises two devices.

3.1.60
regular transmission
data transmission of non safety related data

3.1.61
repeater
connection at the Physical Layer between bus segments, providing an extension of the bus beyond the limits permitted by passive means

Note 1 to entry: The connected segments operate at the same speed and with the same protocol. The delay introduced by a repeater is in the order of one bit duration.

3.1.62
residual error rate
probability of integrity breach (unrecognized wrong bit) per transmitted bit

3.1.63
ring topology
active network where each node is connected in series to two other nodes

Note 1 to entry: Ring may also be referred to as loop.

[SOURCE: IEC 61918:2013,3.1.63]

3.1.64
router
connection between two buses at the Network Layer, which forwards telegrams from one bus to another on the base of their network address

3.1.65
safe data transmission
data transmission of safety related data

3.1.66
service
capabilities and features of a sub-system (provided to a user)

3.1.67
sporadic data
transmission of data on a demand basis

3.1.68
subscriber
one of the sinks of a broadcast dataset (see publisher)

3.1.69
switch
MAC bridge

Note 1 to entry: As defined in IEEE 802.1D.

3.1.70
topology
possible cable interconnection and number of devices in a given network

3.1.71
topography counter
unique identifier of a train composition

3.1.72
train
composition of one or a set of Consists, where the configuration can change during normal operation

3.1.73
train communication network
data communication network for connecting programmable electronic equipment on-board rail vehicles

3.1.74
train communication system
train communication network plus its communication and application profile for the exchange of information trainwide

3.1.75
train control and monitoring system
collection of train on-board services and functions for the operation of a train.

Note 1 to entry: A definition of those services is given in IEC 61375-2-4.

3.1.76
train backbone
bus connecting the consists of a train

Note 1 to entry: Conforms to IEC 61375-2-1 (WTB) or IEC 61375-2-5 (ETB).

3.1.77
train backbone node
device on the train backbone which receives a train backbone node number during inauguration.

Note 1 to entry: A train backbone node can be used to connect an End Device or a Consist Network to the Train Backbone.

3.1.78
train backbone node number
node address
node number
number assigned to each train backbone node during inauguration, which indicates the position of the train backbone node on the train backbone

3.1.79
train backbone view
train seen as a sequence of consist networks determined by ETB inauguration

3.1.80
train directory
data structure describing the train composition after a train inauguration

Note 1 to entry: The train directory is a part of the train topology database.

3.1.81
train network address
dynamic network address, which is used to address communication devices in other consist networks.

Note 1 to entry: This address can change after each inauguration.

3.1.82
train network management
services of the network management for TCN

3.1.83
train directory computation
protocol to determine the actual train composition as a set of consists containing vehicles containing function devices

Note 1 to entry: Optional closed trains represented as consists can be detected and interpreted as a set of consists. Result of the train directory computation is the train directory data structure which is a part of the train topology database.

3.1.84
train reference direction
direction determined by inauguration

3.1.85
train topology database

data base collecting all the information about the actual train composition

Note 1 to entry: An instance of this database is computed within each consist of the train.

3.1.86
train view
train seen as sequence of consists determined by the ETB inauguration

3.1.87
transport layer
layer of the OSI model responsible for end-to-end flow control and error recovery

3.1.88
UIC vehicle identification number
12-digit number for the unique identification of vehicles

Note 1 to entry: As defined by the UIC.

3.1.89
unspecified PDU
TRDP PDU with ComId=0

Note 1 to entry: The user data content of an unspecified PDU is undetermined for the communication profile. Different PDUs and their content can be distinguished only at application level.

3.1.90
vehicle
single coach or locomotive

3.1.91
vehicle gap
gap
vehicle listed in the confirmed vehicle list of the operational train directory which is not reachable over ETB

Note 1 to entry: A vehicle gap can be at one train end or in the middle of a train. There may also exist more than one vehicle gap in a train.

Note 2 to entry: Vehicle gaps may be determined by the driver while comparing the indicated unconfirmed train composition with the real train composition, leading to a correction of the train composition.

3.2 Abbreviations and acronyms

ALG	Application Layer Gateway
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CN	Consist Network
ComId	Communication Identifier
CRC	Cyclic Redundancy Check
CST	Consist
CSTINFO	Consist Information telegram
DC	Direct Current
DEV	Device
DNS	Domain Name System
DoS	Denial of Service
FCS	Frame Check Sequence
ECN	Ethernet Consist Network
ECSC	ETB Control Service Client
ECSP	ETB Control Service Provider
ED	End Device
ETB	Ethernet Train Backbone
ETBCTRL	ETB Control packet

ETBN	Ethernet Train Backbone Node
FCT	Function
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
LEN	Length
MAC	Medium Access Control
MC	Multicast
MD	Message Data
MD-PDU	Message Data Protocol Data Unit
MIB	Management Information Base
MTU	Maximal Transmission Unit
MVB	Multifunction Vehicle Bus
NAT	Network Address Translation
NOK	Not OK
OID	Object Identifier
OMTS	Onboard Multimedia and Telematic Subsystems
OSI	Open System Interconnection
PD	Process Data
PD-PDU	Process Data Protocol Data Unit
PICS	Protocol Implementation Conformance Statement
PROP	Properties
QoS	Quality of Service
RFC	Request For Comments
SDSINK	Safe Data Sink
SDSRC	Safe Data Source
SDTv2	Safe Data Transmission version 2
SID	Source Identifier
SIL	Safety Integrity Level
SMI	Safe Message Identifier
SNMP	Simple Network Management Protocol
SSC	Safety Sequence Counter
STC	Safe Topography Counter

TCN	Train Communication Network
TCMS	Train Control and Monitoring System
TCP	Transmissiont Control Protocol
TP	Test Procedure
TSS&TP	Test Suite Structure and Test Purpose
TRDP	Train Real Time Data Protocol
TTDB	Train Topology Database
TTL	Time to Live
TTDP	Train Topology Discovery Protocol
UDP	User Datagram Protocol
UIC	Union Internationale des Chemins de Fer
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
UuT	Unit under Test
VDP	Vital Data Packet
VLAN	Virtual Local Area Network
WTB	Wire Train Bus
XML	Extended Markup Language
XSD	XML Schema Definition

3.3 Conventions

3.3.1 Base of numeric values

This part of IEC 61375 uses a decimal representation for all numeric values unless otherwise noted.

Analog and fractional values include a comma.

EXAMPLE The voltage is 20,0 V.

Binary and hexadecimal values are represented using the ASN.1 (ISO/IEC 8824) convention.

EXAMPLE Decimal 20 coded on 8 bits = ‘0001 0100’B = ‘14’H.

3.3.2 Character strings and citations

Character strings are placed within double quotes.

EXAMPLE “ComProfTestAppl”.

Single quotes are used for citations (quotations).

EXAMPLE parameter ‘etbTopoCnt’.

Sometimes character strings are parameter values, in which case they can be written as citations.

EXAMPLE telegram type ‘Mn’.

3.3.3 Naming conventions

Keywords are written with a capital letter at the beginning.

If the keyword name is composed, the different parts of the name are united with a space, and all parts begin with a capital letter.

EXAMPLES “Train Backbone”, “Consist”, “Consist Network”.

Parameters are written with a capital letter at the beginning.

If the parameter name is composed, the different parts of the name are united without a space, and all parts begin with a capital letter except for the first part which begins with a lower case letter.

EXAMPLE ‘etbTopoCnt’.

3.3.4 Diagram conventions

Whenever appropriate, UML diagram types are used. In particular, the following UML diagram types are applied:

- UML state machines
- UML sequence diagram

For conditions used in these diagram types the following conventions are applied:

operators	=	Assignment operator: set a value
	==	Equality operator: equal
	!=	Equality operator: not equal
	OR	Logical operator: {expression} OR {expression}
	AND	Logical operator: {expression} AND {expression}
boolean	FALSE	Antivalent2 value 0: 01_b
	TRUE	Antivalent2 value 1: 10_b

For triggers and guards used in these diagram types the following conventions are applied:

- A trigger is an external event, e.g. a user command, which triggers a state transition
- In the special case that a trigger is the reception of a (periodic) ETBCTRL packet, the control flags within the ETBCTRL packet determine the state transition. For those cases only the guards are indicated in the UML state machines

3.3.5 Annotation of data structures

3.3.5.1 General

Data structures are defined following the presentation and encoding of transmitted and stored data as ruled in IEC 61375-2-1, which bases on ISO ASN.1 syntax.

All data within a data structure are organized in big-endian format (most significant octet of a data item first).

All data within a data structure are naturally aligned (stored at offset address which is a multiple of their size).

Data values of parameters within data or telegram structures which are marked ‘application defined’ are either defined by IEC 61375-2-4 or are left to the user to be defined.

3.3.5.2 Additional data types

In addition to the data types defined in IEC 61375-2-1, the following data types are used:

```
(1)    LABEL::= ARRAY [16] OF CHAR

(2)    AntivalentType::= ANTIVALENT8
```

with the encoding:

7	6	5	4	3	2	1	0	interpretation
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
0	0	0	0	0	0	0	1	FALSE
0	0	0	0	0	0	1	0	TRUE
0	0	0	0	0	0	1	1	UNDEFINED
any other value								ERROR

3.3.5.3 Data type keywords

In addition to the keywords/notations defined in IEC 61375-2-1 as an extension to ASN.1, some shorter, but equivalent keywords/notations are used within this part of IEC 61375 in order to improve the readability of data structures. These keywords are (see Table 1):

Table 1 – Data type keywords and notations

Keyword/notation used in this part of IEC 61375	Equivalent keyword/notation used in IEC 61375-2-1
UINT#	UNSIGNED#
INT#	INTEGER#
CHAR	CHARACTER8
Type [IndexList]	ARRAY [IndexList] OF Type
(# = any unsigned integer)	

4 Architecture

4.1 General

This clause defines the train related communication architecture, which is the basis for the communication profile. The train related communication architecture is defined from two different viewpoints, the physical train architecture, and the logical train architecture. The physical train architecture provides a system breakdown by considering components and interfaces in between, whereas the logical train architecture is based on functional breakdown dealing with functions and their interactions. Both views are necessary for defining important

communication profile concepts, like for instance the definition of the train topology database or the definition of a train wide logical addressing concept based on URIs.

4.2 Physical train architecture (system breakdown)

4.2.1 General

The physical train architecture defines a train as composed of closed trains, consists, vehicles, devices and the interfaces in between.

4.2.2 Train network architectures

4.2.2.1 Train structure

IEC 61375-1 defines a train as a composition of closed trains and consists, each consist having one or several vehicles, and each closed train having one or several consists as shown in Figure 2.

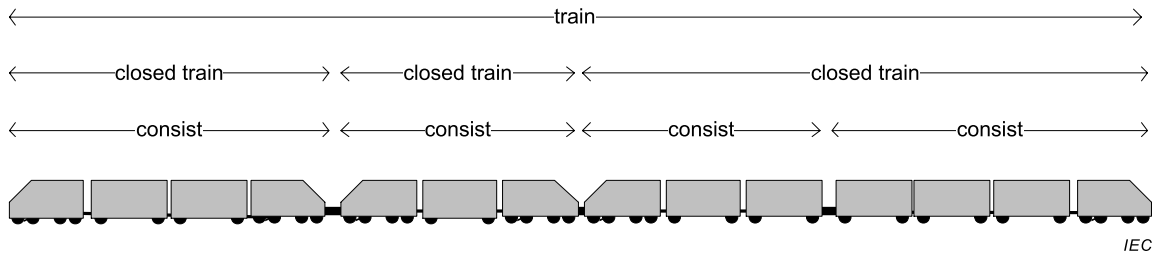


Figure 2 – Train structure in accordance to IEC 61375-1 (example)

For this part of the standard, the following restriction is made:

Although a closed train is a composition of one or several consists (closed train consists), it shall be seen as one operational consist for the scope of this communication profile. In this sense, a train is only composed of (operational) consists as shown in Figure 3, whereby some of the operational consists may be a closed train. Consequently, every closed train has also to behave like a consist with respect to this communication profile.

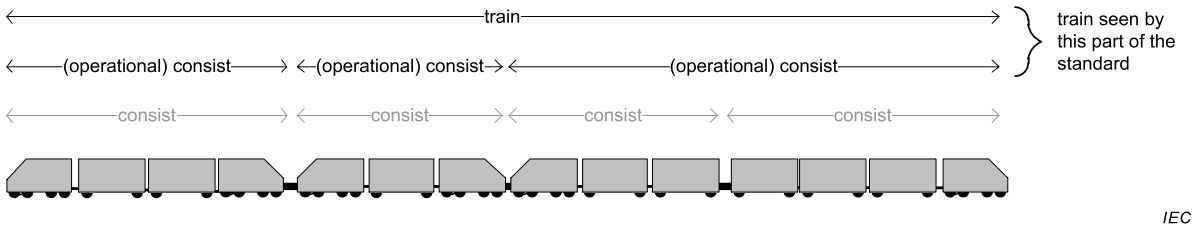


Figure 3 – Train structure seen from viewpoint of the communication profile (example)

NOTE 1 Communication inside a closed train, e.g. between the closed train consists which constitute the closed train, is not in the scope of this standard.

This communication profile is applicable for trains with a maximum number of 63 vehicles.

NOTE 2 How vehicles are allocated to consists depends on the chosen consist design. For example, each vehicle in a train can be a consist, in which case a train may also contain 63 consists, or two trainsets with 32 and 31 vehicles form each one consist, in which case the train contains only 2 consists.

4.2.2.2 Train network allocation

IEC 61375-1 defines a train wide network as a composition of consist networks interconnected by a common train backbone as shown in the example of Figure 4, where one or several consist networks per consist are connected to the ETB via a redundant pair of ETBN.

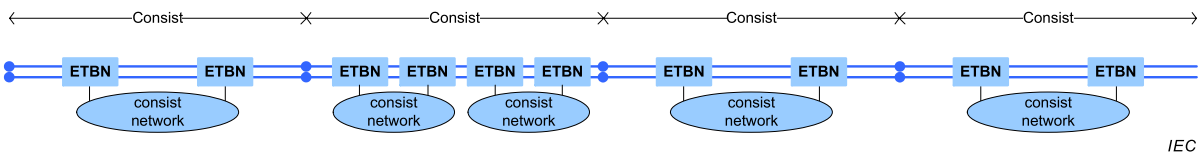


Figure 4 – Train network (example)

Furthermore, IEC 61375-1 allows applying up to 4 train backbones in parallel.

This communication profile defines the usage of the train backbones in the following way:

- One ETB (with subordinated consist networks) shall be used as operational network, supporting TCMS services.
- No, one or several ETBs (with subordinated consist networks) shall be used as multimedia network supporting OMTS services

The operational ETB may also be used for OMTS services.

NOTE 1 To have TCMS services and OMTS services sharing one (operational) ETB may be used for simple applications.

NOTE 2 If one physical network is used for both TCMS services and OMTS services, a separation can be done on an upper layer, e.g. on link layer by using VLAN. The expectation is, however, that requirements related to safety and security are much harder to fulfill for such type of network.

4.2.2.3 ETBN redundancy

At least one ETBN shall be provided per consist network. For redundancy purposes, it is recommended that two ETBNs are at least provided on traction units and for the backbone connection of the driving cab equipment of driving trailers (this also applies to multiple units).

4.2.2.4 Safety and security aspects

The operational network shall be a category 1 (preferred) or a category 2 transmission system as defined by IEC 62280.

4.2.2.5 IP Address space

Both operational network and multimedia network shall share one common train network address space as defined in IEC 61375-2-5 (i.e. 10.128/9). The train network address space for each network shall be identified.

NOTE If both networks are physically separated, the backbone id, 10^{th} and 11^{th} bit in train network address, is used for identification.

4.2.2.6 Interconnecting operational network and multimedia network

4.2.2.6.1 ETB Dependency

In order to ensure interoperability between trains with multiple ETBs, all functions connected to those ETBs shall share a common train view and a common operational train view as defined in 4.2.4.1.

NOTE The impact of this requirement on the TTDB is defined in 5.3.4.2.

4.2.2.6.2 Network couplings

There are several ways of coupling operational network and multimedia network (see Figure 5).

- Both networks can share a common CN, in which case the two networks are coupled on OSI layer 2.
- Both networks can be interconnected by an IP router (eventually with firewall functions), thus providing a coupling on OSI layer 3.
- Both networks can be separated by an application layer gateway (ALG), meaning a coupling on OSI layer 7.

NOTE Modern IP routers with firewall functionality allow also a filtering on (application data) content. In this case the networks are coupled on both layer 3 and layer 7.

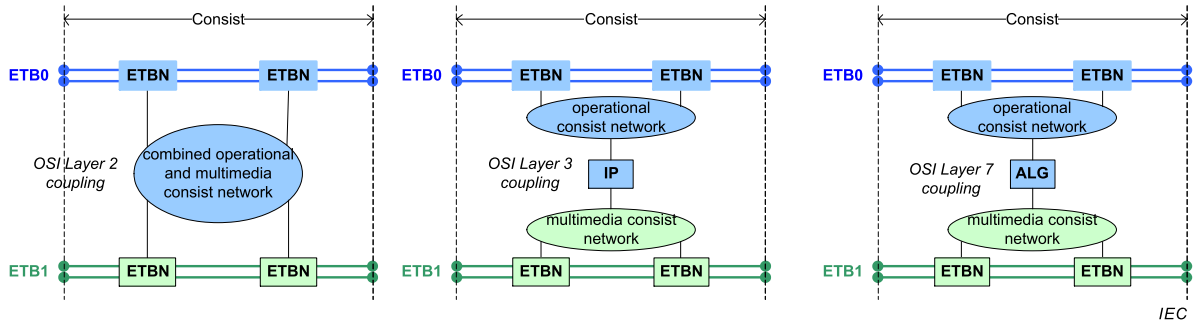


Figure 5 – Possible couplings of operational network and multimedia network

If there is a coupling on at least OSI layer 3, EDs connected to the operational consist network are said to be related to the operational ETB (ETB0), and EDs connected to the multimedia consist network are said to be related to the multimedia ETB (ETB1).

In case of OSI layer 2 coupling this relationship is undetermined.

NOTE The relationship of EDs is an information that is relevant for IP MC group assignments, see 5.4.5.2.

4.2.2.6.3 Data exchange between operational network and multimedia network

The way of data exchange between the operational network and multimedia network depends on the coupling of the two networks. While couplings on OSI layer 2 and 3 allow a data exchange on IP telegram base that is mostly handled within the communication layers of modern operating systems, a coupling on OSI layer 7 provides a better application control of exchanged data.

The following example describes a possible ALG design of an operational/multimedia gateway:

EXAMPLE

The gateway for interconnecting the operational network and the multimedia network (Figure 6) belongs formally to the operational network. This gateway is not only needed for the reasons of security and closed system. It also serves as entry point for multimedia network applications to operational network applications and vice versa, offering services to the respective side, which allows exchanging information with the respective, other side. This also means that the gateway has to provide two different interfaces: the interface to operational section based on the communication protocols used in the operational section, and the interface to the multimedia section implementing the protocols of the multimedia section. As an example, the gateway may receive the train directory information from the operational section ECSP within a TRDP Message Data packet, and offers this information in a web service to the multimedia section participants. Other interface service examples are to provide data streaming (video and audio) or direct exchange of TRDP data packets. So both client/server related interface services as well as real-time data interface services might be needed. As real-time data requires a high throughput, those data are normally routed on OSI layer 3 rather than processed on layer 7. In order to cope with the security, additional security measures are then required as for instance firewall functionality.

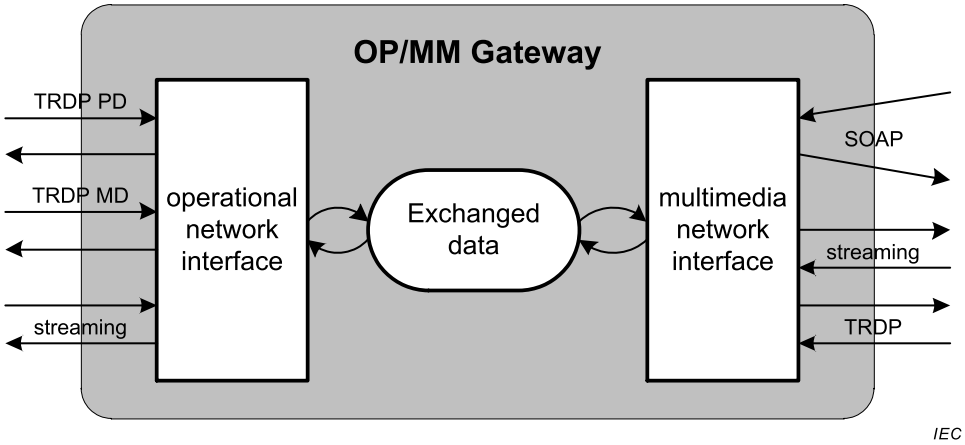


Figure 6 – Gateway between operational network and multimedia network (example)

4.2.2.7 Safety and security aspects

An architecture with a coupled operational network and multimedia network shall not compromise the transmission system categorization of the operational network as defined in 4.2.2.4.

In case the multimedia network is a category 3 transmission system in accordance to IEC 62280, at least a coupling on OSI layer 3 with firewall capabilities shall be envisaged, better a coupling on OSI layer 7.

It is not within the scope of this standard to define the necessary capabilities of the firewall as this is highly related to the system design and its security requirements.

The suitability of the coupling to preserve the transmission system categorization of the operational network shall be demonstrated within the system safety case.

4.2.2.8 Network Reconciliation

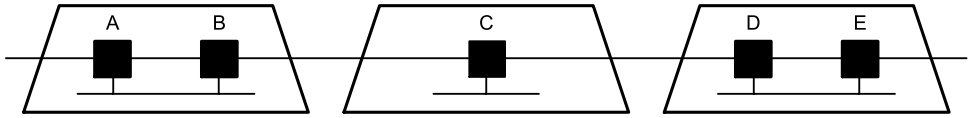
For applications of the multimedia network, it will be necessary to have the same operational train view (see 4.2.4.1) as for an application of the operational network, such as the sequence of consists and the position of the leading vehicle, in order to realize an unambiguous addressing scheme within the train.

As a common rule, the operational train view determined by the operational network and manifested within the operational train directory of the TTDB as defined in 5.3 shall be authoritative for the multimedia network. Hence, consist sequence and position of the leading vehicle shall be adopted from the operational network.

4.2.3 Closed Trains

4.2.3.1 General

As defined in IEC 61375-1, a closed train is composed by consists. The example in Figure 7 shows three coupled consists with UUID 8608, 1761, 2900. Each consist takes part in the ETB inauguration as defined in IEC 61375-2-5.



TTDP TOPOLOGY

cstUuid	8608		1761	2900			
ownEtbnNb	1	2	1	2	1		
etbnNodeRole	EtbnRoleMaster	EtbnRoleBackup	EtbnRoleNotRedundant	EtbnRoleBackup	EtbnRoleMaster		
nEtbnCst	2		1	2			
nCnCst	1		1	1			
cnToEtbnList	cn01	cn01	cn01	cn01	cn01		

IEC

NOTE The table TTDP TOPOLOGY in Figure 7 contains only the relevant elements.

Figure 7 – Example: three coupled Consists

Consist with UUID 8608 contains two ETBNs “A” and “B” with the static relative positions 1 and 2 in the consist. ETBNs “A” and “B” are redundant. Both ETBNs are connected to the first and only consist network.

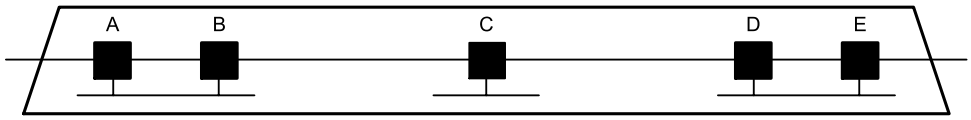
Consist with UUID 1761 contains the ETBN “C” with the static relative position 1 in the consist. ETBN “C” is non-redundant. The ETBN is connected to the first and only consist network.

Consist with UUID 2900 contains two ETBNs “D” and “E” with the static relative positions 1 and 2 in the consist. ETBNs “D” and “E” are redundant. Both ETBNs are connected to the first and only consist network.

4.2.3.2 Closed Train Representation in ETB inauguration

For the ETB inauguration according to IEC 61375-2-5, the closed train shall be represented as a single consist.

For the composition of a closed train the parameters for the TTDP TOPOLOGY frame contents as defined in IEC 61375-2-5 shall be adapted. Figure 8 shows the external view of the closed train example and the relevant TTDP TOPOLOGY frame elements.



TTDP TOPOLOGY

cstUuid	5171				
ownEtbnNb	1	2	3	4	5
etbnNodeRole	EtbnRoleMaster	EtbnRoleBackup	EtbnRoleNotRedundant	EtbnRoleBackup	EtbnRoleMaster
nEtbnCst	5				
nCnCst	3				
cnToEtbnList	cn01	cn01	cn02	cn03	cn03

IEC

Figure 8 – Example: Closed Train

In the example in Figure 8 the consist with UUID 5171 contains the five ETBNs “A”, “B”, “C, “D” and “E” from Figure 7 with the static relative positions 1, 2, 3, 4 and 5 in the Consist, respectively. ETBNs ”A”/”B” and “D”/“E” are redundant. The total number of ETBNs in the Consist is five. The total number of Consist Networks in the Consist is three. The ETBNs ”A”

and ”B” are connected to the first Consist Network. The ETBN ”C” is connected to the second Consist Network. The ETBNs ”D” and ”E” are connected to the third Consist Network.

During ETB inauguration each ETBN of the closed train (i.e. consist with UUID 5171) sends its TTDP TOPOLOGY frame. Therefore another externally coupled consist knows all about the network topology of the closed train.

4.2.3.3 Deficit of information about train composition

Since the closed train in Figure 8 is represented as a single consist, the information about the mapping from ETBNs and connected consist networks to vehicles is lost. This loss of information can be recovered during computation of the TTDB.

4.2.4 Directions

4.2.4.1 Train views

A train can be viewed from different perspectives, which depend on the role of the viewer.

In the train backbone view, a train is composed of consist networks which are connected to one or more ETBs via ETBNs. This view represents the network topology of the train. A train backbone topology will be changed if ETBNs are connected or removed. This view is defined in IEC 61375-2-5.

In the train view, a train is composed of consists and the consists are composed of vehicles. A TCN train composition will be changed if consists are removed or appended. Its defined extremities determine the directions in the train.

In the operational train view (driver’s and attendance view), a train is composed of a series of consists as well as a series of vehicles. Directions in the train are defined by the main view direction of the driver’s cab in the leading vehicle. An operational train view will be changed if the leading cab is changed or if consists are appended or removed and there is no leading cab.

4.2.4.2 Train reference directions

The TCN directions and orientation within a train, consist and vehicle are specified in IEC 61375-1.

IEC 61375-1 specifies as well the numbering of vehicles in a consist and the numbering of consists within a train. The numbering is related to the Extremity 1 of a vehicle, a consist, or a train.

IEC 61375-1 leaves open how the extremities will be assigned. The following rules for assigning the extremities shall apply:

- Extremity 1 of a vehicle shall be statically predefined by the design of that vehicle.
- Extremity 1 of a consist shall be statically predefined by the design of that consist.
- Extremity 1 of a train is the train end, which is pointed to by the ETB reference direction 1 as defined in IEC 61375-2-5.

4.2.4.3 Operational directions

From operator/driver point of view, one end of a train is the train head and the opposite end the train rear, which are determined by the operational train direction. The operational train direction points from the train rear to the train head. The operational train directions are defined according to the rules a) to e).

a) If there is a leading vehicle in the train, the operational train direction is determined by the viewing direction of the driving cab of the leading vehicle.

If there is no leading vehicle, then the operational train direction shall be determined by the following rules:

b) If there was a previously leading vehicle, then this vehicle together with its previously operational train direction determines the operational train direction.

c) If there were several previously leading vehicles, e.g. after a train lengthening, and all of these have the same viewing direction, then this viewing direction determines the operational train direction.

d) If there are multiple consists in the train and a traction consist at one end, then this end defines the train head and traction consist includes vehicle 01 and determines the operational train direction.

e) In all other cases, the operational train direction shall correspond to the ETB reference direction according to IEC 61375-2-5.

4.2.5 Consist and vehicle basic properties

4.2.5.1 General

Consists and vehicles shall be described by a basic set of consist and vehicle properties, which will be stored within the Train topology database. For this a distinction should be made between static, topological and dynamic properties:

– Static properties are features built into the vehicle or consist and thus only dependent on its class or design, e.g. “vehicle identifier”, “vehicle has first class seats”, “traction unit with electric drive”.

– Topological properties are properties, which present the physical configuration of train structure, e.g. vehicle sequence, vehicle orientation.

– Dynamic properties are properties which define a train assignment, e.g. “leading vehicle”, “vehicle sequence number” or “train reference direction”.

4.2.5.2 Vehicle and consist type information

Vehicles and consists may receive a type classifier which makes it easier for applications to check whether vehicles or consists have common static properties. Those type classifiers are defined as follows:

```
VehType          LABEL          -- Type of the vehicle
                                   Zero terminated string

CstType          LABEL          -- Type of the consist
                                   Zero terminated string
```

The definition of type classifiers is application specific. For the scope of this communication profile, the type classifier is only informal.

4.2.5.3 Static Properties

A list of static consist and vehicle properties is provided in IEC 61375-2-4.

4.2.5.4 Identification

A consist shall be uniquely identified by a consist UUID as defined in IEC 61375-2-5.

An application defined vehicle identifier shall uniquely identify a vehicle.

EXAMPLE UIC defined a 12-digit wagon number for the unique identification of coaches, as for example the number “50 80 31 – 34 025 – 2”.

4.3 Logical Train Architecture (Functional Breakdown)

4.3.1 General

The logical train architecture defines a train as composed of services and the information exchanged between the services and between user and services.

4.3.2 Service classification

Services can be classified in (see Figure 9):

- Operational services, as defined in this part of IEC 61375 and in IEC 61375-2-4.
- Multimedia services, as defined in IEC 62580.

Subject of this part of the standard are ETB control services and shared ETB system services. Operational services are defined in IEC 61375-2-4.

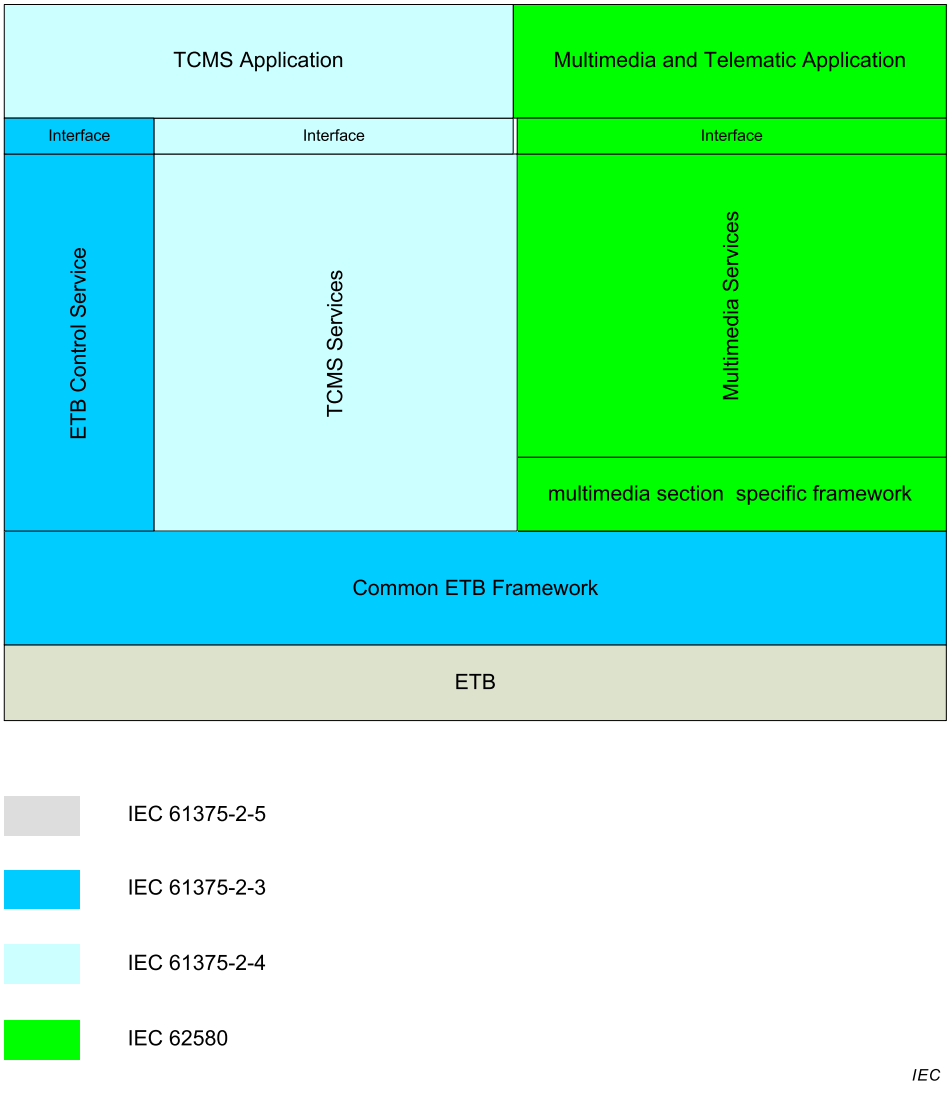


Figure 9 – Service classification

All the services use functions, protocols, and definitions from a common ETB framework, which is defined in Clause 5.

For multimedia services, a multimedia specific framework may be added which provides multimedia specific functions, protocols, and definitions. The definition of the multimedia specific framework is not within the scope of this standard.

4.3.3 Operational Services Overview

Besides TCMS services as specified in IEC 61375-2-4, this part of the standard defines in Clause 6 an ETB control service whose service functions are listed in Table 2.

NOTE The functional breakdown of TCMS services is related to EN 15380-4.

Table 2 – ETB control service

Function	Short form	Classifi-cation	Location	Related subclause	Description
Manage leading vehicle information	Function Leading	control	Consist	6.5	Set or reset leading vehicle on request and handle leading vehicle conflicts
Confirm train configuration	Function Confirmation/Correction	control	Consist	6.6	Allow a user to confirm or to correct the indicated train composition
Manage energy saving mode Manage battery protection mode	Function Sleep Mode	control	Consist	6.8	Set train to low power mode, wake up on command or on ETB activity

4.3.4 Service Provider

The ETB control service shall be provided by the ECSP (see also 6.2).

In consists with several consist networks, the ETB control services shall only be provided within one consist network. This will be the consist network which also hosts the end device responsible for ETB control (ECSC). The reason for this restriction is to avoid control conflicts within a consist.

For redundant ECSP only one ECSP is actively providing the service.

5 Common ETB framework

5.1 General

5.1.1 Overview

The common ETB framework defines basic functions and basic concepts, which together provide the infrastructure the services are bedded in. In particular, it defines:

– CSTINFO Telegram (see 5.2)
– Train topology database TTDB (see 5.3)
– Service addressing (see 5.4)
– TCN-DNS Server (see 5.5)
– Data exchange (see 5.6)
– Service discovery (see 5.6.4)
– Train Info service (see 5.8)

5.1.2 Interoperability

Telegrams exchanged on ETB between consists and their related data structures are versioned using a data structure for version information which is defined as:

```
VERSION::= RECORD
  {
  mainVersion      UINT8     -- main version, incremented for
                                incompatible changes



        value range: 1..255 (0 = reserved)
  subVersion       UINT8     -- sub version, incremented for
                                compatible changes
                                value range: 0..255
  }
```

In order to ensure interoperability between consists, all changes affecting normative clauses of this standard shall be treated as incompatible changes, leading to an increment of parameter ‘mainVersion’, unless otherwise specified.

This standard does not define a “negotiation” of the supported versions when two consists are coupled implementing different versions of the standard’s normative clauses. Such a negotiation would be needed if compatible changes were allowed.

Compatible changes might be applied for informal parts, like for instance the data exchange between ECSC and ECSP defined in Annex E.

5.2 CSTINFO telegram

5.2.1 General

As defined in part IEC 61375-2-5, following an ETB train inauguration, at least one entitled device in each consist shall send out a CSTINFO telegram over ETB to all other entitled devices within the train to inform about consist properties.

For the sake of simplicity, the ECSP shall be entitled to send and receive the CSTINFO telegrams.

5.2.2 Closed train support (Option)

For the ETB inauguration a closed train is represented as a single consist (see 4.2.3.2). Therefore at least one ECSP within the closed train shall send a CSTINFO telegram which describes the closed train as one consist (consist info class 2, see 5.2.4).

In order to learn the consist structure of a closed train, at least one ECSP within a closed train consist shall send a CSTINFO telegram (consist info class 3, see 5.2.4).

5.2.3 Protocol

The protocol for sending CSTINFO telegrams is defined in 5.3.2.

5.2.4 CSTINFO classes

CSTINFO classes are defined to distinguish the CSTINFO telegrams sent by consists, closed trains and closed train consists. The following three classes are defined:

- CSTINFO class 1: CSTINFO telegrams sent by consists which are no closed trains

- CSTINFO class 2: CSTINFO telegrams sent by closed trains
- CSTINFO class 3: CSTINFO telegrams sent by closed train consists

Consists which are no closed trains shall be able to receive and process CSTINFO classes 1 and 2, but need not be able to receive consist info class 3 telegrams.

Only CSTINFO class 2 telegrams shall provide a closed train consist list (array “cltrCstList” CSTINFO telegram). For CSTINFO classes 1 and 3 this array shall be omitted (parameter “cltrCstListCnt” set to 0, see 5.2.5).

5.2.5 CSTINFO Notification Message

CSTINFO messages shall be sent within the payload of a Train Realtime Data Protocol (TRDP) message as defined in Clause A.7.

The CSTINFO size shall be restricted to the maximal DatasetLength parameter value of a MD-PDU as defined in Clause A.7.

TRDP MD Notification telegram parameters:

MsgType:	‘Mn’
Protocol:	UDP
ComId:	2
SourceURI (user part):	void
DestinationURI (user part):	void
DestinationURI (host part):	grpECSP.anyVeh.aCst.aClTrn.lTrn
SourceURI (host part):	-
Dataset:	CSTINFO

The TRDP MD UserDataset is defined as (see Figure 10):

```
CSTINFO::= RECORD
  {
  cstInfo     CONSIST_INFO  -- consist information as defined in
                               subclause 5.3.3.2.6
  }
```

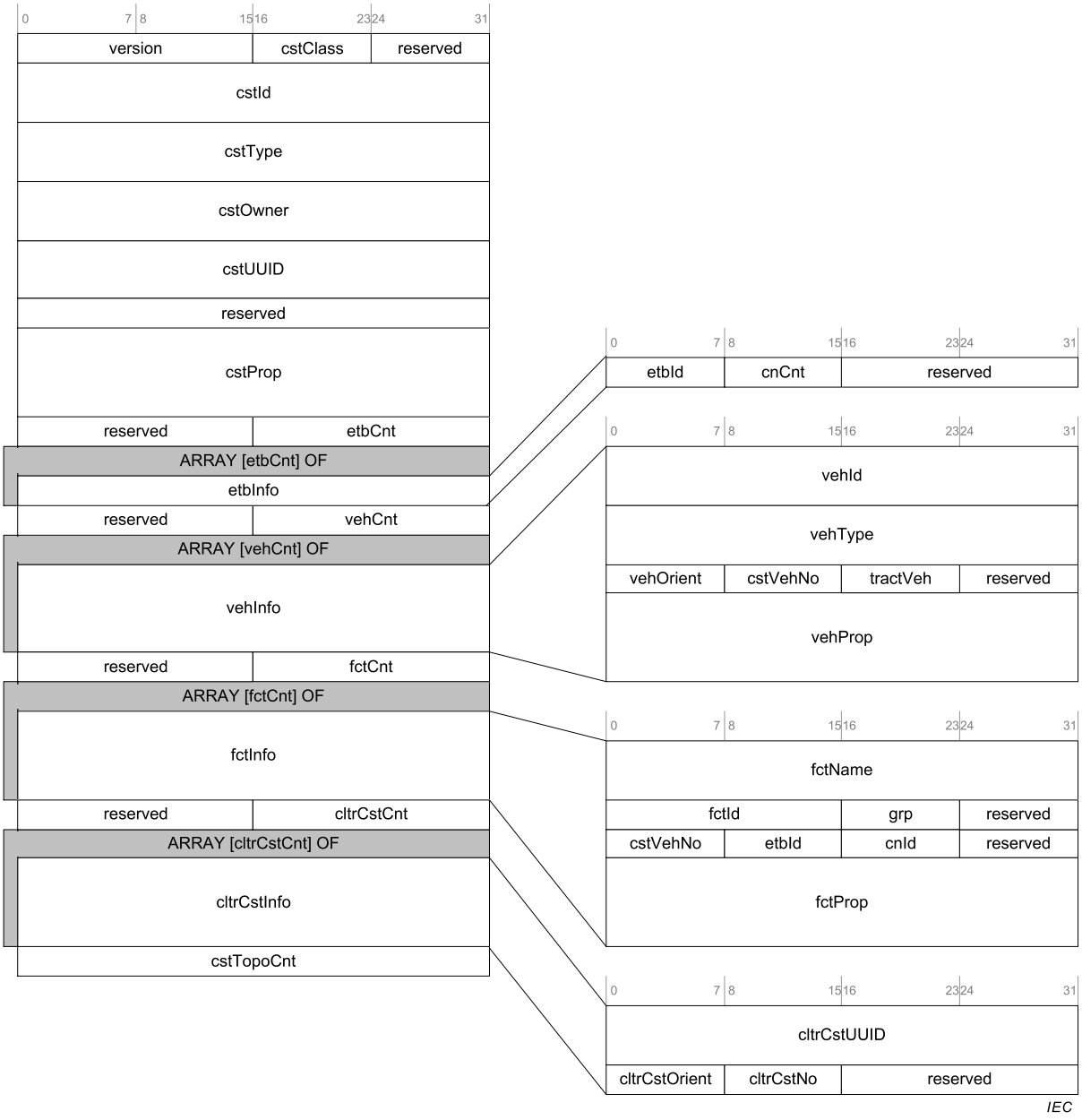


Figure 10 – CSTINFO notification data

5.2.6 CSTINFO Request

CSTINFO telegram sending can be controlled and requested by sending a CSTINFOCTRL notification message as depicted in Figure 11.

The protocol for sending CSTINFOCTRL notification messages is defined in 5.3.2.

CSTINFOCTRL notification message shall be sent within the payload of a Train Realtime Data Protocol (TRDP) message as defined in Clause A.7.

CSTINFOCTRL notification messages support two control modes (differentiated with parameter ‘trnCstNo’):

trnCstNo > 0	Used to request the sending of CSTINFO telegrams. The receiving ECSP shall memorize the parameter value

	'trnTopoCnt'. This mode shall be used by the ECSP responsible for the CSTINFO exchange within a consist (see 5.3.2.3)
trnCstNo = 0	Used to request the sending of CSTINFO telegrams, but without tracking the 'trnTopoCnt' parameter value. This mode shall be used by all devices interested in CSTINFO except for the ECSP responsible for the CSTINFO exchange.

TRDP MD Notification telegram parameters:

MsgType:	'Mn'
Protocol:	UDP
ComId:	3
SMI:	3 (only needed if SDTv2 is used)
UserDataVersion:	'0100'H (only needed if SDTv2 is used)
SourceURI (user part):	void
DestinationURI (user part):	void
DestinationURI (host part):	grpECSP.anyVeh.aCst.aClTrn.lTrn
SourceURI (host part):	-
Dataset:	CSTINFOCTRL

The TRDP MD UserDataset is defined as (see Figure 11):

```
CSTINFOCTRL::= RECORD
  {
  version      VERSION        -- version of the telegram
                                 mainVersion = 1
                                 subversion = 0
  trnCstNo     UINT8          -- train consist number
                                 0 = without trnTopoCnt tracking
                                 > 0 = with trnTopoCnt tracking
  cstCnt       UINT8          -- number of consists in list
  cstList      ARRAY[] OF CONSIST
                              -- consist list.
                                 If trnCstNo > 0 this shall be an
                                 ordered list starting with trnCstNo = 1
                                 (exactly the same as in structure
                                 TRAIN_DIRECTORY).
                                 If trnCstNo = 0 it is not mandatory to
                                 list all consists (only consists which
                                 should send CSTINFO telegram).
                                 parameters 'trnCstNo' and 'cstOrient'
                                 are optional and can be set to 0.
  trnTopoCnt   UINT32         -- trnTopoCnt value
```

```
                                trnCstNo > 0: actual value
                                trnCstNo = 0: set to 0
safetyTrail ETBCTRL_VDP     -- ETB-VDP trailer
                                parameter 'safeTopoCount' = 0
                                completely set to 0 = SDTv2 not used
}
```

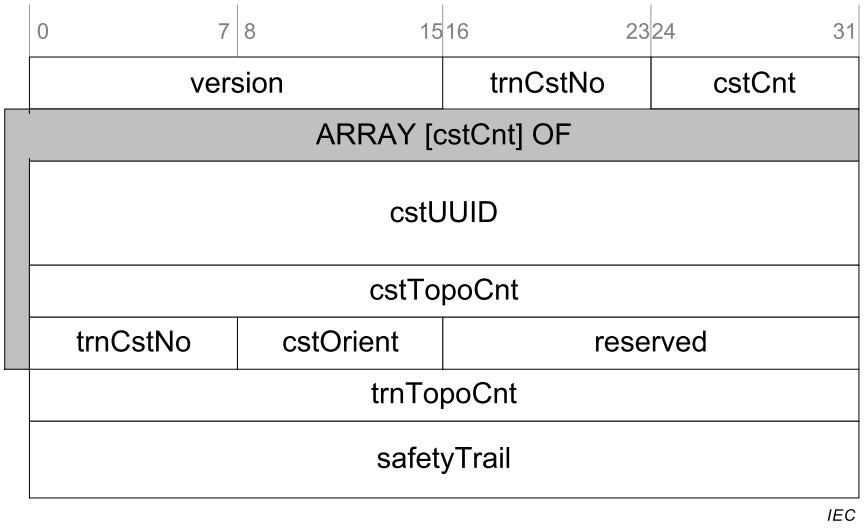


Figure 11 – CSTINFOCTRL telegram

5.3 Train topology database

5.3.1 General

The train topology database (TTDB) is a repository for all the information related to the actual train composition and the actual ETB state.

An instance of the TTDB shall be available at least once in each consist.

NOTE In consists with multiple consist networks it might be an advantage to have multiple TTDB instances, one for each consist network

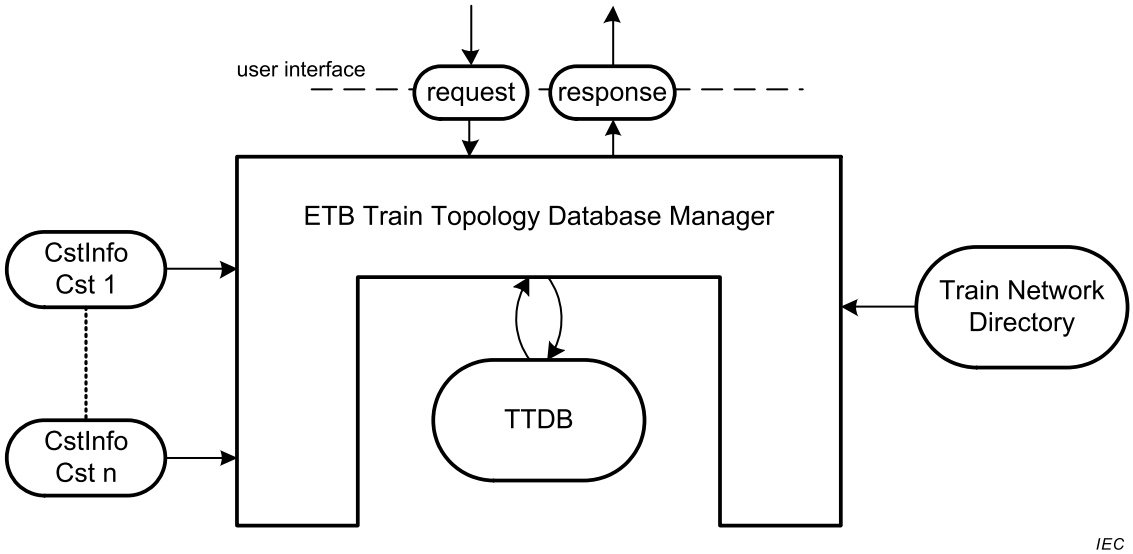


Figure 12 – TTDB management block diagram

The TTDB is maintained by a TTDB Manager function as shown in Figure 12, which reads in consist information from other consists (provided by the CSTINFO telegrams) and the result from ETB train inauguration obtained from the ETBN (Train network directory as defined in IEC 61375-2-5) and stores this information to the TTDB. Towards the application services, it provides a user interface, which allows retrieving parts or all the information from the TTDB (see Annex E and Annex G). The TTDB manager shall always keep the TTDB up-to-date.

The TTDB consists of four associated data blocks as it is depicted in Figure 13.

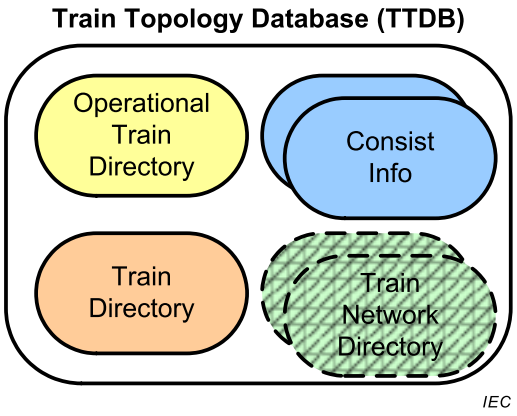


Figure 13 – TTDB Content

Consist Info Contains the descriptions of each consist as they are distributed after ETB train inauguration within CSTINFO telegrams. This includes also a list of functions provided by the consist and all the consist related information which a TCN-DNS server needs for TCN-URI address resolution. This part is statically preconfigured for each consist. In trains with multiple ETBs, the consist info is available for each ETB and the content of the different consist info data blocks content can be different for each individual ETB (see 5.3.4 for details).

Closed trains are represented as consists in the consist info data block, and closed train consist infos may be added optionally. Those entries are identifiable by the consist info class parameter (see 5.2.4).

Train Network Directory Contains the description of the network topology of one ETB. This is the result of the ETB train inauguration as defined in IEC 61375-2-5. The train network directory is hallmarked by unique signature called etbTopoCnt. The TTDB may contain up to four train network directories representing the maximal four ETBs of the train network.

Train Directory is (re-) computed each time there is a change in the train composition indicated by a change of the etbTopoCnt as defined in IEC 61375-2-5. The train directory provides an ordered list of all consists which have been discovered over the ETB the train directory is associated to.

Operational Train Directory is (re-) computed each time there is a change in the train directory or upon a user request for leading (see 6.5) or for correction (see 6.6). This data block adds information about leading and correction to the TTDB and represents the operational train view as defined in 4.2.4.1. In trains with multiple ETBs only one operational train directory for all ETBs will be generated.

Each of these data blocks is protected by a signature, which allows checking the integrity of the stored data, see 5.3.3 for a detailed definition of these data blocks.

5.3.2 Computation of the TTDB

5.3.2.1 General

The TTDB is computed in several steps and by different functions as it is shown in Figure 14:

- ETB Inauguration, as defined in IEC 61375-2-5.
- Train directory computation on the base of the train network directory and the consist info.
- Operational train directory computation on the base of leading and correction information.

The different functions are defined in more detail in the following paragraphs.

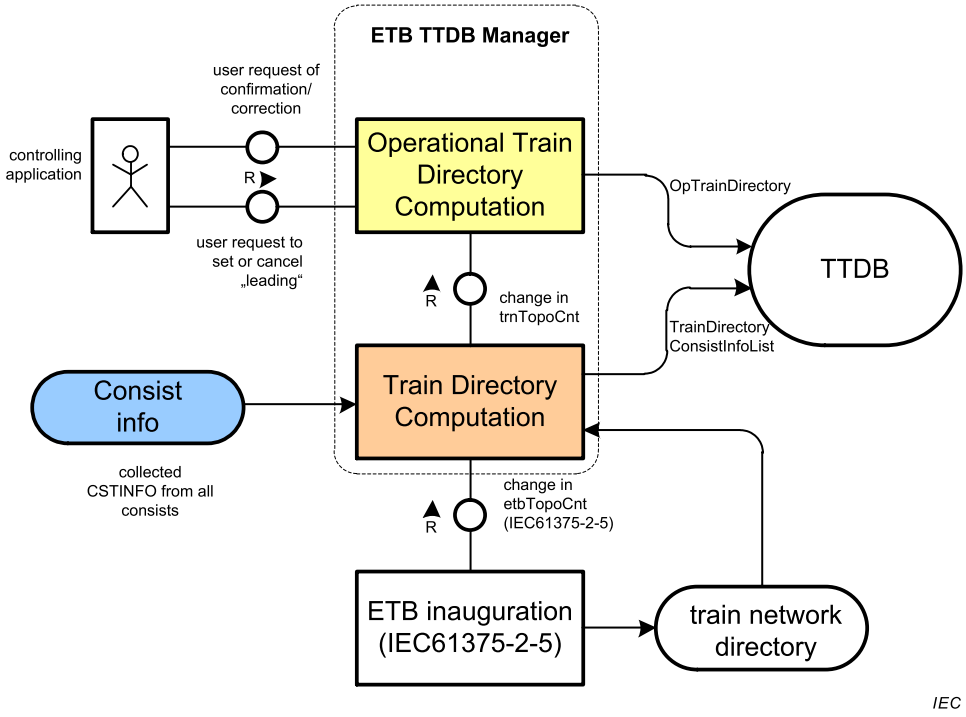


Figure 14 – TTDB computation block diagram

5.3.2.2 ETB train inauguration

Basis of the TTDB is the train network directory, which is computed in accordance to the algorithm defined in IEC 61375-2-5, and which represents the topology of the ETB as a sequence of ETBN belonging to consists. A discovered ETB network topology will be uniquely identified by its topography counter (etbTopoCnt).

5.3.2.3 Computation of the train directory

A (re-)computation of the train directory shall be executed each time there is a change of the etbTopoCnt.

During train directory computation, the TTDB shall be enriched by consist information collected from the consists in the train (CSTINFO telegrams).

The process of train directory computation is defined by the state diagram shown in Figure 15 and further described in Table 3 to Table 5.

NOTE 1 A recomputation of the train directory will be prevented if train inaugurations are inhibited as defined in IEC 61375-2-5.

For the optimization of CSTINFO and CSTINFOCTRL message exchange (network load reduction) a set of flags controlling the sending of CSTINFO and CSTINFOCTRL messages shall be supported, which are defined as:

```
ciTxFlag     BOOLEAN        -- CSTINFO send control flag
                               FALSE: CSTINFO not to send
                               TRUE: CSTINFO to send
cicTxCnt1    UINT8          -- CSTINFOCTRL send control flag (counter)
                               value range: 0..1
cicTxCnt2    UINT8          -- CSTINFOCTRL send control flag (counter)
                               value range: 0..2
```

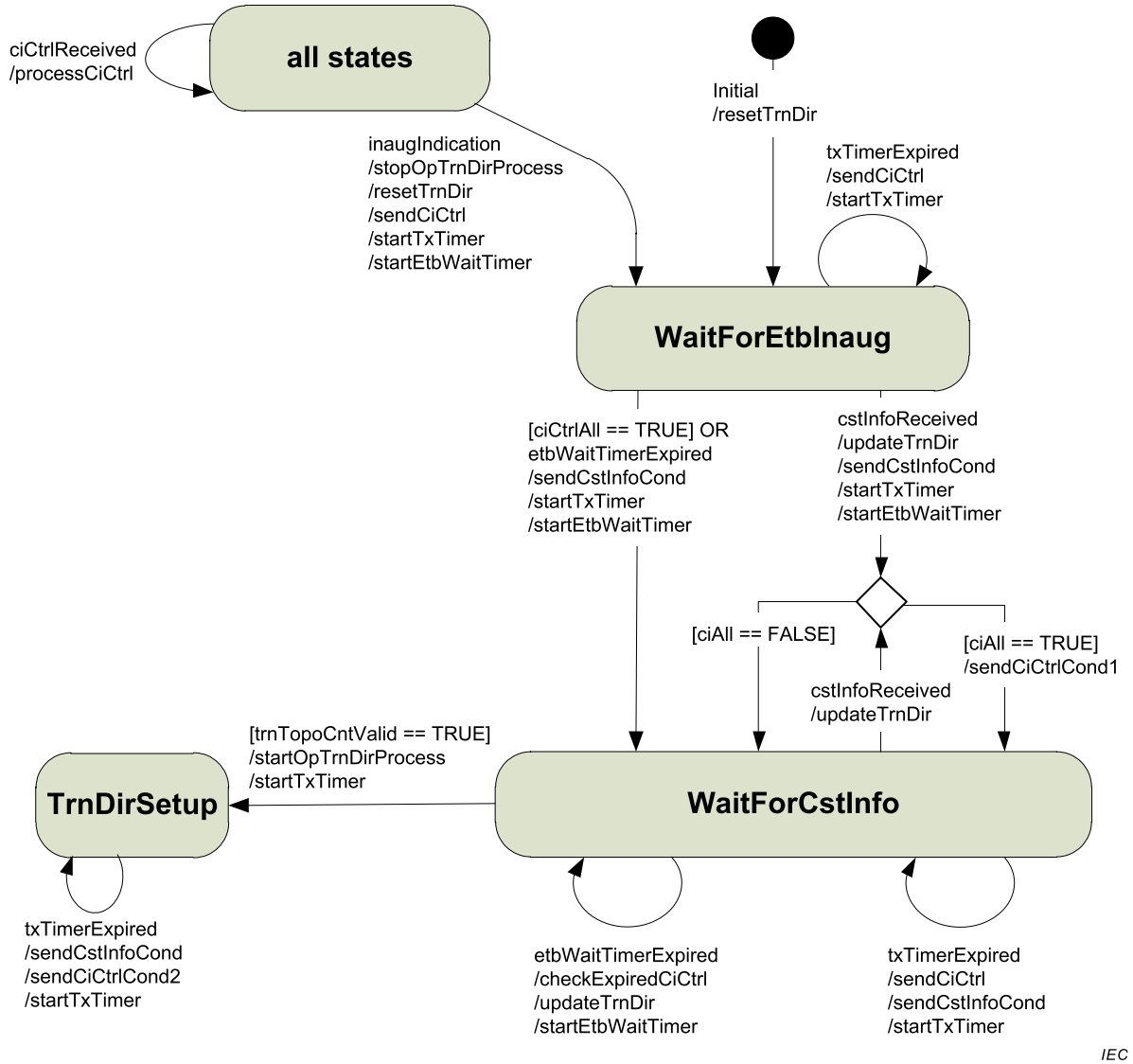


Figure 15 – Train directory computation state diagram

NOTE 2 For some transitions in the UML state machine on Figure 15 a trigger is the reception of a CSTINFOCTRL packet. In those cases only the guards are indicated in the UML state machine transition.

Table 3 – Train directory computation – triggers

Trigger	Description
Initial	Power-up or reboot of ECSP
inaugIndication	Indication of a new train inauguration provided by the local ETBN in accordance to IEC 61375-2-5.
ciCtrlReceived	CSTINFOCTRL telegram has been received.
cstInfoReceived	CSTINFO telegram has been received.
etbWaitTimerExpired	CSTINFOCTRL wait timer expiration – indicating that some consist listed in Train Network Directory is not sending CSTINFOCTRL packets, i.e. ECSP in this consist is not operating correctly.
txTimerExpired	Timer for periodic transmission of CSTINFO and CSTINFOCTRL telegram has expired. Expiration of this timer indicates that previously transmitted CSTINFO or CSTINFOCTRL has not been received by all ECSP.

Table 4 – Train directory computation – guards

Guard	Description
[trnTopoCntValid == TRUE]	CSTINFOCTRL has been received from all consists listed in the train directory with parameter 'cstTopoCnt' value != 0 and all consist from which CSTINFOCTRL has been received have identical trnTopoCnt value. NOTE 1 The trnTopoCnt value is known from the CSTINFOCTRL telegrams received with `trnCstNo` > 0. NOTE 2 This means that all active ECSP share the same input data for the calculation of Operational Train Directory.
[CiAll == TRUE]	CSTINFO message received from all consists listed in the train directory
[CiAll == FALSE]	CSTINFO message not (yet) received from all consists listed in the train directory
[CiCtrlAll == TRUE]	CSTINFOCTRL message received from all consists listed in the train directory
[CiCtrlAll == FALSE]	CSTINFOCTRL message not (yet) received from all consists listed in the train directory

Table 5 – Train directory computation – actions

Action	Description
startEtbWaitTimer	(Re-)Start one-shot timer with a time value of (5,0 ± 0,2) s. Within this time CSTINFOCTRL telegrams from all consists listed in Train Directory should be received.
startTxTimer	(Re-)Start one-shot timer with a time value (1,0 ± 0,1) s. This timer drives the periodic retransmission of CSTINFOCTRL and CSTINFO telegrams in case of some network failure. NOTE CSTINFO is retransmitted only if requested by some CSTINFOCTRL telegram.
resetTrnDir	Reset content of train directory: • train directory contains all consists identified by ETB inauguration (listed in Train Network Directory). • cstTopoCnt in all CONSIST structures are set to 0, except for own

Action	Description
	consist. • compute the trnTopoCnt Clear the consist info list except for the entry of the own consist Reset send control flags: ciTxFlag = FALSE cicTxCnt1 = 1 cicTxCnt2 = 0
updateTrnDir	Store CSTINFO in ConsistInfoList. Update train directory: • Update CONSIST structures in train directory with cstTopoCnt from received CSTINFO • Calculate new value of trnTopoCnt.
startOpTrnDirProcess	Start the operational train directory processing by triggering the event 'startOpTrnDirProcess' in the ETBCTRL processing state machine, see 6.4.5
stopOpTrnDirProcess	Stop the operational train directory processing by triggering the event 'stopOpTrnDirProcess' in the ETBCTRL processing state machine, see 6.4.5

sendCstInfoCond	Conditional sending of CSTINFO telegram:
	IF (ciTxFlag == TRUE) THEN
	send CSTINFO telegram;
	ciTxFlag = FALSE;
	END;
	NOTE The conditional sending of CSTINFO aims to reduce the network load

sendCiCtrl	Send CSTINFOCTRL telegram

sendCiCtrlCond1	Conditional sending of CSTINFOCTRL telegram:
	IF (cicTxCnt1 > 0) THEN
	send CSTINFOCTRL telegram;
	cicTxCnt1 = cicTxCnt1 – 1;
	END;
	NOTE The conditional sending of CSTINFOCTRL aims to reduce the network load

sendCiCtrlCond2	Conditional sending of CSTINFOCTRL telegram:
	IF (cicTxCnt2 > 0) THEN
	send CSTINFOCTRL telegram;
	cicTxCnt2 = cicTxCnt2 – 1;
	END;
	NOTE The conditional sending of CSTINFOCTRL aims to reduce the network load

processCiCtrl	Process received CSTINFOCTRL telegram.
	IF (CSTINFOCTRL parameter 'trnCstNo' > 0) THEN
	memorize the CSTINFOCTRL parameter 'trnTopoCnt';
	memorize the time when CSTINFOCTRL has been received;
	END;
	IF ((CSTINFOCTRL parameter 'trnCstNo' > 0) AND (state == TrainDirSetup) AND (CSTINFOCTRL parameter 'trnTopoCnt != trnTopoCnt value in local train directory)) THEN
	cicTxCnt2 = 2;

Action	Description

	END;
	IF (CSTINFOCTRL message does contain an entry of parameter 'cstTopoCnt' with a value == 0 for the local consist, which is the consist the receiving ECSP belongs to) THEN
	ciTxFlag = TRUE;
	END;
	NOTE This means that the sending ECSP has not received the CSTINFO of the local consist and is requesting now to (re)send this CSTINFO message.

checkExpiredCiCtr	Check for expired CSTINFOCTRL messages. For each consist from which the last received CSTINFOCTRL message is older than the value of etbWaitTimer: • Discard the CSTINFOCTRL message together with the memorized trnTopoCnt value. • Remove the consist info from the consist info list . • Set consist parameter 'cstTopoCnt' in train directory to 0.

NOTE The process of train directory computation is triggered by the ETBN after train inauguration. Subsequently, the ECSPs exchange CSTINFOCTRL and CSTINFO telegrams for the computation of the train directory. ETBN and ECSP need to be synchonized in a way that the ECSP can start the train directory computation process immediately after the train inauguration indication. If the time difference is too large (e.g. greater than EtbWaitTime) this consist will not be listed in the train directory with a cstTopoCnt != 0 and will subsequently not be listed in the operational train directory.

5.3.2.4 Computation of the operational train directory

A computation of the operational train directory shall be executed on user request and each time there is a change of the train directory associated to the operational network (ETB0).

If the user requests to set leading, the leading vehicle shall be determined and indicated in the database. This is further specified in 6.5.

If the user requests to cancel leading, the leading vehicle indication shall be removed from the database. This is further specified in 6.5.

As long as there is no user request for confirmation or correction, the TTDB shall be in state "unconfirmed".

If the user requests to correct the TTDB, the operational train directory shall be updated and the TTDB shall change to state "confirmed" if there is no conflict or error, otherwise it shall remain in state "unconfirmed". This is further specified in 6.6.

If the TTDB is in state "confirmed" and the train directory changes, it shall be checked whether the new train directory is in disagreement with the confirmed and/or corrected train composition. If there is a disagreement, the TTDB state shall be changed to "unconfirmed", this is further specified in 6.6.

NOTE This will for instance occur after the coupling of two trains.

Further specification on the computation of the operational train directory is given in 6.7.

5.3.3 Data structure

5.3.3.1 Data model

The detailed TTDB structure can be modeled as shown in Figure 16.

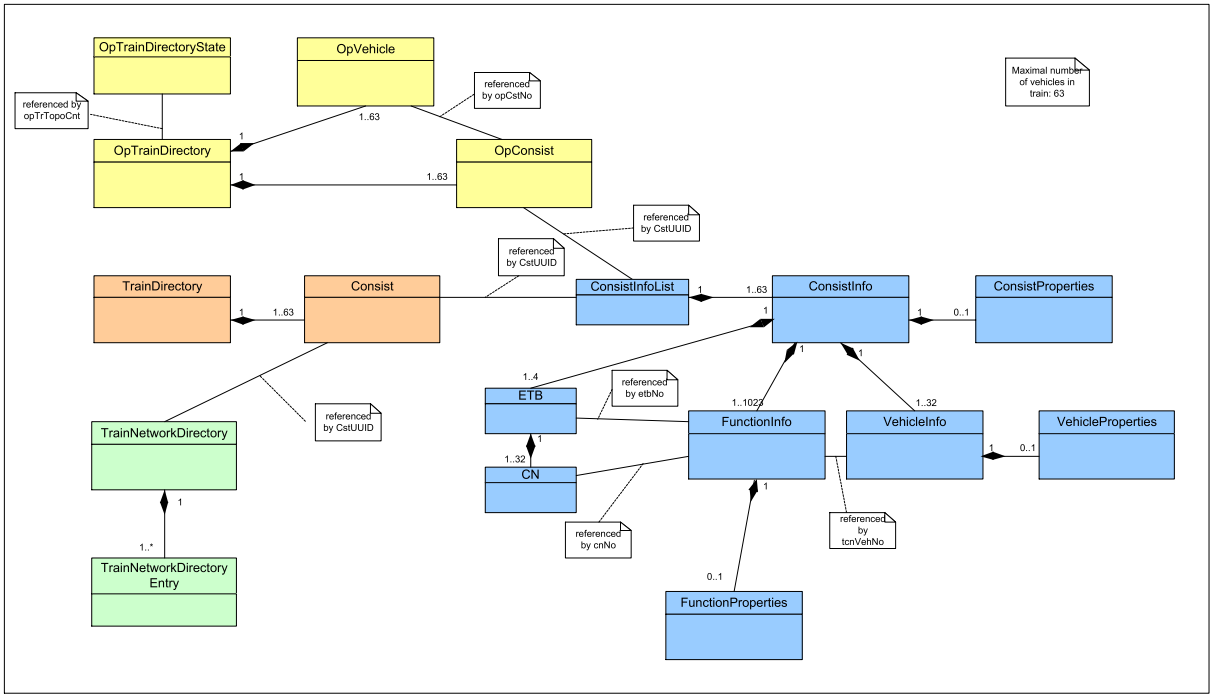


Figure 16 – TTDB class diagram (example)

The four TTDB parts listed in 5.3.1 are separated by aggregation associations.

NOTE An application interface for retrieving data from the TTDB is defined in Annex E.

5.3.3.2 Data structures

5.3.3.2.1 Structure ETB_INFO – ETB information

The data structure ETB_INFO is defined as:

```
ETB_INFO::= RECORD
  {
  etbId        UINT8          -- identification of the ETB
                                 0 = ETB0 (operational network)
                                 1 = ETB1 (multimedia network)
                                 2 = ETB2 (other network)
                                 3 = ETB3 (other network)
  cnCnt        UINT8          -- number of CN within
                                 consist connected to this ETB,
                                 value range 1..16,
                                 referring to cnId 0….15 acc. IEC 61375-2-5
  reserved01   UINT16         -- reserved for future use (= 0)
  }
```

5.3.3.2.2 Structure CLTR_CST_INFO – info on closed train composition

The data structure CLTR_CST_INFO is defined as:

```
CLTR_CST_INFO::= RECORD
  {
  cltrCstUUID      UINT8[16] -- UUID of the consist

  cltrCstOrient    UINT8     -- consist orientation
```

```
                                ‘01’B = same as closed train direction
                                ‘10’B = inverse to closed train direction
  cltrCstNo        UINT8     -- sequence number of the consist within the
                                closed train, value range 1..32
  reserved01       UINT16    -- for future use (= 0)
  }
```

5.3.3.2.3 Structure PROPERTIES – application defined properties

The data structure PROPERTIES is defined as:

```
PROPERTIES::= RECORD
  {
  version     VERSION       -- properties version information,
                               application defined
  len         UINT16        -- properties length in number of octets,
                               application defined, shall be a multiple
                               of 4 octets for alignment reasons
                               value range: 0..32768
  prop        UINT8[]       -- properties, application defined
  }
```

5.3.3.2.4 Structure FUNCTION_INFO – function attributes

This dataset describes the functions which are defined for a consist.

The data structure FUNCTION_INFO is defined as:

```
FUNCTION_INFO::= RECORD
  {
  fctName     LABEL         -- function device or group label as defined
                               in 5.4.4.6.2
  fctId       UINT16        -- host identification of the function
                               device or group as defined in
                               IEC 61375-2-5, application defined.
                               Value range:
                               Devices = 1..16382
                               consist-limited/ETB-related group
                               = 1..254
                               ETB-related group = 256..16382
                               All-train group = 256..32767
  grp         BOOLEAN8      -- is a function group (will be resolved
                               to an IP multicast address)
                               0 = no
                               1 = yes
  reserved01  UINT8         -- reserved for future use (= 0)
  cstVehNo    UINT8         -- sequence number of the vehicle within the
                               consist the function belongs to
                               value range: 1..32
                               0 = not defined
  etbId       UINT8         -- identification of the related ETB
                               0 = ETB0 (operational network)
                               1 = ETB1 (multimedia network)
                               2 = ETB2 (other network)
                               3 = ETB3 (other network)
                               255 = related to all ETB (all-train group)
  cnId        UINT8         -- identification of connected consist network
```

```
                                in the consist, related to the etbId
                                value range: 0..32
                                0 = not relevant (e.g. for groups)
  reserved02   UINT8          -- reserved for future use (= 0)
  }
```

The combination of parameters ‘fctName’, ‘fctId’, ‘cnId’ and ‘etbId’ shall be unique.

This part of IEC 61375 defines only that part of the function information, which is needed for function identification and function addressing. More detailed information about the function itself, and the way to retrieve this information over the network, shall be defined within the related application profile.

NOTE The application profile can define a TRDP message and related service primitives to retrieve function information. Good practise is to define a standard way which is common for all services.

5.3.3.2.5 Structure VEHICLE_INFO – static vehicle attributes

The data structure VEHICLE_INFO is defined as:

```
VEHICLE_INFO::= RECORD
  {
  vehId        LABEL          -- vehicle identifier label,application defined
                                 (e.g. UIC vehicle identification number)
                                 vehId of vehNo = 1 can be used also as
                                 consist identifier
  vehType      LABEL          -- vehicle type,application defined
  vehOrient    UINT8          -- vehicle orientation
                                 ‘01’B = same as consist direction
                                 ‘10’B = inverse to consist direction
  cstVehNo     UINT8          -- sequence number of vehicle within the
                                 consist(1..32)
  tractVeh     ANTIVALENT8    -- vehicle is a traction vehicle
                                 ‘01’B = vehicle is not a traction vehicle
                                 ‘10’B = vehicle is a traction vehicle
  reserved01   UINT8          -- for future use (= 0)
  vehProp      PROPERTIES     -- static vehicle properties
  }
```

5.3.3.2.6 Structure CONSIST_INFO– static consist attributes

This dataset is filled with the information, which has been received with the corresponding CSTINFO telegram.

The data structure CONSIST_INFO is defined as:

```
CONSIST_INFO::= RECORD
  {
  version      VERSION        -- CONSIST_INFO data structure version,
                                 mainVersion = 1
                                 subVersion = 0
  cstClass     UINT8          -- consist info classification
                                 1 = (single) consist
                                 2 = closed train
                                 3 = closed train consist
  reserved01   UINT8          -- reserved for future use (= 0)
  cstId        LABEL          -- consist identifier, application defined
                                 (e.g. UIC identifier)
```

```
  cstType      LABEL          -- consist type,application defined
  cstOwner     LABEL          -- consist owner,application defined
                                 (e.g. "trenitalia.it")
  cstUUID      UINT8[16]      -- UUID of the consist
  reserved02   UINT32         -- reserved for future use (= 0)
  cstProp      PROPERTIES     -- static consist properties
  reserved03   UINT16         -- reserved for future use (= 0)
  etbCnt       UINT16         -- number of ETB's, range: 1..4
  etbInfoList  ARRAY [] OF ETB_INFO
                              -- ETB information for ETBs in the consist
                                 Ordered list starting with lowest etbId
  reserved04   UINT16         -- reserved for future use (= 0)
  vehCnt       UINT16         -- number of vehicles in consist
                                 value range: 1..32
  vehInfoList  ARRAY [] OF VEHICLE_INFO
                              -- vehicle info for the vehicles in the consist
                                 Ordered list starting with cstVehNo = 1
  reserved05   UINT16         -- reserved for future use (= 0)
  fctCnt       UINT16         -- number of consist functions
                                 value range: 0..1024
  fctInfoList  ARRAY [] OF FUNCTION_INFO
                              -- function info for the functions in consist
                                 lexicographical ordered by fctName
  reserved06   UINT16         -- reserved for future use (= 0)
  cltrCstCnt   UINT16         -- number of closed train consists
                                 value range: 0..32
                                 0 = consist is no closed train
  cltrCstInfoList ARRAY [] OF CLTR_CST_INFO
                              -- info on closed train composition
                                 Ordered list starting with cltrCstNo = 1
  cstTopoCnt   UINT32         -- consist topography counter
                                 from the beginning of the record until the
                                 last element before the topography counter
                                 computed as defined in 5.3.3.2.16
                                 (seed value: 'FFFFFFFF'H)
}
```

5.3.3.2.7 Structure CONSIST_INFO_LIST – list of consist info

The consist info list provides a collection of CONSIST_INFO elements as defined in 5.3.3.2.7 which are filled with the content of received CSTINFO telegrams. It cannot be guaranteed that a consist info is available for all consists within a train, especially if there are consists inserted by correction or temporarily after a train inauguration. The consist info is identified by the cstUUID value of the consist.

Within the consist info list all data received from CSTINFO classes 1 and 2 telegrams shall be stored. Data received from CSTINFO class 3 telegrams need not be stored.

The data structure CONSIST_INFO_LIST is defined as:

```
CONSIST_INFO_LIST::= RECORD
  {
  version          VERSION   -- consist info list data structure version
                                 parameter 'mainVersion' shall be set to 1.
  cstInfoCnt       UINT16    -- number of consists in consist info list;
                                 value range: 1..63
  cstInfoList      ARRAY [] OF CONSIST_INFO
```

```
                                -- consist info collection with cstInfoCnt
                                   elements
  }
```

5.3.3.2.8 Structure CONSIST – dynamic consist attributes

This dataset provides the dynamic attributes for each consist discovered during the ETB inauguration. The parameter “cstUUID” allows to reference to the corresponding consist information stored within the data structure CONSIST_INFO_LIST.

Closed trains are represented as single consist. For more details on the closed train’s structure the corresponding entry in the CONSIST_INFO_LIST, which is a CSTINFO class 2 dataset, can be consulted.

Closed train consists are not referenced within this data structure. However, the CSTINFO class 3 datasets representing the close train consists are stored in the CONSIST_INFO_LIST as well.

The data structure CONSIST is defined as:

```
CONSIST::= RECORD
  {
  cstUUID      UINT8[16]      -- UUID of the consist, provided by ETBN
                                 (TRAIN_NETWORK_DIRECTORY)
                                 Reference to static consist attributes
  cstTopoCnt   UINT32         -- consist topography counter provided with
                                 the CSTINFO
                                 0 = no CSTINFO available
  trnCstNo     UINT8          -- Sequence number of consist
                                 in train (1..63)
  cstOrient    UINT8          -- consist orientation
                                 ‘01’B = same as train reference direction
                                 ‘10’B = inverse to train reference
                                 Direction
  reserved01   UINT16         -- for future use (= 0)
  }
```

5.3.3.2.9 Structure TRAIN_DIRECTORY

This dataset provides information about the train composition discovered during the ETB inauguration.

The data structure TRAIN_DIRECTORY is defined as:

```
TRAIN_DIRECTORY::= RECORD
  {
  version      VERSION        -- train directory data structure version
                                 mainVersion = 1
                                 subVersion = 0
  etbId        UINT8          -- identification of the ETB the train
                                 directory is computed for
                                 0 = ETB0 (operational network)
                                 1 = ETB1 (multimedia network)
                                 2 = ETB2 (other network)
                                 3 = ETB3 (other network)
  cstCnt       UINT8          -- number of consists in train
                                 range: 1..63
```

```
  cstList ARRAY [] OF CONSIST
                              -- dynamic consist list
                                 ordered list starting with trnCstNo = 1
  trnTopoCnt   UINT32         -- Train topography counter
                                 computed as defined in 5.3.3.2.16
                                 from the beginning of the record until the
                                 last element before the topography counter
                                 (seed value: etbTopoCnt)
  }
```

5.3.3.2.10 Structure OP_VEHICLE – operational vehicle attributes

The data structure OP_VEHICLE is defined as:

```
OP_VEHICLE::= RECORD
  {
  vehId        LABEL          -- unique vehicle identifier, application
                                 defined (e.g. UIC Identifier)
  opVehNo      UINT8          -- operational vehicle sequence number in train
                                 value range: 1..63
  isLead       ANTIVALENT8    -- vehicle is leading
  leadDir      UINT8          -- vehicle leading direction
                                 0 = not relevant
                                 1 = leading direction 1
                                 2 = leading direction 2
  trnVehNo     UINT8          -- vehicle sequence number within the train
                                 with vehicle 01 being the first vehicle
                                 in ETB reference direction 1 as defined in
                                 IEC 61375-2-5
                                 value range: 1..63
                                 a value of 0 indicates that this vehicle
                                 has been inserted by correction
  vehOrient    UINT8          -- vehicle orientation
                                 '0'B = not known (corrected vehicle)
                                 '01'B = same as operational train direction
                                 '10'B = inverse to operational train
                                 direction
  ownOpCstNo   UINT8          -- operational consist number the vehicle
                                 belongs to
  reserved01   UINT8          -- reserved for future use (= 0)
  reserved02   UINT8          -- reserved for future use (= 0)
  }
```

5.3.3.2.11 Structure OP_CONSIST – operational consist attributes

The data structure OP_CONSIST is defined as:

```
OP_CONSIST::= RECORD
  {
  cstUUID      UINT8[16]      -- Reference to static consist attributes
                                 0 = not available (e.g. correction)
  opCstNo      UINT8          -- operational consist number in train (1..63)
  opCstOrient  UINT8          -- consist orientation
                                 '0'B = not known (corrected vehicle)
                                 '01'B = same as operational train direction
                                 '10'B = inverse to operational train
                                 direction
```

```
  trnCstNo    UINT8         -- Sequence number of consist
                               in train with consist 01 being the first
                               consist in ETB reference direction 1
                               as defined in IEC 61375-2-5
                               value range 0..63
                               a value of 0 indicates that this consist
                               has been inserted by correction
  reserved01  UINT8         -- reserved for future use (= 0)
  }
```

5.3.3.2.12 Structure OP_TRAIN_DIRECTORY_STATE

The data structure OP_TRAIN_DIRECTORY_STATE is defined as:

```
OP_TRAIN_DIRECTORY_STATE::= RECORD
  {
  version          VERSION   -- TRAIN_DIRECTORY_STATE data structure version
                               mainVersion = 1
                               subVersion = 0
  reserved01       UINT8     -- reserved for future use (= 0)
  reserved02       UINT8     -- reserved for future use (= 0)
  etbId            UINT8     -- identification of the ETB the TTDB is
                               computed for
                               0 = ETB0 (operational network)
                               1 = ETB1 (multimedia network)
                               2 = ETB2 (other network)
                               3 = ETB3 (other network)
  trnDirState      UINT8     -- TTDB status
                               1 = UNCONFIRMED
                               2 = CONFIRMED
  opTrnDirState    UINT8     -- operational train directory status
                               1 = INVALID
                               2 = VALID
                               4 = SHARED
  reserved03       UINT8     -- reserved for future use (= 0)
  trnId            LABEL     -- train identifier,application defined
                               (e.g. “ICE75”, “IC346”), informal
  trnOperator      LABEL     -- train operator, e.g. “trenitalia.it”,
                               informal
  opTrnTopoCnt     UINT32    -- operational train topography counter
                               set to 0 if opTrnDirState = INVALID
  crc              UINT32    -- SC-32 computed over record
                               (seed value: ‘FFFFFFFF’H)
  }
```

5.3.3.2.13 Structure OP_TRAIN_DIRECTORY

The operational train directory represents the operational train view as defined in 4.2.4.1.

The data structure OP_TRAIN_DIRECTORY is defined as:

```
OP_TRAIN_DIRECTORY::= RECORD
  {
  version     VERSION       -- OpTrain data structure version
                               mainVersion = 1
                               subVersion = 0
  etbId       UINT8         -- identification of the ETB the train
```

```
                                directory computed for
                                0 = ETB0 (operational network)
                                1 = ETB1 (multimedia network)
                                2 = ETB2 (other network)
                                3 = ETB3 (other network)
  opTrnOrient UINT8          -- operational train orientation
                                '00'B = unknown
                                '01'B = same as train direction
                                '10'B = inverse to train direction
  reserved01  UINT8          -- reserved for future use (= 0)
  reserved02  UINT8          -- reserved for future use (= 0)
  reserved03  UINT8          -- reserved for future use (= 0)
  opCstCnt    UINT8          -- number of consists in train (1..63)
  opCstList   ARRAY [] OF OP_CONSIST
                             -- dynamic operational consist list
                                ordered list starting with opCstNo = 1
  reserved04  UINT8          -- reserved for future use (= 0)
  reserved05  UINT8          -- reserved for future use (= 0)
  reserved06  UINT8          -- reserved for future use (= 0)
  opVehCnt    UINT8          -- number of vehicles in train (1..63)
  opVehList   ARRAY [] OF OP_VEHICLE
                             -- dynamic operational vehicle list
                                ordered list starting with opVehNo = 1
  opTrnTopoCnt UINT32        -- operational train topography counter
                                computed as defined in 5.3.3.2.16
                                from the beginning of the record until the
                                last element before the topography counter
                                (seed value: trnTopoCnt)
  }
```

The data structure OP_TRAIN_DIRECTORY (the same is the case for the data structures TRAIN_DIRECTORY and CONSIST_INFO) is defined with dynamic arrays in order to reduce telegram length for transmission and also to have an easy definition of the parameters included in the topography counter computation. For implementation, static arrays could be used instead, which allow an easier access to array member parameters. Topography counter calculation has then to be adjusted in order to give the same result as defined herein.

5.3.3.2.14 Structure TRAIN_NETWORK_DIRECTORY_ENTRY (informal)

The train network directory entry represents one entry in the trnNetDir list of the train network directory. For a more detailed description see IEC 61375-2-5.

The data structure TRAIN_NETWORK_DIRECTORY_ENTRY is defined as:

```
TRAIN_NETWORK_DIRECTORY_ENTRY::= RECORD
  {
  cstUUID     UINT8[16]      -- unique consist identifier
  cstNetProp  UINT32         -- consist network properties
                                bit 0..1: consist orientation
                                bit 2..7 = 0
                                bit 8..13 = ETBN Id
                                bit 14..15 = 0
                                bit 16..21 = Subnet Id
                                bit 24..29 = CN Id
                                bit 30..31 = 0
  }
```

5.3.3.2.15 Structure TRAIN_NETWORK_DIRECTORY (informal)

The train network directory represents the network topology of one ETB as defined in IEC 61375-2-5.

The data structure TRAIN_NETWORK_DIRECTORY is defined as:

```
TRAIN_NETWORK_DIRECTORY::= RECORD
  {
  reserved01   UINT16          -- reserved for future use (=0)
  entryCnt     UINT16          -- number of entries in train net directory
  trnNetDir    ARRAY [entryCnt] OF TRAIN_NETWORK_DIRECTORY_ENTRY
  etbTopoCnt UINT32            -- train network directory CRC
  }
```

5.3.3.2.16 Computation of topography counter values

The TTDB defined three topography counters, which are used for different purposes:

cstTopoCnt:	unique signature of the consist info
trnTopoCnt:	unique signature of the train directory
opTrnTopoCnt:	unique signature of the operational train directory

For the computation of those topography counters, the following general rules shall apply:

a) First, a SC-32 checksum (see Clause B.7) over the specified data range and with the specified seed value shall be calculated.

b) If the calculation results in a value different to 0, the topography counter shall be set to the value of the SC-32 checksum.

c) If the calculation results in a value of 0, the topography counter shall be set to a value of $(2^{32} – 1)$.

d) Setting the topography counter to a value of 'FFFFFFFF'H declares the topography counter to be invalid.

e) The topography counter value shall be stored in big-endian format.

NOTE 1 Using a value of 0 to declare an invalid topography counter avoids the definition of an additional validity flag.

NOTE 2 The probability P_{crc} to not detect a modification of a data packet protected by a SC-32 CRC is about $P_{crc} \approx 2^{-32}$. The probability P_{tc} to not detect a modification of a data packet signed by a topography counter as specified in this part of IEC 61375 is about $P_{tc} \approx (2^{-32} + 2^{-63})$, a difference which is negligible.

5.3.4 Train Topology Database for multiple ETBs (Option)

5.3.4.1 General

The TTDB is built on the base of the train network directory, which represents the train backbone view of one particular ETB. This train backbone view can be different for the different ETBs within the train. If a TTDB were computed for each ETB, the consequence would be that a different train directory and operational train directory would be computed for each ETB, potentially leading to different train views dependent on which ETB is used for reference.

In order to avoid ambiguousness in the train views of trains with multiple ETBs (see 4.2.2.6.1), the train directory and the operational train directory shall be computed by ETB0 (operational ETB) only and shall be adopted by the other ETBs.

5.3.4.2 Adoption of the TTDB within dependent ETBs

Adopting the train directory and the operational train directory from ETB0 can be defined as a 5 step process (for illustration see Figure 17):

a) Each ETB executes the train inauguration and generates the train network directory as defined in IEC 61375-2-5.

b) Based on the inauguration result, IP addresses of ETBNs and the subnet addresses of the connected CNs are assigned as defined in IEC 61375-2-5.

c) CSTINFO telegrams are exchanged on each ETB as defined in 5.2.3 and stored within the consist info list data block. The train directory is established and received CSTINFO telegrams are registered.

d) ETB0 computes the operational train directory.

e) The other ETBs copy or reference the operational train directory computed by ETB0.

NOTE 1 Instead of copying the operational train directory, it is also possible to provide access from other ETBs to the operational train directory of the ETB0 ('referencing').

All ETBs shall continuously monitor the opTrnTopoCnt of the ETB0, and in case of a change, shall update their TTDB.

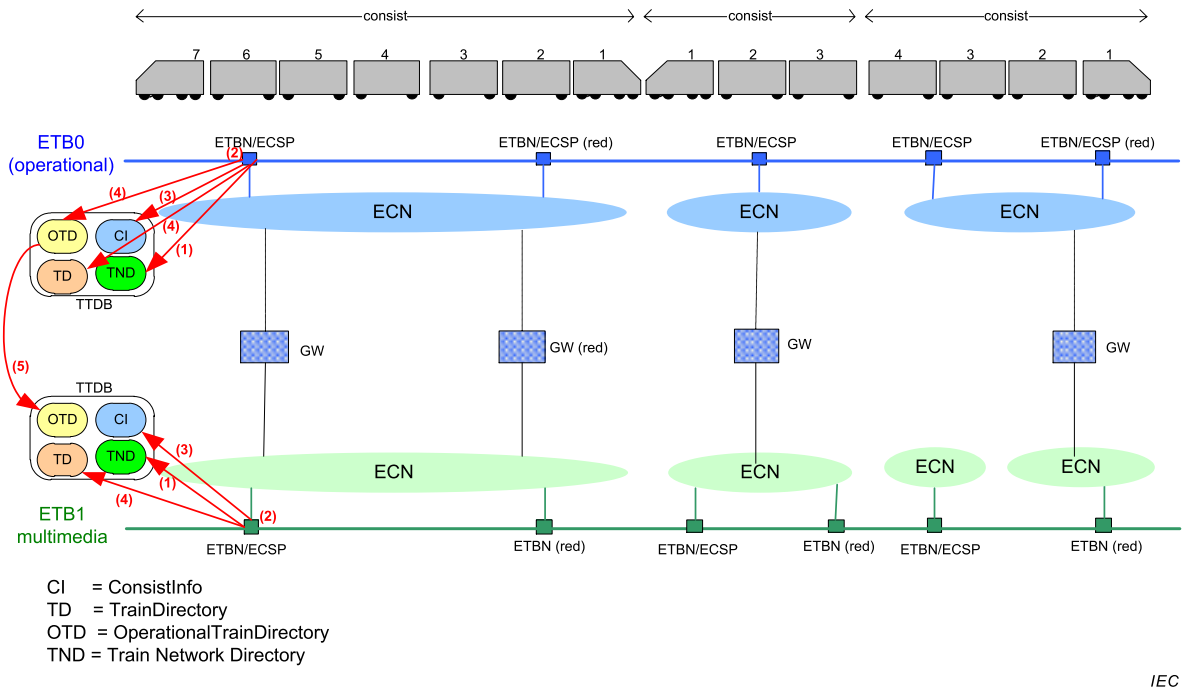


Figure 17 – TTDB adoption (in this example shown for the first consist)

NOTE 2 A consequence will be that a consist which is not reachable on ETB0, but available on multimedia network, is not listed within the TTDB of the multimedia network.

5.3.4.3 Restrictions on network architecture

In order to ensure that CSTINFO telegrams are sent by all consists on all ETBs, each consist in the train should be connected to all the ETBs.

5.3.4.4 Restrictions on CSTINFO telegrams

The CSTINFO telegrams sent by one consist on the different ETBs shall indicate the same number, the same sequence, the same orientation, and the same identity of contained vehicles.

Functions which shall be accessable from multiple ETBs shall be listed in all the ETB related CSTINFO telegrams.

NOTE A typical use case is that a function of the operational network sends data to a remote (other consist) function on the multimedia network. Then it finds all information of the remote function in its local TTDB.

All other data within the CSTINFO telegram may be different as they are not relevant for interoperability.

5.4 Service Addressing

5.4.1 General

Service addressing is a substantial part of a communication system as it identifies the endpoints of a communication relationship and as such the communication partners. Communication, and related to that the addressing, can be defined between physical entities, like physical devices or human beings, or can be defined between more abstract entities like application or system functions.

A schema for the train wide addressing of physical entities is defined in IEC 61375-2-5.

This part of the standard defines an addressing scheme, which shall be used for the train wide addressing of onboard train functions.

On functional level, a more abstract addressing scheme based on TCN domain names (see 5.4.3) is defined which is equivalent to the domain name system defined for the Internet. The main reasons for introducing an additional addressing scheme on top of IP addresses are:

- Usable to address abstract functions instead of device related IP addresses. For example, the same function might be accessible via different IP addresses in different consists, but has the same name in all consists.
- IP addresses can be changed without impacting the domain name.
- Names can be better understood and interpreted by humans. This might become more important in future when IPv6 addresses are in use.

Functional addressing is related to logical functions.

Sources of information can be:

- a function within a vehicle or a consist

Destinations of information can be:

- a function within a vehicle or a consist
- a function within multiple or all consists

NOTE A function can be implemented by one or by several end devices in a consist.

5.4.2 TCN Domain Name System (TCN-DNS)

5.4.2.1 General

The TCN Domain Name System defines a hierarchical, domain based naming scheme and a distributed database system for implementing this naming scheme. Its primary use is to map host names to IP addresses.

NOTE The concept of TCN-DNS is strongly related to the Internet DNS as defined in RFC 1034 and RFC 1035.

5.4.2.2 TCN Domain Name Space and Zones

The TCN Domain name space is defined by a tree structure with the local train ('lTrn') being the (local) root. The next level underneath the root is defined by the closed train sub-domains (cltr01 … cltrNN). The closed train sub-domains are further structured into the consist sub-domains (cst01 … cstNN), the vehicle sub-domains (veh01 … vehNN) finally ending in the hosts, which are the devices hosting the logical functions (fctDevXX ... fctDevYY).

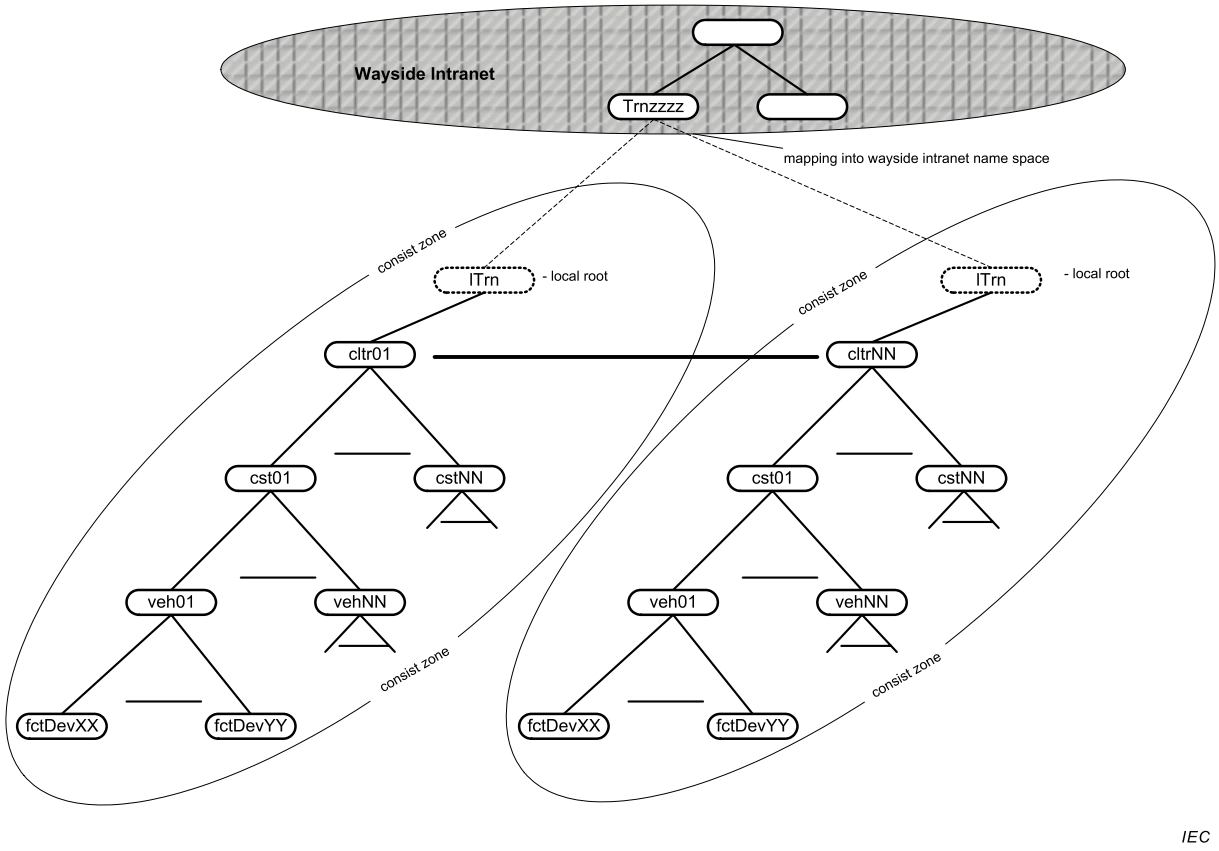


Figure 18 – TCN-DNS name space with division into zones

For onboard-to-ground communication, the whole TCN-DNS name space can be mapped into superior intranets owned for example by the operator or maintainer of the train (this however is not covered by this part of the standard). In this case, the mapped TCN-DNS name space may be identified by an operator's specific root name (e.g. 'Trnzzzz').

The TCN-DNS name space is divided into zones, where each zone is administrated by one authoritative DNS name server.

One possibility is to define consist sub-domains as one zone as it is shown in Figure 18, and one authoritative DNS name server is responsible for this zone. In consists with several consist networks it might be advantageous to subdivide the consist sub-domain in several zones, and to let the consist DNS name server delegate these zones to (subordinated) DNS name servers.

5.4.3 TCN Domain Names

Communicating entities (sources and destinations) shall be identified by a fully qualified TCN domain name which is derived from the TCN domain name space.

5.4.4 TCN-URI Scheme

5.4.4.1 General

The TCN domain name space syntax is defined by an URI scheme following the recommendations given in RFC 3986.

5.4.4.2 TCN URI schema syntax

The syntax of the TCN URI is defined in accordance to RFC 3986 using the Backus Naur Form (BNF) notation. The specific definition of the TCN URI is subject of the following subclauses.

The TCN URI uses some basic elements for syntax definition, which are defined as listed in Table 6.

Table 6 – TCN URI basic syntax

```
label = alpha 1{uchar}13 alphanum
uchar = alphanum | mark
alphanum = alpha | digit
alpha = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" |
        "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" |
        "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"
        "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" |
        "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |
        "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
mark  = "-"
```

NOTE This label definition deviates from RFC 1035 with respect to:

– Restriction to maximally 15 characters (RFC 1035: 63 characters allowed)

– First character cannot be a digit (RFC 1035 allows a digit).

5.4.4.3 TCN URI general schema

The TCN URI Identifier shall be used to identify the source function/device and the destination function/device of information. The general schema of the TCN URI is defined in Table 7.

Table 7 – General schema syntax

```
TCN-URI = [scheme ":" "//"] user "@" host
scheme  = "trn"
user    =  usr
host    = fctdev "." vehicle "." consist "." [cltrain "."]train
["."]
```

According to this definition, the TCN-URI Identifier can be written as (Figure 19):

Figure 19 – TCN-URI Schema

NOTE 1 The host part of the TCN-URI is resolvable to an IP address by the TCN-DNS service.

NOTE 2 The label 'fctdev' identifies the logical device which is implementing the addressed service function. "Logical" device means that it is not known how this device is implemented, e.g. it could be a single physical device implementing the function or the function could be distributed over several physical devices.

For addressing a physical device, the physical addressing scheme defined in IEC 61375-2-5 shall be used.

5.4.4.4 TCN URI Scope

Within a communication relationship, both the source and the destination of information are identified by TCN-URI. Whether a specific TCN URI is related to a source or to a destination is called its scope. TCN-URI syntax rules may be different if the TCN-URI is used for a source or for a destination.

5.4.4.5 TCN URI User part (optional)

The user part can be used for the application profiles defined in IEC 61375-2-4 to distinguish different functions and function instances, which are located on the same physical device.

This part of IEC 61375 defines a character string of 32 octets which is reserved for the user part.

5.4.4.6 TCN URI Host part

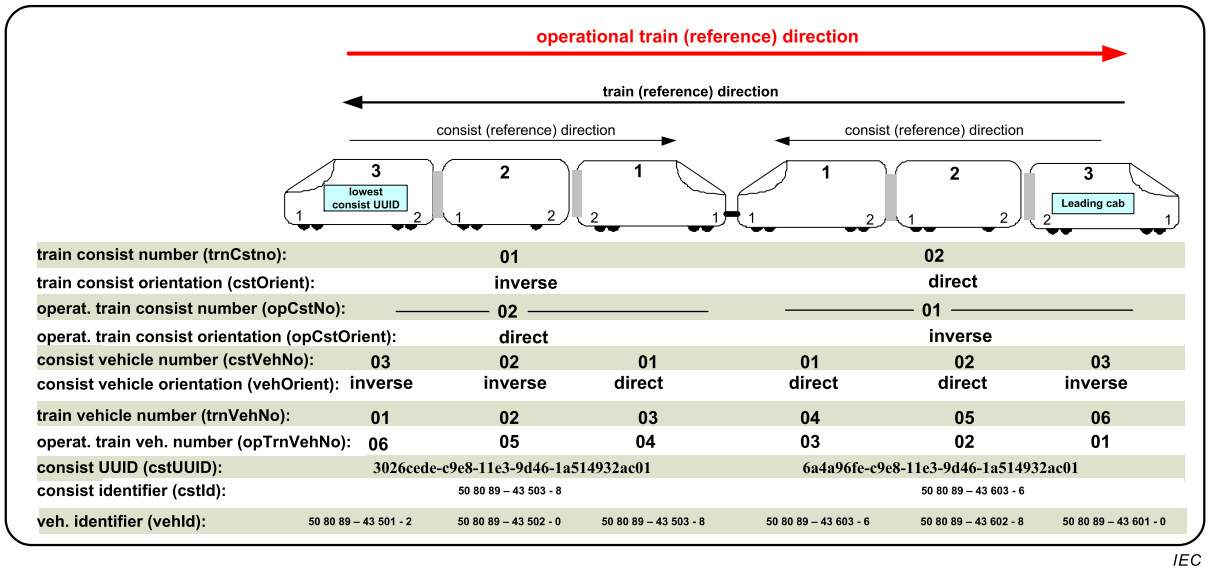
5.4.4.6.1 Directions, orientations and numbers in a train (informative)

Navigation in a train network requires a precise definition of the directions and numbering schemes in a train. Directions and numberings are defined in different parts of IEC 61375. Figure 20 intends to show a summary of those directions and numberings.

NOTE 1 The consist reference direction is based on the construction of the consist. Vehicle 1/cab 1 is always end 1 (front) of the consist. The consist reference direction is used for the vehicle numbering within the consist.

NOTE 2 The TCN reference direction is based on the UUID of the consists as defined in IEC 61375-2-5. The consist with the lowest UUID is always the TCN consist 1.

NOTE 3 The operational direction is based on the leading vehicle/leading consist. The end 1 (front end) of the train is defined depending on the the viewing direction of the driving cab of the leading vehicle (see 4.2.4.3). The operational direction can be used for the management of operational direction depending signals.



IEC

Figure 20 – Directions, orientations and numbers in train

5.4.4.6.2 Function/Device Label

Function devices shall be identified by labels following the syntax given in Table 8:

Table 8 – Device label syntax

```
fctDev = ”lDev” | deviceName | group
devicename = label
group = ”grpAll” | grpDev
grpDev = ”grp” 1{uchar}12
```

The meaning of the different device labels is defined in Table 9.

Table 9 – Device label definition

Label <Device>	Scope S=source D=destination	Manda-tory(M)/ Optiona(O)l	Definition
”lDev”	D	O	The destination function is located on the local ED. To be used for addressing functions residing on the own device.
deviceName	S,D	M	Defines the function device (logical device) . EXAMPLE ‘fctDoorCtrl’, ‘fctBrake, devHMI’. This will be resolved to an IP unicast address.
grpDev	D	M	Defines a group of function devices (logical device). EXAMPLE ‘grpDoorCtrl’, ‘grpBrake’, ‘grpETBN’, ‘grpECSC’. This will be resolved to an IP multicast address.
”grpAll”	D	M	All function devices (logical device) or the physical device. This will be resolved to an IP multicast address.

Label <Device>	Scope S=source D=destination	Manda-tory(M)/ Optiona(O)l	Definition
anyDev	S	O	The location of the source function is unknown. This can be used for TRDP subscription based on ComId and source consist or source closed train. Shall be used in conjunction with 'anyVeh' and will resolve to an IP address range covering all CNs in a consist or a closed train. EXAMPLE 1 TCN-URI 'anyDev.anyVeh.cst05.anyClTrn.lTrn' can be used to subscribe to a specific ComId in consist 05 EXAMPLE 2 TCN-URI 'anyDev.anyVeh.leadCcst.anyClTrn.lTrn' can be used to subscribe to a specific ComId in the leading consist

NOTE It is also possible to define alias names for devices, meaning that the same physical device appears with different device labels.

It is recommended to use defined prefixes for device names, e.g. 'fct' for a function device (logical device) and 'dev' for a physical device.

To support abstract functional addressing device labels or device group labels can be standardized. For instance, the function "brake" can be defined to be addressed via a device label "fctBrake", which is standardized for all TCN applications. It is however not in the scope of this standard to define this.

5.4.4.6.3 Vehicle Label

Vehicles shall be identified by labels following the syntax given in Table 10:

Table 10 – vehicle label syntax

```
veh = ″lVeh″ | cstVehNo | opVehNo | ″leadVeh″ | ″aVeh″ |
        idVeh | ″anyVeh″
cstVehNo = ″veh″ digit digit
opVehNo = ″opVeh″ digit digit
idVeh = label
```

The meaning of the different vehicle labels is defined in Table 11.

Table 11 – Veh (vehicle) label definition

Label <veh>	Scope S=source D=destination	Manda-tory(M)/ Optiona(O)l	Definition
lVeh	D	O	Function in the own vehicle.
aVeh	D	M	Functions located in some or all vehicles. The vehicle label "aVeh" can only be used in a combination with a group device label ("grpName"). End devices which are members of a device group can be spread over the Consist, or, for train wide groups, over the whole train.

Label <veh>	Scope S=source D=destination	Manda-tory(M)/ Optiona(O)l	Definition
opVehNo	S,D	M	Function in the vehicle with operational number x ϵ {01,…,63}. This label can only be used in conjunction with the label “anyCst”. The ‘opVehno’ is unique within the train and in reference to the leading vehicle (operational direction). EXAMPLE ‘opVeh24’
cstVehNo	S,D	M	Function in the consist vehicle with cstVehNo ϵ {01,…,32} EXAMPLE ‘veh05’
leadVeh	S,D	M	Function is in the leading vehicle. This label can only be used in conjunction with the label “leadCst”
idVeh	S,D	M	function in a vehicle with unique identifier (e.g. the 7/12 digit UIC vehicle number) EXAMPLE ”UIC948002343044” // vehicle with UIC identifier 94800234304-4
anyVeh	D	M	function located in one of the vehicles. This label can be used when the location of the device in a consist is irrelevant. In this case, the device label shall be unique in the consist.

5.4.4.6.4 Consist Label

Consists shall be identified by labels following the syntax given in Table 12:

Table 12 – Consist label syntax

```
consist = ”lCst” | ”aCst” | ”anyCst” | ”leadCst” | cstNo
          | opCstNo | idCst
cstNo = ”cst” digit digit
opCstNo = ”opCst” digit digit
idCst = label
```

The meaning of the different consist labels is defined in Table 13.

Table 13 – Consist label definition

Label <consist>	Scope S=source D=destination	Manda-tory(M)/ Optiona(O)l	Definition
lCst	D	O	Function located in the own (local) consist
cstNo	S,D	M	Functions located in the consist with trnCstNo ϵ {01,…,63}. EXAMPLE ‘cst05’ This is the consist sequence number obtained from the train directory which is not respecting corrections. For addressing functions in an operated train the label ‘opCstNo’ may be used instead which also respects correction.

Label <consist>	Scope S=source D=destination	Manda-tory(M)/ Optiona(O)l	Definition
opCstNo	S,D	O	Functions located in consist with operational number opCstNo x ϵ {01,…,63}. This label can only be used in conjunction with the label “anyVeh”. This label is unique within the train and in reference to the leading Consist (operational direction). EXAMPLE ‘opCst29’
aCst	D	M	functions located in some or all Consists. Can only be used in combination with a device group label.
anyCst	D	M	Function located in one of the Consists. This label can be used when the location of the device in the train is irrelevant, in which case, the device label shall be unique in the train, or this label shall be used if an operational vehicle is addressed. This label can only be used in combination with label “anyVeh” or in combination with label opVehNo.
idCst	S,D	M	unique identifier of a Consist EXAMPLE ‘UIC948002343044’ // Consist identified by its consist vehicle with UIC identifier 94 80 0234 304-4
leadCst	S,D	M	destination function is in the consist which contains the leading vehicle

5.4.4.6.5 Closed train Label (Optional)

Closed trains shall be identified by labels following the syntax given in Table 16:

Table 14 – Closed train label syntax

```
cltrain = ”lClTrn” | ”anyClTrn” | ”aClTrn”
```

The meaning of the different closed train labels is defined in Table 17.

Table 15 – Closed train label definition

Label <cltrain>	Scope S=source D=destination	Manda-tory(M)/ Optiona(O)l	Definition
lClTrn	S,D	O	destination/source function in the own closed train
anyClTrn	D	M	function located in one of the closed trains. This label may also be used if there is no closed train.
aClTrn	D	M	function located in all closed trains. This label may also be used if there is no closed train.

5.4.4.6.6 Train Label

Trains shall be identified by labels following the syntax given in Table 16:

Table 16 – Train label syntax

```
train = ”lTrn”
```

The meaning of the different train labels is defined in Table 17.

Table 17 – Train label definition

Label <train>	Scope S=source D=destination	Manda-tory(M)/ Optiona(O)l	Definition
lTrn	S,D	M	destination/source function in the own train

NOTE The URI scheme described herein is only defined for onboard train addressing, but is can be easily extended (and is prepared) for train-to-wayside and inter-train addressing. An extension can for instance look like:

trdp://instance.function@fctdev.vehicle.consist.cltrain.train.fleet.operator

5.4.5 Mapping TCN-URI to IP address

5.4.5.1 General

TCN-URI shall be mapped to IP addresses in accordance to the IP addressing scheme defined in IEC 61375-2-5.

The resolution of URIs to IP addresses is in general managed by the TCN-DNS service as specified in 5.4.2.

5.4.5.2 Mapping to IP MC group addresses

IEC 61375-2-5 assigns the IP address range 239.192.0.0/14 for train level IP MC group addressing. To use this address range in conjunction with the defined TCN-URI scheme, especially for supporting groups, which are limited to one consist, it is necessary to define a decomposition of the assigned IP MC group address range.

The general decomposition is defined as:

11101111.110000rr.zzzzzzzz.zzzzzzzz

with fields described in Table 18.

Table 18 – General decomposition of IP MC groups addresses

Subnet number part:	
[r]	Defines the range. ‘00’B = all-train groups ‘01’B = ETB-related groups ‘10’B = consist-limited groups ‘11’B = open for other use
[z]	For further decomposition

All-train groups are groups with group members related to different ETBs[1] in the train.

The decomposition for all-train groups is defined as:

```
11101111.11000000.gggggggg.gggggggg
```

with fields described in Table 19.

Table 19 – Decomposition of all-train groups

Host number part:	
[g]	All train group number (value range: 0 .. 65535) 0 .. 255 = as specified in IEC 61375-2-5 256 .. 65534 = usable for user defined all-train groups 65535 = reserved for future use

EXAMPLE An IP MC group containing EDs from the operational network and the multimedia network is an all-train group.

An all-train group address shall be assigned to a function if FUNCTION_INFO parameter ‘etbId’ is set to 255 (see 5.3.3.2.4).

NOTE 1 Usage of TRDP for communication to all-train groups is problematic as all-train groups may contain EDs related to different ETBs, and etbTopoCnt values are different for the ETBs. The etbTopoCnt is contained in the TRDP header and used by destination EDs to verify the correctness of the received message, which will fail if an etbTopoCnt value related to another ETB is contained. Using a gateway in between ETB altering the etbTopoCnt value might be a solution, but is complex.

ETB-related groups are groups with group members related to one ETB.

The decomposition for ETB-related groups is defined as:

```
11101111.11000000.bbgggggg.gggggggg
```

with fields described in Table 20.

[1] For the definition of ‘… related to ETB’ see 4.2.2.6.2.

Table 20 – Decomposition of ETB-related groups

Subnet number part:	
[b]	ETB Identifier (etbId) value = 0..3
Host number part:	
[g]	Group number (value range: 0 .. 16383) 0 = all ED 1 = all ECSP (ETB0 only) 2..16382 = usable for user defined ETB-related groups 16383 = reserved for future use

EXAMPLE An IP MC group containing EDs from the operational network only is a ETB-related group. An IP MC group containing EDs from the multimedia network is another ETB-related group.

Consist-limited groups are ETB-related groups where all group members belong to one consist.

The decomposition for consist-limited groups is defined as:

11101111.11000001.bbcccccc.gggggggg

with fields described in Table 21.

Table 21 – Decomposition of consist-limited groups

Subnet number part:	
[b]	ETB Identifier (etbId) value = 0..3
[c]	trnCstNo (value range: 1 .. 63) 0 = local consist
Host number part:	
[g]	Group number (value range: 0 .. 254) 0 = all ED in all CN of the consist connected to the ETB 1 = ECSP within consist (ETB0 only) 2 .. 254 = usable for user defined consist-limited groups 255 = reserved for future use

Group identifiers need to be unique inside a consist. The same 8-bit group identifier value range is used for both vehicle-limited groups and consist-limited groups, so it must be shared. For example, the value of grpHMI is different for this group defined on consist level (“grpHMI.aVeh.lVeh”) and for this group defined on vehicle level (e.g. “grpHMI.veh02.lCst”).

Notice that “all ED” is coded by setting all grp-id bits to 0, and not to 1. This convention is used in order to avoid confusion, because grp-id bits set to 1 could also mean “all groups”.

5.4.5.3 Joining IP multicast groups

For joining IP MC groups some aspects should be considered.

a) All-train and ETB-related groups have defined IP MC addresses which will not change when train composition changes, so it is sufficient for EDs to join the group when the system is booting.

b) Consist-limited groups are dynamic IP Multicast groups possessing IP multicast group addresses which may change after each train inauguration. Due to their dynamic nature, all ED joining such a group need to leave the group with a new train inauguration, and join again with the new IP MC address. Another, recommended, solution is to translate IP MC

addresses while routing IP telegrams from ETB to ECN (ETBN). During this network address translation (NAT, see RFC2663, RFC2766, RFC3022) the [c] bits representing the train consist number within the IP MC destination address can be substituted with 0 (local consist). In that case it is sufficient for EDs to join the group when the system is booting.

c) EDs, which should be capable to receive TRDP telegrams addressed to both consist-limited as well as ETB-realted or all-train groups, should join both IP MC groups.

EXAMPLE The function 'fctDoor' may be both train wide (ETB0-related) addressable (e.g. TCN-URI 'fctDoor.anyVeh.aCst.aCltrn.lTrn') as well as consist-limited (e.g. TCN-URI 'fctDoor.anyVeh.cst01.anyClTrn.lTrn'). If we assume that NAT is used and that the fctId=50 for the fctDoor, the following two IP MC addresses need to be joined: 239.193.0.50 and 239.194.1.50.

5.4.5.4 Well known TCN-URI

Some TCN-URI mappings to IP addresses are predefined and do not require an explicit resolving of the URI via TCN-DNS. Those TCN-URI are listed in Table 22.

Table 22 – Well-known TCN-URI

TCN-URI (host part)	Scope	IP address	Description
grpAll.aVeh.aCst.aClTrn.lTrn	D	239.193.0.0	Broadcast to all end devices related to ETB0.
grpAll.aVeh.lCst.lClTrn.lTrn	D	239.194.0.0	Broadcast to all end devices related to ETB0 within the local consist
grpAll.aVeh.lCst.lClTrn.lTrn	D	239.255.0.0	Alternative broadcast to all end devices within the local consist if all ED are connected to the same CN (only one CN connected to the ETB) NOTE this address is defined in IEC 61375-3-4
lDev.lVeh.lCst.lClTrn.lTrn	S,D	127.0.0.1	Own device (local loop-back)
grpECSP.anyVeh.aCst.aClTrn.lTrn	D	239.193.0.1	Broadcast to all ECSP

EXAMPLE

The following example gives some hints for the implementation of grpAll.aVeh.lCst.lClTrn.lTrn in a consist with multiple CN.

Every ED in a consist subscribes to 239.194.0.0.

ETBNs use ETB inauguration result to add a rule to their MC routing table for address 239.194.X.0 where X represents their own consist (X = trnCstNo + (etbId × 64)). All other addresses from range 239.194.X.0 are not routed to ECN.

ETBNs translate MC address 239.194.X.0 to 239.194.0.0 when routing it from ETB to ECN.

In case of multiple CNs in one consist: packet with destination address 239.194.0.0 has to be handled by ETBNs in this consist as well in order to deliver it to all CNs. One solution is to route it from source CN to ETB and translate the destination address to 239.194.X.0. Another solution is that the ETBN in the source CN tunnels the packet to other ETBNs in the consist using IP unicast, and the destination ETBNs then retransmit the packet as multicast into their CN. The first approach is easier to implement but the packet is sent to all ETBNs in the train. Tunneling approach requires more support from ETBNs, on the other hand the packet never leaves consist boundary.

5.4.5.5 IP routing in networks with multiple ETBs

Basic rules for dynamic IP routing management are defined in IEC 61375-2-5.

The following additional rule shall be observed:

For transmitting IP packets to a destination ED, which is related to another ETB than the ETB the source ED is related to, the IP packet shall be transferred over the ETB the destination ED is related to.

NOTE 1 For a definition of 'ED related to ETB' see 4.2.2.6.2.

NOTE 2 The reason for this rule is that a sending ED cannot expect that an IP packet can be routed between the ETBs within the destination consist. It therefore needs to be routed within the sender's consist to the ETB the destination ED is related to.

EXAMPLE This example uses the train architecture shown in Figure 21. If the source ED in consist 'B' wants to send to the ED with fctDevNo = 10020 in consist 'E', it has to route the packet first to its local multimedia ECN and then via the ETB1 to consist 'E'.

5.4.6 Support of other URI schemas

Besides the defined TCN-URI schema, which is mainly targeting the functional addressing inside trains, also other URI schemas like the commonly used uniform resource locators (URL) with the syntax[2]

```
scheme://domain:port/path?.....″
```

may be used. The 'domain' corresponds to the TCN-URI host part and may be resolved by the TCN-DNS service. This way, other URI schemas than TCN-URI can be supported, which might be requested for specific OMTS applications.

EXAMPLES

HTTP request: http://devECSP.anyVeh.ICst.ITrn:80

Mail service: mailto:bob@devECSP.anyVeh.ICst.ITrn

5.5 TCN-DNS Server

5.5.1 General

TCN-URIs host part are resolved to IP addresses with the aid of the TCN-DNS as defined in 5.4.2. The interaction between DNS clients (the "resolver") and the local DNS server is specified in RFC 1034 and RFC 1035.

A more efficient interface to the local DNS server especially suited to resolve multiple TCN-URIs is described in Annex E.

5.5.2 Architecture

Each consist shall provide at least one consist DNS name server which is authoritative for this consist domain.

NOTE The consist domain may be subdivided into several zones where each zone has its own DNS name server. A special case is a consist with separated operational and multimedia network, in which case two dedicated DNS name servers, one for the operational network and one for the multimedia network, can be provided.

5.5.3 Functional address resolution

The consist DNS server shall be able to resolve TCN-URI host parts as defined in 5.4.4 to the corresponding IP address as defined in IEC 61375-2-5.

EXAMPLES

[2] A URL is technically a type of uniform resource identifier (URI) as defined in RFC3986.

Given a train with two train backbones, ETB0 (operational network ‘OP’) and ETB1 (multimedia network ‘MM’), composed of three consists, as shown in Figure 21. TCN reference directions are equal for ETB0 and ETB1. Due to the position of the leading cab the operational consist numbers are opposite to the TCN consist numbers of ETB0. The consist with UUID=’E’ has two ECN connected to ETB0 and one ECN connected to ETB1. The other consists have only one ECN for each ETB. Furthermore, it is assumed that there are two DNS name servers, one for the operational network and one for the multimedia network, and that end devices connected to either of the two networks use the related DNS name server for address resolution. There exists a TTDB for each ETB with partly shared information follwing the rules defined in 5.3.4.

The task will be for a source ED located in consist with UUID = ‘B’and connected to the operational network to resolve the TCN-URIs of the two shown destination end devices:

- ED doorCTRL@fdDoor.veh02.cst02.anyClTrn.lTrn in consist with UUID = ‘M’
- ED publicAnnounce@fdPubAddr.anyVeh.leadCst. anyClTrn.lTrn in consist with UUID = ‘E’ (leading vehicle)

and in addition, to resolve the group addresses of the door controller group (only shown for consist ‘E’ in Figure 21), as a train wide group and as a consist-limited group:

- group doorCTRL@fctDoor.anyVeh.aCst.aClTrn.lTrn in the train
- group doorCTRL@fctDoor.anyVeh.cst01.anyClTrn.lTrn in consist ‘E’

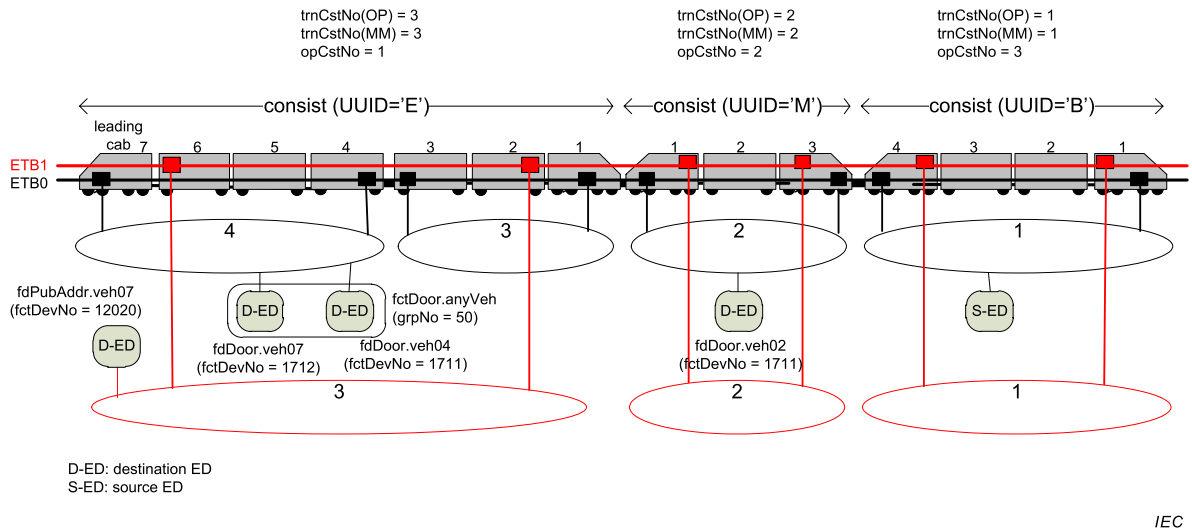


Figure 21 – TCN-URI resolving in a train

EXAMPLE 1

For resolving the TCN-URI ‘doorCTRL@fdDoor.veh02.cst02.anyClTrn.lTrn” the ETB0 DNS server has to execute the steps as listed in Table 23.

Table 23 – TCN-URI resolving – Example 1

Step	Action	Result (example)
1	Isolate the TCN-URI host part and strip off parts not needed	= “fdDoor.veh02.cst02”
2	Lookup ‘cstUUID’ of label ‘cstNo’ = 2 in the train directory of the local TTDB	= ”M”
3	Search for ‘cstUUID’ in the ConsistInfoList	Located CONSIST_INFO record for consist ”M”
4	Search for [‘fctName’ = “fdDoor”] in conjunction with [cstVehNo = 02] in the function info array of the located CONSIST_INFO record and retrieve the value of parameters ‘fctId’, ‘etbNo’ and ‘cnNo’.	fctId = 1711 etbNo = 0 cnNo = 1
5	The value of ‘subnet id’ can be derived from the train network directory associated to the ETB0	subnet id = 2
6	Compute the IP address from the parameter values ‘etbNo’,	IP = 10.128.134.175

Step	Action	Result (example)
	‘subnet id’ and ‘fctId’ in accordance to IEC 61375-2-5	

EXAMPLE 2

For resolving the TCN-URI ‘publicAnnounce@fdPubAddr.anyVeh.leadCst.lTrn” the ETB0 DNS server has to execute the steps as listed in Table 24:

Table 24 – TCN-URI resolving – Example 2

Step	Action	Result (example)
1	Isolate the TCN-URI host part and strip off parts not needed	= “fdPubAddr.anyVeh.leadCst”
2	Lookup the leading vehicle in the operational train directory and retrieve the values of parameters ‘opVehNo’ and ‘ownOpCstNo’.	opVehNo = 1 ownOpCstNo = 1
3	Lookup ‘cstUUID’ of ‘opCstNo’ = 1 in the operational train directory	= ”E”
4	Determine the consist orientation with respect to the operational train orientation by reading parameter ‘opCstOrient’ in the operational train directory	= inverse
5	Search for ‘cstUUID’ in the ConsistInfoList	Located CONSIST_INFO record for consist ”E”
6	Determine the number of vehicles within that consist (parameter ‘vehCnt’) and compute the ‘cstVehNo’ of the leading vehicle.	Number of vehicles = 7. As the consist is inverse to the operational train direction, the value of ‘cstVehNo’ of the leading vehicle with opVehNo=1 is cstVehNo = 7
7	Search for [‘fctdevName’ = “fdPubAddr” AND ‘cstVehNo = 7] in the function info array of the located CONSIST_INFO record and retrieve the value of parameters ‘fctId’, ‘etbNo’ and ‘cnNo’. NOTE The function fdPubAddr is associated to the multimedia network. To make it visible in the ETB0 TTDB this function is listed both in the CONSIST_INFO record of the ETB0 TTDB and in the CONSIST_INFO record of the ETB1 TTDB, see 5.3.4.4.	fctId = 12020 etbNo = 1 cnNo = 1
8	The value of ‘subnet id’ can be derived from the train network directory associated to the ETB1	subnet id = 3
9	Compute the IP address from the parameter values ‘etbNo’, ‘subnet id’ and ‘fctId’ in accordance to IEC 61375-2-5	IP = 10.160.238.244

EXAMPLE 3

The following example shows how a TCN-URI can be resolved to a train wide multicast group address. Here it is assumed that the IP MC group ‘fctDoor’ has been defined as a train wide group with a group Id = 270. For resolving the TCN-URI ‘doorCTRL@fctDoor.anyVeh.aCst.aCITrn.lTrn” the ETB0 DNS server has to execute the steps as listed in Table 25.

Note that train wide multicast group addresses must be unique train wide.

Table 25 – TCN-URI resolving – Example 3

Step	Action	Result (example)
1	Isolate the TCN-URI host part and strip off parts not needed	= “fctDoor”
2	Search for the first match of [‘fctName’ = “fctDoor”] in the consist info list by iterating through the CONSIST_INFO records and retrieve the value of parameters ‘fctId’.	fctId = 50
3	Compute the IP address from the parameter value ‘fctId’ in accordance to the scheme defined above	IP = 239.193.0.50

EXAMPLE 4

The following example shows how a TCN-URI can be resolved to a consist-limited multicast group address. Here it is assumed that the two door controllers in consist E form a multicast group with group Id = 5. For resolving the TCN-URI ‘doorCTRL@fctDoor.anyVeh.cst01.anyClTrn.lTrn” the ETB0 DNS server has to execute the steps as listed in Table 26:

Table 26 – TCN-URI resolving – Example 4

Step	Action	Result (example)
1	Isolate the TCN-URI host part and strip off parts not needed	= “fctDoor.anyVeh.cst01”
2	Lookup ‘cstUUID’ of label ‘cstNo’ = 1 in the train directory of the local TTDB	= ”E”
3	Search for ‘cstUUID’ in the ConsistInfoList	Located CONSIST_INFO record for consist ”E”
4	Search for [‘fctName’ “fctDoor”] in the function info array of the located CONSIST_INFO record and retrieve the value of parameter ‘fctId’.	fctId = 50
5	Compute the IP address from the parameter values ‘trnCstNo’ and ‘fctId’ in accordance to the scheme defined above	IP = 239.194.1.50

5.5.4 Protocol

The consist DNS name server shall implement the DNS protocols as specified in RFC 1034 and RFC 1035.

The consist DNS name server shall accept recursive client queries as shown in the example of Figure 22.

TCN-URIs addressing destinations within the local or remote consists shall be resolved within a time of 1,0 s.

NOTE The time is measured from the query reception until sending the answer to the client is completed.

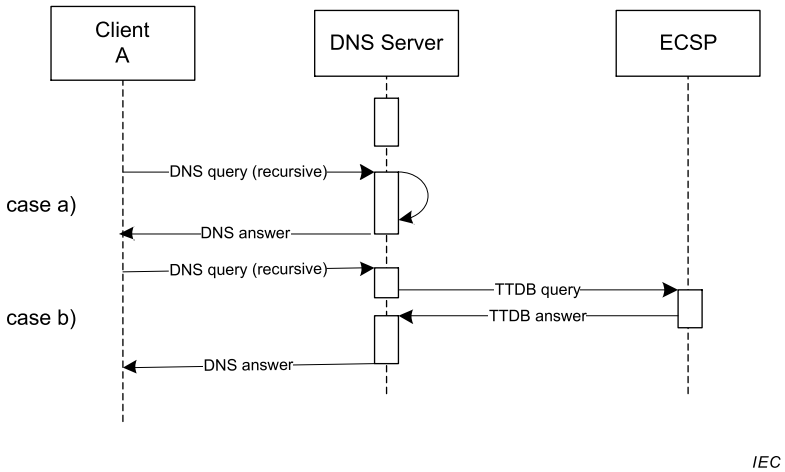


Figure 22 – DNS protocol (case a without, case b with TTDB interrogation)

5.5.5 Multiple ETBs

The mapping of TCN-URI to IP addresses is ETB specific for inter-consist communication as it depends on the train network directory of the specific ETB. Therefore, a TCN-DNS server shall be provided for each ETB.

A function shall use the TCN-DNS server for TCN-URI address resolution, which is related to the ETB the functions wants to communicate.

NOTE Addressing a function in a different consist may result in different IP addresses depending on the ETB which is used for communication.

Take care that the return path of a inter-consist communication will use the same ETB if for returned telegrams the IP address of the original sender is used as destination address.

5.6 Data exchange

5.6.1 General

This subclause defines the data exchange between functions and devices over ETB. Dependant on the involved functions, different communication protocols might be applied. In this standard, only communication protocols are defined which are used by the services specified herein.

5.6.2 Operational network communication

All services of the operational network shall use the TRDP protocol as specified in Annex A (with supplementary information in Annex C and Annex D) for the transfer of process data and message data between operational network functions.

For the transfer of streaming data (audio/video), appropriate streaming data protocols may be used, as for instance (list not exhaustive):

RTSP	Real Time Streaming Protocol	RFC 2326
RTCP	Real Time Control Protocol	RFC 3605
RTP	Real Time Transport Protocol	RFC 3550
SRTP	Secure Real Time Transport Protocol	RFC 3261
SIP	Session Initiation Protocol	RFC 3711
TLS	Transport Layer Security Protocol	RFC 5246

For non-operational services like file transfer or service access, appropriate data protocols may be used, as for instance (list not exhaustive):

FTP	File Transfer Protocol	RFC 956
SFTP	SSH (Secure) File Transfer Protocol	RFC diverse
SNMP	Simple Network Management Protocol	RFC diverse
SSH	Secure Shell	RFC diverse
SCP	Secure Copy Protocol	-

5.6.3 OMTS network communication

In principle, the same protocols as used within the operational network can also be used for the OMTS network. Due to the special nature of the OMTS network as a platform for multimedia and telematics services, additional protocols may be used, as for instance (list not exhaustive):

HTTP	Hypertext Transfer Protocol	RFC 2616
SOAP	Simple Object Access Protocol	World Wide Web Consortium (W3C)

5.6.4 Quality of Service (QoS)

IEC 61375-1 defines a set of data classes, which shall be supported by ETB, and IEC 61375-2-5 defines a minimum of 4 priorities for the data transfer over ETB. The assignment of priorities to data classes shall be made, if not otherwise defined, in accordance to Table 27.

Table 27 – Data class priorities

Data Class	Priority	Remarks
Process Data	3	
Message Data	0 .. 1	
Stream Data	1 .. 2	The choice of priorities may depend on the type of the stream, e.g. an audio stream might require a higher priority than a video stream
Best Effort Data	0	
Supervisory Data	3	This data class is only defined in IEC 61375-2-5
3 = highest priority, 0 = lowest priority		

5.7 Service discovery

Each consist shall announce the functions it supports within its CSTINFO telegram. All supported functions can then be retrieved from the TTDB.

5.8 Train Info Service

The trainInfo service defines a consist internal service which allows to retrieve data from the TTDB. It is not within the scope of this standard to define consist internal services, but a useful set of function primitives is given in Annex E for information.

6 Services of the communication profile – ETB Control Service

6.1 General

This subclause specifies the service functions which are basically needed for enabling user data transfer over ETB. These functions are:

- Function Leading
- Function Sleep Mode
- Function Confirmation/Correction

In case of multiple ETBs, e.g. one operational network and one multimedia network, those functions shall only be activated on the operational network.

NOTE This restriction avoids contradicting comma.

6.2 Communication model

ETB control functions follow the client-server model as depicted in Figure 23. The ETB control service client (ECSC) is requesting some function primitive from the ETB control service provider (ECSP), and the ECSP is responding the function execution result.

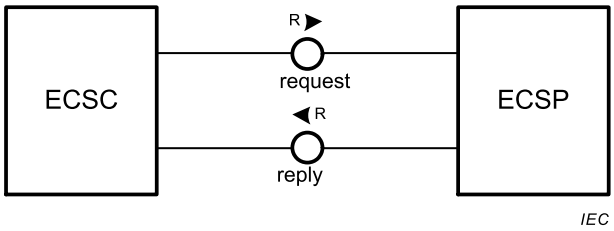


Figure 23 – ETB control service model

Only one ECSP shall be active at one time within a consist.

It is recommended to implement multiple ECSP, e.g. combined with anl ETBN, within a consist and to select one of the ECSP within the consist to become the active ECSP.

This part of the standard makes no assumption about the client of this service. For the specification of the service, an abstract client (ECSC) is modeled which is invoking the service functions and also retrieving the function results.

In general it is recommended to have only one device within each consist dedicated as ETB control service client in order to avoid conflicting server operations, for example one device requesting a certain correction of the train composition and another device requesting another correction.

6.3 ECSP Supervision

The ECSP shall supervise the availability of the ECSC.

A failure of the ECSC shall be detected latest after a time of T_{ECSC_fail} = 5,0 s. If a failure is detected, the ECSP shall react as defined for the individual ETB control service functions.

NOTE Annex E defines an interface between ECSC and ECSP for the services specified herein.

6.4 ECSP Interconnection

6.4.1 General

All ETB control functions require data exchange between all the ECSPs within a train. These are all the ECSPs which are related to a consist listed in the train directory with a valid cstTopoCnt.

For data exchange between the ECSPs, each ECSP sends cyclically a ETBCTRL telegram to all ECSPs within the train as shown in Figure 24. The data exchange can be ‘regular’ for non-safety relevant application or ‘safe’ for safety related applications.

In case of multiple ETBs, e.g. one operational network and one multimedia network, ETBCTRL telegrams shall only be exchanged on the operational network (ETB0).

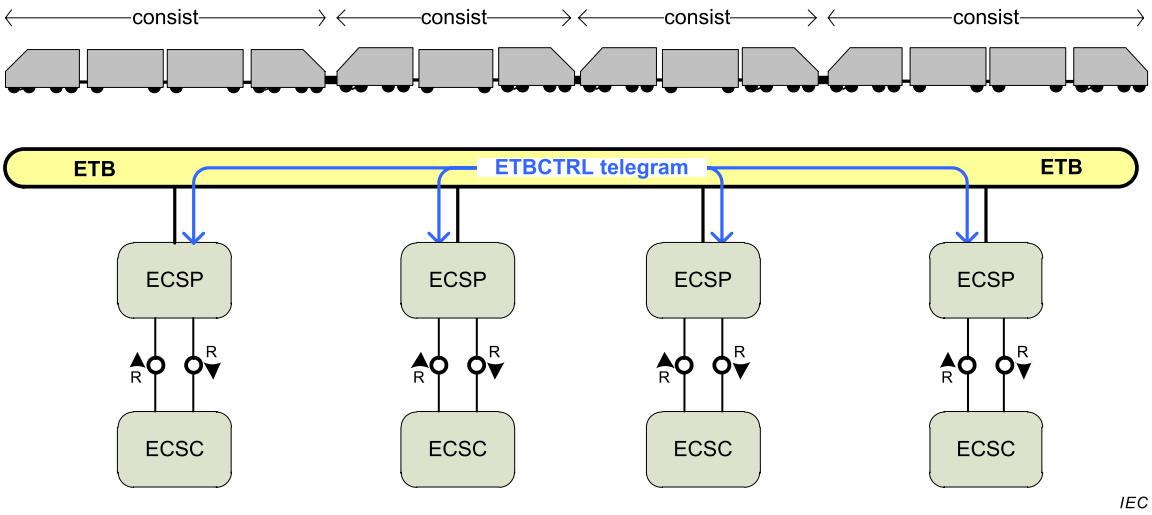


Figure 24 – ETBCTRL telegram exchange

6.4.2 ETBCTRL telegram exchange selection

An ECSP shall support either the regular ETBCTRL telegram transmission or the safe ETBCTRL telegram transmission using SDTv2 as defined in Annex B.

An ECSP implementing the safe ETBCTRL telegram transmission may also process received regular ETBCTRL telegrams.

6.4.3 ETBCTRL telegram transmission

TRDP PD telegram parameters:

Message type:	‘Pd’
ComId:	1
SMI:	1 (only needed if SDTv2 is used)
UserDataVersion:	‘0100’H (only needed if SDTv2 is used)
DestinationURI (host part):	grpECSP.anyVeh.aCst.aClTrn.lTrn
Dataset:	ETBCTRL_TELEGRAM
Cycle time:	(0,5 ± 0,1) s
Timeout:	3,0 s

6.4.4 Structure of the ETBCTRL telegram

The structure of the ETBCTRL telegram is shown in Figure 25.

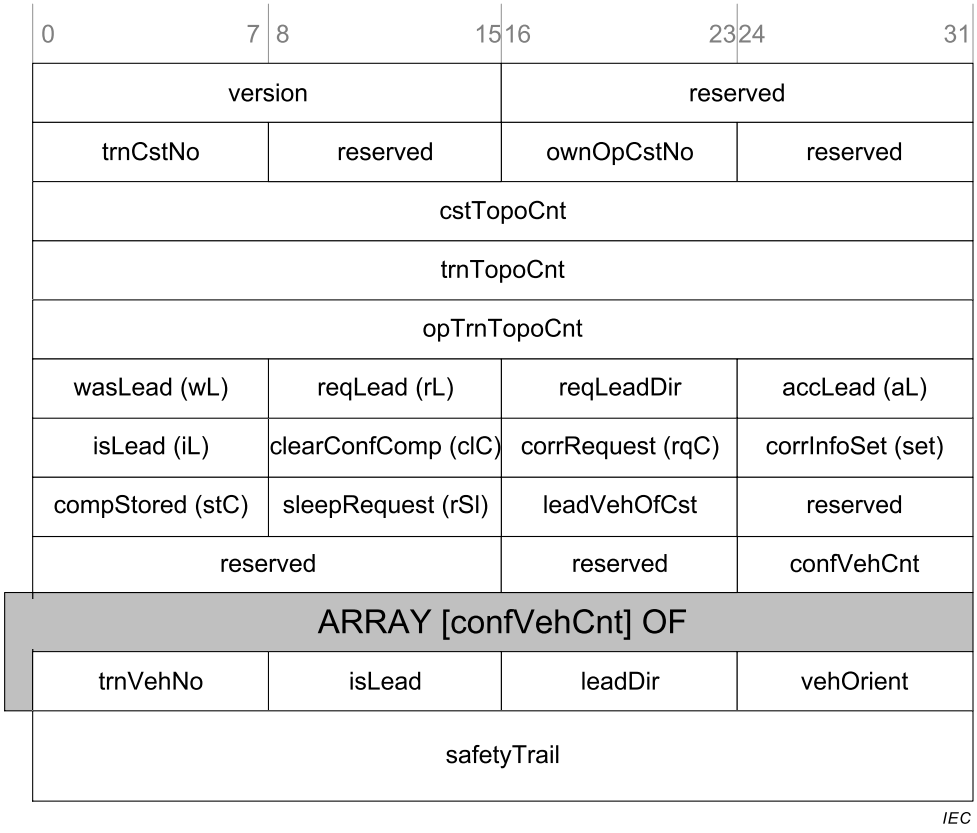


Figure 25 – ETBCTRL telegram

```
CONFIRMED_VEHICLE_::= RECORD
  {
  trnVehNo     UINT8          -- vehicle sequence number within the train
                                 with vehicle 01 being the first vehicle
                                 in ETB reference direction 1 as defined in
                                 IEC 61375-2-5
                                 value range: 0..63
                                 a value of 0 indicates that this vehicle
                                 has been inserted by correction
  isLead       ANTIVALENT8    -- vehicle is leading, informal
                                 may be used for plausibility checks
  leadDir      UINT8          -- vehicle leading direction, informal
                                 0 = not relevant
                                 1 = leading direction 1
                                 2 = leading direction 2
                                 may be used for plausibility checks
  vehOrient    UINT8          -- vehicle orientation, informal
                                 0 = not known (corrected vehicle)
                                 1 = same as operational train direction
                                 2 = inverse to operational train
                                 direction
                                 may be used for plausibility checks
  }
```

Informal parameters in the telegram may be used for a comparison between the own vehicle list and the indicated vehicle list as a plausibility check in addition to the opTopoCnt value check.

```
ETBCTRL_VDP::= RECORD
  {
  reserved01       UINT32    -- reserved, shall be set to 0
  reserved02       UINT16    -- reserved, shall be set to 0
  userDataVersion  VERSION   -- version of the vital ETBCTRL telegram
                             mainVersion = 1
                             subVersion = 0
  safeSequCount    UINT32    -- safe sequence counter, as defined in B.9
  safetyCode       UINT32    -- checksum, as defined in B.9
  }

ETBCTRL_TELEGRAM::= RECORD
  {
  version          VERSION       -- data structure version
                                 parameter 'mainVersion'
                                 shall be set to 1.
  reserved01       UINT16        -- reserved (=0)
  trnCstNo         UINT8         -- own train consist number
                                 value range: 1..63
  reserved02       UINT8         -- rsv: reserved (=0)
  ownOpCstNo       UINT8         -- own operational number
                                 value range: 1..63
                                 0 = unknown (e.g. after inauguration)
  reserved03       UINT8         -- rsv: reserved (=0)
  cstTopoCnt       UINT32        -- consist topography counter
  trnTopoCnt       UINT32        -- train directory topography counter
  opTrnTopoCnt     UINT32        -- operational Train directory topography
                                 -- counter
  wasLead          ANTIVALENT8   -- wL: consist was leading control flag
                                 '01'B = FALSE
                                 '10'B = TRUE
  reqLead          ANTIVALENT8   -- rL: leading request control flag
                                 '01'B = FALSE
                                 '10'B = TRUE
  reqLeadDir       UINT8         -- rLD: (request) leading direction
                                 0 = not relevant
                                 1 = consist direction 1
                                 2 = consist direction 2
  accLead          ANTIVALENT8   -- aL: accept remote leading request
                                 control flag
                                 '01'B = FALSE, not accepted
                                 '10'B = TRUE, accepted
  isLead           ANTIVALENT8   -- iL: consist contains leading vehicle
                                 control flag
                                 '01'B = FALSE
                                 '10'B = TRUE
  clearConfComp    ANTIVALENT8   -- clC: clear confirmed composition
                                 control flag
                                 '01'B = 0:FALSE
                                 '10'B = 1:TRUE
  corrRequest      ANTIVALENT8   -- rqC: request confirmation control flag
                                 '01'B = 0:FALSE
                                 '10'B = 1:TRUE
  corrInfoSet      ANTIVALENT8   -- set: correction info set control flag
                                 '01'B = 0:FALSE
                                 '10'B = 1:TRUE
  compStored       ANTIVALENT8   -- stC: corrected composition stored
```

```
                                    control flag
                                    '01'B = 0:FALSE
                                    '10'B = 1:TRUE
  sleepRequest      ANTIVALENT8   -- rSl: request sleep mode control flag
                                    '01'B = FALSE
                                    '10'B = TRUE
  leadVehOfCst      UINT8         -- position of leading vehicle in consist
                                    value range 0..32
                                    0 = not defined
                                    (1 = first vehicle in consist in
                                    Direction 1, 2 = second vehicle, etc.)
  reserved04        UINT8         -- rsv: reserved (=0)
  reserved05        UINT16        -- rsv: reserved (=0)
  reserved06        UINT8         -- rsv: reserved (=0)
  confVehCnt        UINT8         -- number of confirmed vehicles in train
                                    value range: 0..63,
                                    0 = no correction requested
  confVehList       ARRAY [confVehCnt] OF CONFIRMED_VEHICLE
                                  -- dynamic ordered list
                                    of confirmed vehicles in train,
                                    starting with the vehicle at
                                    train extremity 1 (see 4.2.4.2)
  safetyTrail       ETBCTRL_VDP   -- ETB-VDP trailer
                                    SafeTopoCount = 0
                                    completely set to 0 = SDTv2 not used
  }
```

NOTE The parameter 'reqLeadDir' is only meaningful if one of the parameters 'reqLead', 'wasLead' or 'isLead' is set to TRUE. This parameter is relevant for operational train direction determination.

6.4.5 Operational train directory computation process

6.4.5.1 Overview

The ETB control service functions "leading" and "confirmation/correction" determine the content of the operational train directory. The computation of the operational train directory is a function on each ECSP connected to ETB0, which is responsible to create and to maintain the operational train directory as defined in 5.3.3.

The principal operation of the operational train directory computation is shown in Figure 26.

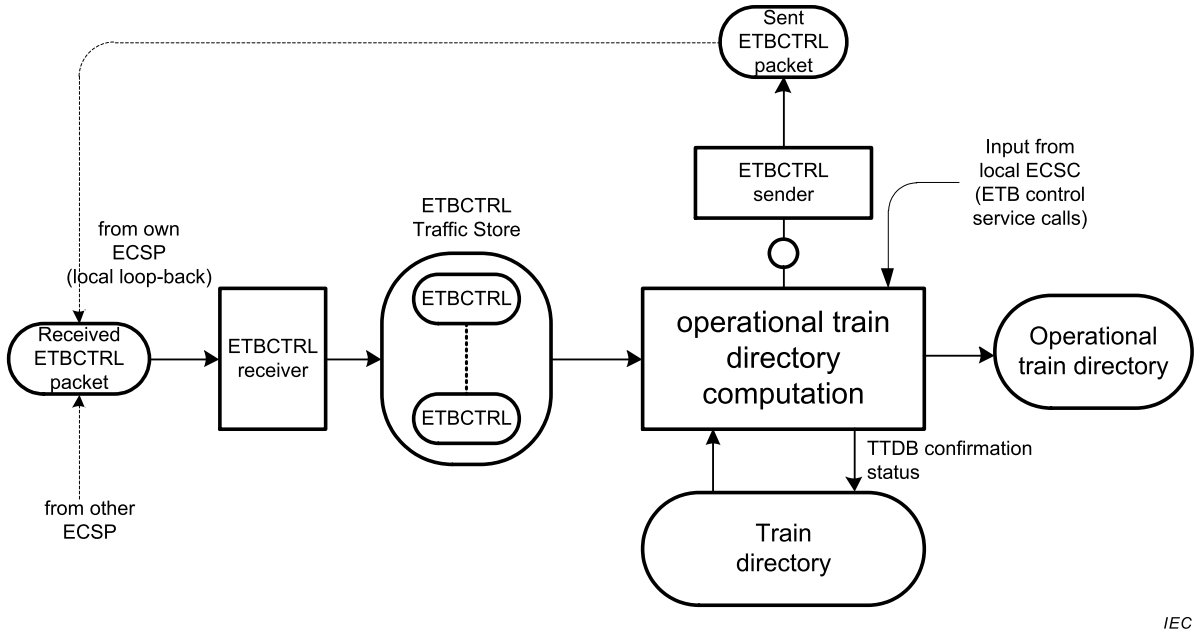


Figure 26 – Operational train directory computation block diagram

Input to the operational train directory computation is the collection of ETBCTRL telegrams received from all the ECSPs. This includes the own (local) ECSP as well as the remote ECSPs (= all the others except the own ECSP).

Those ETBCTRL telegrams might for instance be stored to an ETBCTRL traffic store for further processing. The operational train directory computation checks the content of the ETBCTRL traffic store for changes with respect to leading vehicle information, correction information and trnTopoCnt.

If a change is detected, a new operational train directory shall be computed in accordance to the rules given later in this subclause and the confirmation status of the TTDB shall be updated (if needed). In addition, actualized input to the ETBCTRL telegram sent by the local ECSP shall be provided.

6.4.5.2 State diagram

The operational train directory computation process is defined by the state diagram shown in Figure 27 and further described in Table 28 to Table 30.

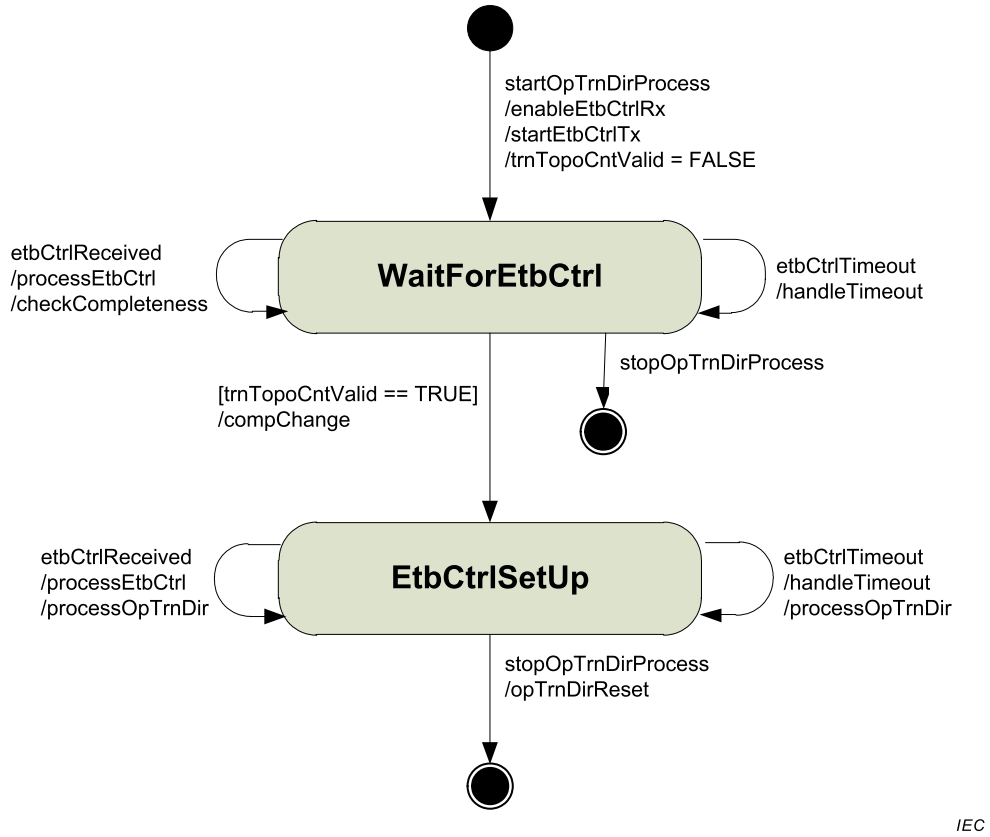


Figure 27 – ETBCTRL processing state diagram

Table 28 – ETBCTRL processing – triggers

Trigger	Description
startOpTrnDirProcess	Start processing of the operational train directory, triggered by the train directory computation state machine, see .5.3.2.
stopOpTrnDirProcess	Stop processing of the operational train directory, triggered by the train directory computation state machine, see .5.3.2.
etbCtrlReceived	ETBCTRL telegram received
etbCtrlTimeOut	ETBCTRL telegram timeout

Table 29 – ETBCTRL processing – guards

Guard	Description
[trnTopoCntValid == TRUE]	ETBCTRL telegrams have been received from all active ECSP listed in the local train directory with cstTopoCnt != 0 and all received valid ETBCTRL telegrams (including own ETBCTRL telegram) have a value of trnTopoCnt identical to the trnTopoCnt value stored in the local train directory.

Table 30 – ETBCTRL processing – actions

Action	Description
opTrnDirReset	The operational train directory shall be reset as defined in 6.7.3 (trigger event ‘opTrnDirReset’)
enableEtbCtrlRx	Prepare to receive ETBCTRL telegrams from all consist ECSPs listed in the train directory with a valid cstTopoCnt and establish individual timeout supervision for each of those consist ECSPs. Set receive timeout to a value of (3,0 ± 0,1) s.

Action	Description
	The timers shall be activated after the reception of the first related ETBCTRL telegram.
startEtbCtrlTx	Start to cyclically send ETBCTRL telegrams Cycle time: (0,5 ± 0,1) s Once ETBCTRL telegram transmission is started it can run forever, meaning that it needs not to be stopped if trigger 'stopOpTrnDirProcess' activates.
processEtbCtrl	Received ETBCTRL telegrams shall be processed as follows: 1) ETBCTRL telegrams originating from an ECSP which is not listed in the train directory (unknown cstTopoCnt value) or listed with cstTopoCnt = 0 shall be discarded. 2) ETBCTRL telegrams with a parameter 'trnTopoCnt' value different to the value of parameter 'trnTopoCnt' stored in the local train directory shall be discarded. 3) Control flags and other relevant parameters from non-discarded ETBCTRL telegrams shall be memorized, e.g. stored in the traffic store
checkCompleteness	trnTopoCntValid shall be set to TRUE if (non-discarded) ETBCTRL telegrams have been received:from all ECSPs listed in the train directory with cstTopoCnt != 0 except from those ECSPs which are marked as being timed out.
processOpTrnDir	Start operational directory computation process: • The leading vehicle function state machine as defined in 6.5. • The confirmation/correction function state machine as defined in 6.6 • Supporting operations as defined in 6.7 This computation process can be executed event driven (as indicated in the state diagram) or cyclic.
handleTimeout	Mark the related source consist as being timed out. Set the (locally stored) control flags of the related source ECSP to deferred values (e.g. in traffic store). Deferred values are defined: • in 6.5 for the leading function • in 6.6 for the confirmation/correction function • in 6.8 for the sleep mode function
compChange	Trigger a 'compChange' event: • in 6.5 for the leading function • in 6.6 for the confirmation/correction function

6.5 Function "Leading"

6.5.1 General

This function allows to elect one of the vehicles within the train to become the leading vehicle. The leading vehicle is typically the vehicle with the activated drivers desk. The function shall ensure, that at any time maximally one vehicle receives the property "leading". In some special cases, as for instance after coupling of two trains, two vehicles may have this property, which leads to a leading conflict. Such a leading conflict needs to be detected and indicated to the user. It is the users responsibility to resolve the leading conflict, e.g. by deactivating one of the active driver's cabs.

6.5.2 Function primitives

The function primitives defined in this section (Table 31 to Table 33) and shown in Figure 28 shall be supported.

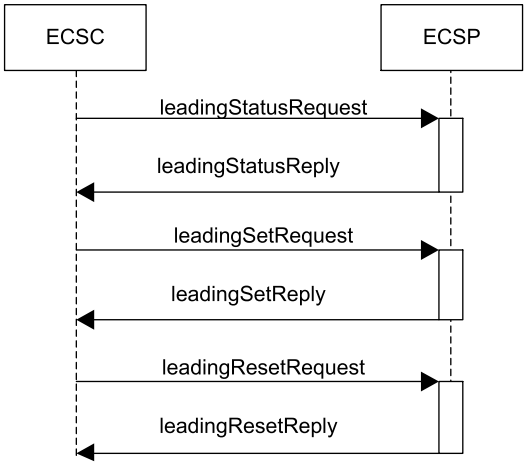


Figure 28 – Leading sequence diagram

Table 31 – Leading function primitives – F_leadingStatusRequest

Function primitive:	F_leadingStatusRequest
Description:	Read the leading status of the own consist
Request parameter:	-
Response parameter:	isLeading = yes/no wasLeading = yes/no leadingDir = Dir1/Dir2 pendingLeadingRequest = yes/no leadingConflict = yes/no leadingVehicle = n^{th} vehicle of leading consist

Table 32 – Leading function primitives – F_leadingSetRequest

Function primitive:	F_leadingSetRequest
Description:	Request to become leading vehicle. Parameter leadingDir indicates the main viewing direction of the driver's cab. NOTE In vehicles with two driver's cabs it also defines at which vehicle end the leading driver's cab is located
Request parameter:	leadingDir = Dir1/Dir2
Response parameter:	-

Table 33 – Leading function primitives – F_leadingResetRequest

Function primitive:	F_leadingResetRequest
Description:	Remove request or remove property "leading"
Request parameter:	-
Response parameter:	-

6.5.3 ECSP to ECSP protocol

6.5.3.1 General

This defines the protocol for service related data exchange between the ECSP peers.

The leading function uses four parameters (control flags) of the ETBCTRL telegram for control as listed in Table 34.

Table 34 – Leading function control flags

control flag	short form	description	deferred value
reqLead	rL	client request or abandon leading property	FALSE
accLead	aL	accept a remote leading request	Don't'care
isLead	iL	consist is leading	FALSE
wasLead	wL	consist was leading previously	unmodified NOTE Under certain conditions this value can change to 'FALSE', see note below
opTrnTopoCnt	-	Computed operational train topography counter value	Don't'care

Key

Unmodified = previously stored value to be kept

Don't care = this value shall not be used in computation (to be ignored)

NOTE In case the leading consists stops sending ECSP telegrams the leading property gets lost and a new operational train directory is computed without a leading consist. However, because the 'wasLead' control flag remains unmodified no change of the operational train direction will occur. If thereafter a new leading vehicle is elected, this control flag is set to 'FALSE', see 6.5.3.2.

The state of the leading vehicle function is determined by the set of flags received from the other consists (remote leading control flags, sent by the consists listed in the train directory with an cstTopoCnt != 0) and the own set of flags (own leading control flags) as it is shown in Figure 29.

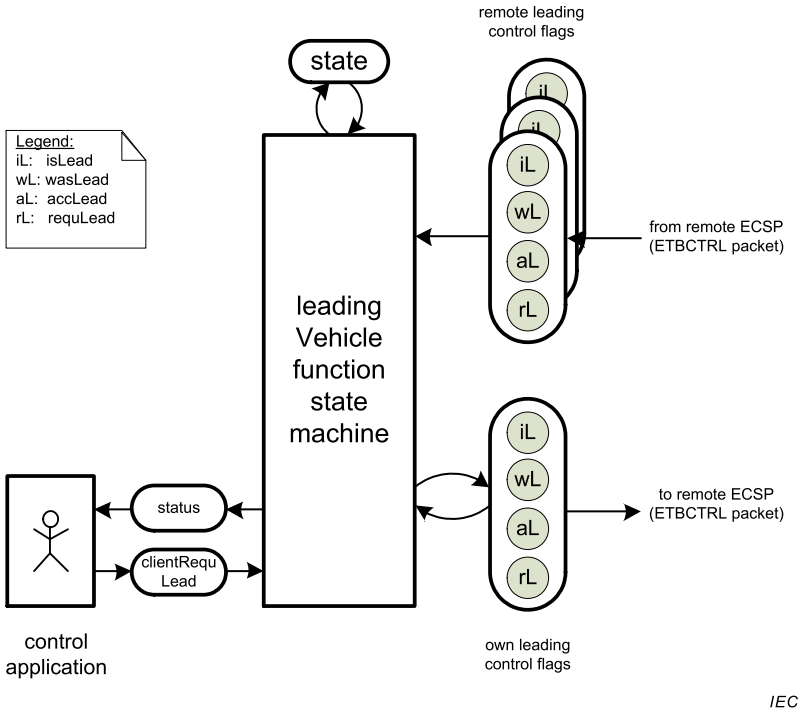


Figure 29 – Leading vehicle function state machine block diagram

All these flags are cyclically exchanged with the ETBCTRL telegram, but it has to be considered that, due to the asynchronous exchange of the ETBCTRL telegrams, the complete set of control flags might not be consistent at one time instance, leading to transient effects.

Locally, the leading vehicle function state machine is triggered by the ECSP control application, which forwards requests for leading to the state machine. The state machine, in return, forwards information about the actual status to the control application. One special information is the indication of a leading conflict which may occur if more than one consist request leading at the same time or if, e.g. after coupling, two leading vehicles exist.

6.5.3.2 State Diagram

The state diagram shown in Figure 30 defines the behavior of an ECSP with respect to the leading function. The related triggers, guards and actions are defined in Table 35 to Table 37.

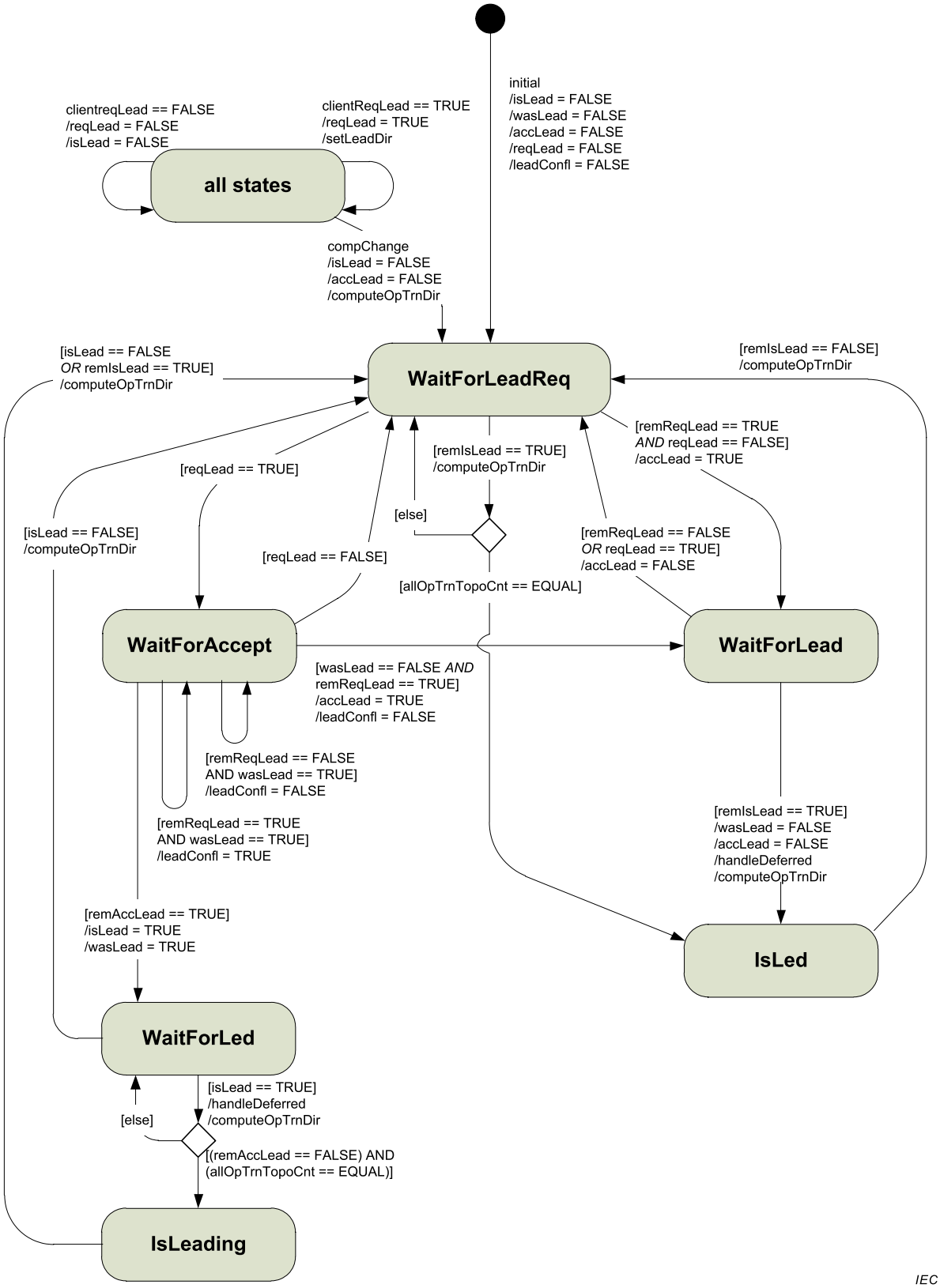


Figure 30 – State diagram of leading function

NOTE 1 For transitions which are triggered by ETBCTRL telegram reception only the guards are shown to improve readibility.

NOTE 2 The guard "remIsLead == TRUE" in state IsLeading should never happen, because if another train with property leading is coupled, then the trigger compChange will fire, because of the inauguration. But it is safer to handle this guard even though.

NOTE 3 A re-inserted ECSP in a composition with a leading vehicle will directly transit through state WaitForLead (guard remReqLead) to state IsLed (guard remIsLead).

NOTE 4 If a late inserted ECSP cannot integrate in a composition, this will lead to a composition change (trigger compChange).

Table 35 – Leading function – triggers

Trigger	Description
Initial	Power-up or re-boot of ECSP
clientReqLead	Local ECSC requests leadership (= TRUE) (service primitive F_leadingSetrequest) or requests to abandon leadership (= FALSE) (service primitive F_leadingResetRequest)
compChange	Train composition has changed. This event is triggered by the ETBCTRL processing state machine, see 6.4.5.

Table 36 – Leading function – guards

Guards	Description
reqLead	Control flag in ETBCTRL indicating, if set, that own consist request leading property NOTE This control flag is set by trigger clientReqLead.
wasLead	Control flag in ETBCTRL indicating, if set, that the own consist previously owned leading property
isLead	Control flag in ETBCTRL indicating, if set, that the own consist is leading
remReqLead	At least one remote consist requests leading, meaning that in at least one received ETBCTRL telegram except the own ETBCTRL telegram the control flag reqLead (‘rL’) is set. NOTE If more than one consist requests leading this might end in a leading conflict
remAccLead	All remote consists accept own leading request. Logical AND over control flags `accLead` of all ECSPs except the own ECSP and except ECSPs marked as being timed out (see 6.4.5.2)
remIsLead	One remote consist is leading;. Control flag ‘`isLead`’ of one ECSP except the own ECSP is set to TRUE
allOpTrnTopoCnt	Comparing the parameter ‘opTrnTopoCnt’ in all (non-discarded and not timed out) ETBCTRL telegrams. EQUAL = identical value in all telegrams

Table 37 – Leading function – actions

Action	Description
computeOpTrnDir	Compute the operational train directory taking into account the correction information obtained with the ETBCTRL telegram and update ETBCTRL telegram parameter ‘opTrnTopoCnt’.., see 6.7.3 for details
setLeadDir	Set ETBCTRL telegram parameter ‘reqLeadDir’ to the leading direction requested by the local ECSC.
leadConfl	Local flag which, if set, indicates to the application that a leading conflict has been detected.
handleDeferred	In case there are ECSPs marked as being timed out, reset (= set to FALSE) the deferred value of control flag ‘wasLead’ of the corresponding ECSP. NOTE This means that the deferred value of control flag ‘wasLead’ changes from ‘unmodified’ to ‘FALSE’.

The parameter ‘wasLead’ is used to prevent a change of the operational direction after a composition change when this consist had the leadership before the composition change.

6.5.3.3 Preconditions

Before issuing an F_leadingSetRequest or F_leadingResetRequest to the ECSP, an ECSC should inhibit ETB train inauguration.

ETB train inauguration inhibit can be cancelled once the leading request transaction is finished.

6.5.3.4 Leading conflict

A leading conflict may occur if two trains are coupled both having a leading consist.

A leading conflict shall be indicated to the ECSC latest 1,0 s after detection.

NOTE 1 A leading conflict is indicated with the F_leadingStatusRequest function primitive.

NOTE 2 This part of IEC 61375 does not specify how a leading conflict can be resolved because this depends highly on the application model. One possibility is to indicate this to the driver so that the driver can deactivate one of the leading consists.

NOTE 3 In case of a leading conflict operation may be restricted until the leading conflict is resolved.

6.5.3.5 ECSC failure

In case of a ECSC failure ((see 6.3) the ECSP shall set telegram parameter ‘reqLead’ to value FALSE.

6.6 Function Confirmation/Correction

6.6.1 General

This function allows confirming the train composition as it has been discovered during train inauguration (see IEC 61375-2-5) or to correct the discovered train composition by adding/inserting ‘gaps’. A gap is one or a sequence of vehicles which are physically present, but which have not been discovered during train inauguration (e.g. if the vehicles are powerless). A confirmation or correction is train-wide valid, meaning that this information will first be coordinated with all the other ECSPs before becoming valid. This also implies that all other ECSPs will add those confirmation and correction information to their local TTDB. A confirmed or corrected train composition will receive a unique signature (opTrnTopoCnt) which may be used in safety related functions to ensure that all train wide communication is based on the same view of the train composition.

NOTE A correction of the discovered train composition will only affect the TTDB operational train directory. The TTDB train directory and the train network directory as defined in IEC 61375-2-5 will not be affected.

6.6.2 Function primitives

Function primitives are provided by the ECSP and may be invoked by the local ECSC for controlling confirmation and correction.

The functions defined in (Table 38 to Table 40) and shown in Figure 31 shall be supported.

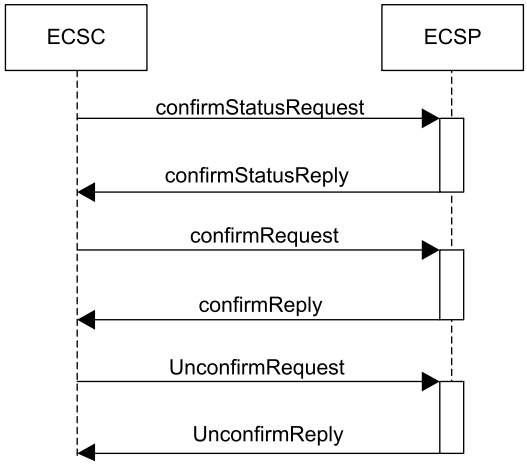


Figure 31 – Confirmation sequence diagram

Table 38 – Confirmation function primitives – F_confirmStatusRequest

Function primitive:	F_confirmStatusRequest
Description:	Read status of the train directory.
Request parameter:	-
Response parameter:	TTDB parameter values: • trnDirState as defined in 5.3.3.2.12 • opTrnDirState as defined in 5.3.3.2.12 • opTrnTopoCnt as defined in 5.3.3.2.12

Table 39 – Confirmation function primitives – F_confirmRequest

Function primitive:	F_confirmRequest
Description:	Correct and confirm a train composition. Correction information shall be entered in the ETBCTRL telegram (parameter ConfVehCnt and ConfVehList). If only confirmation is requested, no confirmed vehicle list shall be filled in (ETBCTRL parameter ConfVehCnt = 0)
Request parameter:	set of confirmation and correction rules (see 6.7.2.2)
Response parameter:	-

Table 40 – Confirmation function primitives – F_unconfirmRequest

Function primitive:	F_unconfirmRequest
Description:	Unconfirm a train composition. This includes also to remove any correction information from the ETBCTRL telegram.
Request parameter:	-
Response parameter:	-

6.6.3 ECSP to ECSP protocol

6.6.3.1 General

This subclause defines the protocol for service related data exchange between the ECSP peers.

The confirmation/correction function uses four parameters (‘control flags’) of the ETBCTRL telegram for control as listed in Table 41:

Table 41 – Confirmation function control flags

control flag / parameter	short form	description	deferred value
corrRequest	rqC	Confirmation/correction requested	FALSE
clearConfComp	clC	Clear confirmed composition	Don’t care
corrInfoSet	set	Correction info set	FALSE
compStored	stC	Composition stored	Don’t care
opTrnTopoCnt	-	Computed operational train topography counter value	Don’t’care
Don’t care = this value shall not be used in computation.			

The state of the confirmation/correction function is determined by the set of flags received from the other consists (remote confirmation/correction control flags, sent by the consists listed in the train directory with an cstTopoCnt != 0) and the own set of flags (own confirmation/correction control flags) as it is shown in Figure 32.

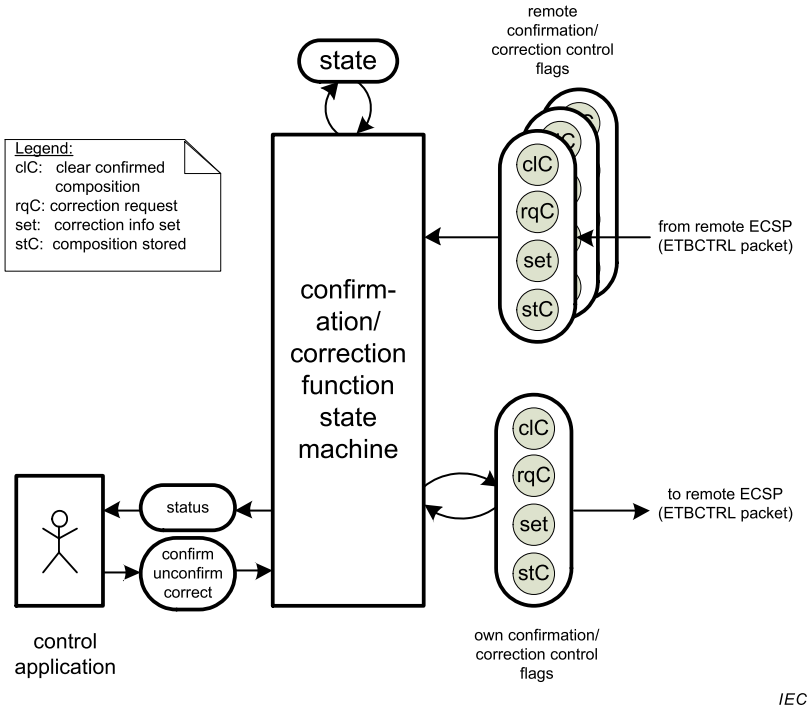


Figure 32 – Confirmation/correction function state machine block diagram

All these flags are cyclically exchanged with the ETBCTRL telegram, but it has to be considered that, due to the asynchronous exchange of the ETBCTRL telegrams, the complete set of control flags might not be consistent at one time instance, leading to transient effects.

Locally, the confirmation/correction function state machine is triggered by the ECSP control application, which forwards a request for confirmation, a request to abandon confirmation (‘unconfirm’) or a request for correction to the state machine. The state machine, in return, forwards information about the actual status to the control application.

6.6.3.2 Correction sequence

The interaction for correction between the ECSP of the different consists in the train is depicted in Figure 33 as an example of a train with two consists A and B.

Initially, the four control flags (stC,set,rqC,clC) are set to logical (0000)[3].

Confirmation with or without correction is initiated by an ECSC (ECSC A in the example) by invoking the F_confirmRequest function primitive. ECSP A will then add the confirmation/correction information to the ETBCTRL telegram, which informs all the ECSPs (including ECSP A) about the required correction or confirmation (control flags set to (0010)).

Correction information comprises:

- The number of confirmed vehicles in the train (parameter ‘confVehCnt’ in ETBCTRL telegram)
- An ordered list of confirmed vehicles in the train (parameter ‘ConfVehList’ in ETBCTRL telegram)

ECSP A will then compute a confirmed/corrected operational train directory and will thereafter set the control flags to (0110).

ECSP B will as well compute a confirmed/corrected operational train directory and will thereafter set the control flags to (0100).

If all ECSPs have computed a confirmed/corrected operational train directory, the ECSPs change to state ‘CompStored”, which is reflected in the control flags set to (1100).

The ECSC can ask for the confirmation status by invoking the F_confirmStatusRequest function primitive.

---

3 ’0’ stands for FALSE, ’1’ stands for TRUE. Here: (0000) stands for (FALSE,FALSE,FALSE,FALSE).

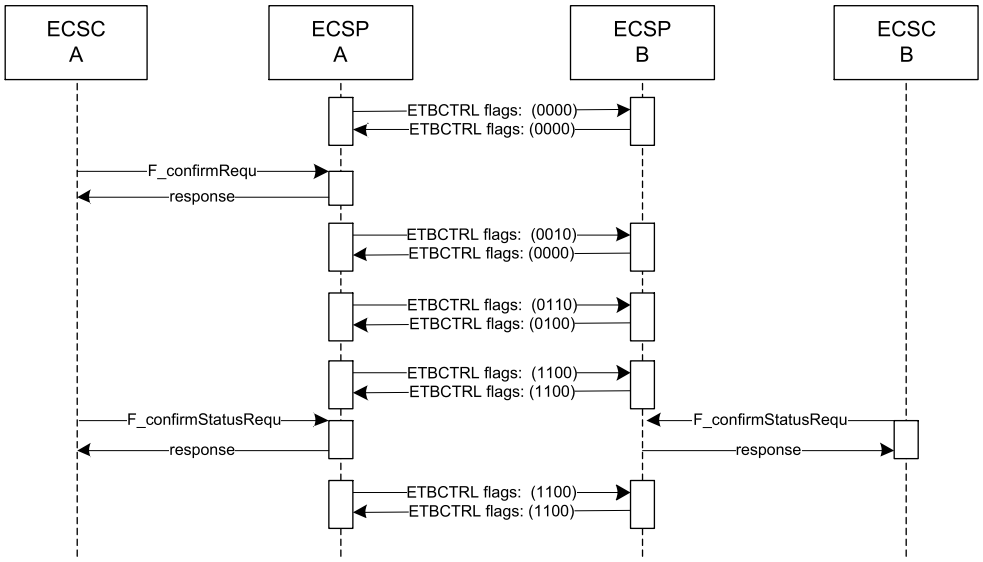


Figure 33 – Correction/confirmation protocol sequence chart (example)

6.6.3.3 Sequence for confirmation of an existing composition

The sequence for confirmation of an existing composition is identical to the correction sequence. The only difference is that the list of confirmed vehicles is empty.

6.6.3.4 Sequence for confirmed composition cancellation

A confirmed composition can be cancelled by invoking the F_unconfirmRequest function primitive, which shall be responded by the ECSP with F_unconfirmReply Upon reception of a cancellation request, the ECSP shall set the ‘clC’ control flag for three consecutive process data cycles, see Figure 34.

A ECSP receiving an ETBCTRL telegram with the ‘clC’ control flag set will acknowledge this by setting its control flags to (0000).

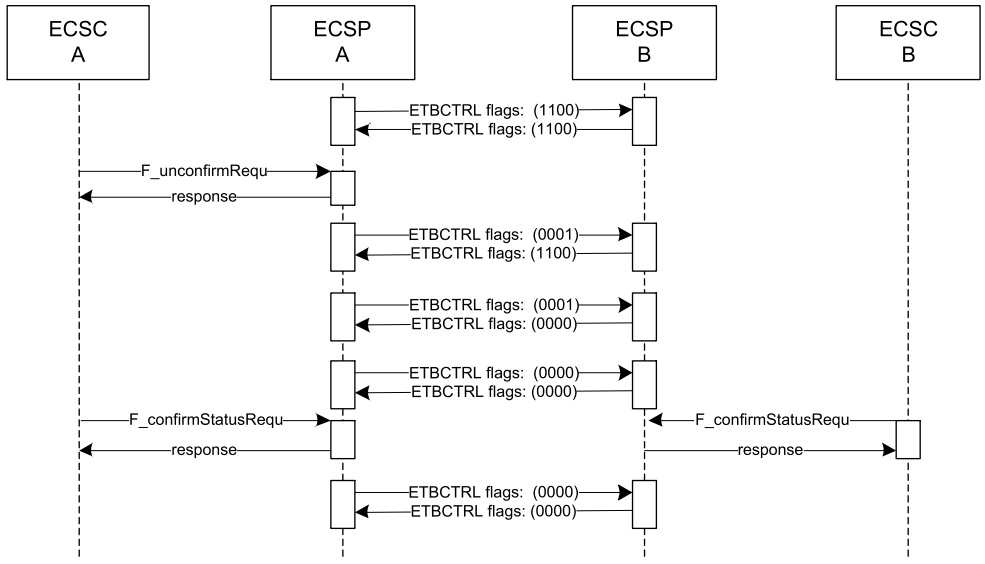


Figure 34 – Unconfirm protocol sequence chart (example)

6.6.4 State diagram

The confirmation/correction function is defined by the state diagram shown in Figure 35 with further explanations given in Table 42 to Table 44.

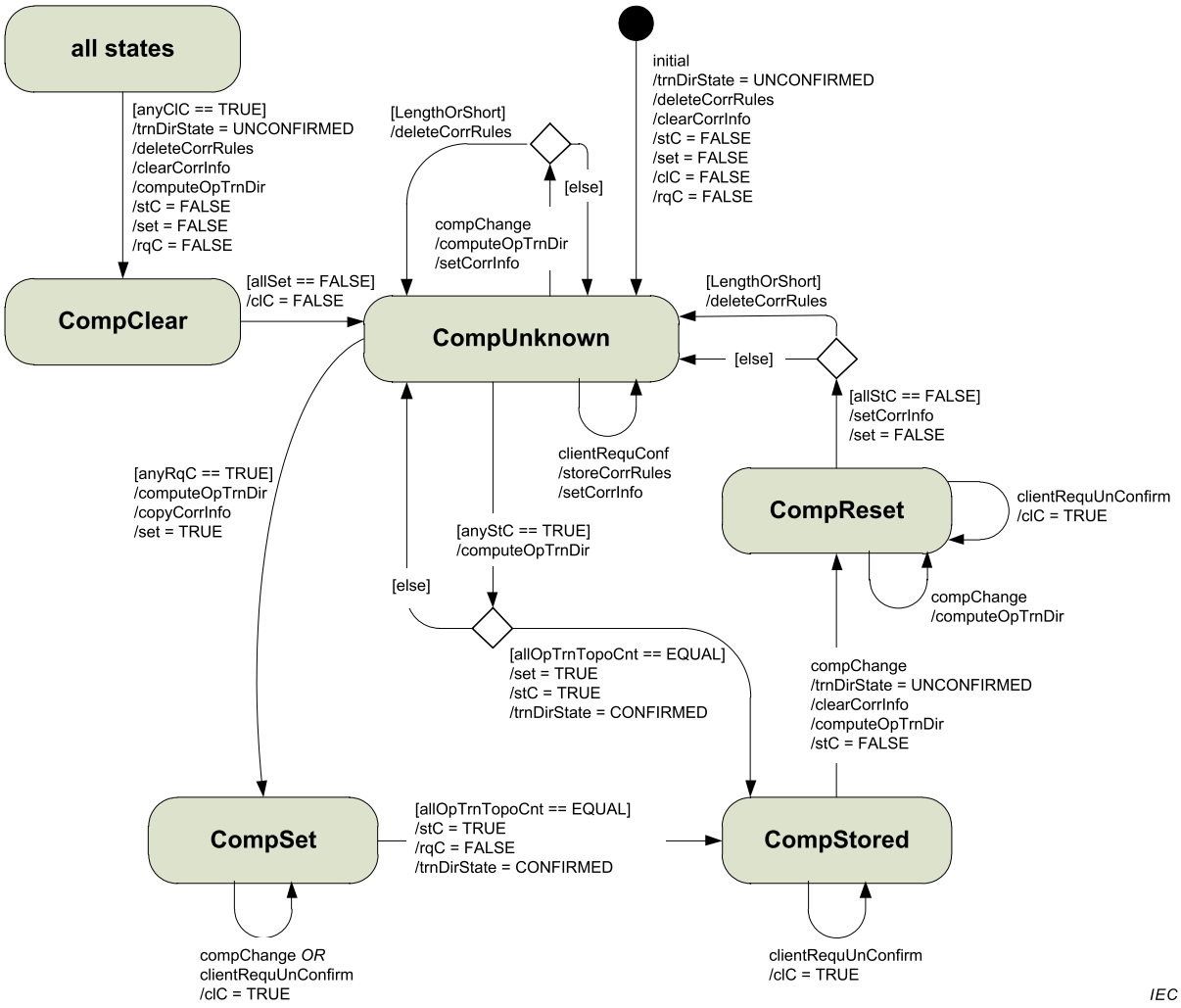


Figure 35 – Confirmation/correction state diagram

NOTE For transitions which are triggered by ETBCTRL telegram reception only the guards are shown to improve readibility.

Table 42 – Confirmation/correction state diagram – Trigger

Trigger	Description
Initial	Power-up or re-boot of ECSP
clientRequConfirm	Confirmation/correction command received from ECSC
clientRequUnConfirm	Command received from ECSC to unconfirm (reset)
compChange	Train composition has changed. This event is triggered by the ETBCTRL processing state machine, see 6.4.5.

Table 43 – Confirmation/correction state diagram – Guard

Guard	Description
anyRqC == TRUE	Logical OR over the ‘rqC’ control flags in all ETBCTRL telegrams equals TRUE
anyClC == TRUE	Logical OR over the ‘clC’ control flags in all ETBCTRL telegrams equals TRUE
allStC == FALSE	Logical AND over the ‘stC’ control flags in all ETBCTRL telegrams equals FALSE

Guard	Description
allSet == FALSE	Logical AND over the ‘set’ control flags in all ETBCTRL telegrams equals FALSE for a time of at least 3 ETBCTRL cycles. NOTE This time condition shall ensure that also reinserted ECSPs properly enter state CompUnknown.
anyStC	Logical OR over the ‘stC’ control flags in all ETBCTRL telegrams
allOpTrnTopoCnt	Comparing the parameter ‘opTrnTopoCnt’ in all ETBCTRL telegrams EQUAL = identical value in all telegrams
LenghtOrShort	The discovered end vehicles of the new composition are not identical to the discovered end vehicles of the last confirmed composition (= train lengthening or shortening)

Table 44 – Confirmation/correction state diagram – Action

Action	Description
computeOpTrnDir	Compute the operational train directory taking into account the correction information obtained with the ETBCTRL telegram and update ETBCTRL telegram parameter ‘opTrnTopoCnt’.., see 6.7.3 for details
trnDirState	Set the TTDB parameter ‘trnDirState’ to defined value.
storeCorrRules	Store the correction rules obtained from user locally
deleteCorrRules	Delete the locally stored correction rules
setCorrInfo	Check consistency of the locally stored correction rules with the operational train directory and add correction information to the ETBCTRL telegram as defined in 6.7.2.
copyCorrInfo	This action is optional. Copy the confirmed vehicle list (ETBCTRL parameters ‘confVehList’ and ‘confVehCnt’) from the ECSP distributing it to the own ETBCTRL telegram (only if this ECSP is not already distributing the confirmed vehicle list). ETBCTRL parameter ‘rqC’ shall not be changed. NOTE By copying this, the correction information is not lost when the ECSP which originally distributed the informations is out of order.
clearCorrInfo	Remove correction information from the ETBCTRL telegram (remove confirmed vehicle list) and set ETBCTRL parameter ‘confVehCnt’ to 0.

6.6.5 ECSC Failure

In case of a ECSC failure (see 6.3) no special action is requested for the ECSP.

6.7 Computation of the operational train directory

6.7.1 General

The computation of the operational train directory is supported by two actions:

- The action “setCorrInfo”
- The action “computeOpTrnDir”

Those two actions are controlled by the leading function (computeOpTrnDir only) and the confirmation/correction function.

6.7.2 Action setCorrInfo

6.7.2.1 General

The purpose of this action is to transfer the correction rules given by the user to a list of confirmed vehicles (confVehList) as it will be appended to the ETBCTRL telegram. This process is shown in Figure 36.

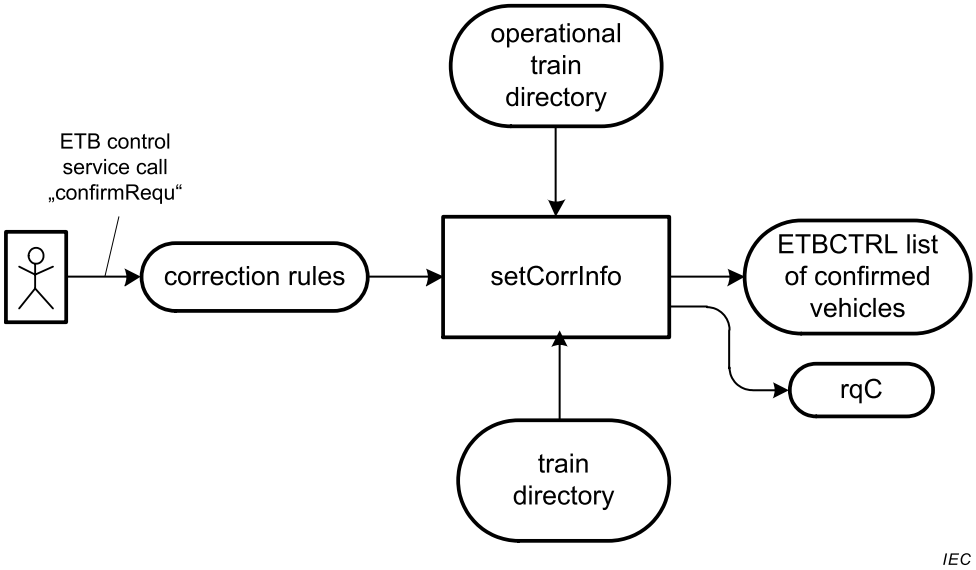


Figure 36 – Action “setCorrInfo” block diagram

For transfer, information from the train directory (e.g. TTDB parameter ‘vehId’) and from the operational train directory (e.g. TTDB parameter ‘trnVehNo’) is needed.

This transfer includes a consistency check, which checks whether the rules can be correctly applied to the given train directory following the correction policy defined in 6.7.2.3.

6.7.2.2 Confirmation/Correction rules

Confirmation/Correction rules are rules which describe which confirmation or correction actions shall be performed on a given train configuration as defined by the train directory.

The rules defined in Table 45 shall be supported.

Table 45 – Confirmation/Correction rules

Rule	Description
Confirm the composition without correction	The train composition as discovered during train inauguration and represented by the train directory is confirmed (no correction necessary)
Insert N vehicles between vehicle with VehId X and vehicle with VehId Y	Correct the discovered train composition by inserting a gap with N vehicles between discovered vehicles X and Y.
Insert N vehicles in front of extremity 1 of the vehicle with VehId X	Correct the discovered train composition by inserting a gap with N vehicles in front of the vehicle.
Insert N vehicles after extremity 2 of the vehicle with VehId X	Correct the discovered train composition by inserting a gap with N vehicles after the vehicle.
Confirm the composition with correction	The train composition including the vehicle gaps shall be confirmed.
Unconfirm the composition	Discard all corrections made and unconfirm the composition

NOTE It is not within the scope of this standard to define a specific coding of those rules. A possible implementation is given in Annex E.

6.7.2.3 Correction Policy

The correction policy defines the conditions under which a correction is accepted and under which a correction is kept following a train inauguration.

a) A correction, meaning the insertion of a gap with N vehicles into a discovered train composition, shall be accepted if the following conditions are fulfilled:
   1) The gap is not inserted within the boundaries of a consist (only insertion between consists is allowed).
   2) The total number of 63 vehicles is not exceeded.

In all other cases shall the user correction request be refused.

b) If after a correction a new train inauguration occurs, the correction shall only persist and the confirmation shall be maintained if the following conditions are fulfilled:
   1) If at the position of a gap discovered vehicles (not present after last train inauguration) appear: the number N of gap vehicles shall equal the number of newly discovered vehicles.
   2) The discovered end vehicles of the new composition are identical to the discovered end vehicles of the old composition (no lengthening or shortening).

In all other cases shall the composition become unconfirmed.

c) If after a new train inauguration previously existing middle vehicles are not anymore discovered, but the discovered end vehicles are identical to the discovered end vehicles before (no train lengthening or shortening), a gap shall be automatically inserted replacing the missing middle vehicles. This however shall only be done if the gap is not inserted within the boundaries of a consist (only insertion between consists is allowed).

6.7.2.4 Function execution

The function setCorrInfo shall behave as follows:

If there is only a rule for composition confirmation without correction, and there was no automatic gap insertion (see 6.7.2.3 item 3), then no confirmed vehicle list (confVehList) shall be appended to the ETBCTRL telegram (ETBCTRL parameter ‘confVehCnt’ = 0) and the ETBCTRL parameter ‘rqC’ shall be set to TRUE.

If there is only a rule for composition confirmation without correction, and there was an automatic gap insertion , then the confirmed vehicle list (confVehList) shall be appended to the ETBCTRL telegram (ETBCTRL parameter ‘confVehCnt’ != 0) and the ETBCTRL parameter ‘rqC’ shall be set to TRUE.

If there are neither correction rules nor a rule to confirm the composition stored, then no confirmed vehicle list (confVehList) shall be appended to the ETBCTRL telegram (ETBCTRL parameter ‘confVehCnt’ = 0) and ETBCTRL parameter ‘rqC’ shall be set to FALSE.

If the correction rules are inconsistent with the correction policy, then no confirmed vehicle list (confVehList) shall be appended to the ETBCTRL telegram (ETBCTRL parameter ‘confVehCnt’ = 0) and ETBCTRL parameter ‘rqC’ shall be set to FALSE.

If there are correction rules stored AND the correction rules are consistent with the correction policy, then the confirmed vehicle list (confVehList) shall be appended to the ETBCTRL telegram (ETBCTRL parameter ‘confVehCnt’ != 0) and the ETBCTRL parameter ‘rqC’ shall be set to TRUE.

NOTE 1 The consistency check presumes that after the successful execution of a confirmation/correction, the confirmed train composition equals the real train composition and that a change of the real train composition is only possible by decoupling or coupling of consists (e.g. the coupling of two trains). This means that if a train composition was wrongly confirmed or corrected (e.g. a non-operable vehicle in the train has been forgotten during confirmation/correction), the operational train directory may also show false results.

NOTE 2 During correction, train composition gaps are announced which indicate locations of unreachable vehicles. At that time it often is only known how many vehicles are in the gap (because they are visible to the driver for example), but not how those vehicles are associated to consists. For example, a gap with 3 vehicles might be filled with a 3-vehicle consist, a 1-vehicle and a 2-vehicle consist or three 1-vehicle consist.

NOTE 3 There exists a related action ‘copyCorrInfo’ which copies correction information obtained from another ETBCTRL telegram to the own ETBCTRL telegram, see 6.6.4. In this case ETBCTRL parameter ‘rqC’ is set to FALSE, but ETBCTRL parameter ‘confVehCnt’ is set to a value != 0. The combination of these two parameters indicates the nature of the confirmation/correction as can be seen from the following truth table:

rqC	confVehCnt	description
FALSE	= 0	Composition unconfirmed
FALSE	!= 0	Correction info copied
TRUE	= 0	Confirmation requested
TRUE	!= 0	Correction requested

EXAMPLES

Some typical examples of consistency check results are shown in Figure 37.

These examples base on a corrected train composition as shown in the beginning of Figure 37. Each vehicle is also a consist. The list of confirmed vehicles (confVehList) which is appended to the ETBCTRL telegram will look like this:

confVehCnt = 9

trnVehNo	isLead	leadDir	vehOrient
0	1	0	0
1	1	0	2
0	1	0	0
2	1	0	1
0	1	0	0
0	1	0	0
3	1	0	1
4	2	1	1
0	1	0	0

Case 1 exactly fits to the confirmed composition, in cases 2 and 4 the number of new vehicles fits to the number of vehicle gaps at that location, so this is consistent. In case three, it is not clear to which location the new vehicle shall be mapped to, it can be E or F, so this is considered inconsistent.

Case 5a depicts a case where a previously present vehicle disappeared after inauguration (creating a gap of three vehicles at that location: two corrected vehicle gaps plus one missing vehicle) and, after a next train inauguration, two new vehicles with unknown vehId are reported at these locations (case 5b). As the gap is not fully filled this is considered inconsistent

Cases 6 and 8 show a gap filling at the train's end, which is inconsistent. It cannot be determined whether the vehicle representing the gap became active or a new vehicle has been appended, in that case increasing the number of vehicles to a total of 10.

Case 7 shows a real train lengthening, but also here it cannot be determined whether one of the new vehicles fills the vehicle gap and only one vehicle has been appended, or the vehicle gap is still present and two vehicles have been appended.

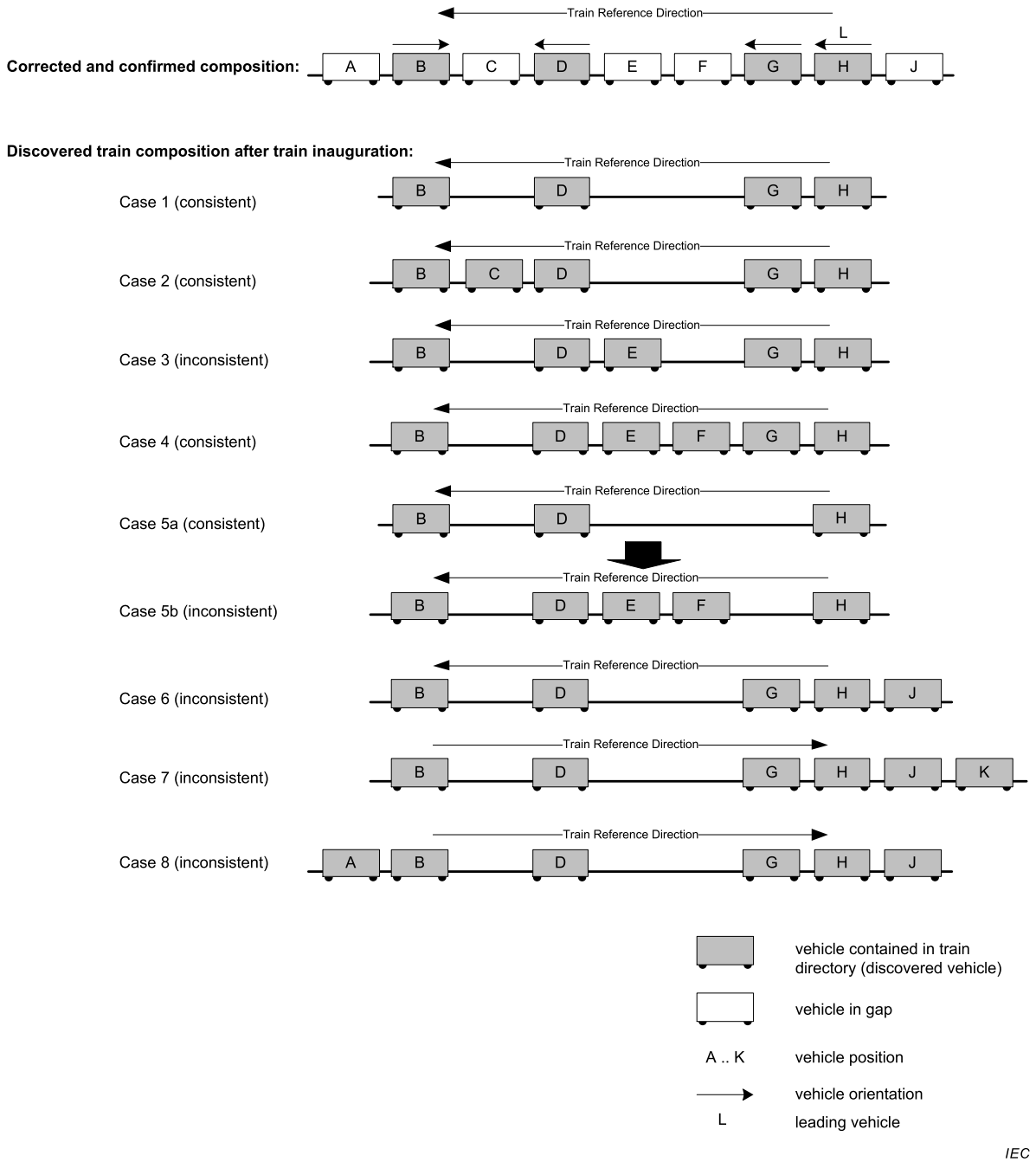


Figure 37 – Train composition consistency check examples

6.7.3 Action computeOpTrnDir

6.7.3.1 General

The action computeOpTrnDir has the task to generate the operational train directory by reading and interpreting information from four input data sources (see Figure 38):

- correction information, taken from the ETBCTRL telegram and provided by the confirmation/correction function , especially the action ‘setCorrInfo’
- the information on a leading vehicle within the train, taken from the ETBCTRL telegram and provided by the leading vehicle function (see 6.5)
- the actual sequence of consists in the train, taken from the train directory
- the number and sequence of vehicles within a consist, taken from the consist info list

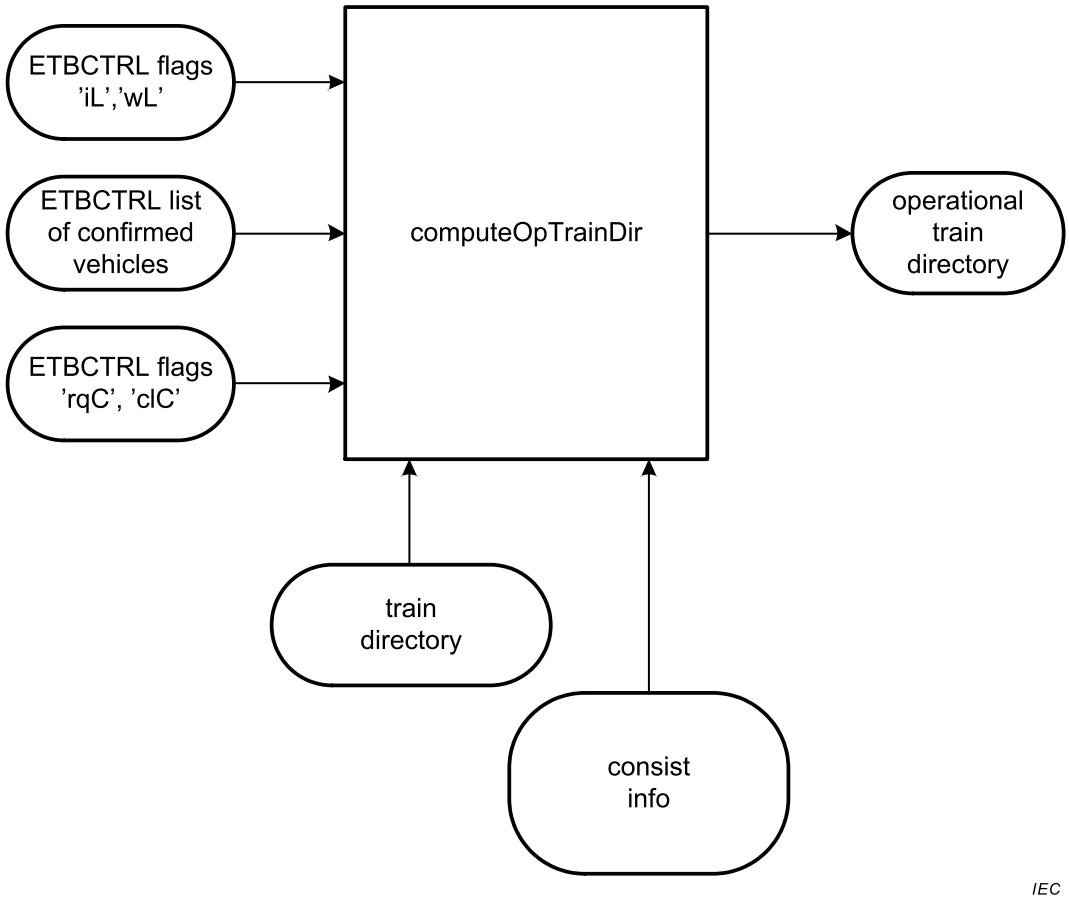


Figure 38 – Computation of the operational train directory

6.7.3.2 Operational train directory computation state diagram

The state diagram shown in Figure 39 and further explained in Table 46 to Table 48 defines the computation of the operational train directory.

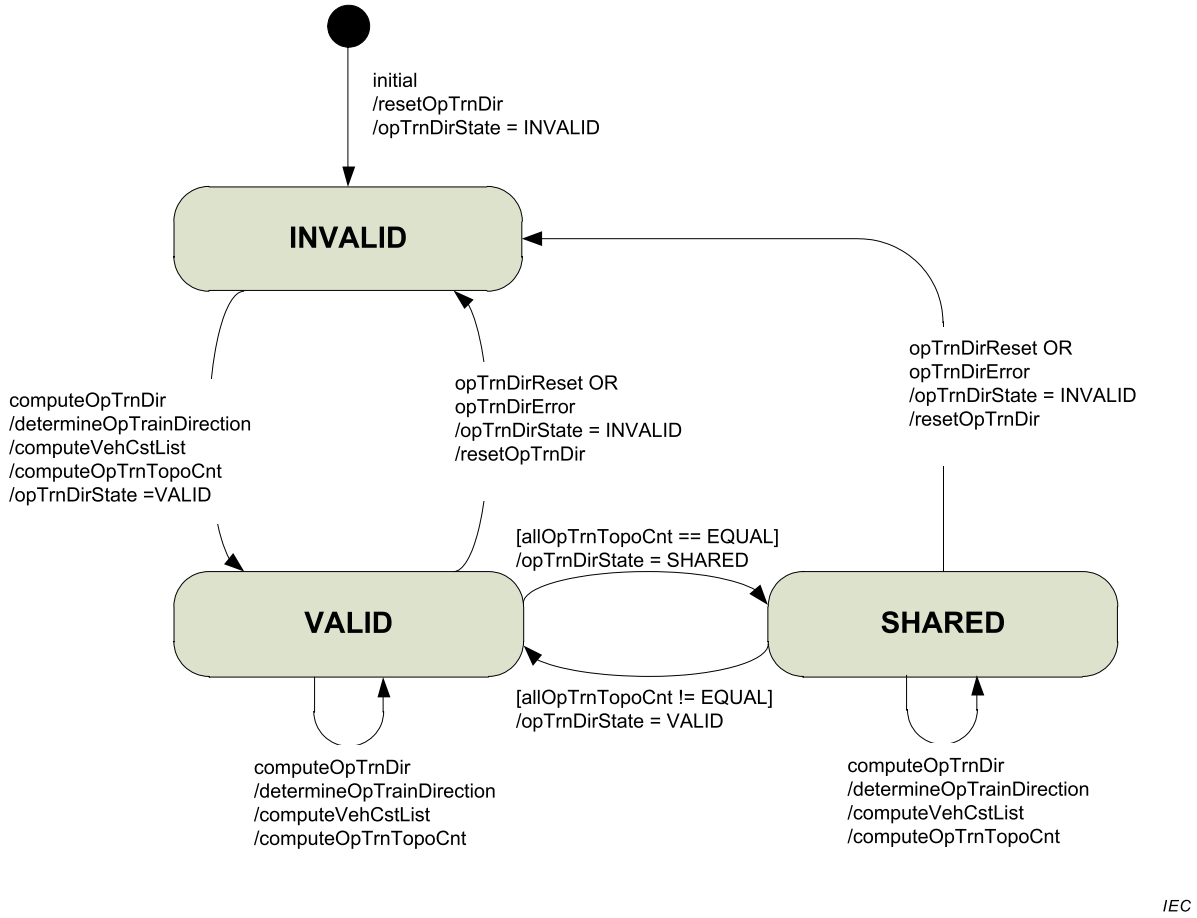


Figure 39 – computeOpTrnDir state chart

Table 46 – Operation Train Directory computation state diagram – Trigger

Trigger	Description
initial	Power-up or reboot of ECSP
computeOpTrnDir	A new computation of the operational train directory has been requested
opTrnDirReset	A reset of the operational train directory has been requested
opTrnDirError	An error has occurred preventing the computation of a valid operational train directory

Table 47 – Operation Train Directory computation state diagram – Guards

Guard	Description
allOpTrnTopoCnt == EQUAL	ETBCTRL telegrams have been received from all active ECSP listed in the local train directory with cstTopoCnt != 0 and all received valid ETBCTRL telegrams (including own ETBCTRL telegram) have an identical value of parameter ‘opTrnTopoCnt’

Table 48 – Operation Train Directory computation state diagram – Action

Action	Description
determineOpTrnDirection	The operational train direction shall be determined as specified in 4.2.4.3.
computeVehCstList	An ordered list of consists (TTDB parameter ‘opCstList’) and vehicles (TTDB parameter ‘opVehList’) shall be generated using information

Action	Description
	from the train directory and the consist info list. If parameter 'confVehCnt' value != 0 then vehicle gaps and missing vehicles, provided with correction information in the ETBCTRL telegram, shall be considered as defined in 6.7.2. As a rule, a vehicle gap shall be counted as a consist containing one vehicle.
computeOpTrnTopoCnt	The operational train topography counter value shall be calculated as defined in 5.3.3.2.13.
resetOpTrnDir	The operational train directory and the operational train directory status shall be invalidated by setting the following TTDB parameter values: opTrnTopoCnt = 0 opTrnOrient = unknown
opTrnDirState	Set TTDB parameter 'opTrnDirState' to defined value

NOTE In state 'SHARED' all ECSPs are synchronized using the same OpTrnTopoCnt value. EDs can profit because EDs can send TRDP telegrams train wide without risking that the telegram will be discarded by the receiver due to a mismatch in the opTrnTopoCnt value.

EXAMPLE

With the corrected train composition obtained from Figure 37 the operational train directory vehicle list will look as shown in Table 49.

Table 49 – Example of operational train directory

vehId	opVehNo	isLead	leadDir	trnVehNo	VehOrient	ownOpCstNo
-	1	1	0	0	0	1
'H'	2	2	1	4	2	2
'G'	3	1	0	3	2	3
-	4	1	0	0	0	4
-	5	1	0	0	0	5
'D'	6	1	0	2	2	6
-	7	1	0	0	0	7
'B'	8	1	0	1	1	8
-	9	1	0	0	0	9

6.8 Function Sleep Mode (Option)

6.8.1 General

This function allows to set a complete train to a low power mode (= sleep mode) and to return to regular operation on demand. Sleep mode will be entered if there is a request from all consists, and will be left if there are demands from at least one consist.

6.8.2 Sleep Mode Use Case (informal)

The following use case for the function sleep mode is taken from the concepts defined in UIC Leaflet 556.

The ETBNs in an individual vehicle should be able to take up the conditions listed in Table 50.

Table 50 – ETBN operating conditions

No	Operating condition	Description
0	OFF	An ETBN is not able to work when the voltage of the DC system of the vehicle concerned falls below 70 % of the nominal voltage.
1	SLEEP MODE	The ETBN is in a low power mode, but can receive signals on the train backbone cable, e.g. Ethernet frames.
2	INAUGURATION	The ETBN determines the train directory
3	REGULAR OPERATION	Information exchange over ETB

The following requirements determine the transitions of the operating and internal conditions of the ETBN:

- With battery charge available, the ETBN goes into the inauguration and regular operation.
- If the ETBN is in the sleep mode (operating condition 1), a dedicated action initiates the transition into the inauguration and regular operation (operating conditions 2 and 3). For example, UIC Leaflet 550-1 defines the action as pushing button "coach lighting on" on the operating switch desk.
- If ETB activity is recognized during sleep mode, the ETBN goes into the inauguration and regular operation (operating conditions 2 and 3).
- 45 min after the termination of battery charging throughout the train (e.g. switching off of the battery charging train line from the traction unit), the ETBN goes into the sleep mode (operating condition 1) corresponding to the sleep mode concept specified for the ETBN in this subclause.
- If the protection of the battery against deep discharge is activated (see also UIC Leaflet 550), the ETBN is switched off (operating condition 0), since all remote-controlled loads are likewise switched off.
- For emergency operation the ETBN can also be switched on below the lowest switch off stage of the minimal voltage, e.g. with the push button "coach lighting on" on the operating desk, provided that the minimum voltage of the DC system (70 % of the nominal voltage, IEC 60571) is available.
- The ETBCTRL telegram control bit "sleepRequest’ (request to change to sleep mode) controls the transition between node operation and waiting in sleep mode.

The corresponding operational states of the ETBN are shown in Figure 40.

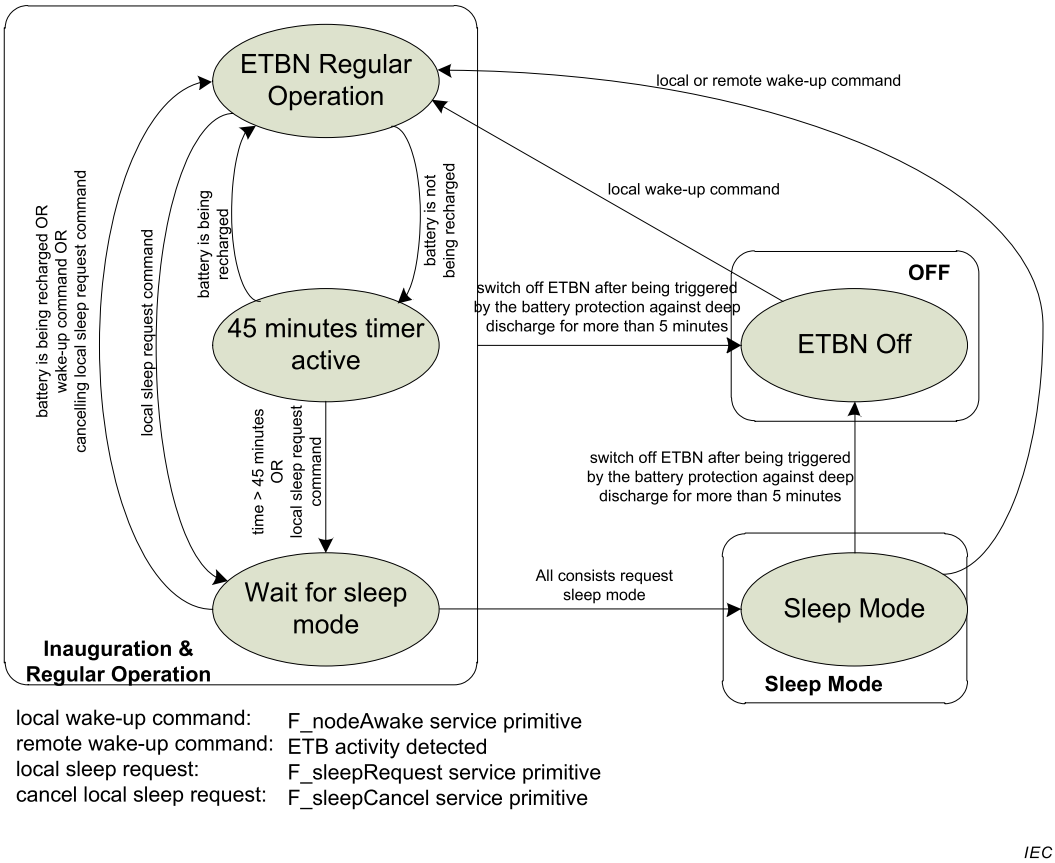


Figure 40 – Use case "sleep mode" state diagram

The handling of state 'ETBN Off' is a consist local matter not affecting interoperability and therefore not further specified in this part of IEC 61375.

Albeit the sleep mode use case described above is related to one ETBN, this part of the standard defines a more generic concept, which shares the sleep mode functionality between ECSP and ETBN:

- ECSP coordinates sleep mode requests between the different consists, handles the timing and provides the control interface to the ECSC
- ETBN supports by stopping transmission, by activating a line sensing function in sleep mode which detects line activity and by awaking the consist in case line activity has been detected

6.8.3 Exclusivity

All consists in a train have to support the sleep mode function in order to make the sleep mode function work. A mixture of sleep-mode aware and sleep-mode unaware consists is not supported.

6.8.4 Function primitives

Function primitives are provided by the ECSP and may be invoked by the local ECSC for controlling sleep mode.

The function primitives defined in this subclause (Table 51 to Table 54) and shown in Figure 41 shall be supported.

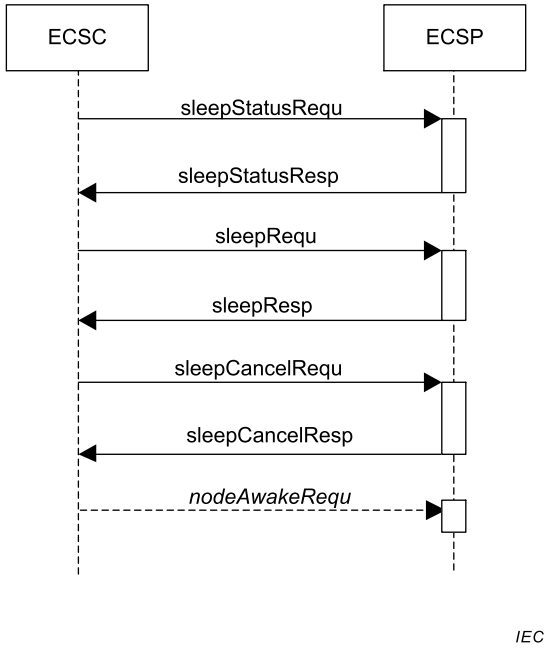


Figure 41 – Sleep control sequence diagram

Table 51 – Sleep mode function primitives – F_sleepStatus

Function primitive:	F_sleepStatus
Description:	Read status of sleep requests
Request parameter:	-
Response parameter:	number of consists requesting sleep own sleep control state

Table 52 – Sleep mode function primitives – F_sleepRequest

Function primitive:	F_sleepRequest
Description:	(Local) request to enter sleep mode (set sleepRequest bit in ETBCTRL-VDP)
Request parameter:	-
Response parameter:	sleepRequest = SET/UNSET

Table 53 – Sleep mode function primitives – F_sleepCancel

Function primitive:	F_sleepCancel
Description:	(Local) cancel sleep mode request (reset sleepRequest bit in ETBCTRL-VDP)
Request parameter:	-
Response parameter:	sleepRequest = SET/UNSET

Table 54 – Sleep mode function primitives – F_nodeAwake

Function primitive:	F_nodeAwake
Description:	Local request to return to regular operation (awake train). This function can be triggered by a local action, as for instance pushing a button "coach lighting on" on the operating switch desk or remotely via wireless ground to train communication. Dependant on the sleep mode implementation this command can be directed to the ECSP or to one or all of the ETBN.
Request parameter:	-
Response parameter:	-

6.8.5 ECSP to ECSP protocol

6.8.5.1 General

This subclause defines the protocol for service related data exchange between the ECSP peers.

The sleep control function uses one parameter (‘flag’) of the ETBCTRL telegram for control as listed in Table 55:

Table 55 – Sleep control function control flags

control flag	short form	description	default value
sleepRequest’	rSl’	Sleep request	FALSE

The state of the sleep control function is determined by the flags received from the other consists (remote sleep control flags, sent by the consists listed in the train directory with an cstTopoCnt != 0) and the own flag (own sleep control flag) as it is shown in Figure 42.

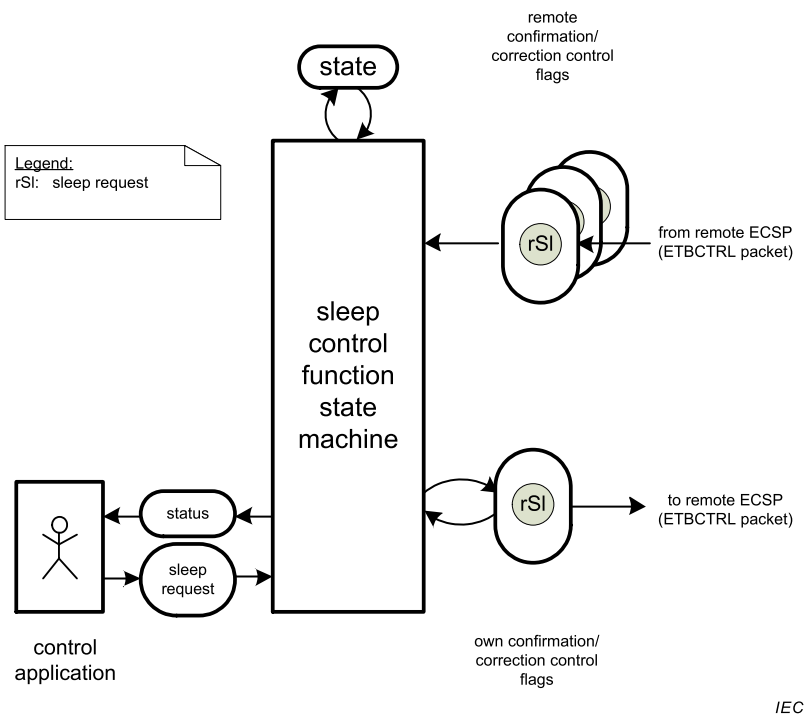


Figure 42 – Sleep control function state machine block diagram

6.8.5.2 Sleep sequence

The interaction for sleep control between the ECSPs of the different consists in the train is depicted in Figure 43 as an example of a train with two consists A and B.

Initially, the control flag for sleep control (rSl) is set to logical (0)[4].

Sleep request is initiated by an ECSC (ECSC A in the example) by invoking the F-sleepRequest function primitive. The local ECSP will then set its 'rSl' control flag, signaling that it is prepared for sleep mode, which informs all the ECSPs (including own ECSP) about the request. The remote ECSC will get this information with its status poll (function primitive F_sleepStatus) and then decides whether to prepare for sleep or not (this is an application decision). If prepared for sleep, the remote ECSC has to demand sleep mode from its local ECSP, which then will set the 'rSl' control flag accordingly. If all ECSPs have the 'rSl' control flag set, sleep mode can be entered.

NOTE 1 This part of IEC 61375 does not define how the ECSP drives a consist to a low power mode (sleep mode). This is task of the consist design.

NOTE 2 A possible sleep control interface between ECSP and ETBN is defined in Annex E.

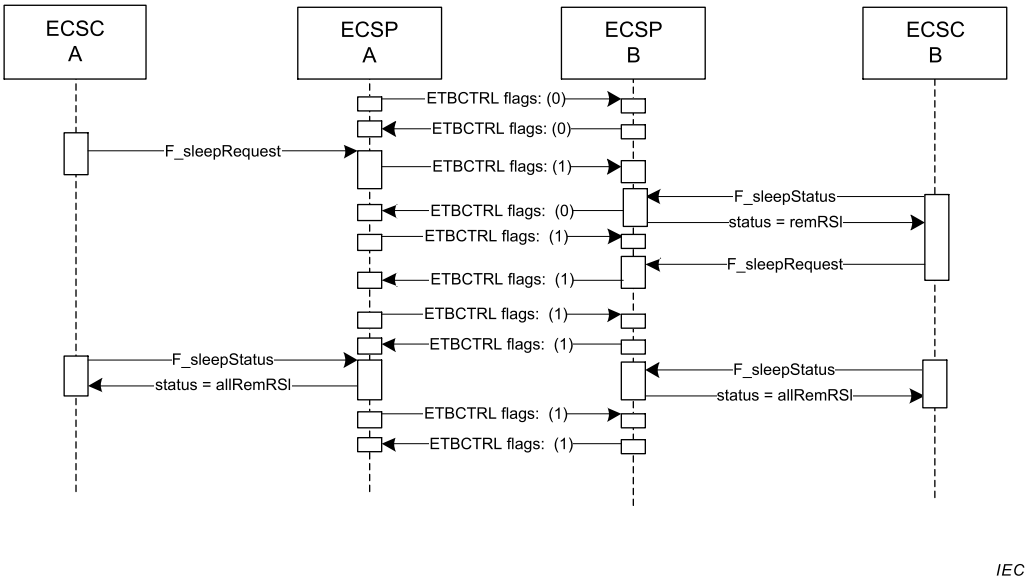


Figure 43 – Sleep request protocol sequence chart (example)

6.8.5.3 State diagram

The ECSP state diagram for sleep control is shown in Figure 44 and Tables 56 to 58.

NOTE An ECSP, which detects all 'sleepRequest' bits set, continues to send ETBCTRL telegram for a while in order to ensure that all other ETBNs have also detected this (state WaitForSleepMode). After this, the ETBN stops sending by changing to active bypass mode (state PrepareForSleepMode). Here it also stays for a while in order to ensure that all other ETBNs have also reached this state. Thereafter it passes to sleep.

4 (0) stands for (FALSE).

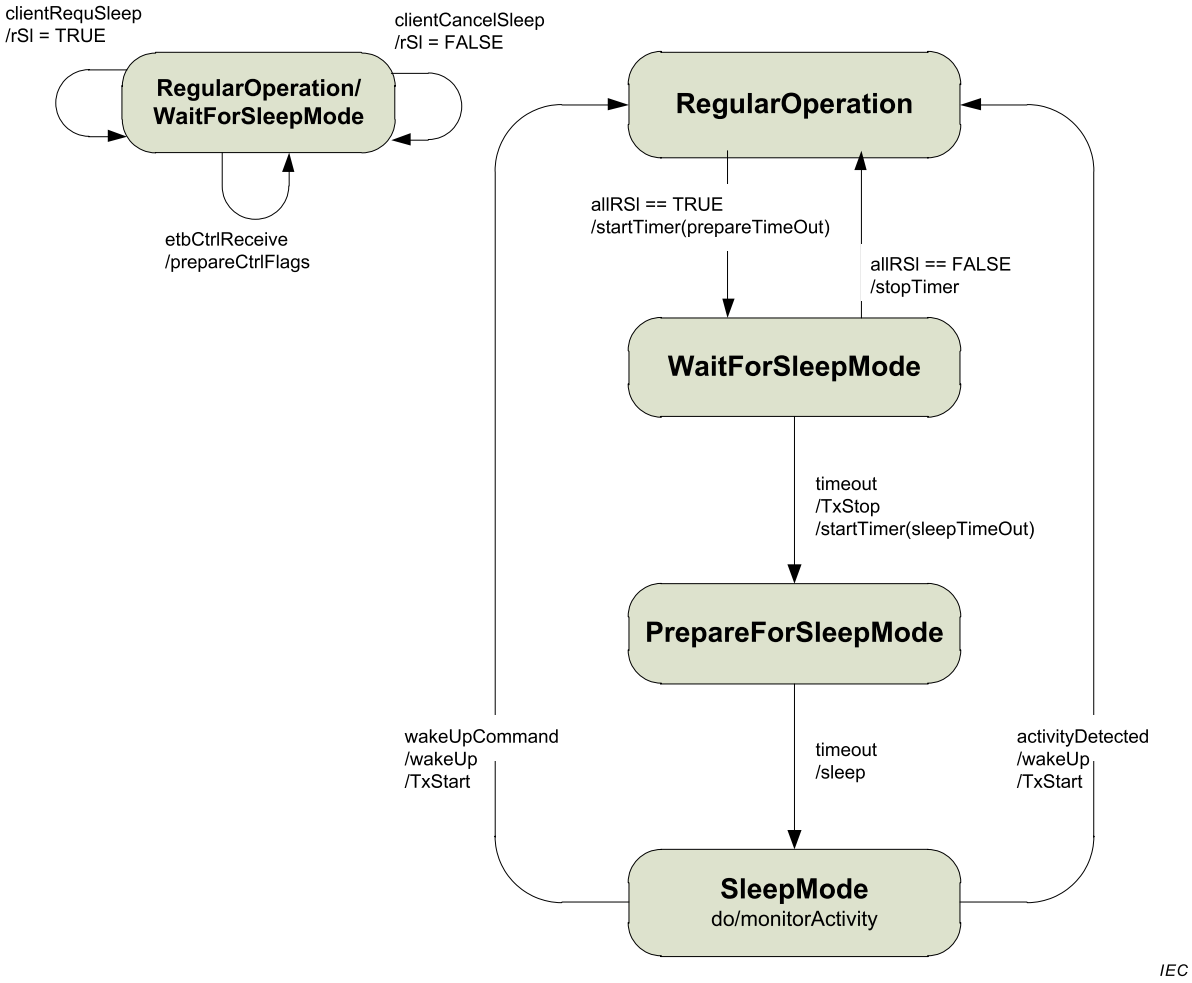


Figure 44 – Sleep control state diagram

Table 56 – Sleep control state diagram – trigger

Trigger	Description
clientRequSleep	Local sleep request command received
clientCancelSleep	Local sleep request cancel command received
activityDetected	Activity on ETB detected (HELLO frames)
wakeUpCommand	Local command to wake up
etbCtrlReceive	New ETBCTRL telegram received
timeout	Timeout

Table 57 – Sleep control state diagram – guards

Guards	Description
allRSl	Logical AND over rSl control flags in all ETBCTRL telegrams

Table 58 – Sleep control state diagram – action

Action	Description
prepareCtrlFlags	Retrieve rSI control flag from received ETBCTRL telegrams and store locally
startTimer	Start a timer with the timeout values: prepareTimeOut = (5,0 ± 1,0) s sleepTimeOut = (10,0 ± 1,0) s
stopTimer	Stop the timer
TxStop	ETBN shall stop sending on ETB, but ETBN in intermediate node setting shall still forward Ethernet frames between direction 1 and direction 2. ETBN shall be able to receive Ethernet frames for detecting line activity. NOTE One implementation possibility is to switch the ETBN in bypass
Sleep	Switch the consist to a low power mode (consist design dependant)
TxStart	ETBN resumes (enables) sending on ETB
monitorActivity	ETBN shall sense activity on ETB (reception of HELLO frames). Upon reception, the ETBN shall trigger ‘activityDetected’
wakeup	Return from low power mode ETBN start train inauguration (sending HELLO and TOPOGRAPHY frames)

6.8.5.4 ECSC Failure

In case of a ECSC failure (see 6.3) no special action is requested for the ECSP.

Annex A
(normative)

Train Real-Time Data Protocol (TRDP)

A.1 General

This annex defines the train real-time data protocol (TRDP) which shall be used for the exchange of TCN process data and TCN message data over ETB. The TRDP is executed by a TRDP layer which is placed on top of the TCP/UDP transport layer (Figure A.1). This layer may optionally be extended by a safety layer which provides safe end-to-end data transmission of safety critical process data and message data between any two end devices in the network.

All TRDP data on the wire is defined as big endian and byte alignment. Possible marshalling of user data needs to be done by a specific service primitive called from application layer before calling the TRDP service primitives to send data. As well a possible unmarshalling of user data needs to be done at application level after receiving data from the TRDP layer.

NOTE Also the optional safety layer uses the data in wire format (big endian and byte alignment).

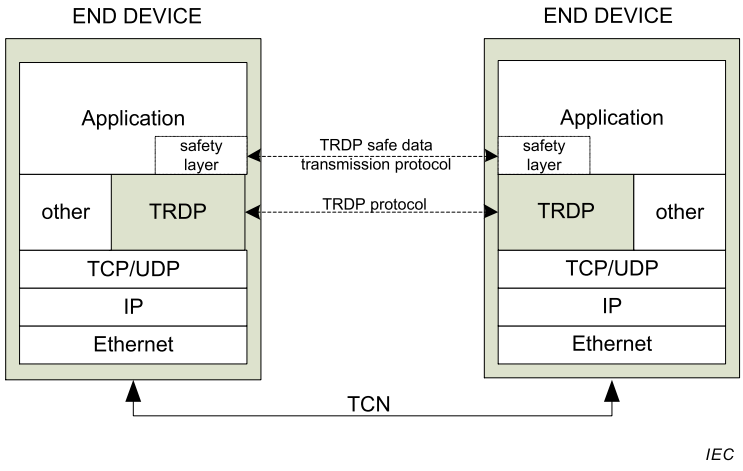


Figure A.1 – Overview of the protocol stack

A.2 Lower Layers

A.2.1 Data link layer

TRDP uses the ETB as defined in IEC 61375-2-5 for data communication within trains.

A.2.2 Network Layer

TRDP is based on IP (RFC 791) for data exchange between functions located in different consists of a train.

IP Addressing of functions and devices is defined in IEC 61375-2-5 and in this part of IEC 61375. IP address assigments may change after train inaugurations.

Data transfer between ETB and CN is provided by a gateway functionality of the ETBN as defined in IEC 61375-2-5. As a recommendation, this gateway should only transfer data between ETB and CN if the data are consistent with the actual train backbone view (defined by

the ‘etbTopoCnt’ value) and (optionally) operational train view (defined by the ‘opTrnTopoCnt’ value). This consistency can be checked by comparing the values of ‘etbTopoCnt’ and ‘opTrnTopoCnt’ contained in TRDP telegrams to the values stored in the ETBN. In case of a discrepancy the telegrams should be discarded.

NOTE 1 The filtering of inconsistent data within the ETBN reduces the probability that inconsistent data are transferred to the destination. But it does not release the destination from the responsibility to check the consistency by itself.

NOTE 2 A possible implementation will be to define corresponding firewall rules for the IP router within the ETBN (if IP router is available)

A.2.3 Transport Layer

TRDP uses the services of the UDP (RFC 768) and TCP (RFC 793) transport layer protocols for data communication.

TRDP process data shall be sent with UDP. TRDP process data packet size shall be restricted to the size of one Ethernet frame.

TRDP message data may be sent with UDP or with TCP. The choice shall be done by the user when the message data transfer is invoked.

NOTE Message data packets exceeding the Ethernet MTU in size (about 1 500 bytes) can be transmitted by TCP in order to avoid problems with packet fragmentation. However, TCP restricts to point-to-point communication.

For interoperable data communication, the destination port assignments (well-known ports) defined in Table A.1 shall be used.

Table A.1 – UDP/TCP port assignments

Protocol	Destination Port
Process Data	20548
Message Data (UDP)	20550
Message Data (TCP)	20550

For receiving any process data telegrams and for receiving UDP message data notification, request and confirm telegrams the well-known port shall be used.

For receiving UDP message data reply telegrams the port the related request was sent from shall be used.

For sending any process data telegrams and for sending UDP message data notification, request and confirm telegrams a private source port different from the well-known port shall be used.

For sending UDP message data reply telegrams any source port different from the one the request was received from can be used.

TCP connections shall be established between a source port different from the well-known port and the well-known port as destination.

Using different well-known port numbers may be allowed for project specific purposes.

The used well-known port numbers should be given as a configuration parameter to the communication stack.

A.3 TRDP FCS Computation

For TRDP FCS CRC computation the CRC32 according to IEEE802.3 is used. The CRC calculation shall be done on the data prepared for transmission – big endian format and byte alignment. The CRC itself shall be appended in little endian format (least significant byte first).

Based on the table below the following algorithm in C programming language notation with start value ‘FFFFFFFF’H is used (see Figure A.2 and Figure A.3):

```
/**********************************************************************************
 *  GLOBAL FUNCTION DEFINITIONS
 */

/**
 * @internal
 * Computes and returns a 32-bit FCS.
 *
 * @param buf   Input buffer
 * @param len   Length of input buffer
 * @param fcs   Initial (seed) value for the FCS calculation
 *
 * @return Computed fcs value
 */
static UNSIGNED32 fcs32(const UNSIGNED8 buf[], UNSIGNED32 len, UNSIGNED32 fcs)
{
    UNSIGNED32 i;

    for (i=0; i < len; i++)
    {
        fcs = (fcs >> 8)^fcstab[(fcs ^ buf[i]) & 0xff];
    }
   return (fcs ^ 0xffffffff);
}
```

Figure A.2 – FCS Computation

```
/* The FCS-32 generator polynomial:
 * x**0 + x**1 + x**2 + x**4 + x**5 + x**7 + x**8 + x**10 + x**11 +
 * x**12 + x**16 + x**22 + x**23 + x**26 + x**32.
*/
static const UINSIGNED32 fcstab[256] =
{
      0x00000000, 0x77073096, 0xee0e612c, 0x990951ba,
      0x076dc419, 0x706af48f, 0xe963a535, 0x9e6495a3,
      0x0edb8832, 0x79dcb8a4, 0xe0d5e91e, 0x97d2d988,
      0x09b64c2b, 0x7eb17cbd, 0xe7b82d07, 0x90bf1d91,
      0x1db71064, 0x6ab020f2, 0xf3b97148, 0x84be41de,
      0x1adad47d, 0x6ddde4eb, 0xf4d4b551, 0x83d385c7,
      0x136c9856, 0x646ba8c0, 0xfd62f97a, 0x8a65c9ec,
      0x14015c4f, 0x63066cd9, 0xfa0f3d63, 0x8d080df5,
      0x3b6e20c8, 0x4c69105e, 0xd56041e4, 0xa2677172,
      0x3c03e4d1, 0x4b04d447, 0xd20d85fd, 0xa50ab56b,
      0x35b5a8fa, 0x42b2986c, 0xdbbbc9d6, 0xacbcf940,
      0x32d86ce3, 0x45df5c75, 0xdcd60dcf, 0xabd13d59,
      0x26d930ac, 0x51de003a, 0xc8d75180, 0xbfd06116,
      0x21b4f4b5, 0x56b3c423, 0xcfba9599, 0xb8bda50f,
      0x2802b89e, 0x5f058808, 0xc60cd9b2, 0xb10be924,
      0x2f6f7c87, 0x58684c11, 0xc1611dab, 0xb6662d3d,
      0x76dc4190, 0x01db7106, 0x98d220bc, 0xefd5102a,
      0x71b18589, 0x06b6b51f, 0x9fbfe4a5, 0xe8b8d433,
      0x7807c9a2, 0x0f00f934, 0x9609a88e, 0xe10e9818,
      0x7f6a0dbb, 0x086d3d2d, 0x91646c97, 0xe6635c01,
      0x6b6b51f4, 0x1c6c6162, 0x856530d8, 0xf262004e,
      0x6c0695ed, 0x1b01a57b, 0x8208f4c1, 0xf50fc457,
      0x65b0d9c6, 0x12b7e950, 0x8bbeb8ea, 0xfcb9887c,
      0x62dd1ddf, 0x15da2d49, 0x8cd37cf3, 0xfbd44c65,
      0x4db26158, 0x3ab551ce, 0xa3bc0074, 0xd4bb30e2,
      0x4adfa541, 0x3dd895d7, 0xa4d1c46d, 0xd3d6f4fb,
      0x4369e96a, 0x346ed9fc, 0xad678846, 0xda60b8d0,
      0x44042d73, 0x33031de5, 0xaa0a4c5f, 0xdd0d7cc9,
      0x5005713c, 0x270241aa, 0xbe0b1010, 0xc90c2086,
      0x5768b525, 0x206f85b3, 0xb966d409, 0xce61e49f,
      0x5edef90e, 0x29d9c998, 0xb0d09822, 0xc7d7a8b4,
      0x59b33d17, 0x2eb40d81, 0xb7bd5c3b, 0xc0ba6cad,
      0xedb88320, 0x9abfb3b6, 0x03b6e20c, 0x74b1d29a,
      0xead54739, 0x9dd277af, 0x04db2615, 0x73dc1683,
      0xe3630b12, 0x94643b84, 0x0d6d6a3e, 0x7a6a5aa8,
      0xe40ecf0b, 0x9309ff9d, 0x0a00ae27, 0x7d079eb1,
      0xf00f9344, 0x8708a3d2, 0x1e01f268, 0x6906c2fe,
      0xf762575d, 0x806567cb, 0x196c3671, 0x6e6b06e7,
      0xfed41b76, 0x89d32be0, 0x10da7a5a, 0x67dd4acc,
      0xf9b9df6f, 0x8ebeeff9, 0x17b7be43, 0x60b08ed5,
      0xd6d6a3e8, 0xa1d1937e, 0x38d8c2c4, 0x4fdff252,
      0xd1bb67f1, 0xa6bc5767, 0x3fb506dd, 0x48b2364b,
      0xd80d2bda, 0xaf0a1b4c, 0x36034af6, 0x41047a60,
      0xdf60efc3, 0xa867df55, 0x316e8eef, 0x4669be79,
      0xcb61b38c, 0xbc66831a, 0x256fd2a0, 0x5268e236,
      0xcc0c7795, 0xbb0b4703, 0x220216b9, 0x5505262f,
      0xc5ba3bbe, 0xb2bd0b28, 0x2bb45a92, 0x5cb36a04,
      0xc2d7ffa7, 0xb5d0cf31, 0x2cd99e8b, 0x5bdeae1d,
      0x9b64c2b0, 0xec63f226, 0x756aa39c, 0x026d930a,
      0x9c0906a9, 0xeb0e363f, 0x72076785, 0x05005713,
      0x95bf4a82, 0xe2b87a14, 0x7bb12bae, 0x0cb61b38,
      0x92d28e9b, 0xe5d5be0d, 0x7cdcefb7, 0x0bdbdf21,
      0x86d3d2d4, 0xf1d4e242, 0x68ddb3f8, 0x1fda836e,
      0x81be16cd, 0xf6b9265b, 0x6fb077e1, 0x18b74777,
      0x88085ae6, 0xff0f6a70, 0x66063bca, 0x11010b5c,
      0x8f659eff, 0xf862ae69, 0x616bffd3, 0x166ccf45,
      0xa00ae278, 0xd70dd2ee, 0x4e048354, 0x3903b3c2,
      0xa7672661, 0xd06016f7, 0x4969474d, 0x3e6e77db,
      0xaed16a4a, 0xd9d65adc, 0x40df0b66, 0x37d83bf0,
      0xa9bcae53, 0xdebb9ec5, 0x47b2cf7f, 0x30b5ffe9,
      0xbdbdf21c, 0xcabac28a, 0x53b39330, 0x24b4a3a6,
      0xbad03605, 0xcdd70693, 0x54de5729, 0x23d967bf,
      0xb3667a2e, 0xc4614ab8, 0x5d681b02, 0x2a6f2b94,
      0xb40bbe37, 0xc30c8ea1, 0x5a05df1b, 0x2d02ef8d
};
```

Figure A.3 – FCS Table

A.4 Interaction between TRDP user and TRDP Layer

The TRDP layer shall provide services for process data and message data communication to the TRDP user at upper layer (application) as shown in Figure A.4. This service interface will be defined in an abstract way in the subsequent clauses for process data and message data communication.

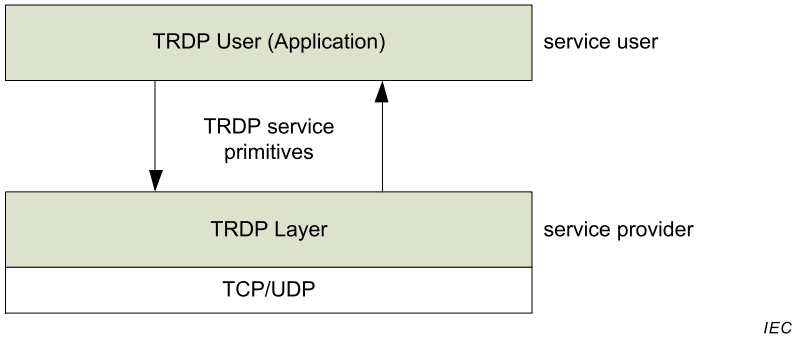


Figure A.4 – TRDP service model

A.5 Communication Identifier (ComId)

All TRDP communication is using a so called ComId transmitted in the header of each PDU. The ComId is a unique identifier for the user data structure of the PDU.

The ComId forms together with the source IP address a unique identifier of the PDU within the train.

The ComId is application dependent but should be standardized in the application profiles. It should be the reference towards the following parameters of the PD telegram:

- structure (data set) (one-to-one relation)
- source URI
- sink URI
- communication parameters (TTL, QoS)
- timeout
- validity behavior (behavior in case of invalid data / timeout) (only for PD_PDU)
- cycle time (only for PD-PDU)
- redundancy (source is redundant) (only for PD-PDU)
- safe data transmission parameters (if used, only for PD-PDU)

The ComId shall be set to zero (ComId = 0) in the following cases:

- for PDUs with unspecified user dataset.
- in message data confirmation messages
- in message data error messages

The ComIds 1..999 shall not be used by the application since they are reserved for specific use as defined in Table A.2.

Table A.2 – Reserved ComIds

comId	Description
0	unspecified PDU
1	ETBCTRL telegram
2	CSTINFO notification message
3	CSTINFOCTRL notification message
4 – 9	Additional ComIds reserved for communication framework and ETB control service
10	TRDP Echo
11 – 29	Additional ComIds reserved for communication framework and ETB control service
30	Additional ComIds reserved for TRDP statistics data
31	TRDP – statistics request command
32-34	Additional ComIds reserved for TRDP statistics data
35	TRDP – global statistics data
36	TRDP – subscription statistics data
37	TRDP – publishing statistics data
38	TRDP – redundancy statistics data
39	TRDP – join statistics data
40	TRDP- UDP listener statistics data
41	TRDP – TCP listener statistics data
50 – 79	Additional ComIds reserved for communication framework and ETB control service
80	Conformance test – control telegram
81	Conformance test – status telegram
82	Conformance test – confirmation request telegram
83	Conformance test – confirmation reply telegram
84	Conformance test – opTrnDir request telegram
85	Conformance test – opTrnDir reply telegram
86	Conformance test – echo request telegram
87	Conformance test – echo reply telegram
88	Conformance test – echo notification telegram
89 – 99	Additional ComIds reserved for conformance test
100	TTDB – operational train directory status telegram
101	TTDB – operational train directory notification
102	TTDB – train directory information request
103	TTDB – train directory information reply
104	TTDB – consist information request
105	TTDB – consist information reply
106	TTDB – train network directory information request
107	TTDB – train network directory information reply
108	TTDB – operational train directory information request
109	TTDB – operational train directory information reply
110	TTDB – train information complete request
111	TTDB – train information complete reply

comId	Description
112 – 119	Additional ComIds reserved for TTDB
120	ECSP – control telegram
121	ECSP – status telegram
122	ECSP – Confirmation/Correction request
123	ECSP – Confirmation/Correction reply
124 – 129	Additional ComIds reserved for ECSP
130	ETBN – control request
131	ETBN – status reply
132	ETBN – train network directory request
133	ETBN – train network directory reply
134 – 139	Additional ComIds reserved for ETBN
140	TCN-DNS – resolving request telegram (query)
141	TCN-DNS – resolving reply telegram
142 – 149	Additional ComIds reseraaved for TCN-DNS
150 – 199	Additional ComIds reserved for communication framework and ETB control service
200 – 999	ComIds reserved for other standards and norms

A.6 Process Data

A.6.1 Communication model

The TRDP process data protocol defines the exchange of PD-PDUs for the transfer of process data.

PD-PDUs shall be cyclically transmitted or transmitted on request between a publisher and one or many subscribers using a connectionless and unconfirmed TRDP service.

A.6.2 Roles

The communication partners in a process data transfer can have different roles:

publisher: the source of process data

subscriber: the sink of process data

requester: requesting the publisher(s) to publish its process data

Publisher and requester coincide in push communication pattern.

A subscriber may also be a requester in pull communication patterns.

A.6.3 Communication pattern

A.6.3.1 Push communication pattern

Process data exchange shall support the following push communication pattern as defined in IEC 61375-1:

a) point to point, cyclic without acknowledge, source knows the sink (Figure A.5);

b) point to multipoint, cyclic without acknowledge, source knows the sink, e.g.redundancy groups (Figure A.6);

c) point to multipoint, cyclic without acknowledge, source does not know the sink (Figure A.6).

NOTE From communication point of view there will be no difference in a point to multipoint communication pattern whether the source knows the sinks or not. It is listed here only for completeness.

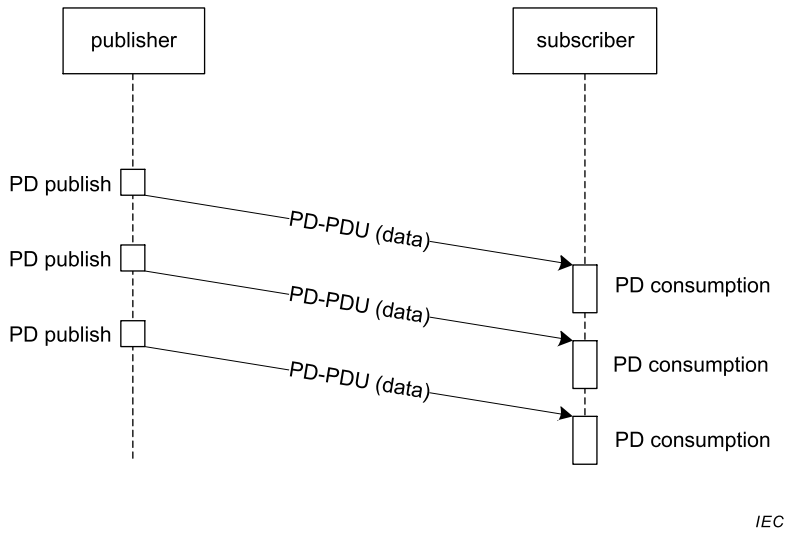


Figure A.5 – PD push pattern (point to point)

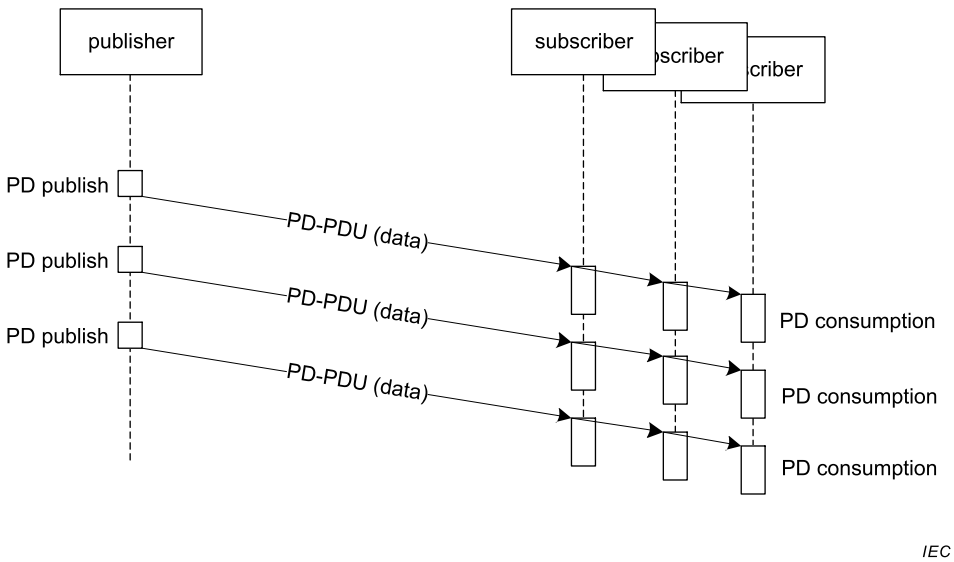


Figure A.6 – PD push pattern (point to multipoint)

A.6.3.2 Pull communication pattern

Process data exchange shall support the following pull communication pattern as defined in IEC 61375-1:

- Point to point, without acknowledge, sink knows the source (Figure A.7).

- Multipoint to point, without acknowledge, sink does not know the source (Figure A.8).
- Point to multipoint, without acknowledge, sink knows the source (Figure A.9). Here, one dedicated subscriber is requesting the known publisher to send its PD-PDU.
- Multipoint to multipoint, without acknowledge, sink does not know the source (Figure A.10). Here, one dedicated subscriber is requesting one or multiple unknown publisher to send their PD-PDU.

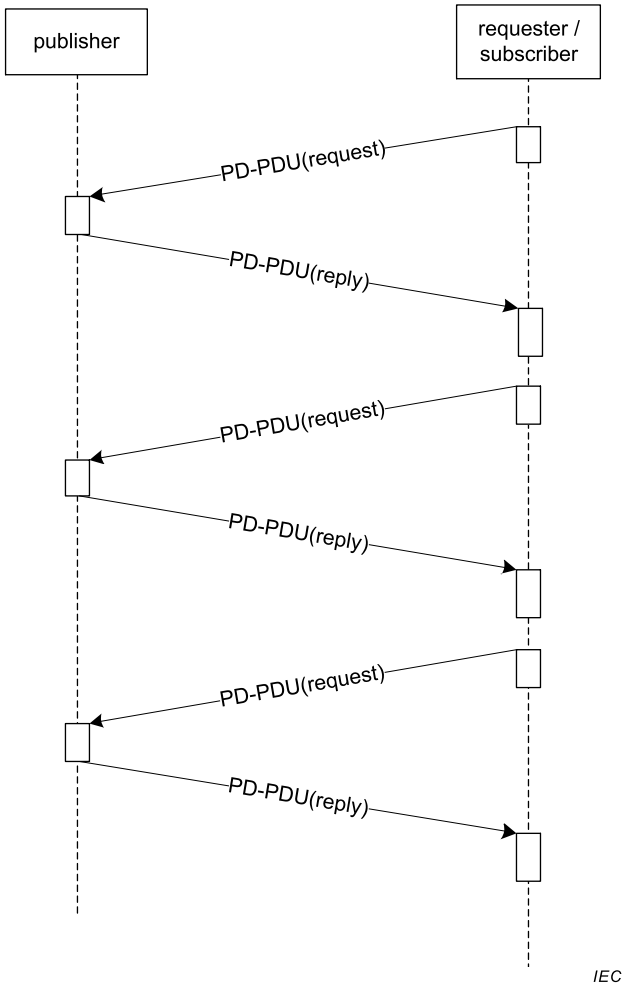


Figure A.7 – PD pull pattern (point to point, sink knows source)

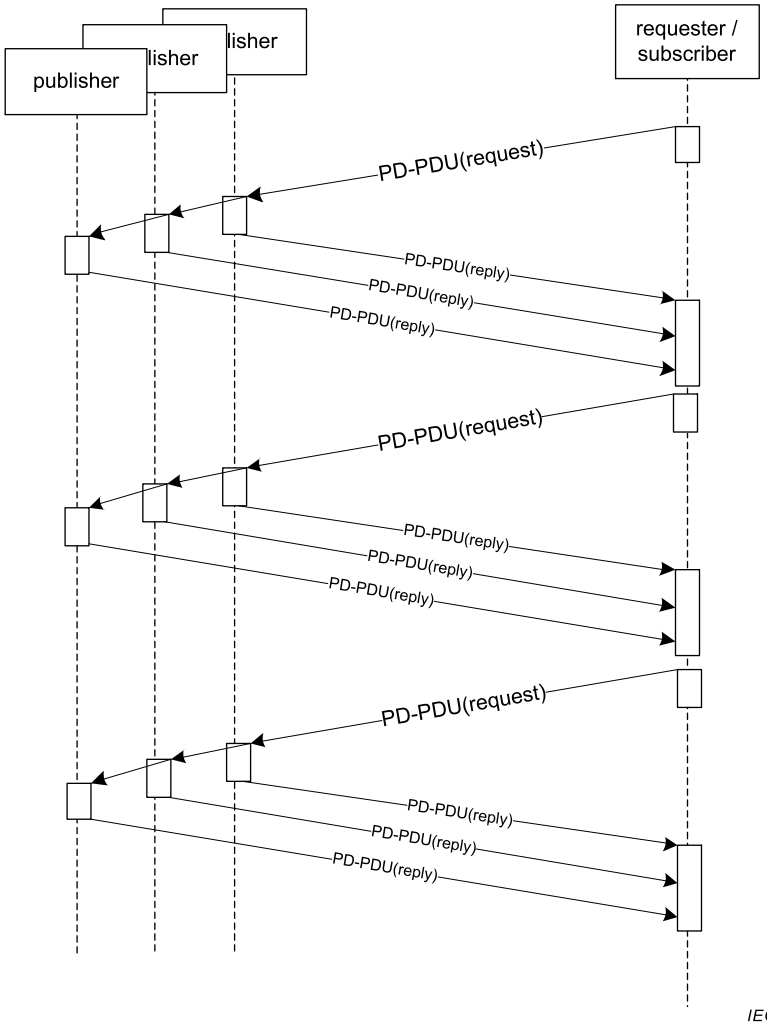


Figure A.8 – PD pull pattern (multipoint to point, sink does not know source)

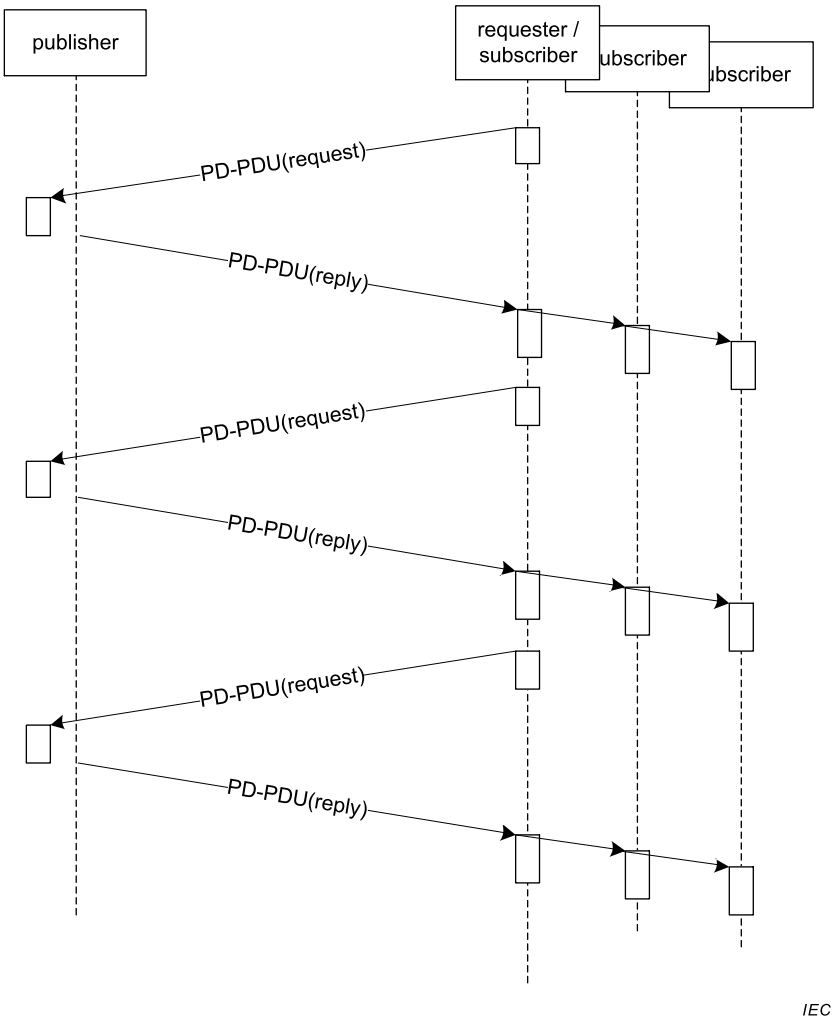


Figure A.9 – PD pull pattern (point to multipoint, sink knows source)

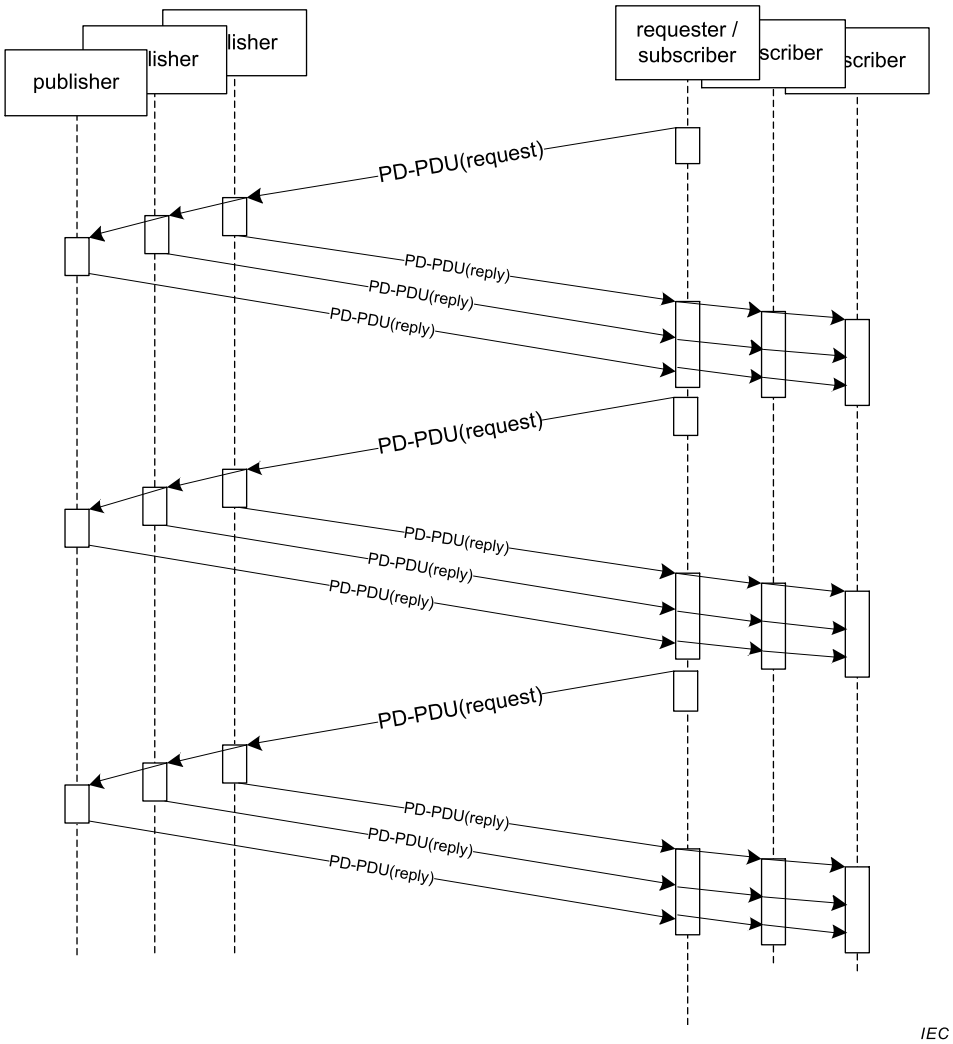


Figure A.10 – PD pull pattern (multipoint to multipoint, sink does not know source)

A.6.4 Addressing

A publisher/subscriber shall use an IP unicast address for addressing a known subscriber/publisher.

A publisher/subscriber shall use an IP multicast address for addressing groups of known subscribers/publishers (e.g. redundancy groups).

A publisher/subscriber shall use an IP multicast address for addressing unknown subscribers/publishers.

Address ranges which can be used for process data communication are defined in IEC 61375-2-5 and in 5.4.5.

A.6.5 PD-PDU

The structure of a PD-PDU is defined in Figure A.11 and in ASN.1 notation, with additional explanation given in Table A.3.

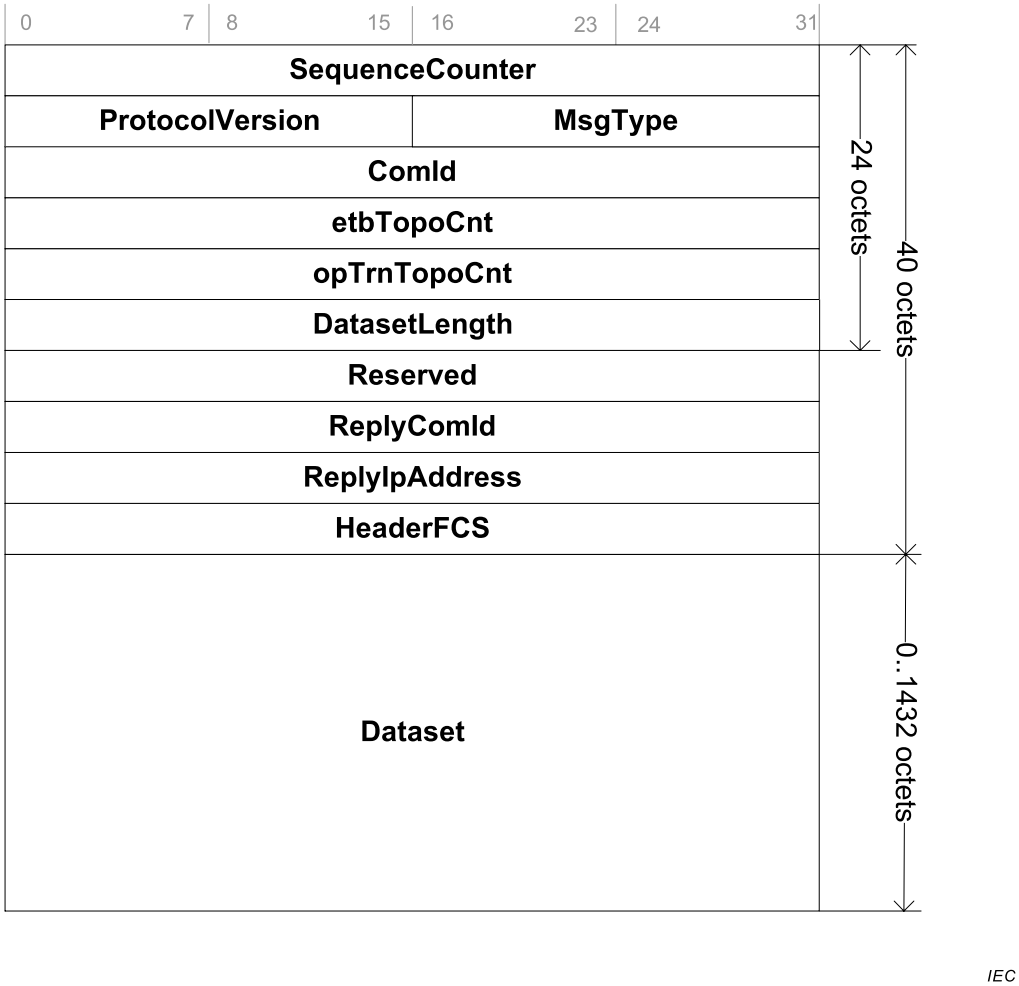


Figure A.11 – PD-PDU

```
PD_PDU::= RECORD
  {
  sequenceCounter  UINT32         -- message sequence counter
  protocolVersion  VERSION        -- protocol version
  msgType          UINT16         -- type of the message
  comId            UINT32         -- communication identifier
  etbTopoCnt       UINT32         -- etbTopoCnt value
  opTrnTopoCnt     UINT32         -- opTrnTopoCnt value
  datasetLength    UINT32         -- length of the array 'dataset'
  reserved01       UINT32         -- reserved for future use (= 0)
  replyComId       UINT32         -- requested communication identifier
  replyIpAddress   UINT32         -- reply IP address
  headerFcs        UINT32         -- header checksum
  dataset          UINT8 [datasetLength]   -- user data set
  }
```

Table A.3 – PD-PDU parameters

Parameter	Description	Value
sequenceCounter	The sequence counter: • Shall be managed for sending process telegrams per ComId/MsgType at each requester/publisher. • Shall be managed (stored) for received process telegrams per SourceIPAddr/ComId/MsgType at each publisher/subscriber. • Shall be incremented with each sending of the process telegram. Telegrams sent in parallel via different subnets are sent with the same sequence counter to detect duplication at receiver side. • Can be used for communication layer surveillance (PD sending). A surveillance that the application is still updating its process data (user data) can be done via the safe data transmission protocol (SDTv2, see Annex B) or can be implemented by the application (e.g. via a life sign).	Computed, start value 0
protocolVersion	The protocol version shall consist of: • Higher significant octet: mainVersion, incremented for incompatible changes • Lower significant octet: subVersion, incremented for compatible changes EXAMPLE – '0102'H = protocol version 1.2	Fixed
msgType	Type of the telegram. 'Pr' = PD Request 'Pp' = PD Reply 'Pd' = PD Data 'Pe' = PD Data (Error) NOTE Coded in ASCII for better reading with open protocol analyzer tools (e.g. Wireshark) MsgType 'Pe' is only for notification purpose and should not be sent over the bus	'5072'H ('Pr') '5070'H ('Pp') '5064'H ('Pd') '5065'H ('Pe')
comId	Identifier of the user dataset. See also A.5.	set by user
etbTopoCnt	The ETB topography counter: • shall be used (train addressing) as defined in IEC 61375-2-5 (parameter 'etbTopoCnt') • shall be set by the user • shall be set for all communication over the ETB • shall be set if a valid opTrnTopoCnt is set • optional in all other cases. Shall be set to 0 (= invalid) if not used.	0..2^{32}-1 set by user
opTrnTopoCnt	The operational train topography counter: • shall be used as defined in 5.3.3 • shall be set by the user • shall be set for communication between functions which use information from the operational train directory (e.g. side selective operations) • shall be set when the source device used the operational train directory to retrieve the destination IP address (e.g. resolving of URI 'vcu.leadVeh.anyCst.anyClTrn.lTrn') • optional in all other cases. Shall be set to 0 (= invalid) if not used.	
datasetLength	The dataset length:	0..1432

Parameter	Description	Value
	• shall be the length of the user data set in number of octets without padding octets. • shall be the primary information about the user data size NOTE In case of fixed size data sets this is redundant information because the dataset length is already defined by the ComId.	Computed
reserved01	Reserved for future use	set to 0
replyComId	The requested ComId: • shall be used only in a PD request • shall be used as ComId in the reply • If set to 0, the ComId of the request shall be used for the reply • If set to 0, and the ComId of the request is also set to 0, the reply shall be sent as an unspecified PDU.	set by user in PD request, in other telegram types set to 0
replyIpAddress	The reply IP address: • shall be used only in a PD request • shall be used as destination IP address in the reply • If set to 0, the source IP address of this request shall be used for the reply.	set by user for request otherwise set to 0
headerFCS	The header frame check sequence: • shall be calculated for the PD-PDU header • shall not include the HeaderFCS itself.	Computed according to A.3
dataset	The user data If the user data length is not a multiple of 4 octets, octets with a value of 0 (zero) should be appended until a multiple of 4 octets is reached (padding bytes).	

A.6.6 Interaction between application and TRDP protocol layer

A.6.6.1 Service primitives

The TRDP layer shall provide a service for process data communication to the TRDP user at upper layer. The service primitives listed in Table A.4 shall be provided.

NOTE The service primitives are defined in an abstract way in order not to restrict implementations. The specification of the service interface itself is not in the scope of this part of the standard.

Table A.4 – TRDP service primitives

Role	Service primitive	Parameter	Description
publisher	PD.publish		TRDP user → TRDP layer service user hands over a new process data packet
		Handle	Handle for this publishing, returned by the TRDP layer
		etbTopoCnt	as defined in Table A.3
		opTrnTopoCnt	as defined in Table A.3
		ComId	as defined in Table A.3
		DatasetLength	Actual user data length to be sent as defined in Table A.3

Role	Service primitive	Parameter	Description
		Dataset	Buffer with user data to be sent as defined in Table A.3
		CycleTime	Cycle time (txTime) for sending PD. Unit: 10^{-6} s (μs)
		DestinationIpAddress[]	Destination IP address(es) to send the PD to.
		redundant	FALSE means that the PD are published immediately, TRUE means that PD are only published if redundant ComIds are activated
	PD.unPublish		TRDP user → TRDP layer Stop publishing
		Handle	Handle which has been given by the TRDP layer
	PD.putData		
		Handle	Handle which has been given by the TRDP layer
		DatasetLength	Actual user data length to be sent as defined in Table A.3
		Dataset	Buffer with user data to be sent as defined in Table A.3
	PD.activateRed		TRDP user → TRDP layer service user activates the publishing of the ComIds marked as redundant
	PD.deactivateRed		TRDP user → TRDP layer service user deactivates the publishing of the ComIds marked as redundant
Requester	PD.request		TRDP user → TRDP layer TRDP layer sends a PD-PDU.request to one or multiple publishers. Cyclically called from application.
		Handle	handle for this subscription of the reply ComId; returned by the TRDP layer
		etbTopoCnt	as defined in Table A.3
		opTrnTopoCnt	as defined in Table A.3
		ComId	ComId to be sent, as defined in Table A.3
		DatasetLength	data set length to be sent, as defined in Table A.3
		DataSet	User data set to be sent, as defined in Table A.3
		DestinationIpAddress[]	Destination IP address(es) to send the PD request to.
		redundant	FALSE means that the PD are published immediately, TRUE means that PD are only published if redundant ComIds are activated

Role	Service primitive	Parameter	Description
		ReplyComId	as defined in Table A.3
		ReplyIpAddress	as defined in Table A.3
Subscriber	PD.subscribe		TRDP user → TRDP layer User subscribes to process data using possible filter for ComId, SourceIpAddress, SourceIpAddress range, DestinationIpAddress.
		Handle	handle for this subscription; returned by the TRDP layer
		etbTopoCnt	as defined in Table A.3
		opTrnTopoCnt	as defined in Table A.3
		ComId	as defined in Table A.3
		DatasetLength	as defined in Table A.3
		Dataset	as defined in Table A.3
		SourceIpAddress	IP source address, generated out of respective URI's using DNS. Defines the lower IP address in case of an IP address range (see 5.4.4.6.2) Set to 0 if no filtering requested
		SourceIpAddress2	Defines the upper IP address in case of an IP address range (see 5.4.4.6.2) IP source address, generated out of respective URI's using DNS Set to 0 if not used
		DestinationIpAddress	IP destination address, generated out of respective URI's using DNS Set to 0 if no filtering requested
		Subnet	Select the subnet(s) to subscribe; designated one or all of the redundant networks.
		Timeout	Timeout (rxTime) for PD to be received Unit: 10^{-6} s (μs)
	PD.unsubscribe		TRDP user → TRDP layer User unsubscribes to process data using possible filter for ComId, SourceIpAddress, DestinationIpAddress.
		Handle	handle returned by the TRDP layer in PD.subscribe
	PD.indicate		TRDP layer → TRDP user TRDP layer indicates new process data or a timeout. The timeout value is given with PD.subscribe.
		Handle	handle returned by the TRDP layer in PD.subscribe
		IndType	• process data ('Pd') • process data reply ('Pp') • process data request ('Pr') • process data error ('Pe')

Role	Service primitive	Parameter	Description
		etbTopoCnt	Received topo count, as defined in Table A.3
		opTrnTopoCnt	Received topo count, as defined in Table A.3
		ComId	Received ComId, as defined in Table A.3
		DatasetLength	Received user data size, as defined in Table A.3
		Dataset	Received user data, as defined in Table A.3
		SequenceCounter	Received sequence counter, as defined in Table A.3
		SourceIpAddress	Received source IP address
		DestinationIpAddress	Received destination IP address, as defined in Table A.3
		ReplyIComId	Received reply ComId, as defined in Table A.3
		ReplyIpAddress	Received reply IP address, as defined in Table A.3
		Status In case of a TRDP error (‘Pe’) the value is supplied by TRDP. • 1 – timeout • 6 – no subscription • 5 – no memory • 7 – data invalid	0 OK < 0 NOK
	PD.poll		TRDP user → TRDP layer User polls for new PD instead of using the indication
		Handle	handle returned by the TRDP layer in PD.subscribe
		IndType	• process data (‘Pd’) • process data reply (‘Pp’) • process data request (‘Pr’) • process data error (‘Pe’)
		etbTopoCnt	Received topo count, as defined in Table A.3
		opTrnTopoCnt	Received topo count, as defined in Table A.3
		ComId	Received ComId, as defined in Table A.3
		DatasetLength	Received user data size, as defined in Table A.3
		Dataset	Received user data, as defined in Table A.3
		SequenceCounter	Received sequence counter, as defined in Table A.3
		SourceIpAddress	Received source IP address
		DestinationIpAddress	Received destination IP

Role	Service primitive	Parameter	Description
		ReplyIComId	Received reply ComId, as defined in Table A.3
		ReplyIpAddress	Received reply IP address, as defined in Table A.3
		Status In case of a TRDP error ('Pe') the value is supplied by TRDP. • 1 – timeout • 5 – no memory • 6 – no subscription • 7 – data invalid	0 OK < 0 NOK

A.6.6.2 PD User access

The TRDP user shall have two possibilities to retrieve PD: either via a poll mechanism, typically used in cyclic user tasks, or via indication mechanism, where the TRDP layer notifies the user when new data are available or when there is a timeout.

A.6.6.3 Interaction sequence – PD Pull Pattern

The sequence chart in Figure A.12 defines the allowed sequence of the service primitives for the PD pull pattern, PD.request service primitive and related request telegram are only called by a requester.

NOTE 1 Dotted lines in the sequence charts are used to indicate options.

Process data are prepared cyclically by the publisher and shall be given to the TRDP layer calling the PD.putData primitive.

To receive a request, the publisher shall subscribe for it.

Until receiving the first request telegram matching to the filter criteria of the subscription, related data in the receive buffer of the replier shall be marked as invalid.

Also a request for already cyclically published PDU shall be possible.

An incoming request telegram with parameter values 'etbTopoCnt' and 'opTrnTopoCnt' different to expected (own locally stored) values shall be discarded.

Each incoming request telegram shall be responded by the TRDP layer using the available process data, the reply IP address and the reply ComId given in the request telegram.

If the reply IP address of the request telegram is 0, the source IP address of the request telegram shall be used as destination IP address for the reply.

To receive the related reply for a request, the requester needs to subscribe for it. Destination IP address, source IP address and source IP address range are possible filter criteria.

Until receiving the first reply telegram matching to the filter criteria of the subscription, related data in the receive buffer of the subscriber shall be marked as invalid. Before sending the request telegram, the related reply data in the receive buffer of the subscriber shall be set to invalid.

NOTE 2 This is to prevent that the user fetches outdated reply data after it has requested new reply data.

Timeout supervision at subscriber TRDP layer shall be restarted after sending the request telegram.

The request can contain user data.

Before sending out a request, the TRDP layer shall check the topopology counters submitted with the request against the actual topology counters. At least one of the cases listed in Table A.5 for the topography counters shall be fulfilled.

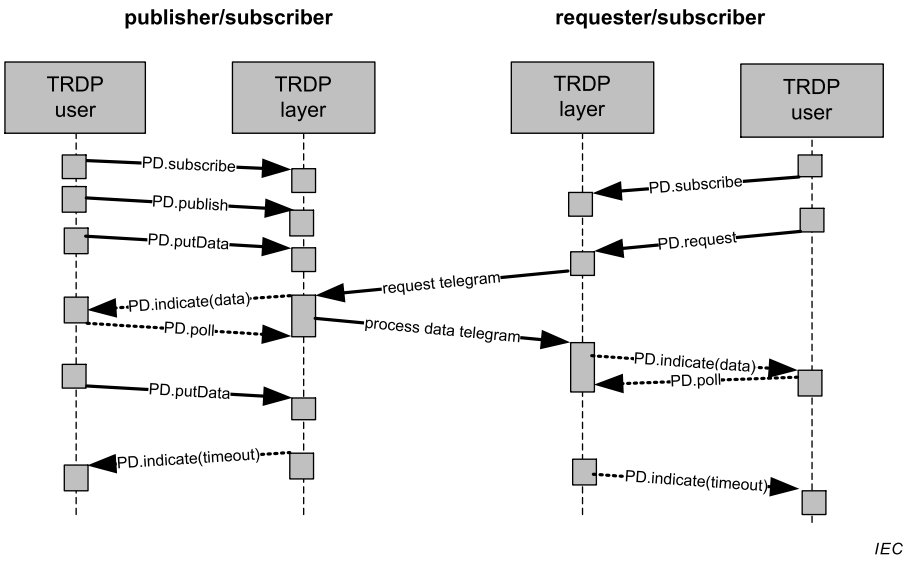


Figure A.12 – Interaction sequence chart for PD pull pattern

A.6.6.4 Interaction sequence – PD Push Pattern

The sequence chart in Figure A.13 defines the allowed sequence of the service primitives for the PD push pattern.

Process data are prepared cyclically by the publisher and shall be given to the TRDP layer calling the PD.putData primitive.

The publisher TRDP layer shall send the data in the configured cycle to the configured address.

The publisher TRDP layer shall send out the data only after checking the locally stored topography counters submitted with the publish against the actual topography counters. At least one of the cases listed in Table A.5 for the topography counters shall be fulfilled.

Any subscriber can subscribe for the process telegram using ComID, destination IP address and source IP address of the process telegram as possible filter criteria.

Timeout supervision at subscriber TRDP layer shall be started after subscription.

Timeout supervision at subscriber TRDP layer shall be restarted after receiving the related PD PDU.

The subscriber TRDP layer shall check the topography counters of the received telegram against the actual topography counters and against the topography counters submitted with the

subscription. At least one of the cases listed in Table A.5 for the topography counters shall be fulfilled.

Until receiving the first valid telegram matching to the filter criteria the data shall be marked as invalid.

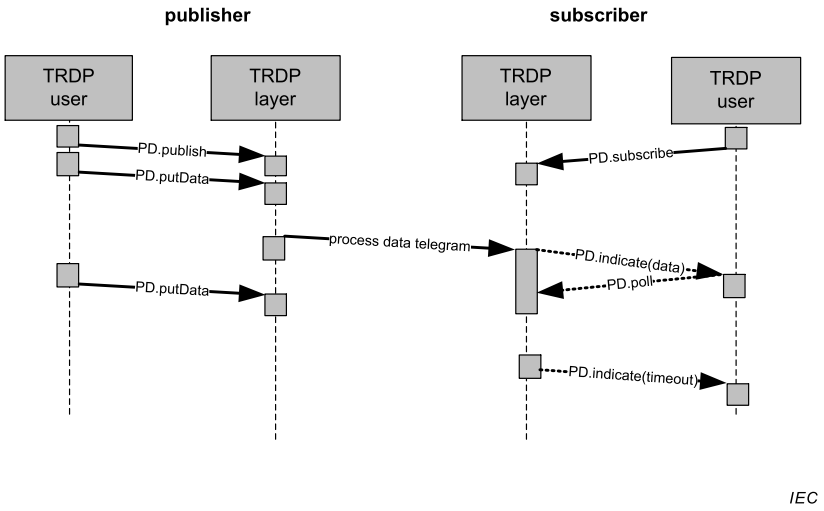


Figure A.13 – Interaction sequence chart for PD push pattern

A.6.6.5 PD Redundancy Handling

The sequence chart in Figure A.14 shows the sequence of the service primitives for redundancy handling of the push pattern.

The service primitives for redundancy handling shall be used in the same way by the publisher for the pull and the push pattern.

If a redundant device enters the redundancy leader state it shall call PD.activate to start publishing process data related to ComIds marked as redundant.

If a redundant device enters the redundancy follower state it shall call PD.deactivate to stop publishing process data related to ComIds marked as redundant.

Starting TRDP, publishers of redundant process data (identified by ComId marked as redundant) shall be initialized in redundancy follower mode (publishing deactivated).

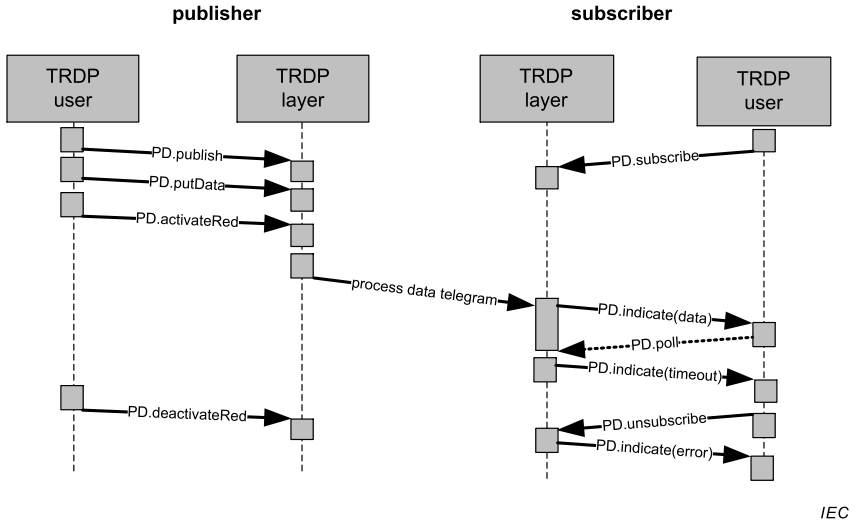


Figure A.14 – Interaction sequence chart for redundant PD handling

A.6.7 Topography counter check

Before sending a telegram, the topography counters of the telegram shall be checked against the actual topography counters to ensure that the sending application shares the actual train backbone view and operational train view.

After reception of a telegram, the topography counter values shall be checked to ensure that publisher/requester and subscriber share an identical train backbone view and operational train view. For this check, the topography counter values received are compared with the topography counter values submitted in the PD.subscribe service primitive.

The topography counter check is passed if at least one of the cases listed in Table A.5 came true.

Table A.5 – Topography counter check

Publisher	**Actual topography counter values**		**Locally stored topography counter values submitted with the publish**	
Subscriber	**Actual topography counter values**		**Topography counter values of the received telegram**	
	Topography counter values of the received telegram		**Locally stored topography counter values submitted with the subscription**	
Case	etbTopoCnt	opTrnTopoCnt	etbTopoCnt	opTrnTopoCnt
1	any	any	0	0
2	0	equal	0	equal
3	equal	any	equal	0
4	equal	equal	equal	equal
5 (1)	0	0	equal	equal

(1) Case applies only for subscriber comparing topography counter values of the received telegram with the locally stored topography counter values submitted with the subscription. Used to receive local telegrams.

Key:

0 = topography counter value set to 0 (don't care)
any = topography counter has any value in the range 0..(2^{32}–1)
equal = local and received topography counter values are valid (1..(2^{32}–1)) and identical

A.6.8 State Machine

A.6.8.1 Publishing PD-PDU(data)

A publisher shall publish PD-PDU(data) when a defined time cycle terminates (push pattern) and/or when it receives a request from a dedicated subscriber or an independent communication device.

Not to overload subscribers, the publisher of pushed PD-PDU shall apply a traffic shaping mechanism for equal distribution of the PD-PDU's over the time.

The publishing of PD-PDU(data) is defined with the state diagram depicted in Figure A.15.

NOTE If there is a pull request for a PD-PDU that is already published using the push pattern, the related reply is sent to the address given by the request independently of the push pattern address and cycle.

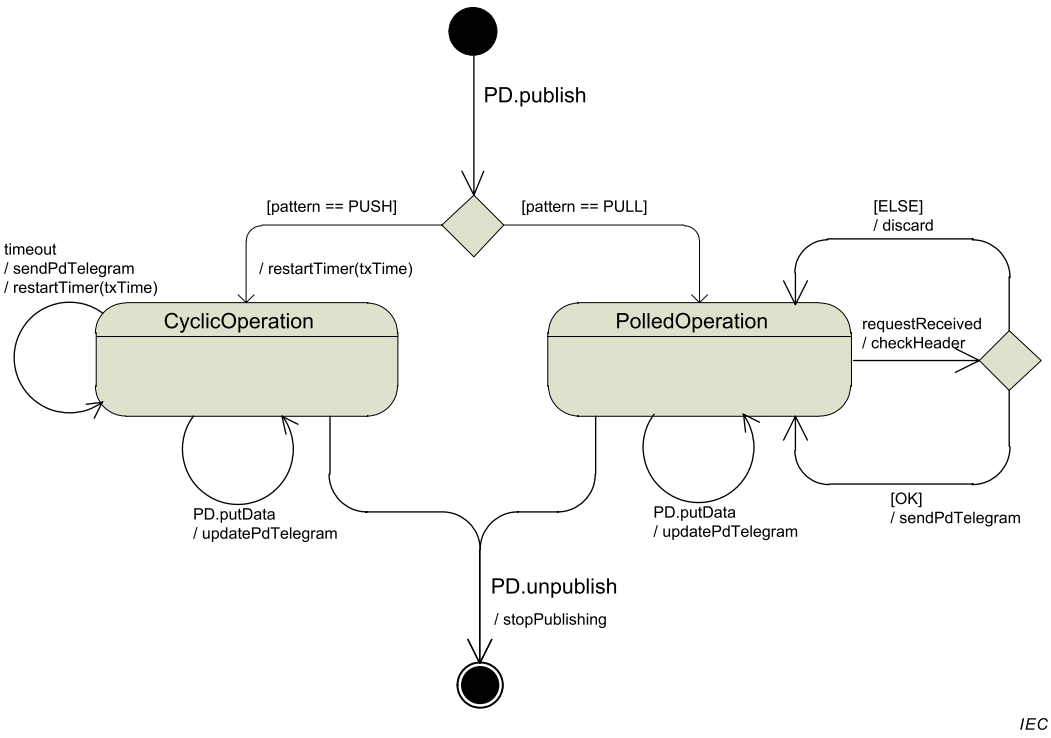


Figure A.15 – PD State diagram publisher

Guards, triggers, actions and states are described in Table A.6 to Table A.9.

Table A.6 – PD publisher state diagram – guards

Guards	Description
pattern	configured communication pattern for the PD-PDU exchange Values: PUSH PULL
checkTopoCounts	Check the locally stored topology counters submitted with the publish against the actual topography counters. At least one of the cases listed in Table A.5 for the topography counters is fulfilled.

Table A.7 – PD publisher state diagram – triggers

Triggers	Description
PD.publish	User called PD.publish service primitive
PD.unpublish	User called PD.unpublish service primitive
PD.putData	New process data packet has been prepared by the application for publishing (service primitive PD.putData)
requestReceived	A request telegram (type 'Pr') has been received
timeout	The time txTime expired

Table A.8 – PD publisher state diagram – actions

Actions	Description
updatePdTelegram	Take process data packet from application and generate process data telegram for publishing
checkHeader	Check validity of received PD request telegram: • HeaderFCS • ProtocolVersion Check existence of a publisher for that ComId If a publisher exists, check that at least one of the cases listed in Table A.5 for the topography counters is fulfilled. OK = all checks passed
sendPdTelegram	publish (send) the process data telegram
restartTimer(txTime)	restart the timer txTime = cycle time for sending PD as defined in PD.publish service primitive
stopPublishing	Stop publishing on request, stop the timer
discard	Discard the telegram

Table A.9 – PD publisher state diagram – states

States	Description
CyclicOperation	The publisher sends PD-PDUs cyclically triggered by a timer
PolledOperation	The publisher sends PD-PDUs triggered by a PD-PDU(request)

A.6.8.2 Request publishing of PD-PDU (reply)

One dedicated subscriber or an independent communication device may request one or many publishers to send their PD-PDUs. The requesting of PD-PDUs is defined with the state diagram depicted in Figure A.16.

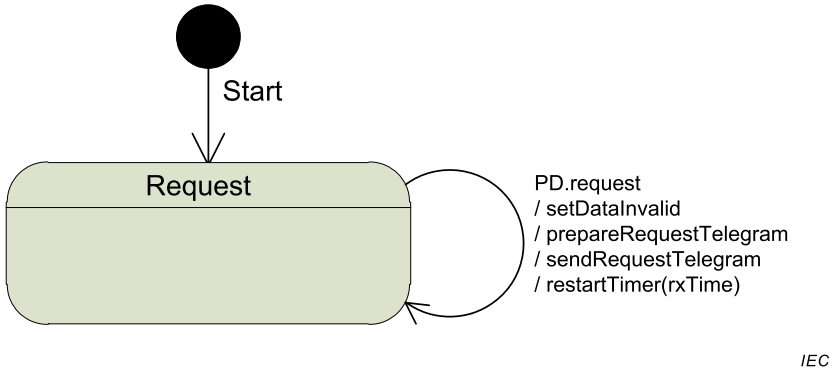


Figure A.16 – PD State diagram requester

Triggers, actions and states are described in Table A.10 to Table A.13.

Table A.10 – PD publisher state diagram – guards

Guards	Description
checkTopoCounts	Check the topography counters submitted with the request against the actual topography counters. At least one of the cases listed in Table A.5 for the topography counters is fulfilled.

Table A.11 – PD requester state diagram – triggers

Triggers	Description
PD.request	PD.request service primitive called by application

Table A.12 – PD requester state diagram – actions

Actions	Description
prepareRequestTelegram	prepare a request telegram to one or many publishers
sendRequestTelegram	send a request telegram to one or many publishers
restartTimer	(re-)start timer time = rxTime (timeout value given in PD.subscribe service primitive)
setDataInvalid	Sets the related data in the receive buffer to initialized/invalid.

Table A.13 – PD requester state diagram – states

States	Description
Request	The requester sends request telegram. Sending of the request telegram is triggered by the application using PD.request.

A.6.8.3 Receiving PD-PDU(reply/data)

See Figure A.17.

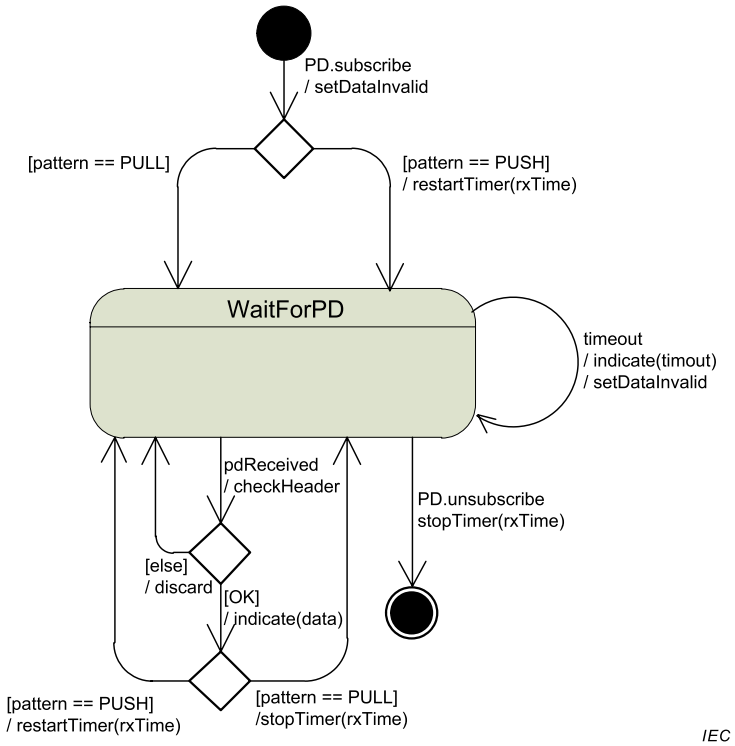


Figure A.17 – PD State diagram subscriber

Triggers, actions and states are described in Table A.14 to Table A.17.

Table A.14 – PD subscriber state diagram – triggers

Triggers	Description
PD.subscribe	PD.subscribe service primitive called by application
pdReceived	A process or request telegram has been received
Timeout	The rxTime expired
PD.unsubscribe	PD.unsubscribe service primitive called by application

Table A.15 – PD subscriber state diagram – guards

Triggers	Description
pattern	configured communication pattern for the PD-PDU exchange Values: PUSH PULL
checkTopoCounts	Check the topography counters of the received telegram against the topography counters submitted with the subscription. At least one of the cases listed in Table A.5 for the topography counters is fulfilled.

Table A.16 – PD subscriber state diagram – actions

Actions	Description
indicate(data)	Notify application about a new received process or request telegram (service primitive PD.indicate)
indicate(timeout)	Notify application about timeout (service primitive PD.indicate)
checkHeader	Check header for correct FCS, version and type. Check filter criteria OK = all checks passed
setDataInvalid	Sets the PD to initialized/invalid
stopTimer(timer)	Stop the timer which is related to the subscription
restartTimer(time)	restart the timer time = rxTime (timeout value given in PD.subscribe service primitive) The timer shall be started first time when the subscription is done. For PUSH pattern it shall be restarted with every received process data telegram. For PULL pattern it shall be restarted with each request.
discard	Discard the telegram

Table A.17 – PD subscriber state diagram – states

States	Description
WaitForPD	Wait until a new process telegram is received

A.7 Message Data

A.7.1 Communication model

The TRDP message data protocol defines the message data exchange between a caller and one or many repliers over a confirmed TRDP service. Since it is thought only for real-time message data, the length of a telegram is limited to 64 kBytes. The caller is sending a request message with or without user data to the replier(s) and the replier(s) shall send a reply message with or without user data in return, if required by the caller. The caller then shall quit reception of the reply by sending a confirmation if required by the replier.

NOTE This gives a “guaranteed” transfer of data as the caller/replier is notified about the correct reception and processing of the message data.

A caller shall be able to define by the request type whether a reply is expected or not.

A replier shall be able to define by the reply type whether a confirmation of its reply is expected or not.

As a consequence, three message data transfer options shall be provided by TRDP (Figure A.18):

a) request without reply (‘notification’);
b) request with reply but without confirmation (‘request without confirmation’);
c) request with reply and confirmation (‘request with confirmation’).

TRDP shall provide two mechanisms to transfer message data (via UDP and via TCP); the different service primitives of the two possibilities shall not be mixed.

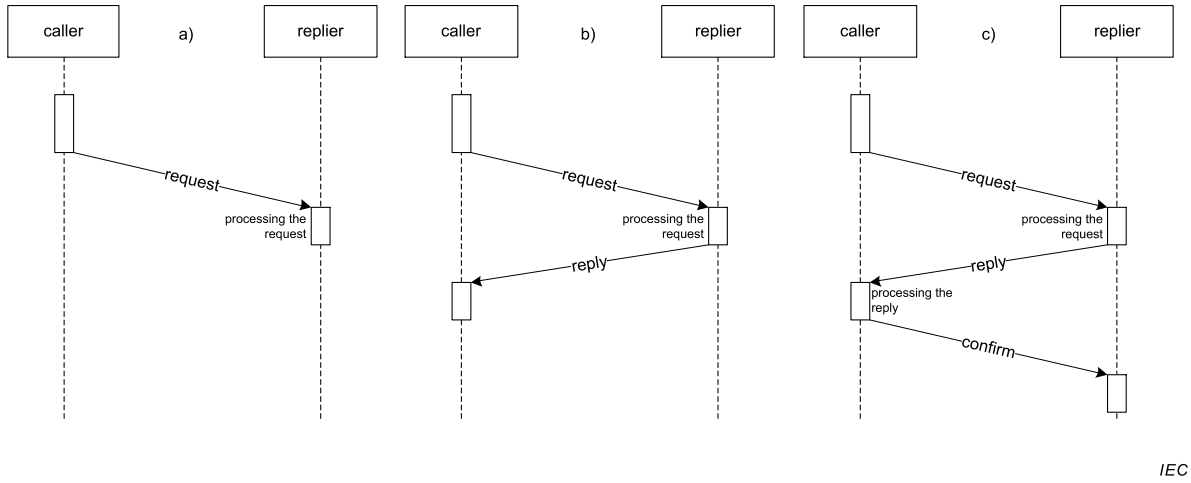


Figure A.18 – Message data transfer options

A.7.2 Roles

The communication partners in a message data transfer can have different roles:

caller: the source of message data in push communication pattern

the sink of message data in pull communication pattern

replier: the source of message data in pull communication pattern

the sink of message data in push communication pattern

A.7.3 Communication pattern

A.7.3.1 Push communication pattern

Message data exchange shall support the following push communication pattern as defined in IEC 61375-1 (for exceptions see note):

a) point to point , sporadic with acknowledge, source knows the sink;
b) point to point , sporadic without acknowledge, source knows the sink;
c) point to multipoint, sporadic with acknowledge, source knows the sink;
d) point to multipoint, sporadic without acknowledge, source knows the sink;
e) point to multipoint, sporadic with acknowledge, source does not know the sink;
f) point to multipoint, sporadic without acknowledge, source does not know the sink.

NOTE e) is not required in IEC 61375-1.

A.7.3.2 Pull communication pattern

Message data exchange shall support the following pull communication pattern as defined in IEC 61375-1 (for exceptions see note):

a) point to point , sporadic with acknowledge, sink knows the source;
b) point to point , sporadic without acknowledge, sink knows the source;
c) point to multipoint, sporadic with acknowledge, sink knows the source;
d) point to multipoint, sporadic without acknowledge, sink knows the source;

e) point to multipoint, sporadic on first acknowledge, sink does not know the source;

f) point to multipoint, sporadic without acknowledge, sink does not know the source.

NOTE e) is not required in IEC 61375-1.

A.7.4 Addressing

A caller shall use an IP unicast address or an IP multicast address for addressing known replier(s).

A caller shall use an IP multicast address for addressing unknown repliers.

A caller may use an IP multicast address for addressing a known replier redundancy group.

A replier shall respond to the caller's unicast address.

Address ranges which can be used for message data communication are defined in IEC 61375-2-5.

A.7.5 MD-PDU

The structure of a MD-PDU is defined in Figure A.19 and subsequently in ASN.1 notation, with additional explanation given in Table A.18.

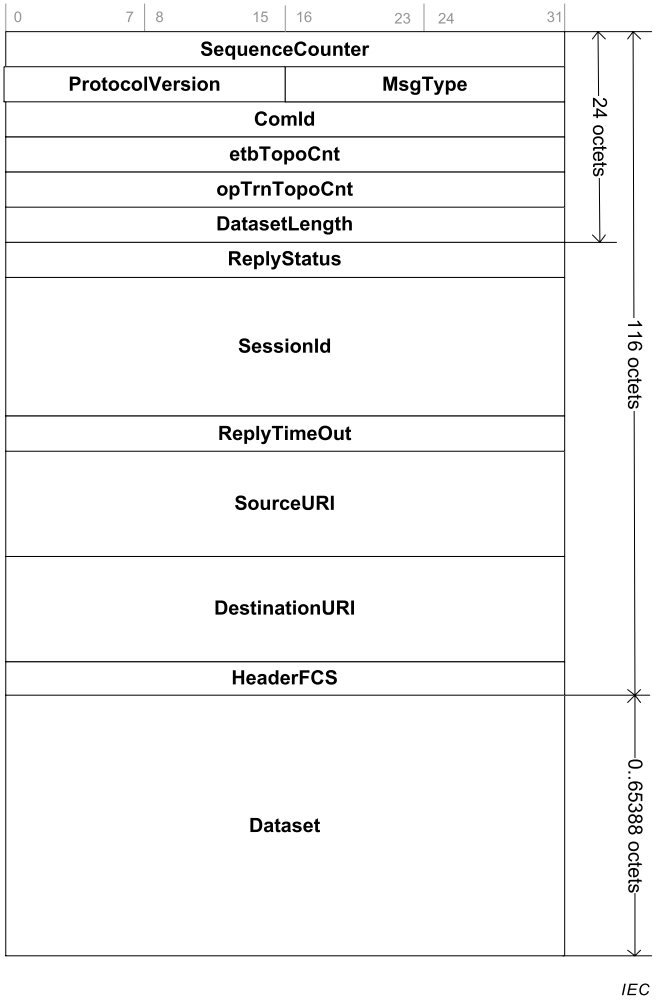


Figure A.19 – MD-PDU

```
MD_PDU::= RECORD
  {
  sequenceCounter  UINT32        -- message sequence counter
  protocolVersion  VERSION       -- protocol version
  msgType          UINT16        -- type of the message
  comId            UINT32        -- communication identifier
  etbTopoCnt       UINT32        -- etbTopoCnt value
  opTrnTopoCnt     UINT32        -- opTrnTopoCnt value
  datasetLength    UINT32        -- length of the array 'dataset'
  replyStatus      INT32         -- reply status indication
  sessionId        UINT32[4]     -- session identifier
  replyTimeout     UINT32        -- reply timeout
  sourceUri        CHAR[32]      -- source URI (user part)
  destinationUri   CHAR[32]      -- destination URI (user part)
  headerFcs        UINT32        -- header checksum
  dataset          ARRAY[]of UINT8    -- user data set
  }
```

Table A.18 – MD-PDU parameters

Parameter	Description	Value
sequenceCounter	The sequence counter • shall be incremented with each repetition of the message (return to 0 on overflow) • shall be returned with the reply message • shall start with value: 0	Computed
protocolVersion	The protocol version shall consist of: • Higher significant octet: MainVersion, incremented for incompatible changes, • Lower significant octet: SubVersion, incremented for compatible changes EXAMPLE – '0102'H = protocol version 1.2	Fixed
msgType	Type of the telegram. 'Mn' = Notification (Request without reply) 'Mr' = MD Request with reply 'Mp' = MD Reply without confirmation 'Mq' = MD Reply with confirmation 'Mc' = MD Confirm 'Me'= MD error NOTE Coded in ASCII for better reading with open protocol analyzer tools (e.g. Wireshark)	'4D6E'H ('Mn') '4D72'H ('Mr') '4D70'H ('Mp') '4D71'H ('Mq') '4D63'H ('Mc') '4D65'H ('Me')
comId	Identifier of the user dataset. See also A.5. Unspecified PDU (ComId=0) shall be supported. NOTE source IP address might be changed by NAT, see IEC 61375-2-5	set by user
etbTopoCnt	The topography counter: • shall be used (train addressing) as defined in standard part IEC 61375-2-5 (parameter 'etbTopoCnt') • shall be set by the user • shall be set for all communication over the ETB • shall be set if a valid opTrnTopoCnt is set	0..2^{32}-1 set by user

Parameter	Description	Value
	• optional in all other cases. Shall be set to 0 (= invalid) if not used.	
opTrnTopoCnt	The topography counter: • shall be used as defined in 5.3.3 • shall be set by the user • shall be set for communication between functions which use information from the operational train directory (e.g. side selective operations) • shall be set when the source device used the operational train directory to retrieve the destination IP address (e.g. resolving of URI 'vcu.leadVeh.anyCst.anyClTrn.lTrn') • optional in all other cases. Shall be set to 0 (= invalid) if not used.	
datasetLength	Length of the user data set in number of octets without padding octets	0 ..65388 Computed
replyStatus	The status value shall be set by the replier to report the execution result of a request message or by the caller sending a confirmation. The execution result is supplied by the replying application and transmitted to the requesting application in addition to the reply message itself. In case of a TRDP error reply ('Me') the value is supplied by TRDP. • –1 – reserved • –2 – session abort • –3 – no replier instance (at replier side) • –4 – no memory (at replier side) • –5 – no memory (local) • –6 – no reply • –7 – not all replies • –8 – no confirm • –9 – reserved • –10 – sending failed	< 0: NOK 0: OK > 0: user reply status
sessionId	The session identification: • shall identify a "request-reply" or a "request-reply-confirm" session which is composed of a caller session and a reply session. • shall identify a "request-error" session when the replier's TRDP layer returns an error message. • shall be computed at caller side and reused at replier (listener) side. • shall be used at caller side to relate a reply or error message to the original request message • shall be used at replier side to identify a retransmission of the request in case the reply message was not received and to identify the confirm message • shall be a UUID according to RFC 4122, time based version	Computed
replyTimeout	The reply timeout value shall be set to define the expected reply time in a request / reply session. Unit: 10^{-6} s (μs) Shall be set to 0 for 'Mn', 'Mp', 'Mc' and 'Me' type telegrams.	1 .. (2^{32}-1) 0 == infinite time
sourceURI	The source URI: • shall be the used for functional addressing	

Parameter	Description	Value
	• shall be a null terminated string • filling bytes at the end shall be set to 0 • shall contain only the “user part” without “host part” and “@” • may be an empty string (all 0)	
destinationURI	The destination URI: • shall be used for functional addressing • shall be a null terminated string • filling bytes at the end shall be set to 0 • shall contain only the “user part” without “host part” and “@” • may be an empty string (all 0)	
headerFCS	The header frame check sequence: • shall be calculated for the MD header • shall not include the HeaderFCS itself	Computed according to A.3
dataset	The user data. If the user data set length is not a multiple of 4 octets, octets with a value of 0 (zero) shall be appended until a multiple of 4 octets is reached (padding bytes).	

A.7.6 Interaction between application and TRDP layer

A.7.6.1 Service primitives

The TRDP layer shall provide services for message data communication to the TRDP user. The service primitives listed in Table A.19 and Table A.20 shall be provided.

NOTE The service primitives are defined in an abstract way in order to not restrict implementation. The specification of the service interface itself is not in the scope of this part of the standard.

Table A.19 – TRDP service primitives – Caller

Service primitive	Parameters	Direction/Description
MD.request		TRDP user → TRDP layer TRDP user asks to send a request message (notification only or requesting a reply)
	MsgType	• send notification message (‘Mn’) • send request message (‘Mr’)
	UserIdentifier	identifier returned with the MD.indication. Used by user to associate the MD.request with the MD.indication
	SessionId	session identifier used and returned by the TRDP layer.
	TransProtocol	UDP or TCP transport layer protocol
	NumRepliers	number of repliers if the request message is sent to known replier(s), e.g. ‘1’ if sent to one replier. Shall be set to ‘0’ if number of repliers is unknown or if notification message is sent.
	etbTopoCnt	as defined in Table A.18
	opTrnTopoCnt	as defined in Table A.18
	ComId	as defined in Table A.18

Service primitive	Parameters	Direction/Description
	DatasetLength	as defined in Table A.18
	Dataset	as defined in Table A.18
	SourceURI	as defined in Table A.18
	DestinationURI	as defined in Table A.18
	DestinationIpAddress	IP destination address, generated out of respective URI's using DNS
	ReplyTimeOut	as defined in Table A.18
	MaxNumRetries	maximum number of retries in case NumRepliers == 1. Value range 0..2.
MD.indicate		TRDP layer → TRDP user TRDP layer informs about an event (received data, timeout or error).
	IndType	• reception of a reply message ('Mp') • reception of a query message ('Mq') • TRDP layer error ('Me')
	UserIdentifier	identifier which was submitted with MD.request
	SessionId	session identifier used by the TRDP layer.
	NoOfRepl	Number of received replies
	etbTopoCnt	etbTopoCnt value obtained from received message. Set to 0 if unknown (error return)
	opTrnTopoCnt	opTrnTopoCnt value obtained from received message. Set to 0 if unknown (error return)
	ComId	as defined in Table A.18
	DatasetLength	as defined in Table A.18
	User Dataset	as defined in Table A.18
	SourceURI	as defined in Table A.18
	DestinationURI	as defined in Table A.18
	SourceIpAddress	IP source address
	DestinationIpAddress	IP destination address
	ReplyTimeOut	as defined in Table A.18
	ReplyStatus	as defined in Table A.18
MD.confirm		TRDP user → TRDP layer TRDP user asks to send a confirm message
	SessionId	session identifier as returned by MD.indication. Used by the TRDP layer to associate the MD.confirm with the session
	ReplyStatus	as defined in Table A.18
MD.abort		TRDP user → TRDP layer TRDP user asks to abort an open session e.g because of a TCN inauguration
	SessionId	session identifier used by the TRDP layer.

Table A.20 – TRDP service primitives – Replier

Service primitive	Parameters	Direction/Description
MD.addListener		TRDP user → TRDP layer TRDP user asks the TRDP layer to prepare for reception of notification/request messages (e.g. by joining defined MC groups and providing telegram buffer). Filters can be set for ComId, SourceIpAddress, SourceIpAddress range, DestinationIpAddress, SourceURI and DestinationURI.
	Handle	Handle for this listener, returned by the TRDP layer
	ComId	Notification/request with that ComId(s) to be accepted
	etbTopoCnt	actual etbTopoCnt value
	opTrnTopoCnt	actual opTrnTopoCnt value
	SourceURI[]	as defined in Table A.18, only notification/request from that source URI(s) to be accepted, only user part taken into account, host part only in case of group addressing See also A.7.6.3
	DestinationURI	as defined in Table A.18, only notification/request with that destination URI to be accepted, only user part taken into account, host part only in case of group addressing See also A.7.6.3
	SourceIpAddress	IP source address, generated out of respective URI's using DNS Defines the lower IP address in case of an IP address range (see 5.4.4.6.2)
	SourceIpAddress2	Defines the upper IP address in case of an IP address range (see 5.4.4.6.2) IP source address, generated out of respective URI's using DNS Set to 0 if not used
	DestinationIpAddress	IP destination address, generated out of respective URI's using DNS
	UserIdentifier	Identifier to be returned with MD.indicate
MD.updateListener		TRDP user → TRDP layer TRDP user asks the TRDP layer to update the listener parameters and eventually MC group memberships after a change of the train topology.
	Handle	Handle for this listener
	etbTopoCnt	actual etbTopoCnt value
	opTrnTopoCnt	actual opTrnTopoCnt value
	SourceIpAddress	IP source address, generated out of respective URI's using DNS Defines the lower IP address in case of an IP address range (see 5.4.4.6.2)
	SourceIpAddress2	Defines the upper IP address in case of an IP address range (see 5.4.4.6.2) IP source address, generated out of respective URI's using DNS Set to 0 if not used

Service primitive	Parameters	Direction/Description
	DestinationIpAddress	IP destination address, generated out of respective URI's using DNS
MD.remListener		TRDP user → TRDP layer TRDP user asks the TRDP layer to remove a listener
	Handle	Handle returned by the TRDP layer in MD.addListener
MD.indicate		TRDP layer → TRDP user TRDP layer informs about an event
	IndType	• reception of a request message ('Mr') • reception of a notification message ('Mn') • TRDP layer error ('Me')
	UserIdentifier	identifier which was submitted with MD.addListener
	SessionId	Session identifier of the session created by the TRDP layer
	etbTopoCnt	etbTopoCnt value obtained from received message. Set to 0 if unknown (error return)
	opTrnTopoCnt	opTrnTopoCnt value obtained from received message. Set to 0 if unknown (error return)
	ComId	opTrnTopoCnt value obtained from received message. Set to 0 if unknown (error return)
	DatasetLength	as defined in Table A.18
	Dataset	as defined in Table A.18
	SourceURI	as defined in Table A.18
	DestinationURI	as defined in Table A.18
	SourceIpAddress	IP source address
	DestinationIpAddress	IP destination address. Can also be an IP multicast address.
	ReplyTimeOut	as defined in Table A.18
	ReplyStatus	as defined in Table A.18
MD.reply		TRDP user → TRDP layer TRDP user asks to send a reply message
	MsgType	• send reply message ('Mp') • send query message ('Mq')
	SessionId	Session identifier as indicated with MD.indication. Used by the TRDP layer to associate the MD.reply with the reply session
	ComId	as defined in Table A.18
	DatasetLength	as defined in Table A.18
	UserDataset	as defined in Table A.18
	SourceURI	as defined in Table A.18
	SourceIpAddress	IP source address, generated out of respective URI's using DNS
	ReplyTimeOut	as defined in Table A.18
	ReplyStatus	as defined in Table A.18
MD.release		TRDP layer → TRDP user TRDP layer informs about a received confirm message,

Service primitive	Parameters	Direction/Description
		related timeout or error.
	MsgType	• reception of a confirm message (‘Mc’) • TRDP layer error (‘Me’)
	SessionId	Session identifier used by the TRDP layer.
	etbTopoCnt	etbTopoCnt value obtained from received message. Set to 0 if unknown (error return)
	opTrnTopoCnt	opTrnTopoCnt value obtained from received message. Set to 0 if unknown (error return)
	ComId	as defined in Table A.18
	SourceURI	as defined in Table A.18
	SourceIpAddress	IP source address
	ReplyStatus	as defined in Table A.18

A.7.6.2 Interaction sequence

The sequence chart in Figure A.20 defines the sequence of the service primitives.

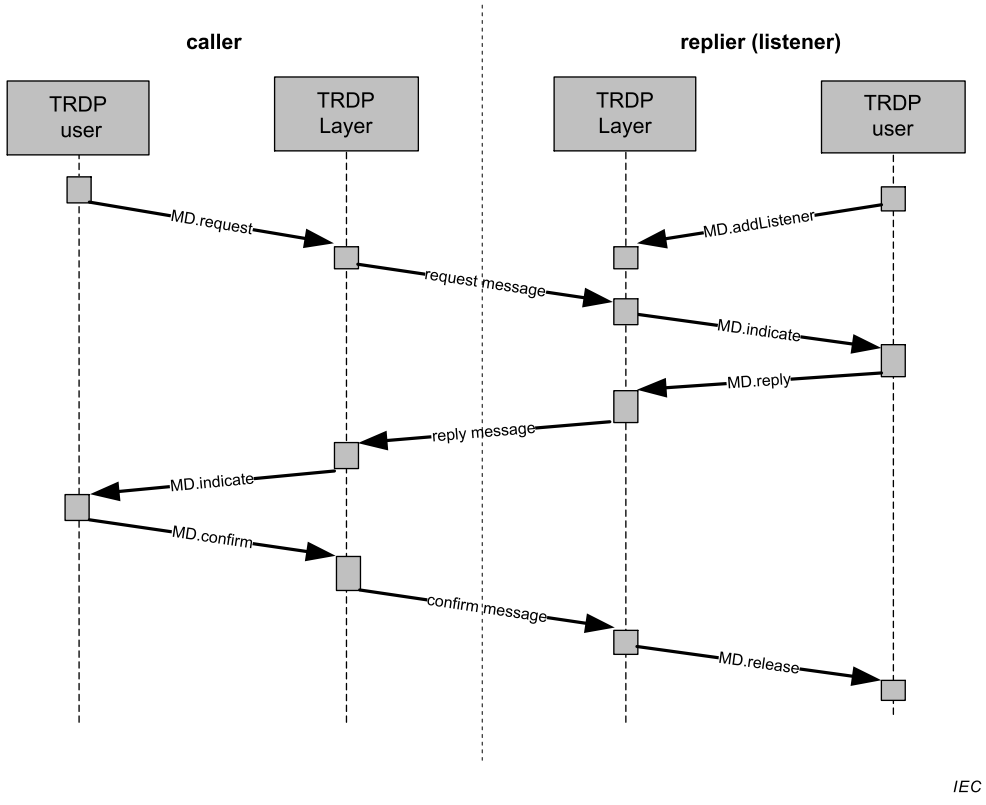


Figure A.20 – Interaction sequence chart

A.7.6.3 Filtering rules

The service primitive MD.addListener allows to optionally define SourceURI and DestinationURI for filtering received MD telegrams. For the filtering, the following rules shall apply:

a) An empty DestinationURI in a MD telegrams shall be received by a listener regardless of any specified DestinationURI.

b) An empty SourceURI in a MD telegrams has no special meaning.

c) An empty DestinationURI filter entry in MD.addListener service primitive means that the listener shall receive MD telegrams regardless of any defined DestinationURI in the MD telegrams.

d) An empty SourceURI filter entry in MD.addListener service primitive means that the listener shall receive MD telegrams regardless of any defined SourceURI in the MD telegrams.

e) Any two listeners shall have a disjunctive set of filter parameters ComId, SourceURI, DestinationURI, SourceIpAddress and DestinationIpAddress. If empty filter parameters are used, any two listeners have to differ in at least one non-empty filter parameter.

f) MD telegrams with empty DestinationURI may be received by several listeners with different DestinationURI filters on one device. TRDP has to handle the duplication of received data for the different listeners.

NOTE A MD telegram sent to an unicast IP destination address with empty DestinationURI is a potential multicast – multiple replies may be received.

A.7.7 Topography counter check

Before sending a telegram, the topography counters of the telegram shall be checked against the actual topography counters to ensure that the sending application shares the actual train backbone view and operational train view.

After reception of a telegram, the topography counter values shall be checked to ensure that caller and replier share an identical train backbone view and operational train view. For this check, the topography counter values received are compared first with the actual topography counter values and then with those submitted in the MD.addListener service primitive.

The topography counter check is passed if at least one of the cases listed in is true.

Table A.21 – Topography counter check

Caller	**Actual topography counter values**		**Topography counter values of the telegram to be sent**	
Replier	**Actual topography counter values**		**Topography counter values of the received telegram**	
	Topography counter values of the received telegram		**Locally stored topography counter values submitted with adding a listener**	
Case	etbTopoCnt	opTrnTopoCnt	etbTopoCnt	opTrnTopoCnt
1	any	any	0	0
2	0	equal	0	equal
3	equal	any	equal	0
4	equal	equal	equal	equal
5(1)	0	0	equal	equal

(1) Case applies only for replier comparing topography counter values of the received telegram with the locally stored topography counter values submitted with adding the listener. Used to receive local telegrams.

Key:

0 = topography counter value set to 0 (don't care)
any = topography counter has any value in the range 0..(2^{32}–1)
equal = local and received topography counter values are valid (1..(2^{32}–1)) and identical

A.7.8 MD protocol state machine

A.7.8.1 TRDP Layer – Caller

If a TRDP user (caller) requests sending a message, its topography counters shall be checked against the actual topography counters. If none of the cases listed in Table A.21 is TRUE, the message shall be discarded.

If a TRDP user (caller) requests sending a notification, a notification message (MsgType ‘Mn’) shall be sent. The session identifier of the notification message shall be set to 0.

If a TRDP user (caller) requests sending a request message, a caller session shall be opened and a request message (MsgType ‘Mr’) shall be sent.

To ensure that the session identifier is unique, each request-reply/request-reply-confirm session shall be identified by a 16 byte UUID according to RFC 4122, time based version. The session identifier is calculated at caller side in the TRDP layer, transmitted within each message and used at caller and replier side to identify the related caller and replier session.

A timeout value for the reply message(s) shall be defined by the TRDP user (caller).

NOTE If retries are selected, the timeout at application layer can be up to (maxNumRetries + 1) × replyTimeout.

During the caller session, the TRDP layer shall wait for incoming reply messages (MsgType ‘Mp’, ‘Mq’ or ‘Me’). All incoming reply messages related to the caller session (identified by the received session id) shall be given immediately to the TRDP user.

If the number of expected incoming replies is reached and there are no more outstanding confirmations from TRDP user (caller), the caller session shall be closed.

If the replier requests a confirmation (MsgType ‘Mq’), the TRDP layer shall start a timer with the given confirm timeout time of the reply and wait for the confirmation from the TRDP user (caller).

The confirmation shall use as destination URI the source URI received in the reply message.

After getting the confirmation from TRDP user (caller), the TRDP layer shall send a confirmation message (MsgType ‘Mc’) to the replier.

A confirmation message shall not contain user data and shall be sent only as unicast.

The TRDP user (caller) shall take care to provide the confirmation in time as indicated by the ReplyTimeOut parameter of the reply.

If the number of incoming replies is reached and the confirm timeout timer expires while waiting for outstanding confirmations from TRDP user (caller), the TRDP user (caller) shall be notified that confirmations are missing and the caller session shall be closed.

If the reply timeout timer of the caller session expires because of a missing reply and the number of expected repliers is 1, the TRDP layer shall, depending on the given parameter value ‘MaxNumRetries’, repeat the request up to two times before it notifies the TRDP user (caller) about the missing reply and closes the caller session.

If the reply timeout timer expires and the number of expected repliers is greater than 1 and less than the expected replies have been received, the TRDP Layer shall notify its TRDP user (caller) about the timeout and the number of missing replies. The caller session shall be closed when all requested confirmations are sent or the confirmation timeout timer has timed out.

If the number of repliers is not known (parameter NoOfRepliers = 0 and the reply timeout timer expires, the total number of replies shall be indicated to the TRDP user (caller) and the caller session shall be closed.

There shall be no retransmission of request messages if TCP is used. If for TCP retransmission or continuing of the transmission is wished, this shall be done in a higher level service or in the application. TRDP shall provide the interface to continue a transmission after the connection was lost.

If the TRDP layer receives a reply message (MsgType ‘Mp’, ‘Mq’ or ‘Me’) without having opened a caller session for the indicated session id, the message shall be discarded.

The topography counter values obtained during caller session opening shall be used throughout the session (for request and confirmation messages).

If a TRDP user receives a reply message with topography counter values different to the expected ones (see Table A.5), the message shall be discarded.

If a TRDP user aborts an (open) session (e.g. after a train topology change), running timers shall be stopped, the session identifier shall be destroyed and the session shall be closed.

The behavior of the caller’s TRDP layer is depicted in Figure A.21 with explanations given in Table A.22 to Table A.25.

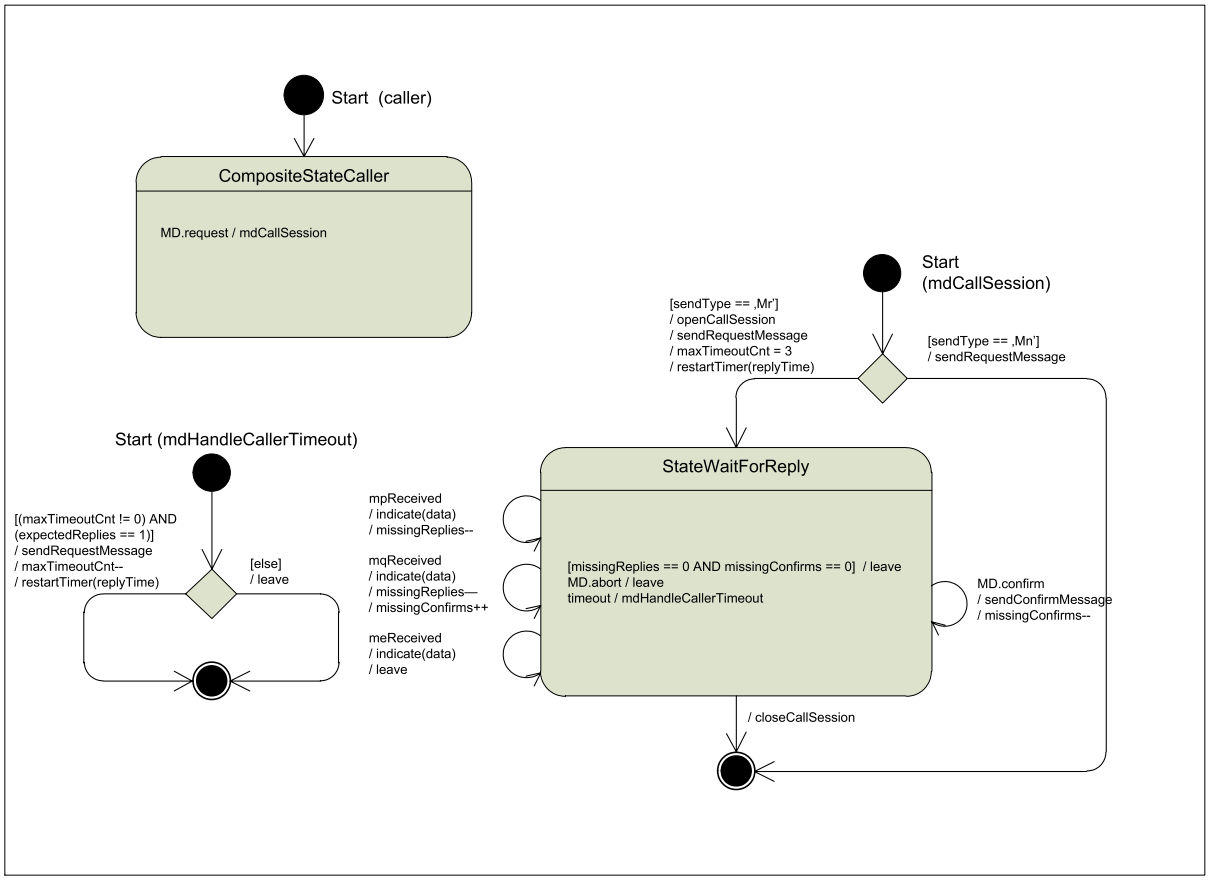


Figure A.21 – TRDP layer MD caller state chart

Table A.22 – MD caller state diagram – triggers

Triggers	Description
MD.request	User called MD.request service primitive
mpReceived	Valid reply without requested confirmation received, see A.7.8.3
mqReceived	Valid reply with requested confirmation received, see A.7.8.3
meReceived	Valid MD error message received, see A.7.8.3
MD.confirm	User called MD.confirm service primitive
MD.abort	User called MD.abort service primitive to close the open session (e.g. after a change of topography counters).
timeout	Reply timeout (no reply message received within replyTimeout)

Table A.23 – MD caller state diagram – guards

Guards	Description
sendType	Type of the MD request telegram as requested by the user in the MD.request service primitive
expectedReplies	Number of expected replies as indicated in MD.request service primitive (parameter ‘NoOfRepliers’)
missingReplies	Counter for missing replies
missingConfirms	Counter for missing confirms
maxTimeoutCnt	Counter for timeouts. Shall be decremented with each timeout.

Guards	Description
checkMessage	Check received message for correct CRC, protocol version, protocol type, length and topography counter (at least one of the cases listed in Table A.21 for the topography counters is fulfilled).
checkTopoCounts	Check the topopology counters of the received message against the actual topography counters. At least one of the cases listed in Table A.21 for the topography counters is fulfilled.

Table A.24 – MD caller state diagram – actions

Actions	Description
mdCallSession	Handles the states of a caller session.
openCallSession	Creates a caller session
closeCallSession	Closes the caller session
mdGetCallSessionState	Returns the state of the caller session for a given sessionId
sendRequestMessage	Sends the request message. If sending the request message is repeated after timeout telegram parameter 'SequenceCounter' shall be incremented.
sendConfirmMessage	Sends the confirm message
Indicate	Inform user about a received reply or error message by invoking MD.indicate service primitive
restartTimer(replyTime)	Start a timer with the user defined replyTimeout value (MD.request service primitive)
discard	discard received message without further processing

Table A.25 – MD caller state diagram – states

States	Description
CompositeStateCaller	Composite state of the caller
StateWaitForReply	State of a caller session that has sent out a request and is now waiting for the reply/replies and optional requested confirmations from the application

A.7.8.2 TRDP Layer – Replier (Listener)

Each TRDP user (replier) that wants to receive MD shall register as listener for MD sent to a specific URI (multicast or unicast) or for MD of a specific ComId.

All incoming messages shall be checked against the actual topography counters. Messages not fitting to the actual topography counters (see Table A.21) shall be discarded.

All incoming messages shall be checked against registered listeners. Any message to a not registered listener or to a listener expecting another value of the topography counters shall be discarded.

In case of a unicast request message to a not registered listener or to a listener expecting another value of the topography counters an error message (MsgType = 'Me') shall be sent indicating the error with replyStatus == -3 (no replier instance).

If a notification message (MsgType 'Mn') is received, the message shall be passed to the related listener.

If a request message (MsgType ‘Mr’) is received, a reply session shall be opened using the received session id, a timeout timer with the received reply timeout shall be started, the message shall be passed to the related listener and the TRDP layer shall wait for the MD.reply of the listener.

If, due to the lack of resources, a reply session can’t be opened, the received message shall be discarded and an error message (MsgType = ‘Me’) shall be sent, indicating the error with replyStatus == -4 (no memory (at replier side)).

If a request message (MsgType ‘Mr’) is received for an open reply session with the received session id while waiting for the MD.reply of the listener, the request message shall be discarded.

After receiving the MD.reply from the TRDP user (listener), the TRDP layer shall send, depending on MD.reply of the listener, a reply message without confirmation (MsgType = ‘Mp’) or a reply message with confirmation (MsgType = ‘Mq’) to the caller.

A reply message shall be sent only as unicast using source IP address and source URI of the related request message as destination IP address and destination URI.

The MD.reply of a listener shall use as source IP address the IP address of the listener and as source URI the unique URI (user part) of the listener.

If a reply message without confirmation (MsgType = ‘Mp’) was sent, the reply session shall be closed.

If a reply message with confirmation (MsgType = ‘Mq’) was sent, the timeout timer of the reply session shall be restarted using the given confirm timeout value.

If a request message (MsgType ‘Mr’) is received with a different sequence counter for an already open reply session, after the TRDP layer has received the MD.reply from the TRDP user (replier), the TRDP layer shall repeat sending the reply message (MsgType = ‘Mq’) to the caller with an incremented sequence counter. After resending the reply message the timeout timer of the reply session shall be restarted using the given confirm timout value.

If a request message (MsgType ‘Mr’) is received for an already open reply session and the same sequence counter like received before, the request message shall be discarded.

If a confirmation message (MsgType ‘Mc’) is received for an open reply session with the received session id, the related listener shall be notified and the session shall be closed.

If a confirmation message (MsgType ‘Mc’) is received without having an open reply session with the received session id, the message shall be discarded.

If the timeout timer of a reply session expires, the reply session shall be closed. In case of a timeout because of a missing confirmation or a missing MD.reply of the listener, the listener shall be notified.

The topography counter values obtained in an MD.addListener service primitive shall be used for upcoming reply sessions unless they are updated by the user (MD.updateListener service primitive).

If a TRDP user receives a request message with topography counter values different to the expected ones (see Table A.21), the message shall be discarded.

If a TRDP user removes or updates a listener (e.g. after a train topology change) during an open session, running timers shall be stopped, the session identifier shall be destroyed and the session shall be closed.

The behavior of the repliersr's TRDP layer is depicted in Figure A.22 with explanations given in Table A.26 until Table A.29.

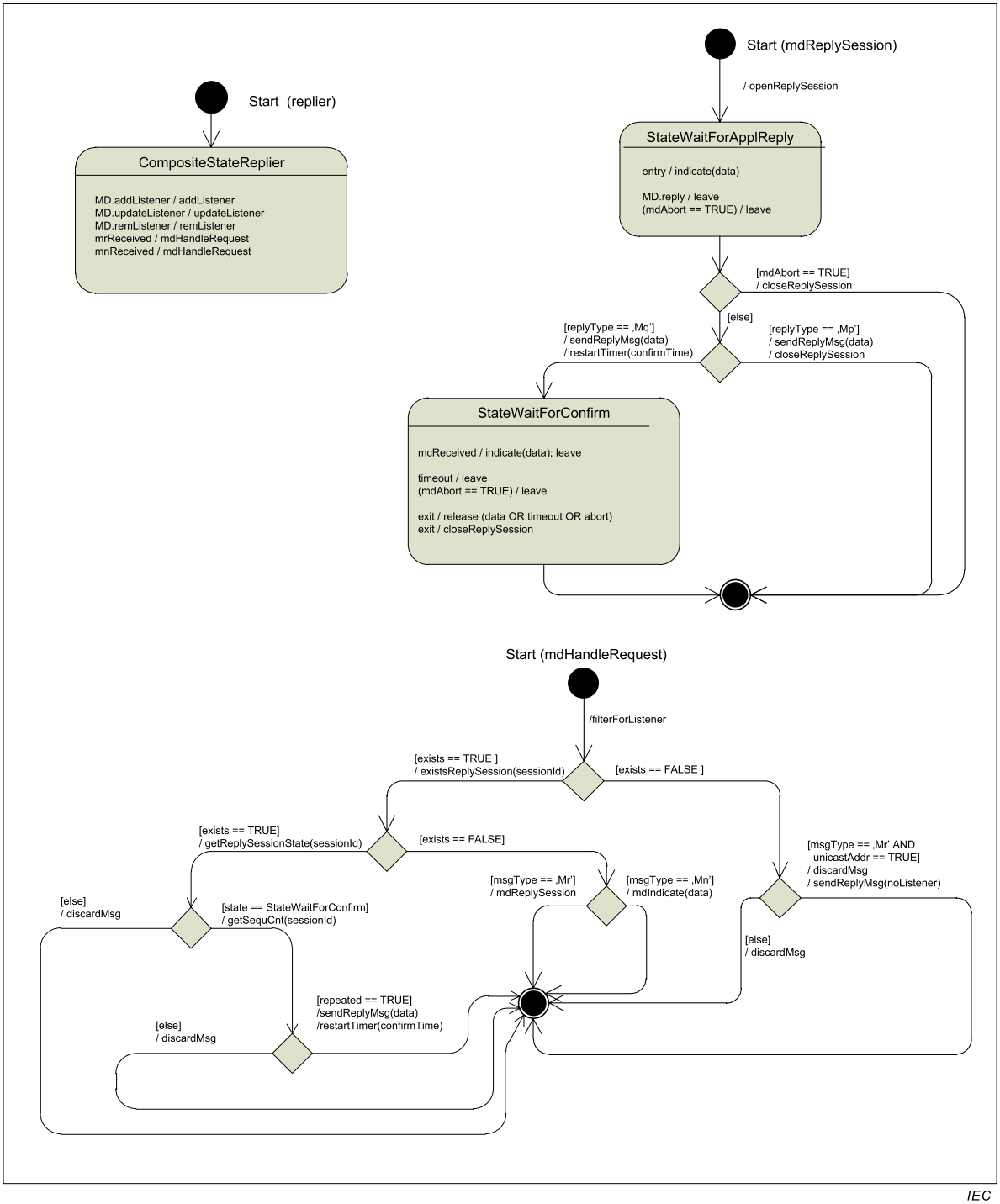


Figure A.22 – TRDP layer MD replier state chart

Table A.26 – MD replier state diagram – triggers

Triggers	Description
MD.addListener	User called MD.addListener service primitive
MD.updateListener	User called MD.updateListener service primitive
MD.remListener	User called MD.remListener service primitive
mrReceived	Valid request received, see A.7.8.3
mnReceived	Valid notification received, see A.7.8.3
mcReceived	Valid confirmation received, see A.7.8.3
MD.Reply	User called MD.reply service primitive
timeout	Confirm timeout (no confirm message received within replyTimeout)

Table A.27 – MD replier state diagram – guards

Guards	Description
existsReplySession (sessionId)	The reply session for a given session id exists
existsListener	A listener for the given request with fitting topography counters (see Table A.21) exists.
replyType	Type of MD reply telegram as requested by the user in the MD.reply service primitive
stateReplySession(sessionId)	replier session state for a given session id
repeated	Indication of request message repetition
unicastAddr	destination address is a unicast address
checkMessage	Check received message for correct CRC, protocol version, protocol type, length and topography counter (at least one of the cases listed in Table A.21 for the topography counters is fulfilled).
checkTopoCounts	Check the topopology counters of the received message against the actual topography counters (at least one of the cases listed in Table A.21 for the topography counters is fulfilled).

Table A.28 – MD replier state diagram – actions

Actions	Description
mdReplySession	Handles the states of a reply session.
mdHandleRequest	Handles the received request message.
openReplySession	Creates a reply session
closeReplySession	Closes the reply session
sendReplyMessage	Sends the reply message
indicate	Inform user about a received request message or a received confirmation message by invoking MD.indicate service primitive
release	Inform user about a received confirm message or a timeout by invoking MD.release service primitive
addListener	Add a listener for given filter criteria mdAbort = FALSE
updateListener	Update listener parameters mdAbort = TRUE
remListener	Remove a listener mdAbort = TRUE
getSequCnt(sessionId)	Checks if the request message has been repeated by the caller (is the case if the sequence counter of the previous received request

Actions	Description
	message is different to the actual received value) repeated == TRUE: sequence counters different repeated == FALSE: sequence counters equal
discardMsg	Discard the received message
restartTimer(confirmTime)	Start a timer with the user defined replyTimeout value (MD.reply service primitive)
sendReplyMsg(param)	Send reply message Param = ‘data’: – Reply type: ‘Mq’ OR ‘Mp’ (user defined) – Reply status: user defined – Reply ComId: user defined – Dataset: user defined
sendReplyErr(replyStatus)	Send error reply message Param = ‘noListener’ – Reply type: ‘Me’ – Reply status: replyStatus – Reply ComId: 0 – Dataset: empty

Table A.29 – MD replier state diagram – states

States	Description
CompositeStateReplier	Composite state of the replier
StateWaitForApplReply	State of a reply session that has sent a received request message requesting a reply to the application and is now waiting for the MD.reply of the application.
StateWaitForConfirm	State of a reply session that has sent out a reply requesting a confirmation and is now waiting for the confirmation.

A.7.8.3 Handling of received messages (action mdHandleReceived)

The state diagram shown in Figure A.23 with explanations given in Table A.30 until Table A.33 describes the handling of received message data telegrams

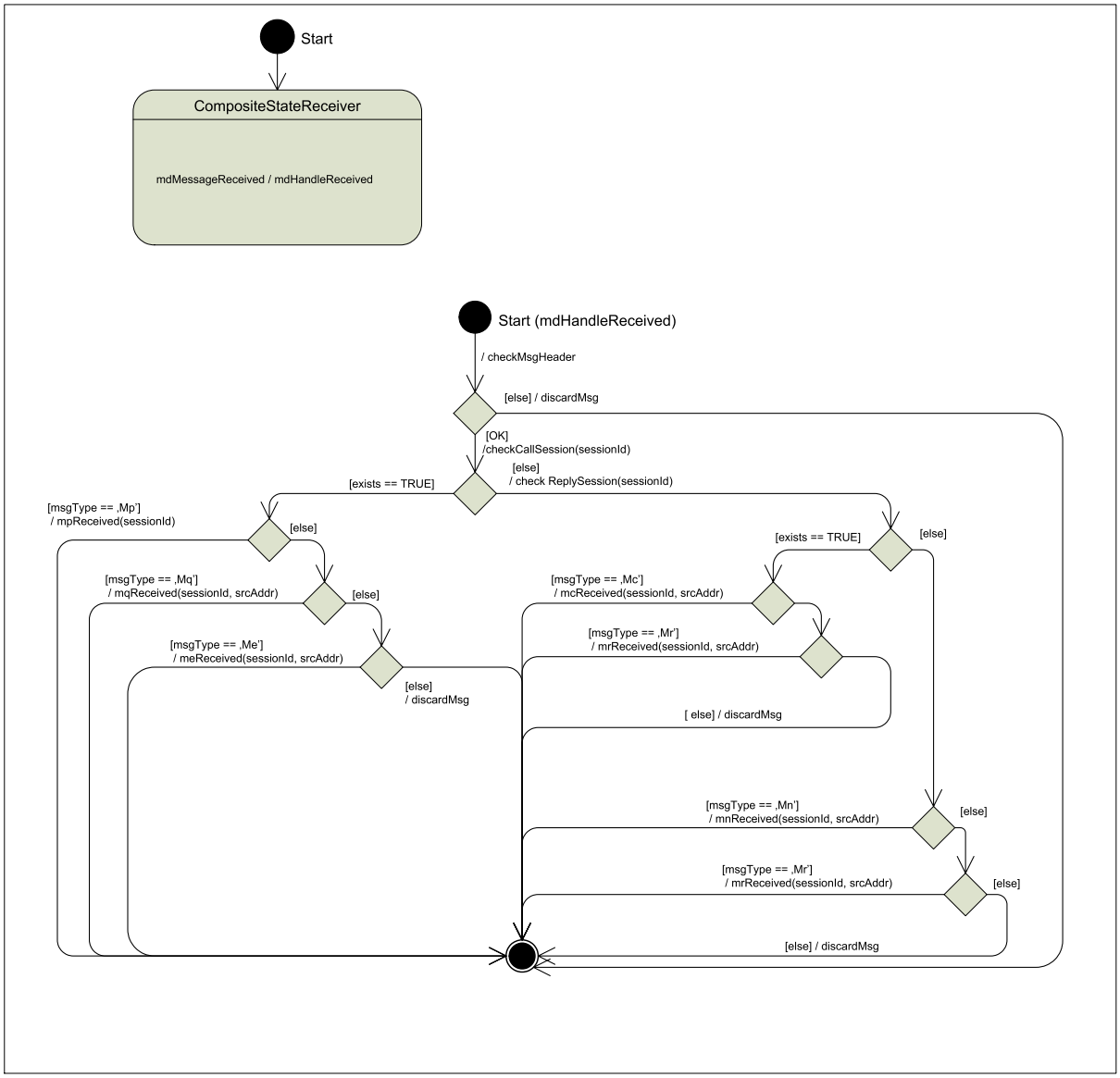


Figure A.23 – TRDP Layer MD telegram reception

Table A.30 – MD receiver state diagram – triggers

Triggers	Description
mdMessageReceived	A MD telegram has been received

Table A.31 – MD receiver state diagram – guards

Guards	Description
msgType	Message telegram type as defined in Table A.18

Table A.32 – MD receiver state diagram – actions

Actions	Description
checkCallSession	Check if a caller session to the given reply exists If a caller session exists, check that at least one of the cases listed in Table A.5 for the topography counters is fulfilled. exists = TRUE: both checks passed
checkReplySession	Check if a reply session to the given request or confirmation exists exists = TRUE: check positive NOTE There may be multiple reply sessions
checkMsgHeader	Check the MD telegram header for FCS, version and sequence counter and discard incorrect telegram
mpReceived	Trigger reception of Mn' type MD telegram
mqReceived	Trigger reception of 'Mq' type MD telegram
mcReceived	Trigger reception of 'Mc' type MD telegram
mrReceived	Trigger reception of 'Mr' type MD telegram
mnReceived	Trigger reception of 'Mn' type MD telegram
discardMsg	Discard the message

Table A.33 – MD receiver state diagram – states

States	Description
CompositeStateReceiver	Composite state of the receiver

A.7.9 TCP Connection Handling

This subclause defines the handling of TCP connections for TRDP message data transmission.

Using TCP for data transmission requires an established TCP connection between the caller and the replier device. Thus the first request to a specified device shall open a connection if there was not yet a connection established before.

As long as a TCP connection to another device in the required direction is open, it should be reused for all further calls to this device.

The caller shall close an existing TCP connection (active end) in the following cases:

- A signal that the TCP connection will be closed has been received.
- TRDP shut down or re-initialization.
- A timeout occurred because the TCP connection has not been used for a defined time.

The replier shall use the connection opened by the caller.

The replier shall close an existing TCP connection (passive end) in the following cases:

- A signal that the TCP connection will be closed has been received.
- TRDP shut down or re-initialization.
- Another TCP connection was opened from the same caller device and the old connection is not used anymore for a defined time.

A.8 Message data echo server (option)

End devices with TRDP message data support can provide a message data echo function. A system that wants to test communication with an end device sends an echo request (ERQ) to the end device by using the MD.request service primitive. The end device activates an echo function that creates an echo reply (ERP), copies data from the ERQ to the ERP and then returns it to the sender.

The Message Data Echo should be implemented in the TRDP layer.

When message data arrive with the echo ComId (see Table A.2) an echo message is created and returned to the source. The user data of the returned message shall be a copy of the received user data.

Annex B
(normative)

Safe Data Transmission (SDTv2)

B.1 General

This annex defines an extension to the communication profile for the transmission of safety related (vital) data over ETB.

B.2 Overview of SDTv2 (informal)

SDTv2 provides a safe communication path between a source of safety related (vital) data (SDSRC) and one or several sinks of those data (SDSINK). This safe communication path is called "SDTv2 channel". Figure B.1 provides a logical model of such a SDTv2 channel. The SDTv2 channel starts and ends at safe applications, and covers the entire path along from the SDSRC application to the SDSINK application(s). This includes the train communication network (TCN) as well as the communication layers related to the TCN which reside on the safe end devices (ED-S) hosting the safe application, and which both are considered not trustable from a safety point of view. A safe end device may contain one or a number of SDSRC or SDSINK, respectively. SDSRC and SDSINK itself can be described as a composition of the safe application and the SDTv2 protocol layer. The SDTv2 protocol layer provides two interfaces: the communication channel interface and the SDTv2 application interface. The communication channel interface is defined by the communication technology underneath (TRDP). This is the interface where SDTv2 protocol data units, called VDPs (= Vital Data Packets) are sent to or received from the TCN. The SDTv2 application interface, which is product specific, is the place where safety related process data are put to or get from the application. The SDTv2 Layer at SDSRC side has mainly the task to add protocol information, in such a way forming a VDP, which is necessary for a safe transfer of those data over the network, prior to sending. On receiver side, the SDTv2 layer validates received VDPs, and if validation is successful, contained vital data are exposed in the SDTv2 application interface.

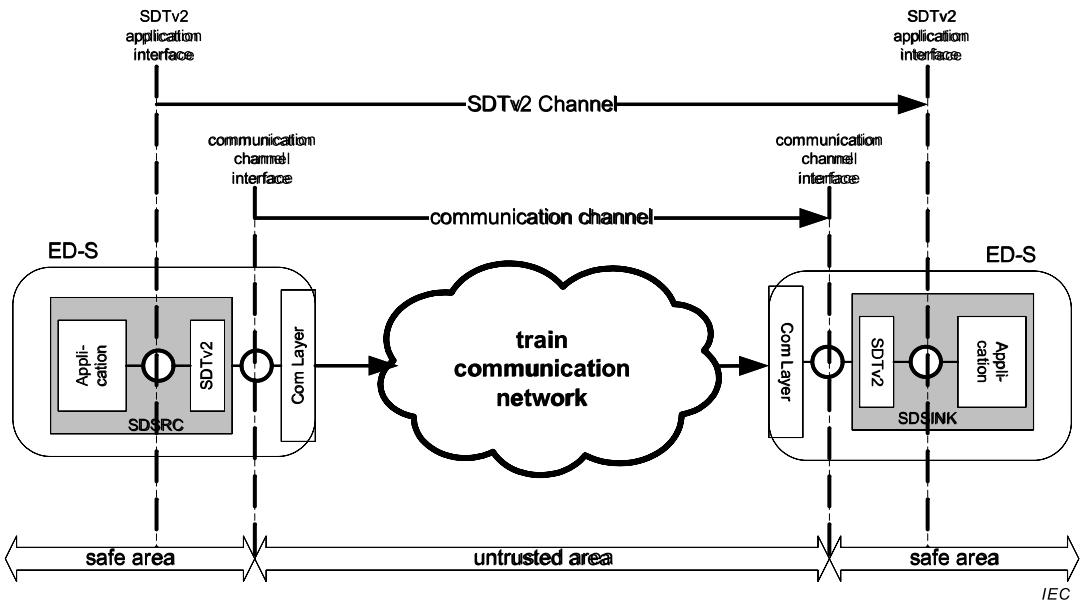


Figure B.1 – SDTV2 Channel

VDPs are specified for cyclic transmission, e.g. transported within the payload of TRDP process data.

B.3 Safety functional requirements

The following requirements have been mandatory for the development of the SDTv2 protocol:

a) Safety communication and standard (regular) communication shall be independent. However, standard devices and safety devices shall be able to use the same communication channel.

b) Safety communication shall be suitable for Safety Integrity Level SIL2 for continuous mode of operation (see IEC 61508-1).

c) Safety communication shall use a single-channel communication system. Redundancy may only be used optionally for increased availability.

d) Implementation of the safe transmission protocol shall be restricted to the communication end devices.

e) Due to the dynamic nature of train compositions with a varying number of consists, a 1:n communication relationship between the safe data source and safe data sinks has to be supported.

f) The transmission duration times shall be monitored.

g) Environmental conditions shall be according to general railway requirements, mainly IEC 60571, if there are no particular product standards.

h) Transmission equipment such as controllers, ASICs, Ethernet switch cores, cables, couplers, etc., shall remain unmodified (black channel). The safety functions shall be above OSI layer 7 (i.e. profile, no standard protocol changes or enhancements).

i) The safety communication shall not reduce the permitted number of devices.

j) The safety communication shall support data exchange over the ETB in dynamically changing train compositions.

B.4 Safety measures

The safety measures listed in Table B.1 for mastering possible transmission errors are one significant component of the SDTv2.

The safety measures shall be processed and monitored within one safety device.

Table B.1 – Deployed measures to communication errors

Communication Error	Safety measures						
	Safe Sequence Counter	Sink-time supervision	Safety Code	Guard time	Source Identi-fier (SID)	Latency Monitoring	Channel Monitoring
Corruption			X				
Unintended repetition	X						
Incorrect sequence	X						
Loss		X					
Unacceptable delay		X				X	
Insertion	X				X		
Masquerade	X				X		
Addressing					X		
Redundancy failure (active redundant sources)				X			
Unacceptable communication channel error rate		X	X				X

B.5 Operational states of the SDTv2 channel

A SDTv2 channel can be in two basic operational states (see Figure B.2):

State RegularCommunication: In this state, transmitted vital process data cannot be considered to be safe

State SafeCommunication: In this state, transmitted vital process data can be considered to be safe

The conditions under which state changes occur are specified in the following subclauses.

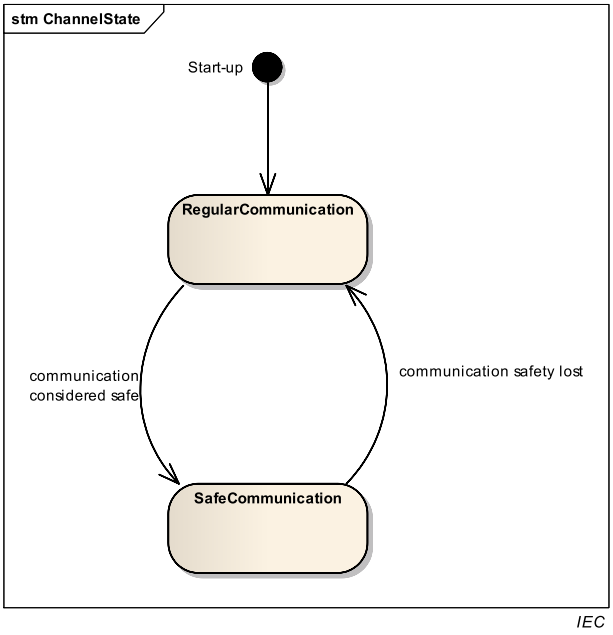


Figure B.2 – SDTv2 Channel States

B.6 Data presentation

All data within SDTv2 shall be transmitted in big-endian format (most significant octet of a data item first).

All data structures used within SDTv2 shall be naturally aligned (data items stored at offset address which is a multiple of their size).

Variable length data structures (open arrays, records) shall not be used.

The VDP data elements of a structured type (record, sequence) shall be arranged in the order they are declared.

B.7 SC-32

SC-32 defines a cyclic redundancy code used for the computation of SID and SafetyCode. The polynomial to be used is '1F4ACFB13' as defined in IEC 61784-3-3. The algorithm in C programming language notation depicted in Figure B.3 (and defined in IEC 61784-3-3) for SC-32 computation should be used, see also Figure B.4.

```
/*****************************************************************************
 *  GLOBAL FUNCTION DEFINITIONS
 */

/**
 * @internal
 * Calculates and returns a 32-bit CRC.
 *
 * @param buf   Input buffer
 * @param len   Length of input buffer
 * @param crc   Initial (seed) value for the CRC calculation
 *
 * @return Calculated CRC value
 */
static UNSIGNED32 sdt_crc32(const UNSIGNED8 buf[], UNSIGNED32 len, UNSIGNED32 crc)
{
    UNSIGNED32 i;

    for (i = 0; i < len; i++)
    {
      crc = crctab32[((UNSIGNED32)(crc >> 24)^buf[i])&0xff]^(crc << 8);
    }

    return crc;
}
```

Figure B.3 – SC-32 Computation

```
/*
 * Table for 32-bit CRC calculation according to IEC 61784-3-3 (2007)
 * CRC of the string "123456789" is 0x1697d06a
 */
static const UNSIGNED32 crctab32[256]={
    0x00000000U, 0xF4ACFB13U, 0x1DF50D35U, 0xE959F626U,
    0x3BEA1A6AU, 0xCF46E179U, 0x261F175FU, 0xD2B3EC4CU,
    0x77D434D4U, 0x8378CFC7U, 0x6A2139E1U, 0x9E8DC2F2U,
    0x4C3E2EBEU, 0xB892D5ADU, 0x51CB238BU, 0xA567D898U,
    0xEFA869A8U, 0x1B0492BBU, 0xF25D649DU, 0x06F19F8EU,
    0xD44273C2U, 0x20EE88D1U, 0xC9B77EF7U, 0x3D1B85E4U,
    0x987C5D7CU, 0x6CD0A66FU, 0x85895049U, 0x7125AB5AU,
    0xA3964716U, 0x573ABC05U, 0xBE634A23U, 0x4ACFB130U,
    0x2BFC2843U, 0xDF50D350U, 0x36092576U, 0xC2A5DE65U,
    0x10163229U, 0xE4BAC93AU, 0x0DE33F1CU, 0xF94FC40FU,
    0x5C281C97U, 0xA884E784U, 0x41DD11A2U, 0xB571EAB1U,
    0x67C206FDU, 0x936EFDEEU, 0x7A370BC8U, 0x8E9BF0DBU,
    0xC45441EBU, 0x30F8BAF8U, 0xD9A14CDEU, 0x2D0DB7CDU,
    0xFFBE5B81U, 0x0B12A092U, 0xE24B56B4U, 0x16E7ADA7U,
    0xB380753FU, 0x472C8E2CU, 0xAE75780AU, 0x5AD98319U,
    0x886A6F55U, 0x7CC69446U, 0x959F6260U, 0x61339973U,
    0x57F85086U, 0xA354AB95U, 0x4A0D5DB3U, 0xBEA1A6A0U,
    0x6C124AECU, 0x98BEB1FFU, 0x71E747D9U, 0x854BBCCAU,
    0x202C6452U, 0xD4809F41U, 0x3DD96967U, 0xC9759274U,
    0x1BC67E38U, 0xEF6A852BU, 0x0633730DU, 0xF29F881EU,
    0xB850392EU, 0x4CFCC23DU, 0xA5A5341BU, 0x5109CF08U,
    0x83BA2344U, 0x7716D857U, 0x9E4F2E71U, 0x6AE3D562U,
    0xCF840DFAU, 0x3B28F6E9U, 0xD27100CFU, 0x26DDFBDCU,
    0xF46E1790U, 0x00C2EC83U, 0xE99B1AA5U, 0x1D37E1B6U,
    0x7C0478C5U, 0x88A883D6U, 0x61F175F0U, 0x955D8EE3U,
    0x47EE62AFU, 0xB34299BCU, 0x5A1B6F9AU, 0xAEB79489U,
    0x0BD04C11U, 0xFF7CB702U, 0x16254124U, 0xE289BA37U,
    0x303A567BU, 0xC496AD68U, 0x2DCF5B4EU, 0xD963A05DU,
    0x93AC116DU, 0x6700EA7EU, 0x8E591C58U, 0x7AF5E74BU,
    0xA8460B07U, 0x5CEAF014U, 0xB5B30632U, 0x411FFD21U,
    0xE47825B9U, 0x10D4DEAAU, 0xF98D288CU, 0x0D21D39FU,
    0xDF923FD3U, 0x2B3EC4C0U, 0xC26732E6U, 0x36CBC9F5U,
    0xAFF0A10CU, 0x5B5C5A1FU, 0xB205AC39U, 0x46A9572AU,
    0x941ABB66U, 0x60B64075U, 0x89EFB653U, 0x7D434D40U,
    0xD82495D8U, 0x2C886ECBU, 0xC5D198EDU, 0x317D63FEU,
    0xE3CE8FB2U, 0x176274A1U, 0xFE3B8287U, 0x0A977994U,
    0x4058C8A4U, 0xB4F433B7U, 0x5DADC591U, 0xA9013E82U,
    0x7BB2D2CEU, 0x8F1E29DDU, 0x6647DFFBU, 0x92EB24E8U,
    0x378CFC70U, 0xC3200763U, 0x2A79F145U, 0xDED50A56U,
    0x0C66E61AU, 0xF8CA1D09U, 0x1193EB2FU, 0xE53F103CU,
    0x840C894FU, 0x70A0725CU, 0x99F9847AU, 0x6D557F69U,
    0xBFE69325U, 0x4B4A6836U, 0xA2139E10U, 0x56BF6503U,
    0xF3D8BD9BU, 0x07744688U, 0xEE2DB0AEU, 0x1A814BBDU,
    0xC832A7F1U, 0x3C9E5CE2U, 0xD5C7AAC4U, 0x216B51D7U,
    0x6BA4E0E7U, 0x9F081BF4U, 0x7651EDD2U, 0x82FD16C1U,
    0x504EFA8DU, 0xA4E2019EU, 0x4DBBF7B8U, 0xB9170CABU,
    0x1C70D433U, 0xE8DC2F20U, 0x0185D906U, 0xF5292215U,
    0x279ACE59U, 0xD336354AU, 0x3A6FC36CU, 0xCEC3387FU,
    0xF808F18AU, 0x0CA40A99U, 0xE5FDFCBFU, 0x115107ACU,
    0xC3E2EBE0U, 0x374E10F3U, 0xDE17E6D5U, 0x2ABB1DC6U,
    0x8FDCC55EU, 0x7B703E4DU, 0x9229C86BU, 0x66853378U,
    0xB436DF34U, 0x409A2427U, 0xA9C3D201U, 0x5D6F2912U,
    0x17A09822U, 0xE30C6331U, 0x0A559517U, 0xFEF96E04U,
    0x2C4A8248U, 0xD8E6795BU, 0x31BF8F7DU, 0xC513746EU,
    0x6074ACF6U, 0x94D857E5U, 0x7D81A1C3U, 0x892D5AD0U,
    0x5B9EB69CU, 0xAF324D8FU, 0x466BBBA9U, 0xB2C740BAU,
    0xD3F4D9C9U, 0x275822DAU, 0xCE01D4FCU, 0x3AAD2FEFU,
    0xE81EC3A3U, 0x1CB238B0U, 0xF5EBCE96U, 0x01473585U,
    0xA420ED1DU, 0x508C160EU, 0xB9D5E028U, 0x4D791B3BU,
    0x9FCAF777U, 0x6B660C64U, 0x823FFA42U, 0x76930151U,
    0x3C5CB061U, 0xC8F04B72U, 0x21A9BD54U, 0xD5054647U,
    0x07B6AA0BU, 0xF31A5118U, 0x1A43A73EU, 0xEEEF5C2DU,
    0x4B8884B5U, 0xBF247FA6U, 0x567D8980U, 0xA2D17293U,
    0x70629EDFU, 0x84CE65CCU, 0x6D9793EAU, 0x993B68F9U
};
```

Figure B.4 – SC-32 Table

B.8 SID

All sources of safety related data (SDSRC) shall be identified by a source identifier (SID). The SID shall be an UNSIGNED32 value which is computed as a SC-32 signature of the following data structure, as shown in Figure B.5. The initial (seed) value shall be 'FFFFFFFF'H.

```
SID_STRUCT::= RECORD
  {
   SMI             UINT32          -- user defined safety message
                                      identifier. Shall be unique within
                                      a consist.
                                      value: 1..'FFFFFFFF'H (0 = reserved)

   reserved01      UINT16          -- reserved, shall be set to 0.

   SDTProtVers     UINT16          -- version of the SDTv2 protocol.
                                      shall be set to '0002'H

   cstUUID         ARRAY[16] OF UINT8
                                   -- unique consist identifier

  SafeTopoCount    UINT32          -- safe topography counter (STC), unique
                                      identification of the actual train
                                      composition. Provided by the
                                      communication subsystem. To be set
                                      to 0 for consist network internal
                                      communication

  reserved02       UINT32          -- reserved, shall be set to 0.
  }
```

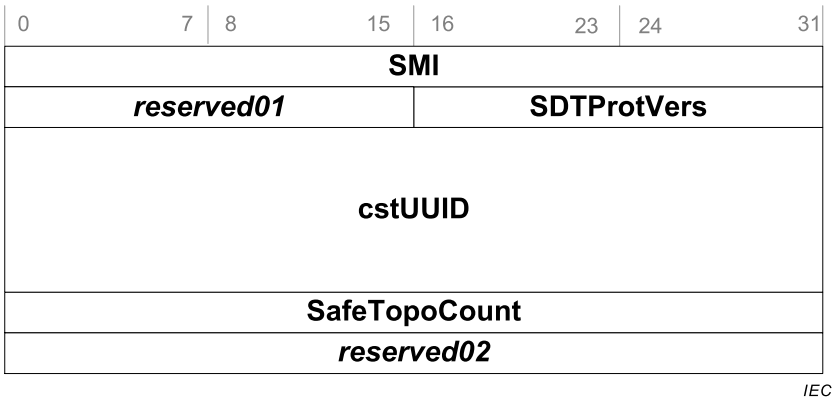


Figure B.5 – SID Computation

The SafeTopoCount value shall equal the opTrnTopoCnt value as specified in 5.3.3.2.13 for train wide communication over ETB unless otherwise specified.

SMI values from 1 until 999 shall be reserved for generic VDPs which are specified within this part of the standard.

SDSINKs can compute the SIDs of expected VDPs with information retrieved from the TTDB. Solely the SMI needs to be preconfigured

EXAMPLE A SDSINK expecting VDPs with SMI = 4861 from the leading vehicle look up the CstUUID of the leading vehicle and the opTrnTopoCnt in the TTDB and compute the SID.

SDSINKs expecting VDPs from a redundant SDSRC have to compute two SIDs because the SMIs must be different, see B.12.3.2.

B.9 Vital Data Packet

A SDSRC shall encapsulate safety critical data in a vital data packet (VDP). VDPs transferred over ETB (ETB-VDP) are transmitted within the user data part of a TRDP process data telegram.

A VDPs may also be transmitted within the user part of a TRDP message data telegram, in which case the restrictions listed in X shall be considered.

The ETB-VDP is defined as shown in Figure B.6:

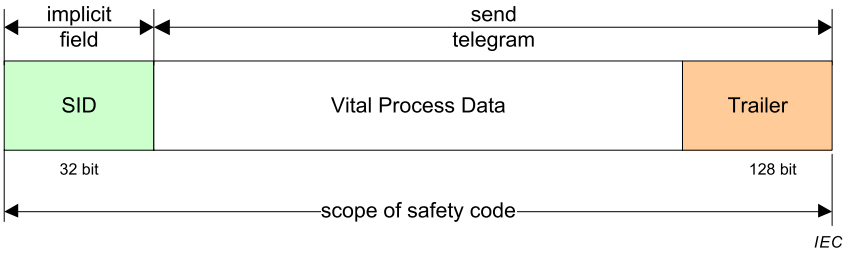


Figure B.6 – ETB-VDP

The ETB-VDP shall be represented by the following data structure, as shown in Figure B.7:

```
ETB_VDP::= RECORD
  {
  VitalProcessData      ARRAY [VDPSize] of UINT8
                                -- user defined data set
                                   with safe and non-safe process data
                                   VDPsize = 0 .. 984 (padding with
                                   trailing bytes may be necessary to
                                   have a total
                                   length of a multiple of 32 bits).

  reserved01       UINT32       -- reserved, shall be set to 0.

  reserved02       UINT16       -- reserved, shall be set to 0.
  UserDataVersion  VERSION      -- version of the vital process data part

  SafeSequCount    UINT32       -- safe sequence counter (SSC),
                                   implemented as a counter which is
                                   incremented each time a new VDP
                                   is generated and stored to the
                                   communication channel interface

  SafetyCode       UINT32       -- SC-32, computed over VDP starting
                                   with most significant byte of
                                   VitalProcessData up to the
                                   SafeSequCount.
                                   Seed value: SID
  }
```

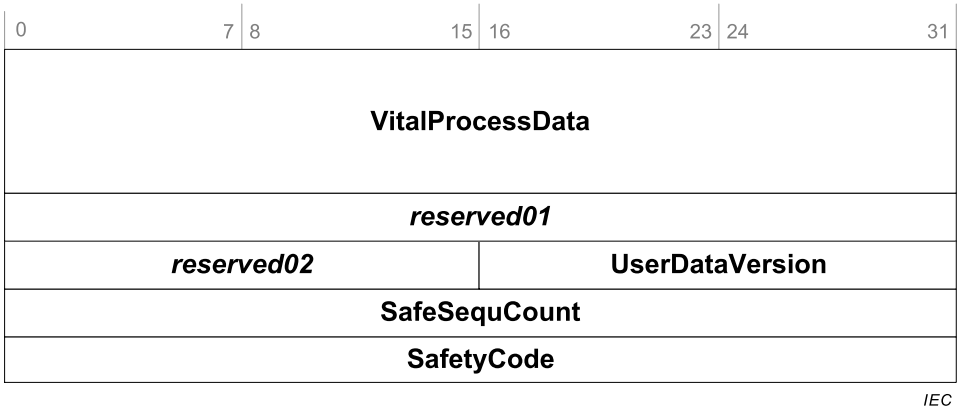


Figure B.7 – Format of ETB-VDP

B.10 Exclusivity

The end device running the SDTv2 protocol shall not implement any other communication protocol which uses the same SC-32 or the same coding of SID.

B.11 Configuration time parameters

SDTv2 requires a set of (configuration) time parameters which are listed underneath. A more detailed specification of those parameters is given in later sections.

T_{tx_period}	Time period for sending VDP, as defined for SDSRC
T_{rx_period}	VDP receive (sampling) period, as defined for SDSINK
T_{rx_safe}	Maximal time for which SDSINK tolerates the absence of new (fresh) vital data
T_{guard}	Time used by SDSINK to detect the undesired presence of more than one active SDSRC in case of redundant SDSRC

B.12 Safe data source (SDSRC)

B.12.1 General

This clause defines protocol requirements on safe data sources. A safe data source has to produce VDPs, meaning that the VDPs are generated and are subsequently passed to the communication layer for transmission.

B.12.2 Safe Data Preparation (Application)

The following requirements define some application conditions. In general, the application is responsible for providing the vital process data to be sent with SDTv2. Two input data classes are distinguished:

a) **Continuous data**. Those data are characterized by changing their value more or less continuously over time (example: speed signal). Only samples of those data need to be transmitted. After sampling, the sampled data value is kept constant until the next sample

(sample and hold principle). The time, during which such a sampled data value is constant, is subsequently called "signal sample duration time" (T_{sig}).

b) **Discrete data**. Those data are characterized by changing their value on event (example: doors close/open signal). All different values of those data need to be transmitted, because otherwise safety related information might get lost. The minimal time during which the signal value is constant is as well referred to as the "signal sample duration time" (T_{sig}). Contrary to continuous input signals where T_{sig} is exactly one sampling period, can T_{sig} be a multiple of a sample period in the case of discrete signals.

By this, T_{sig} defines the time during which a data value sample is kept constant within the SDTv2 application interface of SDSRC.

In a VDP containing more than one discrete data item which might be typically the case, the signal sample duration time can be different for the individual data items, because it is a property of the source data items itself. The value of T_{sig_min} defines the lowest signal sample duration time occurrence of all the data items within a VDP.

The application shall ensure that all value changes of a discrete input data item (signal) are sampled and that the samples are kept stable in the SDTv2 application interface for a time T_{sig}.

Sampling rate of the application for sampling the input data item needs to be higher than the change rate (frequency) of the input data item.

NOTE T_{sig} can be different for the different input data items.

B.12.3 Safe data sending

B.12.3.1 Protocol

Safety related data shall be sent within the VitalProcessData part of Vital Data Packets (VDP).

The producer (SDSRC) of VDP shall produce the VDP periodically with a cycle time of T_{tx_period}.

NOTE 1 T_{tx_period} will be defined by application, e.g. within IEC 61375-2-4.

The selection of T_{tx_period} shall comply with the following condition:

$$T_{tx\_period} \leq T_{sig\_min} / 3$$

with T_{sig_min} being the lowest signal sample duration time of a sampled input data item (signal) exposed in the SDTv2 application interface.

NOTE 2 This ensures that all signal values are transmitted even in the case of two subsequent VDP losses during transmission.

VDPs shall not be produced if no valid SID can be computed.

NOTE 3 A valid SID can for instance not be computed if SID input parameters are missing.

VDPs shall not be produced if the end device hosting the SDSRC is not a safety device.

The SDSRC shall increment the SSC for each produced VDP:

$$SSC(i+1) = (SSC(i) + 1) \mid_{mod2^k}$$

with $i = 0 .. \infty$ and k being the cardinality of the safety sequence counter used (ETB-VDP: k = 32).

B.12.3.2 Redundant SDSRC

This subclause defines specific requirements on redundant safe data sources. The redundancy principle is to have two redundant source devices (SDSRC-A and SDSRC-B) forming one redundancy group. The input signal is read by both source devices, but only one device (redundancy leader) is actively sending to the sink (SDSINK), the other device (redundancy follower) is not sending. Both source devices supervise each other, and if the redundancy follower detects a failure of the redundancy leader it starts actively sending to SDSINK, see Figure B.8.

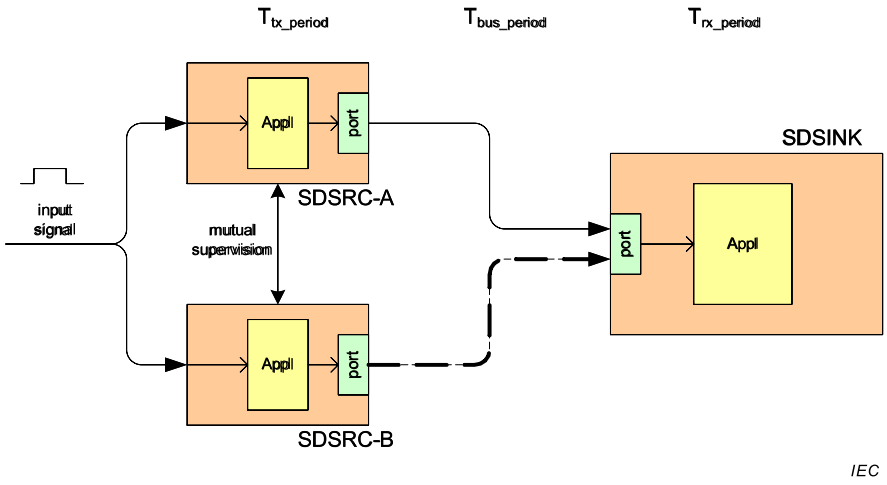


Figure B.8 – Redundancy Group (Example with 2 SDSRC)

NOTE 1 It is not defined herein how the mutual supervision between a redundancy leader and a redundancy follower, and how the switch-over from a redundancy leader to a redundancy follower is realized. This is an implementation choice.

In a redundant producer group of SDSRC (e.g. hosted by a redundant group of safety devices forming a redundancy group) only one SDSRC shall send VDPs.

All SDSRC of a redundancy group shall use different safety message identifiers (SMI).

The time span T_{red} between the redundancy leader ceasing to send VDPs and the redundancy follower start to send VDPs shall not exceed a value of:

$$T_{red} \leq T_{rx\_safe} - 2 \times \max(T_{tx\_period}, T_{rx\_period})$$

NOTE 2 A violation of this rule may trigger the sink time supervision of SDSINK.

NOTE 3 For the definition of T_{rx_safe} and T_{rx_period} see below.

NOTE 4 T_{rx_period} might be longer than T_{tx_period} in case of undersampling.

It has to be considered that there might be multiple SDSINK connected to SDSRC, in which case the lowest value of T_{red} has to be used.

NOTE 5 By fault both SDSRC may send VDPs. This will be detected by SDSINK with the guard time check (see below).

B.13 Safe data sink (SDSINK)

B.13.1 General

This clause defines specific protocol requirements on safe data sinks. SDSINK has to receive ("sample") VDPs, to validate the VDPs, and to expose received process data in the SDTv2

application interface. "Sampling" of VDPs in this context means that the most recent VDP is read from the communication channel interface. The terms “sampling” and “receiving” are used synonymously.

B.13.2 Definitions

B.13.2.1 Variables

For the subsequent specification, the following set of variables is used:

SSC	this is the non-stored SSC value of the actually sampled VDP
SSC_i	this is the SSC value of the last valid VDP
$SSC_{received}$	SSC value of the actually sampled correct VDP
$SSC_{initial}$	SSC value of the initial VDP
SSC_{last}	this is the SSC value of the previously sampled valid VDP
$SID_{initial}$	SID value of an initial VDP

B.13.2.2 VDP Classification

B.13.2.2.1 Duplicate VDP

A VDP is considered a **duplicate** if it is identical to the VDP received before, meaning that the computed SafetyCode of that VDP is identical to the computed SafetyCode of the VDP received before.

NOTE 1 For VDPs already validated with correct SafetyCode and correct user data main version, it is sufficient to compare the SSC in order to identify a duplicate: if SSC = SSC_i or SSC = $SSC_{initial}$ then the VDP is a duplicate.

NOTE 2 ”Computed” SafetyCode means that the SafetyCode computed by the SDSINK during reception is used, not the SafetyCode value written in the VDP.

B.13.2.2.2 Correct VDP

A received VDP is considered **correct**, if:

– SafetyCode is correct (computed SafetyCode value is identical to the SafetyCode value contained in the VDP);
– UserDataMainVersion is correct (equals the expected user data version value).

B.13.2.2.3 Initial VDP

A received VDP is considered **initial** in one of the following cases:

a) it is not a duplicate;
b) it is the first correct VDP received after power-up/reset;
c) it is the first correct VDP received after a communication loss as indicated by the sink time supervision;
d) it is a correct VDP, but where the SafetyCode evaluation has been done with the alternative SID of the redundant SDSRC (not the stored $SID_{initial}$ value of the previously received initial VDP).

NOTE A VDP with a different SID than $SID_{initial}$ may for instance be received when a redundancy shift occurs within a SDSRC redundancy group.

B.13.2.2.4 Fresh VDP

A VDP is considered **fresh** if:

– it is correct;
– the VDP is not the initial VDP;

- the SID of the VDP is identical to $SID_{initial;}$
- it is a real successor to the initial or fresh VDP received before, meaning that
  $SSC \varepsilon \{(SSC_i + 1)|_{mod(2^k)}, \ldots, (SSC_i + N_{SSC})|_{mod(2^k)}\}$,
  with $N_{SSC} = (T_{rx\_safe} / T_{tx\_period})$, rounded up to an integer value, and k being the cardinality of the SSC.

B.13.2.2.5 Valid VDP

A received VDP is considered **valid**, if it is a fresh VDP or a duplicate of the fresh VDP received before.

In all other cases it shall be considered invalid.

B.13.2.2.6 Discarded VDP

To **discard** a VDP means to not expose its data to the application.

B.13.3 SDSINK States

The state diagram shown in Figure B.9 defines the two possible main states a SDSINK can be in. The related triggers, guards and operations are defined in Table B.2 to Table B.4.

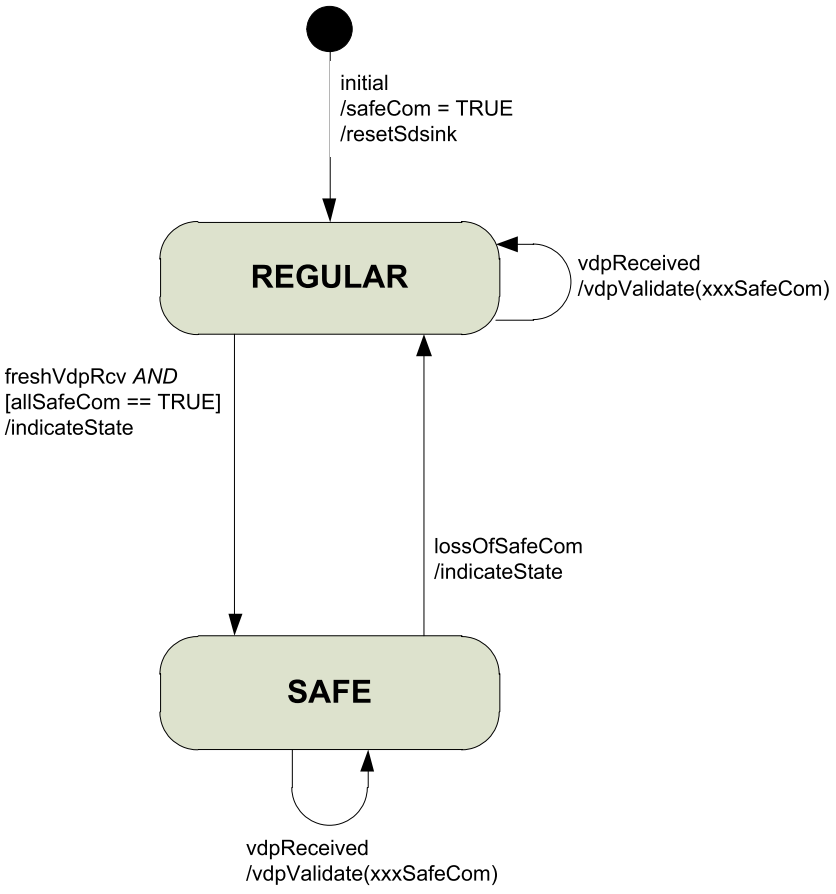


Figure B.9 – SDSINK state diagram

Table B.2 – SDSINK state diagram – triggers

Trigger	Description
Initial	Power-up or re-boot of SDSINK
vdpReceived	The most recent VDP is read from the communication channel interface, see B.13.4.
freshVdpRcv	A fresh VDP has been received
lossOfSafeCom	During VDP validation a loss of safe communication has been detected

Table B.3 – SDSINK state diagram – guards

Guard	Description
allSafeCom	Logical AND over the check results (variable 'xxxSafeCom', see below)

Table B.4 – SDSINK state diagram – operations

Operation	Description
vdpValidate	Validation of received VDP, in particular: • VDP integrity check, see B.13.5 • Sink time supervision check, see B.13.6 • Guard time check (result: gtcSafeCom), see B.13.7 • Latency monitoring (result: lmcSafeCom), see B.13.8 • Channel monitoring (result: cmcSafeCom), see B.13.9 The returned check results can have the following values: xxxSafeCom=TRUE: SDTv2 channel communication is considered safe xxxSafeCom=FALSE: SDTv2 channel communication is considered not safe (regular) (xxx stands for 'gtc', 'lmc' or 'cmc')
resetSdsink	Reset the SDSINK
IndicateState	Indicate a state change to the application

B.13.4 VDP Sampling

A configurable time T_{rx_safe} shall be defined for detecting the loss of safe communication. By default, T_{rx_safe} shall be set in a way that the loss of two subsequently send VDPs is tolerated, e.g. $T_{rx\_safe} := 3 \times T_{tx\_period}$.

Tolerances in timing should be considered for real implementations (e.g. T_{rx_safe} can be set to $3{,}5 \times T_{tx\_period}$ to compensate tolerances in T_{tx_period} and to respect transmission jitter).

The default setting of T_{rx_safe} might be changed in the case of under-sampling, see below.

A configurable time $\mathbf{T_{rx\_period}}$ shall be defined which specifies the cycle in which SDSINK reads (samples) VDPs from the communication channel interface.

NOTE 1 $\mathbf{T_{rx\_period}}$ defines the time between two samplings of the communication channel interface. The model of a periodic sampling of the communication channel interface is used for the purpose of this specification, but it does not imply a specific implementation. A real implementation can also use an event-driven VDP reception procedure.

The period T_{rx_period} of reading VDPs should be shorter than T_{tx_period} of the SDSRC.

NOTE 2 This is necessary to sample all received VDPs.

Undersampling, meaning that T_{rx_period} is longer than T_{tx_period}, can be configured. If so, it shall be defined whether information loss is allowed or not.

Under-sampling without information loss might for instance be used if the SDSINK is only interested in specific discrete vital process data items (signals) within the received VDP, which will change their value less frequently as other data items (signals) in the same VDP.

Under-sampling with information loss might for instance be used when continuous signals are sampled (e.g. train speed signal) and a lower granularity is sufficient.

In case of SDSINK undersampling without data loss, the time T_{rx_safe} shall be shorter than the symbol sample duration time of the signal the SDSINK is interested in, but longer than T_{rx_period}.

NOTE 3 If T_{rx_safe} were longer than the symbol sample duration time, a signal change would get lost undetected.

B.13.5 VDP Integrity Check

B.13.5.1 General

The VDP integrity check aims to filter VDPs which are not correct, meaning that data are corrupted or the user data main version is not the expected one. Data within those 'invalid' VDPs cannot be used (consumed) by the application.

After a power-up, reset, redundancy shift or a loss of safe communication, the receiver waits for the reception of an “initial” VDP. This initial VDP is used to “synchronize” the SDSINK with the SDSRC. Receiving the initial VDP is however not sufficient to indicate the reception of valid and safe data to the application: this will be done only with the next received fresh VDP, which matches the “window of expected SSC“. This window defines a range of allowed SSC values which have to be matched by the VDPs following the initial VDP, in order to ensure that the received VDPs are in correct sequence. This window is shifted to the right with each received VDP carrying a new matching SSC (see Figure B.10), so subsequent VDPs need to match the shifted window. In the example below, a VDP with SSC = 9 has been received. The next VDP is expected to have a SSC in the range of 09 to 13. If the next received VDP has a SSC of 09, it is a duplicate to the previously received VDP. If it has a SSC of 10 to 13, it will be a fresh VDP. All VDPs matching the window are called “valid”, but only those with a new SSC value are called “fresh”. After receiving the VDP with SSC = 10, the window is shifted by one covering now the range of 10..14 (not shown).

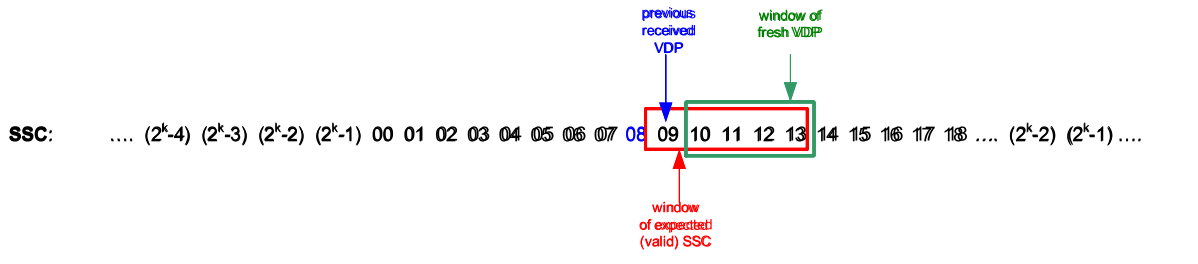


Figure B.10 – Window of expected SSC (example)

B.13.5.2 VDP processing

After reading a VDP from the communication channel interface, SDSINK shall first check the correctness of the VDP.

If VDPs from a redundancy group are expected, the correctness check shall be made with the expected SIDs associated to the SDSRC of the redundancy group.

NOTE In a redundancy group of two SDSRC, first a check can be made with the SID already known from the previously received VDP, and only if this fails it will be done with the second SID.

SDSINK shall check whether a received VDP is initial. Upon reception of an initial VDP, SSC_i and $SSC_{initial}$ shall be set to the SSC of this initial VDP and $SID_{initial}$ shall be set to the SID value used for the validation.

SDSINK shall check whether a received VDP is valid.

SDSINK shall check whether a received VDP is fresh. If the received VDP is a fresh VDP, SSC_i shall be set to the value of the SSC of the received fresh VDP.

This means that the receiver needs only to implement one sequence counter.

User data contained in fresh or valid VDPs shall be exposed to the application for consumption if SDSINK is in state ‘SAFE’.

User data contained in invalid VDPs shall not be exposed to the application for consumption (shall be discarded).

The application may consume the user data received from the most recently received valid VDP in those cases as long as data is indicated as safe.

B.13.6 Sink time supervision

With the reception of an initial VDP, the receiver shall start a timer which expires after a time T_{rx_safe}. ("sink time supervision timer").

The sink time supervision timer shall be retriggered with the reception of a fresh VDP.

If the sink time supervision timer expires, a loss of safe communication shall be indicated (trigger ‘lossOfSafeCom’) and the SDSINK shall wait for the next received VDP which is not a duplicate. This VDP shall be treated as an initial VDP.

B.13.7 Guard time check

B.13.7.1 General

The guard time check intends to detect two redundant active SDSRC (both sending VDPs). For this, a “guard time” is introduced which shall start with the reception of an initial VDP and shall last for a multiple of T_{rx_safe} (configurable). If a VDP with another SID than expected is received during that time, SDSINK assumes that both redundant SDSRC became active and shall indicate loss of communication safety. This situation is depicted in Figure B.11. Here, SDSINK receives first VDPs with SID=A and then VDPs with SID=B. So SDSINK assumes that a redundancy shift at SDSRC side has taken place and expects to receive only VDP with SID=B further on. If it receives a VDP with SID=A now during the time T_{guard}, SDSINK assumes that the SDSRC sending SID=A is still active, which is a redundancy fault at SDSRC side. Such an event is called a "guard time violation". As mentioned, the guard time starts with the reception of an initial VDP. In the example, there are four initial VDPs: the first at the very beginning and the second after the redundancy shift. The VDP with the unexpected SID=A is as well interpreted as an initial VDP according to the rules defined for initial VDPs. This means, that the guard time supervision is retriggered in that case. The same is the case for the next received VDP with SID=B again. Consequently, guard time will be retriggered all the way and practically never expires if there is a mixed reception of VDPs with SID=A and SID=B. The guard timer will only expire if there is a stable reception from one SDSRC (again). So, with expiring guard timer, a loss of safe communication can be negated.

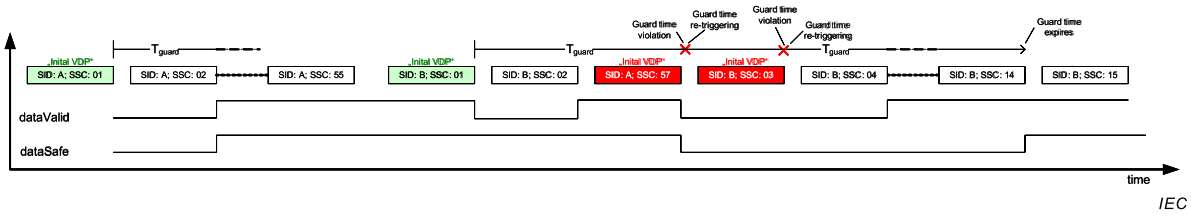


Figure B.11 – Guard time violation (example)

B.13.7.2 Requirements

With the reception of an initial VDP with a SID different to $SID_{initial}$, the receiver shall start a timer, which expires after a time T_{guard} (guard timer).

The receiver shall indicate a loss of safe communication (trigger 'lossOfSafeCom', 'gtcSafeCom = FALSE') if it receives, during the time the guard timer is active, an initial VDP with a SID different to $SID_{initial}$. This is called a "guard time violation".

NOTE 1 The intention with a guard time supervision is to detect two active redundant SDSRC sending VDPs, which normally never should happen.

The parameter T_{guard} shall be defined in a range of:

$$2 \times T_{rx\_safe} \leq T_{guard} \leq 1000 \times T_{rx\_safe}$$

NOTE 2 A good practical value will be $T_{guard} = 10 \times T_{rx\_safe}$.

In case of a guard time violation, the guard timer shall be retriggered in order to start a new guard time supervision time interval.

The guard time supervision shall cancel the loss of safe communication indication (gtcSafeCom = TRUE) if the guard timer expires.

B.13.8 Latency monitoring

B.13.8.1 General

The intention of latency monitoring is to supervise the latency of VDP transmissions. The latency time T_L is the VDP transmission time from SDSRC to SDSINK. In a fault free and properly configured network, this latency time will vary around a constant mean value T_{L_mean}. In case of a network fault[5], this latency time mean value may increase over time, especially in networks with high queuing capabilities, as it is the case in networks with gateways or IP routers. This can be seen in the Figure B.12. Here, beginning with the VDP SSC=04, latency is slowly increasing (extension phase). The increase is not that much that sink time supervision triggers. With VDP SSC=06 the latency time T_L becomes higher than T_{rx_safe}, meaning that received VDPs are outdated (latency violation). The system can then return to normal condition with a contraction phase, during which (queued) VDPs are received closely. During the contraction phase, vital data may get lost, because the receiving frequency may be much higher than the sampling rate.

5 A possible network fault is when a high priority VDP is stored by fault within a low priority queue which carries low priority bulk data like for instance stream data. The probability of such a fault is mainly determined by the hardware of the communication channel.

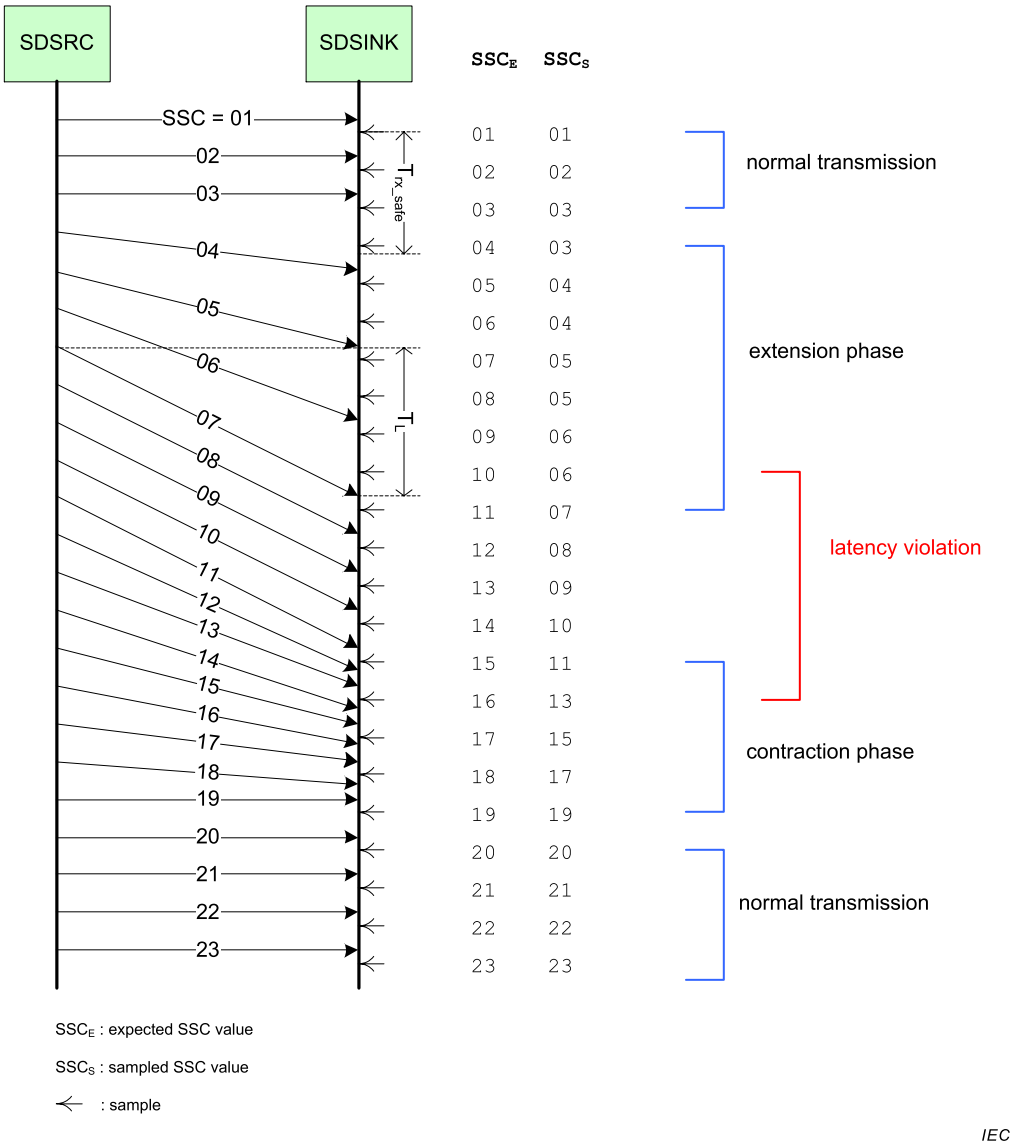


Figure B.12 – Latency violation sequence chart (example)

The latency time can be expressed by the following formula[6]:

$T_L = T_{L\_mean} + K_2 \times t$,

with

T_{L_mean}: mean latency time (in absence of an extension)
K_2: gradient of latency time increase/decrease,
t: time, expressed in T_{tx_period} cycles

In a fault free network, $K_2 = 0$.

The latency cannot be measured directly because a VDP does not contain a time stamp[7]. But it can be measured indirectly when the SDSINK knows the sending period T_{tx_period} of the

6 For this formula a linear increase/decrease of latency is assumed.

7 A time stamp is not used because maintaining a safe synchronized clock with a dynamically changing TCN is considered to be not manageable.

SDSRC. It then can predict which SSC value is expected at a certain point in time. For doing so, the expected SSC values of VDPs (SSC_E) to be received can be predicted and can be compared with the SSC value of the actually received SSC (SSC_S).

SSC_E can be predicted with the calculation:

$$SSC_E = SSC_0 + (t - t_0) / T_{tx\_period}$$

with t_0, SSC_0 being the time and the SSC value at begin of the measurement, t being the actual time and '$(t - t_0) / T_{tx\_period}$' rounded up to an integer value.

In order to compensate for tolerances in SDSRC and SDSINK time base, the latency measurement can be executed in subsequent measurement intervals of a limited duration (measurement time T_M), where at the begin of each interval the measurement is calibrated. This calibration is done by setting SSC_0 to the SSC value received with the first VDP after measurement start. For detecting latency violations, T_M needs to be higher than T_{rx_safe}/k_2 and lower than T_{rx_safe}/ξ, with ξ being the precision of the time base.

EXAMPLE In order to detect a latency increase of $k_2 = 0,1$, the duration of the measurement interval shall be at least 3,0 s, if $T_{rx\_safe} = 0,3$ s, and maximally 300,0 s for an assumed time base tolerance of $\xi = 1$ ‰. In praxis it is recommended to choose a value close to the minimal value, e.g. $T_M = 4,0$ s.

NOTE he measurement calibration at the beginning of a measurement interval eventually hides a latency violation detected during the measurement interval before, although the latency violation is still persisting. This is a known limitation of the procedure. This limitation can be reduced or even abolished by using more accurate time bases or by synchronising the time bases.

The time $T_{L\_violation}$ defines the time span between start of latency increase until it is detected.

B.13.8.2 Procedure

With the reception of an initial VDP, SDSINK shall start to monitor the latency time increase.

SDSINK shall indicate a loss of safe communication (trigger ‘lossOfSafeCom’, guard ‘lmcSafeCom = FALSE’) if the following condition is fulfilled:

$\{SSC_E(t) - SSC_S(t)\}|_{mod(2\hat{}k)} \times T_{tx\_period} \geq T_{rx\_safe}$, with t being the SDSINK time

SDSINK shall be able to detect a latency violation in case of a latency time increase with a constant gradient of $K_2 \geq 0,1$ within a time of:

$$T_{L\_violation} \leq 2 \times (1 + 1/K_2) \times T_{rx\_safe}$$

EXAMPLE $K_2 = 0,1$, $T_{tx\_period} = 0,1$ s, $T_{rx\_safe} = 0,3$ s. Then the mean latency T_{L_mean} increases with 0,01 s for each transmitted VDP. It then should take maximally 6,6 s until latency monitoring triggers.

SDSINK shall cancel the loss of safe communication indication (guard ‘lmcSafeCom = TRUE’) if the following condition is fulfilled:

$\{SCC_E(t) - SCC_S(t)\}|_{mod(2\hat{}k)} \times T_{tx\_period} < T_{rx\_safe}$, with t being the SDSINK time

B.13.9 Channel monitoring

B.13.9.1 General

Channel monitoring intends to detect a sudden increase of the transmission failure rate within the SDTv2 channel which might follow a hardware or software fault in one of the components belonging to the SDTv2 channel. With an increased transmission failure rate, the probability that a corrupted VDP slips through the SafetyCode checker undetected grows and might

become unacceptable. Those failures might be permanent, in which case a repair action will be required, or temporary, in which case the system might recover itself.

B.13.9.2 Procedure

The channel monitoring shall indicate a loss of safe communication (trigger 'lossOfSafeCom', result 'cmcSafeCom = FALSE') if the number of VDPs received with incorrect SafetyCode exceeds a threshold defined by a configurable parameter K_3 (channel monitoring threshold).

Channel monitoring should cancel the loss of safe communication indication ('cmcSafeCom = TRUE') if the rate of VDPs received with incorrect SafetyCode is lower than K_3.

VDP duplicates shall not be considered for channel monitoring (shall be filtered out).

By default, K_3 shall be set to a value of $K_3 = 43$ per hour.

NOTE K_3 is the quotient of the tolerated hazard rate R_{H3} for undetected corrupted VDPs and the performance $p_{us} \approx 2^{-32}$ of the SafetyCode SC-32 for detecting corrupted VDPs (see IEC 62280). For $R_{H3} = 10^{-8}$ this results in a value of $K_3 = 43$.

B.13.9.3 Activation

With the creation of a SDTv2 channel, channel monitoring shall be started and shall be active as long as the SDTv2 channel is setup.

In case a SDTv2 channel is relinquished or reconfigured, channel monitoring should be aborted and restarted again when reconfiguration is completed.

NOTE During the reconfiguration of a SDTv2 channel spurious VDPs with incorrect SafetyCode might be detected because the SDSINK may use another value of the SafeTopoCount for VDP validation than the SDSRC. This is typically the case during train composition changes or changes in the operational train directory.

EXAMPLE If the operational train directory changes (e.g. change of leading vehicle), existing SDTv2 channels between a function in the leading vehicle and another function are relinquished and are setup again when the operational train directory is updated.

B.13.9.4 Channel monitoring algorithm (informative)

There are surely different ways to implement channel monitoring, with different characteristics and performance. The following algorithm allows an easy implementation of channel monitoring:

First filter out all duplicated VDPs (before checking the SafetyCode for correctness).

Provide a counter Z which:

- Increments each time it receives a VDP with incorrect SafetyCode by a value of f_M / K_4, until a maximum value of $(f_M \times K_3 / K_4)$.
- Decrement each time it receives a VDP with correct SafetyCode by 1, until 0 is reached.

f_M is the frequency of VDPs sent per hour: $f_M = 3600 / T_{tx\_period}$

Parameter K_4 to be set to $K_4 = 36$, for the case that $K_3 = 43$

If Z exceeds a threshold CMMAX = $(f_M / K_4) \times (K_3 - K_4)$, a loss of safe communication is indicated.

The indication of a loss of safe communication is canceled if Z reaches 0.

B.13.10 SDTv2 Application Interface

SDSINK state transitions from state REGULAR to state SAFE and vice versa shall be indicated to the application.

Vital process data exposed in the application interface shall be invalidated (set to 0) in SDSINK state REGULAR.

B.13.11 Change of operational train composition

B.13.11.1 General

In case of a change of the operational train directory, the SDSRC will thereafter use another SID value in the computation of the safety code. This means that SDSINK also has to change the value of the expected SID(s). There are two use cases:

a) SDSRC is the same as before, only SID changed because there is a new SafeTopoCount value

b) SDSRC is another than before, e.g. in the case the SDSRC belongs to a function of the leading car and the leading car position itself changed.

The second case is comparable to a redundancy shift of a redundant SDSRC couple. Consequently, SDSINK has to behave in the same manner – after getting informed about new SID value(s) SDSINK shall treat the next received correct VDP, which then uses the new SID, as an initial VDP.

B.13.11.2 Interrupting channel monitoring

Special attention has to be paid to the potential asynchrony between a SID change at SDSRC and SDSINK side.

In both cases, either when SDSRC changes first and then the SDSINK, or vice versa, there will be a certain time where SDSINK detects safety code errors, which could trigger the channel monitoring.

Therefore, SDSINK shall stop channel monitoring as soon as it gets informed by its local ETBN about a new train inauguration.

SDSINK shall resume channel monitoring with the initial VDP received after the SID change.

B.14 Diagnosis and statistics

The statistic counter listed in Table B.5 should be provided.

Table B.5 – SDTV2 statistic counters

Statistic counter	Provided by
statistic counter of produced VDPs. NOTE How statistic counters are made accessible has to be defined on application level.	SDSRC
statistic counter of received fresh VDP	SDSINK
statistic counter of received non-duplicated VDP with a SafetyCode error	SDSINK
statistic counter of received VDP with a SID different to the expected SID NOTE This is the case when a redundancy shift at SDSRC occurs	SDSINK
statistic counter of VDP received "out of sequence".	SDSINK
statistic counter of received VDP duplicates	SDSINK

Statistic counter	Provided by
statistic counter of latency gap occurance	SDSINK
statistic counter of received VDP with wrong UserDataMainVersion	SDSINK
statistic counter of communication safety losses caused by SDTV2 channel monitoring	SDSINK

SDSINK statistic counters should be maintained individually for each expected (configured) SID.

Statistic counter should be implemented as 32-bit unsigned integer

SDSINK statistics counter shall be reset at power up and device reset.

SDSINK statistic counters shall stop counting after reaching their maximal value (no overflow)

B.15 Safe data transmission over MVB (informative)

B.15.1 General

Although SDTv2 is basically defined for the safe communication over ETB and ECN, there might still be end devices connected to MVB which have to communicate safely. In principle, the same protocol can be used, but due to some limitations on MVB some modifications of the SDTv2 protocol are necessary. These deviations from SDTv2 are defined within the next subclauses.

B.15.2 MVB-VDP

The MVB-VDP is defined as shown in Figure B.13 and Figure B.14:

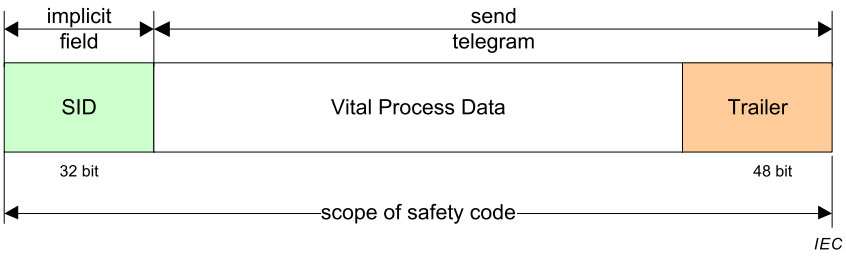


Figure B.13 – MVB-VDP

The MVB-VDP is defined as follows:

```
MVB_VDP::= RECORD
  {
  VitalProcessData  ARRAY [VDPSize] of WORD8
                                  -- user defined data set with
                                     safe and non-safe process data.
                                     VDPsize = 2, 10 or 26
                                     (dependant on MVB frame data size)

  UserDataMainVersion    UNSIGNED4 -- main version of the vital process
                                     data part, shall be set by
                                     the application
                                     Value: 1..15 (0 = reserved)

  reserved01             UNSIGNED4 -- reserved value, shall be set to 0
```

```
SafeSequCount          UNSIGNED8 -- safe sequence counter (SSC),
                                    implemented as a counter which is
                                    incremented each time a new MVB-VDP
                                    is produced to the MVB traffic store port

SafetyCode             UNSIGNED32 -- SC-32, computed over VDP starting
                                    with most significant byte of
                                    VitalProcessData
                                    up to and including the SafeSequCount.
                                    Start value: SID
}
```

Figure B.14 – Format of MVB-VDP

B.15.3 SDTV2 protocol deviations for MVB

Vital MVB process data shall be packed into MVB-VDPs for communication.

a) The parameter ConsistId of the SID as defined in Clause B.8 shall be set to 0 since there is only intra consist communication via MVB.

b) The parameter SafeTopoCount of the SID as defined in Clause B.8 shall be set to 0 since there is only intra consist communication via MVB.

c) Latency monitoring as specified in B.13.8 is only optional for MVB sink.

B.16 SDTv2 with TRDP message data

Safety related data may be transmitted within the payload of TRDP message data telegrams, if the restrictions defined in this clause are considered. Contrary to process data are message data transmitted event based, and there is no guarantee that data sent by SDSRC are safely received by SDSINK in time, nor is there a safe feedback from SDSINK to SDSRC that data have arrived. To cope with those shortages, the following requirements shall be applied:

a) A SDSRC shall encapsulate safety critical data in an ETB-VDP as defined in B.9, with ETB_VDP parameter ‘SafeSequCount’ = ‘FFFFFFFF’H.

b) ETB-VDP shall only be sent on a SDTv2 channel on which a parallel safe SDTv2 connection based on TRDP process data is existing.

This shall ensure that the SDTv2 channel is supervised with respect to availability (sink time supervision), redundancy (guard time supervision), latency (latency monitoring) and error rate (channel monitoring).

c) SDSINK shall inform SDSRC about the reception of the ETB-VDP. This can be done directly, e.g. by sending a TRDP-MD reply telegram with a reply VDP, or indirectly via another mechanism).

EXAMPLE If the CSTINFOCTRL telegram is not received by the SDSINK ECSP, the SDSRC will indirectly recognize this because it cannot transit from train directory computation state ‘WaitForCstInfo’ to state ‘TrnDirSetup’.

Annex C
(informative)

Train Real-Time Data Protocol Configuration (TRDP)

C.1 General

This annex defines, as an extension to the communication profile, an XML format that can be used to configure the TRDP.

In order to exchange data over TCN, TRDP needs to be configured, e.g. with information on how to handle the data, where to send data, etc. This information is stored in a **Configuration Database**.

The database can be populated from configuration file(s) in XML-format or with the aid of API functions as it is shown in Figure C.1. The database is usually read in at start up from a configuration file in XML-format. The file(s) is created by an off-line tool and downloaded as a DLU (Downloadable Unit).

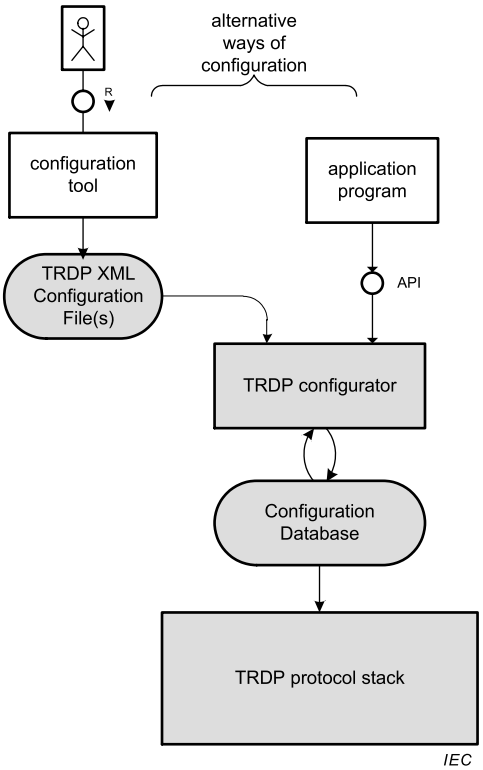


Figure C.1 – TRDP configuration block diagram

The structure of this XML file can be described in a XML Schema Definition (.XSD). By use of XML schema definition, it is possible to validate a configuration file. The schema definition describes only the formal structure of the configuration and can be used to validate the configurations by TRDP.

The definition files can be common for different communication concepts. In the clauses below, only features relevant to TRDP are described. The TRDP configuration includes:

- Device Configuration Parameters
- Bus Interface Parameters including related Telegram Parameters (Exchange Parameters)
- Mapped device parameters
- Communication Parameters
- Dataset Parameters
- Debug Parameters

NOTE The column “Optional/Required” in the tables below is used for what is optional and required for TRDP and not what is optional and required for the XML file or for other use of the XML file, e.g. tools.

The overall structure of the XML configuration description is defined as:

```
<device attributes>
   <device-configuration attributes/>
   <bus-interface-list>
      <bus-interface attributes>
         <telegram attributes >
          :
         </telegram>
          :
      </bus-interface>
       :
   </bus-interface-list>
   <mapped-device-list>
      <mapped-device attributes>
         <mapped-bus-interface attributes>
            <mapped-telegram attributes/>
              :
         </mapped-bus-interface>
           :
      </mapped-device>
         :
   </mapped-device-list>
   <com-parameter-list>
      <com-parameter attributes/>
       :
   </com-parameter-list>
   <data-set-list>
      <data-set attributes >
       :
      </data-set>
       :
   </data-set-list>
   <debug attributes/>
</device>
```

C.2 Device Parameters

The parameters given with the device can be used to initialize the TRDP stack of a device.

XML Element:

```
<device >
     :
</device>
```

The tag “device-configuration” is optional.

XML Attributes:

All attributes that can be specified for the tags are described in the following Table C.1.

Table C.1 – Attributes for device tag

Name	Data Type	Optional/Required	Description
host-name	STRING	Required	Device host name
leader-name	STRING	Optional	Leader host name, depending on redundancy concept
type	STRING	Optional	Device type, for information only

C.3 Device Configuration Parameters

The parameters given with the device configuration can be used to initialize the TRDP stack of a device.

Device configuration parameters may be provided for:

- Memory configuration

XML Element:

```
<device >
    <device-configuration attributes />
     :
</device>
```

The tag “device-configuration” is optional.

XML Attributes:

All attributes that can be specified for the tags are described in the following Table C.2.

Table C.2 – Attributes for device-configuration tag

Name	Data Type	Optional/Required	Description
memory-size	UINT32	Optional	Size of TRDP total memory

C.4 Bus Interface List

C.4.1 General

The Bus interface list contains the configuration parameter needed for the specific interfaces.

XML Element:

```
<device>
    <bus-interface-list>
        <bus-interface attributes>
          <trdp-process attributes/>
          <pd-com-parameter attributes/>
          <pd-com-parameter attributes/>
            <telegram attributes >
            :
            </telegram>
          :
        </bus-interface>
        :
   </bus-interface-list>
</device>
```

C.4.2 Bus Interface Configuration

C.4.2.1 General

The parameters given with the bus interface configuration can be used as parameters to initialize a specific TRDP session.

For each bus interfaces parameters may be provided for:

- Interface configuration
- Thread/task configuration
- Default PD Communication
- Default MD Communication
- Telegram configuration

XML Element:

```
<device>
    <bus-interface-list>
        <bus-interface attributes>
          <trdp-process attributes/>
          <pd-com-parameter attributes/>
          <pd-com-parameter attributes/>
            <telegram attributes >
               :
            </telegram>
          :
        </bus-interface>
          :
    </bus-interface-list>
</device>
```

XML Attributes:

All attributes that can be specified for the tag “bus-interface” are described in the following Table C.3.

Table C.3 – Attributes for bus-interface tag

Name	Data Type	Optional/Required	Description
network-id	UINT8	Required	IP interface (1..4) on the device
name	STRING	Required	Interface name
host-ip	UINT32	Optional	Host IP address
leader-ip	UINT32	Optional	Leader IP address in case of redundancy

C.4.2.2 Process Configuration

At start up the TRDP main process is spawned for sending and receiving MD and PD. The thread/task priority and, for cyclic thread/task, the cycle time may be specified in the XML configuration file. If not specified TRDP default values will be used.

XML Element:

```
<device>
    <bus-interface-list>
        <bus-interface attributes>
          <trdp-process attributes/>
          <pd-com-parameter attributes/>
          <pd-com-parameter attributes/>
            <telegram attributes >
               :
            </telegram>
          :
        </bus-interface>
         :
     </bus-interface-list>
</device>
```

The tag “trdp-process” is optional.

XML Attributes:

All attributes that can be specified are described in Table C.4:

Table C.4 – Attributes for trdp-process tag

Name	Data Type	Optional/Required	Description
cycle-time	UINT32	Optional	Cycle time–in 10^{-6} s (µs). Only for cyclic threads/tasks. For default value see Table C.5.
blocking	STRING	Optional	NO – non – blocking, YES – blocking For default value see Table C.5.
traffic-shaping	STRING	Optional	OFF – no traffic shaping, ON – traffic shaping For default value see Table C.5.
priority	UINT32	Optional	Priority for trdp process task. Values:1-255, 1 = highest For default value see Table C.5.

Default values are defined in Table C.5:

Table C.5 – Default values for thread/task

Name	Value
cycle-time	10000
blocking	NO
traffic-shaping	ON
priority	64

C.4.2.3 PD Communication Parameters

There are a number of parameters that may be configured for the PD communication in the XML configuration file. If not specified TRDP default values will be used.

XML Element:

```
<device>
    <bus-interface-list>
        <bus-interface attributes>
          <trdp-process attributes/>
          <pd-com-parameter attributes/>
            <telegram attributes >
               :
            </telegram>
          :
        </bus-interface>
        :
    </bus-interface-list>
</device>
```

The tag “pd-com-parameter” is optional.

XML Attributes:

All attributes that can be specified for the tag “pd-com-parameter” are described in Table C.6.

Table C.6 – Attributes for pd-com-parameter tag

Name	Data Type	Optional/Required	Description
timeout-value	UINT32	Optional	Timeout value in 10^{-6} s (µs), before considering received process data as invalid. Disabled if 0 or if not specified. For default value see Table C.7.
validity-behaviour	STRING	Optional	Behaviour when received process data is invalid. ZERO = zero values KEEP = keep last value For default value see Table C.7.
ttl	UINT32	Optional	Default time To live for PD. For default value see Table C.7.
qos	UINT32	Optional	Default quality of service for PD. For default value see Table C.7.
marshall	STRING	Optional	ON/OFF = enable/disable internal marshalling/unmarshalling For default value see Table C.7.

Name	Data Type	Optional/Required	Description
callback	STRING	Optional	ON/OFF = enable/disable call back For default value see Table C.7.
port	UINT32	Optional	Port to be used for PD communication For default value see Table C.7.

Default values are defined in Table C.7:

Table C.7 – Default values for pd-com-parameter

Name	Value
timeout-value	100000
validity-behaviour	ZERO
ttl	64
qos	5
marshall	OFF
callback	OFF
port	20548

C.4.2.4 MD Communication Parameters

There are a number of parameters that may be configured for the MD communication in the XML configuration file. If not specified TRDP hard coded default values will be used.

The configured default values herein are used if nothing else is configured for the ComId.

The default time-out time for waiting for a confirm message is used when no confirm time-out value is specified for a ComId.

The default time-out time for waiting for a reply message(s) is used when no reply time-out value is specified for a ComId.

TRDP stores received sequence numbers per source IP address, to detect resend messages that already have been received. The maximum number of stored sequence numbers per IP address may be configured.

XML Element:

```
<device>
    <bus-interface-list>
        <bus-interface attributes>
            <trdp-process attributes/>
            <pd-com-parameter attributes/>
            <md-com-parameter attributes/>
            <telegram attributes >
               :
            </telegram>
          :
        </bus-interface>
        :
```

```
    </bus-interface-list>
</device>
```

The tag “md-com-parameter” is optional.

XML Attributes:

All attributes that can be specified for the tag “md-com-parameter” are described in Table C.8.

Table C.8 – Attributes for md-com-parameter tag

Name	Data Type	Optional/Required	Description
confirm-timeout	UINT32	Optional	Default time-out time in 10^{-6} s (μs) for receiving a confirm message For default value see Table C.9.
reply-timeout	UINT32	Optional	Default time-out time in μs for receiving reply message(s) For default value see Table C.9.
connect-timeout	UINT32	Optional	Default time-out time in μs to close a not used TCP connection For default value see Table C.9.
ttl	UINT32	Optional	Default time to live for MD. For default value see Table C.9.
qos	UINT32	Optional	Default quality of service for MD. For default value see Table C.9.
protocol	STRING	Optional	TCP/UDP = default protocol For default value see Table C.9.
marshall	STRING	Optional	ON/OFF = enable/disable internal marshalling/unmarshalling For default value see Table C.9.
callback	STRING	Optional	ON/OFF = enable/disable call back For default value see Table C.9.
udp-port	UINT32	Optional	Port to be used for UDP MD communication For default value see Table C.9.
tcp-port	UINT32	Optional	Port to be used for TCP MD communication For default value see Table C.9.
num-sessions	UINT32	Optional	Maximal number of replier sessions to prevent DoS attacks For default value see Table C.9.

Default values are defined in Table C.9:

Table C.9 – Default values for md-com-parameter

Name	Value
confirm-timeout	1 000 000 μs
reply-timeout	5 000 000 μs
connect-timeout	60 000 000 μs
ttl	64
qos	3
protocol	UDP
marshall	OFF
callback	ON
udp-port	20550
tcp-port	20550
num-sessions	1000

C.4.2.5 Telegram Configuration (ExchgPar)

C.4.2.5.1 General

The telegram configuration contains the central elements for data exchange – the exchange parameters. It is identified by the central key **ComID**, see Figure C.2.

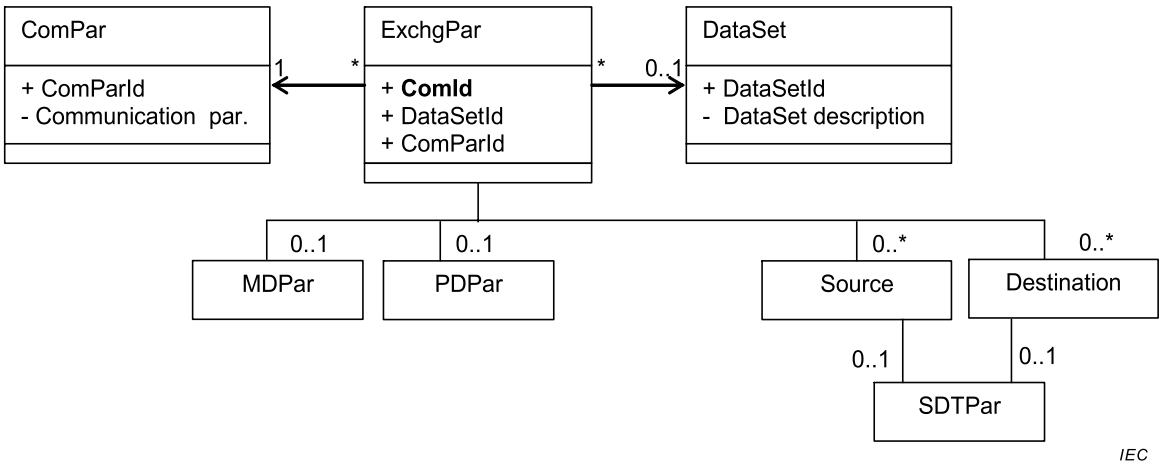


Figure C.2 – Exchange Parameters with the central key ComId

Each Exchange Parameter references one or no DataSet and one Communication Parameter. (Many Exchange Parameters can reference the same DataSet and Communication Parameter.) The Exchange Parameter also describes how to handle any Process and/or Message Data.

An Exchange Parameter is described in the XML file as a “telegram” tag.

XML Element:

```
<device>
    <bus-interface-list>
       <bus-interface attributes>
           <trdp-process attributes/>
           <pd-com-parameter attributes/>
```

```
            <md-com-parameter attributes/>
            <telegram attributes >
                <md-parameter attributes />
                <pd-parameter attributes />
                <source attributes />
                <destination attributes />
            </telegram>
            :
        </bus-interface>
      :
    </bus-interface-list>
</device>
```

The tags “sdt-parameter“, “md-parameter”, “pd-parameter”, “source” and “destination” are optional.

The tags “sdt-parameter”, “md-parameter” and “pd-parameter” can only be used one time for each Comid.

For each ComId the tags “source” and “destination” may be used several times with different values.

For some attributes there are default values.

XML Attributes:

All attributes that can be specified for the telegram tag are described in Table C.10.

Table C.10 – Attributes for telegram tag

Name	Data Type	Optional/ Required	Description
name	STRING	Optional	Required for tooling and debugging
com-id	UINT32	Required	ID for exchange parameter NOTE 1 Only the first found configuration of a ComId will be considered. NOTE 2 ComId’s 1-1000 are reserved for special purpose.
data-set-id	UINT32	Required for PD Optional for MD	ID for dataset to be exchanged.
com-parameter-id	UINT32	Optional	ID for communication parameter to be used, if not given the default parameters for MD/PD communication will be used

C.4.2.5.2 MD Parameters

The MD telegram configuration defines the parameters for MD telegram exchange.

XML Element:

```
<device>
    <bus-interface-list>
        <bus-interface attributes>
            :
            <telegram attributes >
                :
                <md-parameter attributes/>
```

```
                :
            </telegram>
            :
        </bus-interface>
    </bus-interface-list>
</device>
```

XML Attributes:

All attributes that can be specified for the md-parameter tag are described in Table C.11.

Table C.11 – Attributes for md-parameter tag

Name	Data Type	Optional/ Required	Description
confirm-timeout	UINT32	Optional	Confirm time-out time (unit: 10^{-6} s (μs)) Default value will be used at absence of the tag, see Table C.9.
reply-timeout	UINT32	Optional	Response time-out time (unit: 10^{-6} s (μs)) Default value will be used at absence of the tag, see Table C.9.
marshall	STRING	Optional	ON/OFF = enable/disable internal marshalling/unmarshalling
callback	STRING	Optional	ON/OFF = enable/disable callback
protocol	STRING	Optional	UDP/TCP

C.4.2.5.3 PD Parameters

The PD telegram configuration defines the parameters for PD telegram exchange.

XML Element:

```
<device>
    <bus-interface-list>
        <bus-interface attributes>
            :
            <telegram attributes >
                :
                <pd-parameter attributes/>
                :
            </telegram>
           :
        </bus-interface>
        :
    </bus-interface-list>
</device>
```

XML Attributes:

All attributes that can be specified for the pd-parameter tag are described in Table C.12.

Table C.12 – Attributes for pd-parameter tag

Name	Data Type	Optional/ Required	Description
timeout	UINT32	Optional	Timeout value (unit: 10^{-6} s (µs)), before considering received process data as invalid. Disabled if 0 or no specified value. Default value will be used at absence of the tag, see Table C.7.
validity-behaviour	STRING	Optional	Behaviour when received process data is invalid. ZERO = zero values KEEP = keep the last value Default value will be used at absence of the tag, see Table C.7.
cycle	UINT32	Required	Cycle time in 10^{-6} s (µs), describing how often a process data should be transmitted. TRDP will round this value to a multiple of the cycle time of the TRDP process thread. Default value will be used at absence of the tag, see Table C.7.
redundant	UINT32	Optional	>0 if process data is redundant, i.e. it should be transmitted in leader mode, and not in follower mode in a redundant system. Default value = 0 (no redundancy);
marshall	STRING	Optional	ON/OFF – enable/disable internal marshalling/unmarshalling Default value will be used at absence of the tag, see Table C.7.
callback	STRING	Optional	ON/OFF = enable/disable callback
offset-address	UINT16	Optional	Offset-address for PD in traffic store for ladder topology

C.4.2.5.4 Source Parameters

The source configuration defines the parameters for the data source.

XML Element:

```
<device>
    <bus-interface-list>
        <bus-interface attributes>
              :
            <telegram attributes >
            :
                <source attributes>
                   <sdt-parameter attributes />
                </source>
            :
            </telegram>
         :
        </bus-interface>
      :
    </bus-interface-list>
</device>
```

XML Attributes:

All attributes that can be specified for the source tag are described in Table C.13.

Table C.13 – Attributes for source tag

Name	Data Type	Optional/ Required	Description
id	UINT32	Required	Source identifier.
uri1	STRING	Required	Source URI for process and message data. Used for PD filtering at receiver side to only receive data from a specific end device(s) or for MD as information provided to the receiving side. Syntax: [[user@]host] User part only required for MD. NOTE If not specified here or overridden in the subscribe call, then no filtering for PD used. If not specified here or overridden in any MD send call, the URI (host part) of the own device is used. NOTE Only URI strings that are resolved as unicast IP addresses can be used.
uri2	STRING	Optional	For process data a second source URI string can be given for source filtering, e.g. for redundant devices. Syntax: [host] NOTE If not specified here or overridden in the subscribe call, then no filtering for PD used. If not specified here or overridden in any MD send call, the URI (host part) of the own device is used. NOTE Only URI strings that are resolved as unicast IP addresses can be used.
name	STRING	Optional	Optional name for the connection

C.4.2.5.5 Destination Parameters

The source configuration defines the parameters for the data destination.

XML Element:

```
<device>
    </bus-interface-list>
        <bus-interface attributes>
              :
            <telegram attributes >
                 :
               <destination attributes
                    <sdt-parameter attributes />
               </destination>
                 :
            </telegram>
           :
        </bus-interface>
        :
    </bus-interface-list>
</device>
```

XML Attributes:

All attributes that can be specified for the source tag are described in Table C.14.

Table C.14 – Attributes for destination tag

Name	Data Type	Optional/ Required	Description
id	UINT32	Required	Destination identifier.
uri	STRING	Required	Destination URI for process and message data. Syntax: [user@]host If not specified here it has to be set/overridden in any send/publish call. The URI user part is used in the MD header frame to address a URI listener. The URI host part is used to resolve the destination IP address used for sending and at receiver side to check if the IP address is a multicast address that has to be joined.
name	STRING	Optional	Optional name for the connection

C.4.2.5.6 SDTv2 Parameters

The SDTv2 configuration defines the parameters for the SDTv2 SDSRC and SDSINK.

XML Element:

```
<device>
   <bus-interface-list>
      <bus-interface attributes>
               :
         <telegram attributes >
               :
            <source attributes
               <sdt-parameter attributes />
            </source>
             :
            <destination attributes
               <sdt-parameter attributes />
            </destination>
             :
         </telegram>
          :
      </bus-interface>
   </bus-interface-list>
</device>
```

The tag “sdt-parameter” is optional.

XML Attributes:

All attributes that can be specified for the SDTv2 are described in Table C.15.

Table C.15 – Attributes for sdt-parameter tag

Name	Data Type	Optional/ Required	Description
smi1	UINT32	Required	Safe message identifier – unique for this message at consist level
smi2	UINT32	Optional	Safe message identifier for a redundant device – unique for this message at consist level
udv	UINT16	Required	User data version

Name	Data Type	Optional/ Required	Description
rx-period	UINT16	Required	Sink cycle time, unit: 10^{-3} s (ms)
tx-period	UINT16	Required	Source cycle time, unit: 10^{-3} s (ms)
n-rxsafe	UINT8	Optional	Timout in number of tx-period cycles
n-guard	UINT16	Optional	Guard time in number of tx-period cycles
cm-thr	UINT32	Optional	Channel monitoring threshold

Default values are defined in Table C.16:

Table C.16 – Default values for sdt-parameter tag

Name	Value
smi2	0
n-rxsafe	3
n-guard	100
cm-thr	43

C.5 Mapped Device Parameters

C.5.1 General

There might be the requirement to have for several identical devices the same configuration structure (e.g. the different door controllers of the consist). This is supported by the mapped device tag, containing the differences for the specific mapped devices.

XML Element:

```
<device>
   <device-configuration>
   :
   </device-configuration>
   <mapped-device-list>
      <mapped-device attributes>
         <mapped-bus-interface attributes>
            <mapped-telegram attributes>
               :
            </mapped-telegram attributes>
               :
         </mapped-bus-interface>
            :
      </mapped device>
         :
   </mapped-device-list>
      :
</device>
```

The tags “mapped-device-list” and “mapped-device” are optional.

XML Attributes:

All attributes that can be specified for the tag “mapped-device” are described in the following Table C.17.

Table C.17 – Attributes for mapped-device tag

Name	Data Type	Optional/Required	Description
host-name	STRING	Required	Device name
leader-name	STRING	Optional	Leader name, optional for redundant devices

C.5.2 Mapped Bus Interface Parameters

For each bus interface of a mapped device the different attributes regarding the process data, sources, destinations and related SDTv2 parameters of the telegrams can be specified.

XML Element:

```
<device>
   <device-configuration>
   :
   </device-configuration>
   <mapped-device-list>
      <mapped-device attributes>
         <mapped-bus-interface attributes>
            <mapped-telegram attributes>
               <mapped-pd-parameter attributes />
               <mapped-source attributes>
                  <mapped-sdt-parameter attributes />
               </mapped-source>
                  :
               <mapped-destination attributes>
                  <mapped-sdt-parameter attributes />
               </mapped-destination>
                  :
            </mapped-telegram>
               :
         </mapped-bus-interface>
            :
      </mapped device>
   </mapped-device-list>
</device>
```

The tag “mapped-bus-interface” is optional.

The tag “mapped-telegram” is optional.

The tag “mapped-pd-parameter” is optional.

The tag “mapped-source” is optional.

The tag “mapped-destination” is optional.

The tag “mapped-sdt-parameter” is optional.

XML Attributes:

All attributes that can be specified for the tag “mapped-bus” are described in Table C.18.

Table C.18 – Attributes for mapped-bus-interface tag

Name	Data Type	Optional/Required	Description
name	STRING	Required	Name for this interface
host-ip	UINT32	Optional	IP address for this interface
leader-ip	UINT32	Optional	Leader IP address in case of redundancy

All attributes that can be specified for the tag “mapped-telegram” are described in the following Table C.19.

Table C.19 – Attributes for mapped-telegram tag

Name	Data Type	Optional/Required	Description
com-id	UINT32	Required	ComId of the mapped telegram
name	STRING	Optional	Optional different name for the telegram

All attributes that can be specified for the tag “mapped-pd-parameter” are described in Table C.20.

Table C.20 – Attributes for mapped-pd-parameter tag

Name	Data Type	Optional/ Required	Description
offset-address	UINT16	Optional	Offset-address for PD in traffic store for ladder topology

All attributes that can be specified for the tag “mapped-source” are described in Table C.21.

Table C.21 – Attributes for mapped-source tag

Name	Data Type	Optional/Required	Description
Id	UINT32	Required	Identifier of the source of the telegram
uri1	STRING	Required	Source URI for process and message data. Used for PD filtering at receiver side to only receive data from a specific end device(s) or for MD as information provided to the receiving side. For process data the source URI string can include up to two comma separated URI strings, e.g. redundant devices. Syntax: [[user@]host[, host]] NOTE If not specified here or overridden in the subscribe call, then no filtering for PD used. If not specified here or overridden in any MD send call, the URI (host part) of the own device is used. NOTE Only URI strings that are resolved as unicast IP addresses can be used.
uri2	STRING	Optional	For process data a second source URI string can be given for source filtering, e.g. for redundant devices. Syntax: [host] NOTE If not specified here or overridden in the subscribe call, then no filtering for PD used. If not specified here or overridden in any MD send call, the URI (host part) of the own device is used.

Name	Data Type	Optional/Required	Description
			NOTE Only URI strings that are resolved as unicast IP addresses can be used.
name	STRING	Optional	Optional different name for the connection

All attributes that can be specified for the tag “mapped-destination” are described in Table C.22.

Table C.22 – Attributes for mapped-destination tag

Name	Data Type	Optional/Required	Description
Id	UINT32	Required	Identifier of the destination of the telegram
uri	STRING	Required	Destination URI for process and message data. Syntax: [user@]host If not specified here it has to be set/overridden in any send/publish call. The URI user part is used in the MD header frame to address a URI listener. The URI host part is used to resolve the destination IP address used for sending and at receiver side to check if the IP address is a multicast address that has to be joined.
name	STRING	Optional	Optional different name for the connection

All attributes that can be specified for the tag “mapped-SDTv2 parameters” are described in Table C.23.

Table C.23 – Attributes for mapped-SDTv2-parameter tag

Name	Data Type	Optional/Required	Description
smi1	UINT32	Required	Required for SDTv2 telegrams.
smi2	UINT32	Optional	Requred for redundant SDTv2 telegrams.

C.6 Communication Parameters (ComPar)

C.6.1 General

The communication parameter information describes in which way the communication should take place. In most cases, one set of parameters is sufficient but it is possible to specify specific communication parameters for special situations.

A Communication Parameter is described in the XML file as a “com-parameter” tag.

XML Element:

```
<device>
   <com-parameter-list>
      <com-parameter attributes/>
   :
   </com-parameter-list>
</device>
```

XML Attributes:

All attributes that can be specified for the tag are described in Table C.24.

Table C.24 – Attributes for com-parameter tag

Name	Data Type	Optional/ Required	Description
id	UINT32	Required	ID for communication parameter NOTE Only the first configuration of a com-parameter-id will be considered.
qos	UINT32	Required	Quality of Service, what priority level data should be sent with, 0..7
ttl	UINT32	Optional	Time To Live, how many jumps a message should live, 0..255, default = 64.
retries	UINT32	Optional	Number of retries used for MD request

C.6.2 Default Communication Parameters

Most communication can be done with a small set of communication parameters. To simplify for the TRDP user there are always two sets of default communication parameters available as listed in Table C.25.

Table C.25 – Default communication parameters

Com-parameter-id	qos	time-to-live	retries	Description
1	5	64	0	Suitable for PD communication
2	3	64	3	Suitable for MD communication

C.7 DataSet Parameters

C.7.1 General

Data communication over the ETB is done with DataSets (Figure C.3).

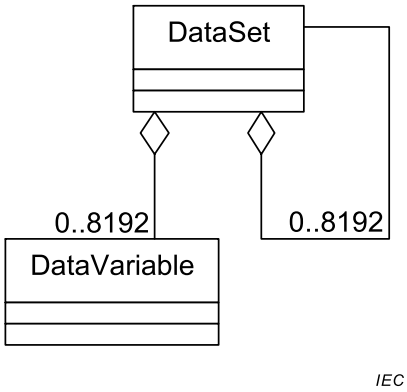


Figure C.3 – DataSet structure

A DataSet is a container of data items, like a *structure* in C language. A DataSet can contain up to 8192 DataVariables or other DataSets. A *DataVariable* is data of one of the basic data types defined in Table C.26.

Table C.26 – Basic data types

Type name	Type #	Description
BOOL8	1	=UINT8, 1 bit relevant (equal zero ➔ false, not equal zero ➔ true)
BITSET8		=UINT8, 8 bit relevant
ANTIVALENT8		=UINT8, 2 bit relevant
CHAR8	2	char, can be used also as UTF-8
UTF16	3	Unicode UTF-16 character
INT8	4	Signed integer, 8 bit
INT16	5	Signed integer, 16 bit
INT32	6	Signed integer, 32 bit
INT64	7	Signed integer, 64 bit
UINT8	8	Unsigned integer, 8 bit
UINT16	9	Unsigned integer, 16 bit
UINT32	10	Unsigned integer, 32 bit
UINT64	11	Unsigned integer, 64 bit
REAL32	12	Floating point real, 32 bit
REAL64	13	Floating point real, 64 bit
TIMEDATE32	14	32 bit UNIX time
TIMEDATE48	15	48 bit TCN time (32 bit seconds and 16 bit ticks)
TIMEDATE64	16	32 bit seconds and 32 bit microseconds

Each DataVariable or DataSet can be single or multiple instances. Multiple instances (arrays) can be of fix or variable size.

The maximum depth of a dataset within a dataset is 5.

The DataSet configuration information describes all datasets to be used in data exchange. Many Exchange Parameters can use one single DataSet.

The DataSet describes the structure of the data, i.e. which data variables and other datasets it contains. By use of this information TRDP can send and receive data in such a way that it complies with the data formats required on the TCN. Applications do not have to do any further adjustment to be able to use the data.

A DataSet is described in the XML file as a “data-set” tag. Inside this there are “element” tags.

XML Element:

```
<device>
   <data-set-list>
       <data-set attributes >
          <element attributes />
       :
       </data-set>
    :
   </data-set-list>
</device>
```

XML Attributes:

All attributes that can be specified for the data-set tags are described in Table C.27.

Table C.27 – Attributes for data-set tag

Name	Data Type	Optional/ Required	Description
name	STRING	Optional	Required for tooling and debugging
id	UINT32	Required	ID for dataset NOTE 1 Only the first found configuration of a data-set-id will be considered. NOTE 2 Data set id’s 1-1000 are reserved for special purpose.

C.7.2 DataSet Element

C.7.2.1 General

To be able to perform marshalling of datasets, all end devices need information about structure and content of the datasets. This information is stored in a Dataset Formatting Table in the configuration database. This table describes for each data item of the DataSet its *type* and *array size*. For visualization purposes each element can have the additional attributes unit, scale and offset.

Each element of a DataSet can be either a DataVariable or DataSet. Up to 8192 elements are possible.

XML Element:

```
<device>
   <data-set-list>
       <data-set attributes >
          <element attributes />
       :
       </data-set>
    :
```

```
   </data-set-list>
</device>
```

XML Attributes:

All attributes that can be specified for an element tag are described in Table C.28.

Table C.28 – Attributes for element tag

Name	Data Type	Optional/ Required	Description
name	STRING	Optional	Required for tooling and debugging
type	STRING	Required	Data type of data variable (BOOL8, BITSET8, …, see basic data types in chapter C.7.1) or dataset identifier (1..)
array-size	UINT32	Optional	0 = Array with dynamic size 1 = Single >1 Number of instances in array Default = 1
unit	STRING	Optional	Unit text for visualisation purposes.
scale	FLOAT	Optional	Factor for visualisation purposes (val = scale × x+offset).
offset	INT32	Optional	Offset for visualisation purposes (val = scale × x+offset).

C.7.2.2 DataSet Element Type

If the type is another DataSet, the *Type* is the ID of this dataset.

For a DataVariable the *Type* tells the marshalling software the size of the data but also how to treat it. A 4-byte string is treated differently as 4-byte integer or real. The basic Datatype constants currently available are described in C.7.1.

The table describes all data types that can be used on the TCN. For IEC 61131-3 software only a sub-set of types with fixed size can be used.

NOTE The C++ type “BOOL” is often not resolved into a UINT8 variable by the compiler because of that here BOOL8 is defined and used.

C.7.2.3 DataSet Element array-size

The *array-size* describes the number of instances for the data item as listed in Table C.29.

Table C.29 – Use of element array size

Array-size	Instances
1	Single
2..n	Array, fixed size 2..n
0	Array, dynamic size

A dynamic sized array is preceded by a UINT8, UNIT16 or UINT32, containing current size of the array, i.e. number of items in the array (not number of bytes). See example in C.7.3.5.

Dynamic sized arrays can only be used for message data and not for process data.

C.7.3 Examples of DataSets

C.7.3.1 General

The examples below show different types of datasets with example of included DataVariables and DataSets, single and multiple instances, fixed or variable.

C.7.3.2 DataSet with DataVariables

This dataset contains single instances of basic data types.

C Declaration	Dataset Formatting Table		
	Type	Number	Comment
`struct DS57 {`			
`UINT32 a;`	UINT32	1	Single
`UINT8 b;`	UINT8	1	Single
`INT16 c;`	INT16	1	Single
`REAL32 d;`	REAL32	1	Single
`};`			

XML configuration example:

```
<device>
   <data-set-list>
       <data-set id=”57” >
          <element type=”UINT32” array-size=”1” name=”a” />
          <element type=”UINT8” array-size=”1” name=”b” />
          <element type=”INT16” array-size=”1” name=”c” />
          <element type=”REAL32” array-size=”1” name=”d” />
       </data-set>
    :
   </data-set-list>
</device>
```

C.7.3.3 DataSet with other DataSet

In this example a dataset DS3 contains another dataset DS2.

To specify that a dataset contain another dataset, the column for Type is set to the ID of that dataset. The marshalling software will recognize this by the positive value. Codes for basic data types are negative.

C Declaration	Dataset Formatting Table		
	Type	Number	Comment
struct DS2 {			
UINT32 a1;	UINT32	1	Single
INT32 b1;	INT32	1	Single
INT32 c1;	INT32	1	Single
};			
struct DS3 {			
UINT32 a;	UINT32	1	Single
INT32 b;	INT32	1	Single
struct DS2 c;	ID of DS2	1	Single
};			

XML configuration example:

```
<device>
   <data-set-list>
       <data-set id=”2” >
          <element type=”UINT32” array-size=”1” name=”a1” />
          <element type=”INT32” array-size=”1” name=”b1” />
          <element type=”INT32” array-size=”1” name=”c1” />
       </data-set>
       <data-set id=”3 >
          <element type=”UINT32” array-size=”1” name=”a” />
          <element type=”INT32” array-size=”1” name=”b” />
          <element type=”2” array-size=”1” name=”c” />
       </data-set>
    :
   </data-set-list>
</device>
```

C.7.3.4 DataSet with Fixed Sized Arrays

In this example, a dataset contains multiple instances of DataVariables and DataSets, i.e. arrays of data.

C Declaration	Dataset Formatting Table		
	Type	Number	Comment
struct DS4 {			
INT32 a[17];	INT32	17	Array of integers
struct DS2 b[3];	ID of DS2	3	Array of datasets
};			

The first data item is an array of 17 instances of INT32;

The second data item is an array of 3 instances of dataset DS2.

XML configuration example:

```
<device>
   <data-set-list>
       <data-set id=”2” >
          <element type=”UINT32” array-size=”1” name=”a1” />
          <element type=”INT32” array-size=”1” name=”b1” />
```

```
            <element type="INT32" array-size="1" name="c1" />
        </data-set>
        <data-set id="4" >
            <element type="INT32" array-size="17" name="a" />
            <element type="2" array-size="3" name="b" />
        </data-set>
     :
   </data-set-list>
</device>
```

C.7.3.5 DataSet with Dynamic sized arrays

In this example a dataset contains instances of arrays, but the size of the arrays is not predefined in the configuration database but dynamically specified at runtime.

A dynamically sized array is preceded with a size of type UINT8, UINT16 or UINT32 that contains the current size of the array. These shall be loaded with current size at run-time. Size should be set to number of items in the array.

Dynamic size of arrays means that there is no information in the configuration database about the size of the array.

NOTE Dynamic sized datasets can only be used in MD communication, not in PD communication.

Declaration	**Dataset Formatting Table**		
	Type	Number	Comment
`struct DS5 {`			
`  UINT32 aSize;`	UINT32	1	Size of array b (=5)
`  INT32 b[5];`	INT32	0	
`};`			

Declaration	**Dataset Formatting Table**		
	Type	Number	Comment
`struct DS6 {`			
`  UINT32 aSize;`	UINT32	1	Size of array b (=5)
`  INT32 b[5];`	INT32	0	
`  UINT16 cSize;`	UINT16	1	Size of array d (=9)
`  struct DS2 d[9];`	ID of DS2	0	
`};`			

XML configuration example:

```
<device>
   <data-set-list>
       <data-set id="2" >
          <element type="UINT32" array-size="1" name="a1" />
          <element type="INT32" array-size="1" name="b1" />
          <element type="INT32" array-size="1" name="c1" />
       </data-set>
       <data-set id="5" >
          <element type="UINT32" array-size="1" name="a" />
          <element type="INT32" array-size="0" name="b" />
       </data-set>
       <data-set id="6" >
          <element type="UINT32" array-size="1" name="a" />
          <element type="INT32" array-size="0" name="b" />
```

```
          <element type="UINT32" array-size="1" name="c" />
          <element type="2" array-size="0" name="d" />
       </data-set>
    :
   </data-set-list>
</device>
```

The example of structure for data set 6 in the figure above is not very useful unless there are several similar structures used for sending with the same dataset ID. When this type of structure is used where a dynamic array is followed by any other variable type the size variables has to be set to exactly the size of the corresponding array. See the code example below for a more useful way to use this type of dynamic dataset. Another possibility, as for data set 5, is to only use one dynamic array in the end of the dataset then the size variable can be set to the current used size of the array. Code example for dataset 6:

```
char buf[2000];
UINT16 *p;
UINT32 len;

p = (UINT16 *) buf;          /* Set pointer to start of buffer */
*p++ = currAsize;            /* Store current size of array a */
memcpy(p, aBuffer, currAsize * sizeof(INT32));
p = (char *) p + currAsize * sizeof(INT32); /* Move pointer */

*p++ = currBsize;            /* Store current size of array b */
memcpy(p, bBuffer, currBsize * sizeof(struct DS2));
p = (char *) p + currBsize * sizeof(struct DS2);

len = (char *) p - buf;      /* length of buffer, no. of bytes */
```

C.7.3.6 DataSet for Transparent Communication

In some cases applications want to exchange data without any interference of marshalling software, e.g for uploading of recorded binary data. This can be achieved by sending a dataset that contains only one data item: a dynamic sized array of bytes, unsigned 8 bit integers or by sending a comId without any specified dataset. As mentioned before dynamic data set can only be sent as message data.

Declaration	**Dataset Formatting Table**		
	Type	Number	Comment
`struct DS5 {`			
`  UINT32 bufSize;`	UINT32	1	Size of array buf (=n)
`  UINT8 buf[n];`	UINT8	0	Array of bytes
`};`			

The sending buffer `buf` shall be preceded with the size of the buffer to send, loaded with the number of items in `buf`, i.e. ‘n’.

Annex D
(informative)

Access to End Device (ED) statistics

D.1 General

This annex describes the access to statistics data of end devices implementing TRDP.

D.2 Structures

D.2.1 General

TRDP memory statistic and configuration information is defined as follows:

```
TRDP_MEM_STATISTICS_T::= RECORD
  {
  total            UINT32        -- total memory size
  free             UINT32        -- free memory
  minFree          UINT32        -- minimal free memory in statistics
                                    interval
  allocBlocks      UINT32        -- number of allocated blocks
  allocErr         UINT32        -- number of allocation errors
  freeErr          UINT32        -- number of freeing errors
  blockSize        UINT32[15]    -- if used, predefined memory block sizes
  usedBlockSize    UINT32[15]    -- if used, number of used memory blocks
                                    for each block size
  }
```

TRDP process data statistic and configuration information is defined as follows:

```
TRDP_PD_STATISTICS_T::= RECORD
  {
  defQos                UINT32    -- default QoS for PD
  defTtl                UINT32    -- default TTL for PD
  defTimeout            UINT32    -- Default timeout for PD
  numSubs               UINT32    -- Number of subscribed ComId's
  numPub                UINT32    -- Number of published ComId's
  numRcv                UINT32    -- number of received PD packets
  numCrcErr             UINT32    -- number of received PD packets with
                                     CRC err
  numProtErr            UINT32    -- number of received PD packets with
                                     protocol err
  numTopoErr            UINT32    -- number of received PD packets with
                                     wrong topo count
  numNoSubs             UINT32    -- number of received PD push packets
                                     without subscription
  numNoPub              UINT32    -- Number of received PD pull packets
                                     without publisher
  numTimeout            UINT32    -- number of PD timeouts
  numSend               UINT32    -- number of sent PD  packets
  }
```

TRDP message data statistic and configuration information is defined as follows:

```
TRDP_MD_STATISTICS_T::= RECORD
  {
  defQos                UINT32    -- default QoS for MD
  defTtl                UINT32    -- default TTL for MD
  defReplyTimeout       UINT32    -- Default reply timeout for MD
  defConfirmTimeout     UINT32    -- Default confirm timeout for MD
  numList               UINT32    -- Number of Listeners
  numRcv                UINT32    -- number of received MD packets
  numCrcErr             UINT32    -- number of received MD packets with
                                     CRC err
  numProtErr            UINT32    -- number of received MD packets with
                                     protocol err
  numTopoErr            UINT32    -- number of received MD packets with
                                     wrong topo count
  numNoListener         UINT32    -- number of received MD packets without
                                     listener
  numReplyTimeout       UINT32    -- number of MD reply timeouts
  numConfirmTimeout     UINT32    -- number of MD confirm timeouts
  numSend               UINT32    -- number of sent MD  packets
  }
```

TRDP general statistic and configuration information is defined as follows:

```
TRDP_STATISTICS_T::= RECORD
  {
  version               UINT32    -- TRDP version
  timestamp             UINT64    -- actual time stamp
  uptime                UINT32    -- time in seconds since last
                                     initialization
  statisticTime         UINT32    -- time in seconds since last reset
                                     of statistics
  hostname              CHAR[16]  -- own host name
  leaderName            CHAR[16]  -- leader host name
  ownIpAddr             UINT32    -- own IP address
  leaderIpAddr          UINT32    -- leader IP address
  processPrio           UINT32    -- priority of TRDP process
  processCycle          UINT32    -- cycle time of TRDP process
                                     (unit: 10-6 s (µs))
  numJoin               UINT32    -- number of joins
  numRed                UINT32    -- number of redundancy groups
  mem                   TRDP_MEM_STATISTICS_T   -- memory statistics
  pd                    TRDP_PD_STATISTICS_T    -- PD statistics
  udpMd                 TRDP_MD_STATISTICS_T    -- MD statistics
  tcpMd                 TRDP_MD_STATISTICS_T    -- MD statistics
  }
```

D.2.2 tlc_getSubsStatistics

TRDP process data subscription statistic information is defined as follows:

```
TRDP_SUBS_STATISTICS_T::= RECORD
  {
  comId             UINT32    -- subscribed ComId
  joinedAddr        UINT32    -- joined IP address
  filterAddr        UINT32    -- filter IP address, i.e IP address of
                                 the sender for this subscription,
                                 0.0.0.0 in case all senders.
  callBack          UINT32    -- reference for call back function if used
  timeout           UINT32    -- time-out value (unit: 10-6 s (µs)).
                                 0 = No time-out supervision
  status            UINT32    -- receive status information TRDP_NO_ERR,
                                 TRDP_TIMEOUT_ERR
  toBehav           UINT32    -- behaviour at time-out.
                                 set data to zero / keep last value
  numRecv           UINT32    -- number of packets received for this
                                 subscription
  }
```

D.2.3 tlc_getPubStatistics

TRDP process data publish statistic information is defined as follows:

```
TRDP_PUB_STATISTICS_T::= RECORD
  {
  comId             UINT32    -- subscribed ComId
  destAddr          UINT32    -- IP address of destination for this
                                 publishing
  cycle             UINT32    -- publishing cycle (unit: 10-6 s (µs)
  redId             UINT32    -- redundancy group id
  redState          UINT32    -- redundant state:
                                 !=0 = Leader or 0 = Follower
  numPut            UINT32    -- number of packet updates
  numSend           UINT32    -- number of packets sent for this publish
  }
```

D.2.4 tlc_getUdpListStatistics, tlc_getTcpListStatistics

TRDP message data listener statistic information is defined as follows:

```
TRDP_LIST_STATISTICS_T::= RECORD
  {
  comId             UINT32    -- subscribed ComId
  uri               CHAR[32]  -- URI to listen to (user part)
  joinedAddr        UINT32    -- joined IP address
  callback          UINT32    -- call back function reference if used
  queue             UINT32    -- queue reference if used
  userRef           UINT32    -- user reference if used
  numRecv           UINT32    -- number of received packets
  }
```

D.2.5 tlc_getRedStatistics

TRDP redundancy statistic information is defined as follows:

```
TRDP_RED_STATISTICS_T::= RECORD
  {
  id                UINT32    -- redundancy id
  state             UINT32    -- leader/follower
  }
```

D.3 ED interface for statistic data access

D.3.1 General

TRDP statistic data can be retrieved either by:

- SNMP, as defined in Clause G.3
- TRDP interface as defined below

D.3.2 TRDP interface

The TRDP interface for retrieving statistic data from EDs is shown in Figure D.1.

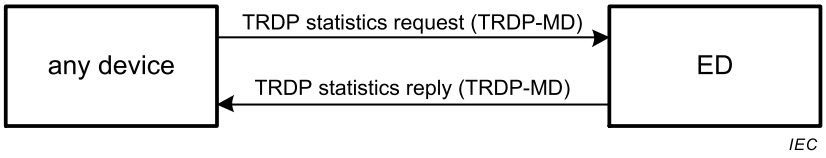


Figure D.1 – TRDP statistics data telegrams

TRDP MD Request telegram parameters:

MsgType:	‘Mr’
ComId:	As defined below
SourceURI (user part):	user defined
DestinationURI (user part):	void
DestinationURI (host part):	selected ED
SourceURI (host part):	user defined
Dataset:	TRDP_STATISTICS_REQUEST
Reply timeout:	3,0 s

TRDP MD Reply telegram parameters:

MsgType:	‘Mp’
ComId:	As defined below
Dataset:	TRDP_STATISTICS_REPLY

TRDP-MD telegram payload:

```
TRDP_STATISTICS_REQUEST::= RECORD
  {                                -- payload empty
  }

TRDP_STATISTICS_REPLY::= ONE_OF
  {
  [GLOBAL]      TRDP_STATISTICS_T         -- comId = 30 (request) 31 (reply)
  [SUBS]        TRDP_SUBS_STATISTICS_T    -- comId = 32 (request) 33 (reply)
  [PUB]         TRDP_PUB_STATISTICS_T     -- comId = 34 (request) 35 (reply)
  [RED]         TRDP_RED_STATISTICS_T     -- comId = 36 (request) 37 (reply)
  [LIST_UDP]    TRDP_LIST_STATISTICS_T    -- comId = 38 (request) 39 (reply)
  [LIST_TCP]    TRDP_LIST_STATISTICS_T    -- comId = 40 (request) 41 (reply)
  }
```

Annex E
(informative)

Service interface

E.1 General

This annex defines a communication interface between the services providers defined in this part of the standard (DNS Server, TTDB Manager, ECSP and ETBN) and its clients.

An overview of the different interfaces is given in Figure E.1. Those interfaces are:

- ECSP interface. This interface is provided by the ECSP and supports the ETB control service. There exists only one ECSP per consist and the interface is between the ECSP and a dedicated client (ECSC) which is entitled to control the communication on the operational ETB (ETB0).
- TTDB Manager interface. The TTDB manager provides this interface and supports read access to the TTDB. The TTDB manager should be available within each consist network.
- DNS server interface. This interface is provided by the DNS server and supports the TCN-DNS service. A DNS server should be available within each consist network.
- ETBN control interface. This interface is provided by the ETBN and supports ETBN control and access to ETB status information, refer to IEC 61375-2-5. An ETBN is available within each consist network.

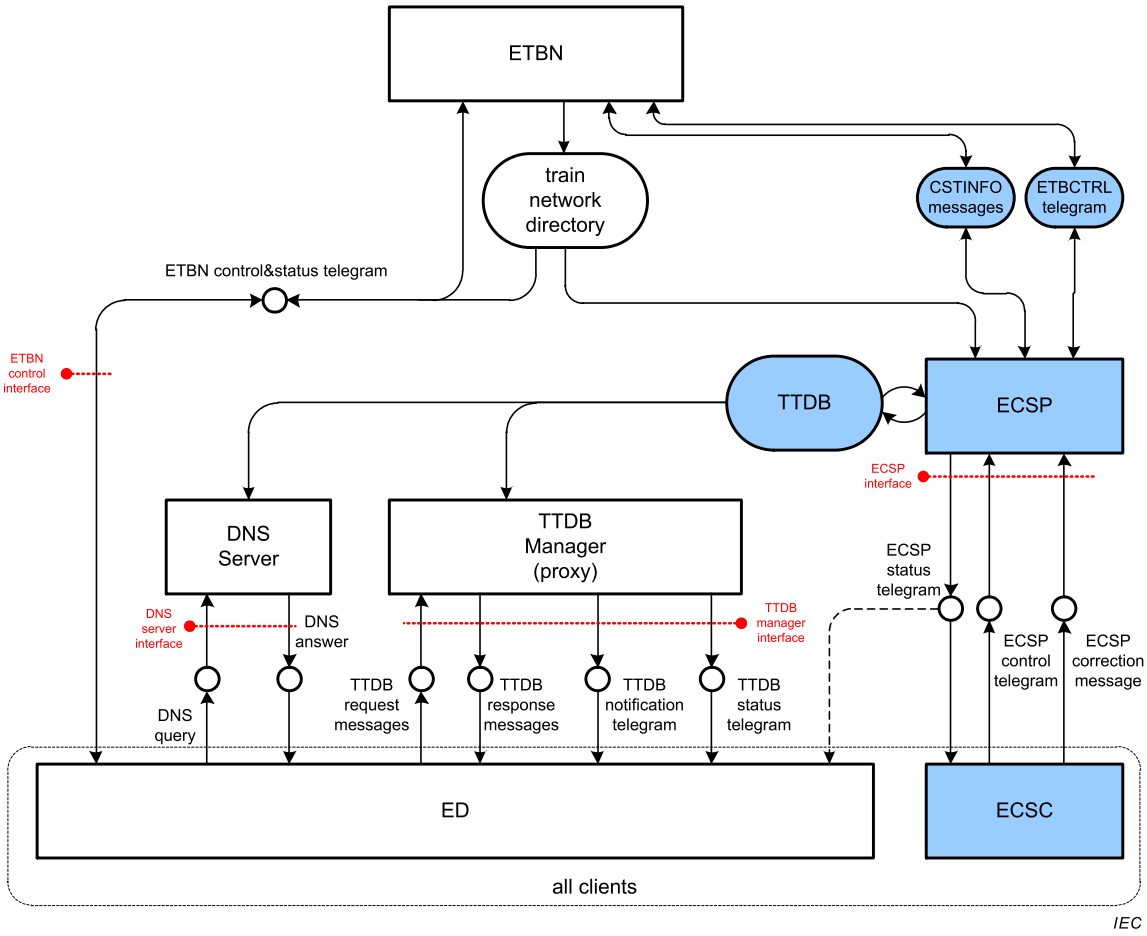


Figure E.1 – Service interfaces block diagram

E.2 Service provider

E.2.1 Proxies

In order to prevent IP routing of service requests in consists with multiple consist networks, there is a possibility to install local instances of a service provider or local proxies to a central service provider. The implementation of local instances or local proxies is a choice of the consist designer.

NOTE Services provided by the ETBN do not require a proxy as there is an ETBN connected to each CN.

E.2.2 Performance

A user of a service based on a TRDP MD request-reply pattern needs to know how fast the server will reply in order to set the TRDP parameter value ‘ReplyTimeOut’ correspondingly. This reply time not only depends on the server itself, but also on the performance of the network. Therefore it is recommended that the consist designer defines the minimum values.

EXAMPLE For retrieving the operational train directory from the local TTDB manager the consist designer defines a minimum reply time of 1,5 s.

E.3 ECSP interface

E.3.1 General

The ECSP interface defines the interaction between the ECSC device and the ECSP for the functions which are defined in Clause 6.

The telegrams exchanged between ECSC and ECSP are shown in Figure E.2.

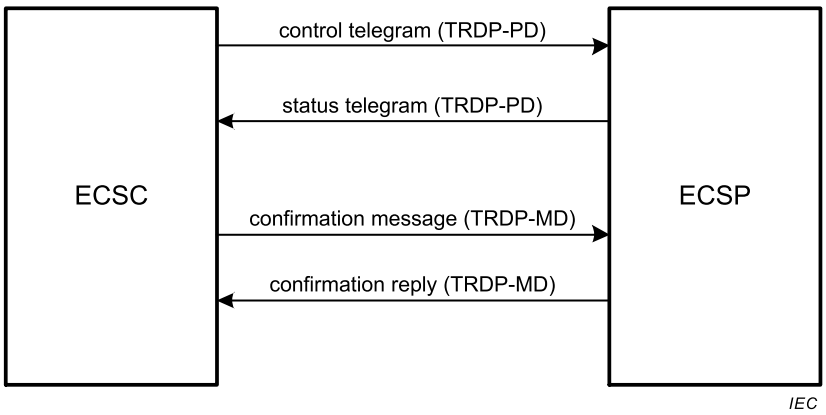


Figure E.2 – ECSP interface telegrams

E.3.2 ECSP control telegram

TRDP PD telegram parameters:

MsgType:	‘Pd’
ComId:	120
SMI:	120 (only needed if SDTv2 is used)
UserDataVersion:	‘0100’H (only needed if SDTv2 is used)

DestinationURI (host part): devECSP.anyVeh.ICst.ICITrn.ITrn

Dataset: ECSP_CTRL

Cycle time: (1,0 ± 0,1) s

Timeout: 5,0 s

The TRDP telegram is defined as shown in Figure E.3.

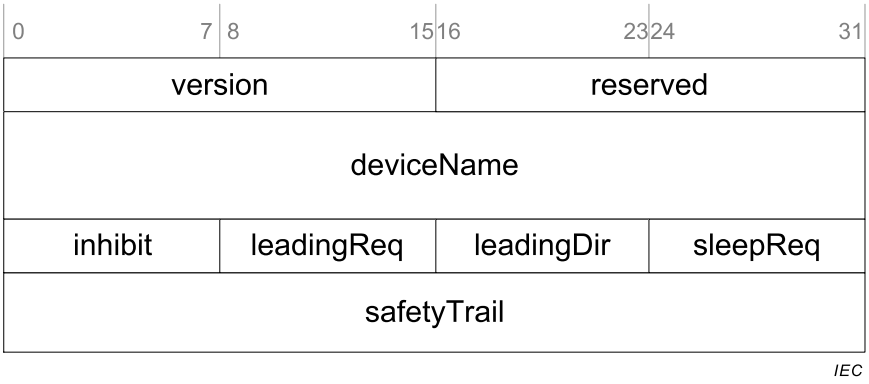


Figure E.3 – ECSP control data

The payload of the TRDP telegram is defined as:

```
ECSP_CTRL::= RECORD
  {
  version          UINT16        -- telegram version information
                                    (vvrr, v-version, r-release)
                                    version = 1
                                    release = 0
  reserved01       UINT16        -- reserved (= 0)
  deviceName       LABEL         -- function device of ECSC which sends
                                    the telegram
  inhibit          UINT8         -- inauguration inhibit
                                    0 = no inhibit request
                                    1 = inhibit request
  leadingReq       UINT8         -- leading request
                                    0 = no leading request
                                    1 = leading request
  leadingDir       UINT8         -- leading direction
                                    0 = no leading request
                                    1 = leading request direction 1
                                    2 = leading request direction 2
  sleepReq         UINT8         -- sleep request
                                    0 = no sleep request
                                    1 = sleep request
  safetyTrail      ETBCTRL_VDP   -- ETB-VDP trailer
                                    completely set to 0 = SDTv2 not used
  }
```

NOTE The control telegram contains also parameters related to the ETBN, like for instance parameter ‘inhibit’. For this interface, it is assumed that the ECSC interfaces only with the ECSP and that ETBN related information is transferred from the ECSP to the ETBN (ECSP acting as proxy). However, it is also possible to use the direct interface to the ETBN instead (see Clause E.6).

E.3.3 ECSP status telegram

TRDP PD telegram parameters:

MsgType:	‘Pd’
ComId:	121
SMI:	121 (only needed if SDTv2 is used)
UserDataVersion:	‘0100’H (only needed if SDTv2 is used)
DestinationURI (host part):	devECSC.anyVeh.ICst.ICITrn.ITrn
Dataset:	ECSP_STATUS
Cycle time:	(1,0 ± 0,1) s
Timeout:	User defined

If the receiving ECSC detects a timeout, the ECSC shall inform the consist control application that the ECSP failed.

The TRDP telegram is defined as shown in Figure E.4.

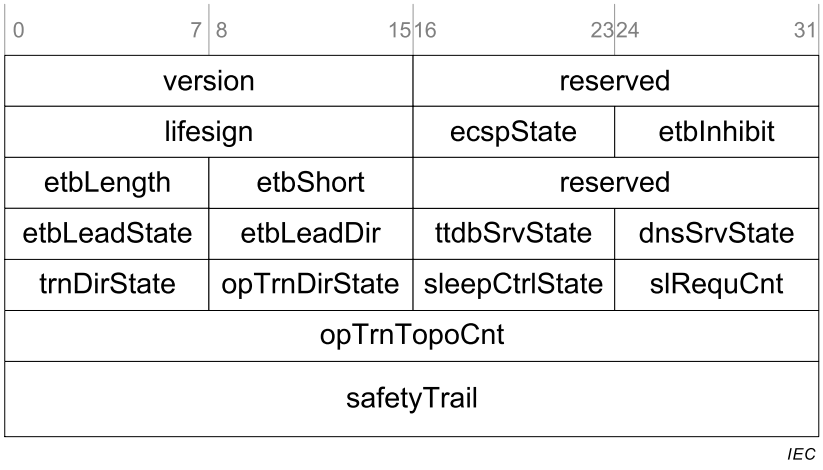


Figure E.4 – ECSP status data

The payload of the TRDP telegram is defined as:

```
ECSP_STATUS::= RECORD
  {
  version          UINT16        -- telegram version information
                                    (vvrr, v-version, r-release)
                                    version = 1
                                    release = 0
  reserved01       UINT16        -- reserved (= 0)
  lifesign         UINT16        -- wrap-around counter, incremented with
                                    each produced telegram.
  ecspState        UINT8         -- ECSP state indication
                                    0 = ECSP not operational(initial value)
                                    1 = ECSP in operation
  etbInhibit       UINT8         -- inauguration inhibit indication
                                    0 = n/a (default)
                                    1 = inhibit not requested on ETB
                                    2 = inhibit set on local ETBN
                                    3 = inhibit set on remote ETBN
                                    4 = inhibit set on local and remote
                                        ETBN
  etbLength        UINT8         -- indicates train lengthening in case
                                    train inauguration is inhibit
                                    0 = no lengthening (default)
                                    1 = lengthening detected
  etbShort         UINT8         -- indicates train shortening in case
                                    train inauguration is inhibit
                                    0 = no shortening (default)
                                    1 = shortening detected
  reserved02       UINT16        -- reserved (= 0)
  etbLeadState     UINT8         -- indication of local consist leadership
                                    5 = consist not leading (initial value)
                                    6 = consist is leading requesting
                                    9 = consist is leading
                                    10 = leading conflict
                                    other values are not allowed
  etbLeadDir       UINT8         -- direction of the leading end vehicle
                                    in thelocal consist
                                    0 = not relevant
                                    1 = TCN direction 1
                                    2 = TCN direction 2
                                    other values are not allowed
  ttdbSrvState     UINT8         -- TTDB server state indication
                                    0 = n/a (initial value)
                                    1 = Leader (default)
                                    2 = Follower
                                    3 = Error
  dnsSrvState      UINT8         -- DNS server state indication
                                    0 = n/a (initial value)
                                    1 = Leader (default)
                                    2 = Follower
                                    3 = Error
  trnDirState      UINT8         -- train directory state
                                    1 = UNCONFIRMED
                                    2 = CONFIRMED
                                    other values are not allowed
  opTrnDirState    UINT8         -- operational train directory state
                                    1 = INVALID
```

```
                                    2 = VALID
                                    4 = SHARED
                                    other values are not allowed
  sleepCtrlState    UINT8         -- sleep control state (option)
                                    0 = option not available
                                    1 = RegularOperation
                                    2 = WaitForSleepMode
                                    3 = PrepareForSleepMode
  sleepReqCnt       UINT8         -- number of sleep requests (option)
                                    value range: 0..63
                                    not used = 0
  opTrnTopoCnt      UINT32        -- operational train topography counter
  safetyTrail       ETBCTRL_VDP   -- ETB-VDP trailer
                                    completely set to 0 = SDTv2 not used

  }
```

E.3.4 ECSP Confirmation/Correction Request

TRDP MD Request message telegram parameters:

MsgType:	‘Mr’
ComId:	122
SMI:	122 (only needed if SDTv2 is used)
UserDataVersion:	‘0100’H (only needed if SDTv2 is used)
SourceURI (user part):	user defined
DestinationURI (user part):	void
DestinationURI (host part):	devECSP.anyVeh.ICst.ICITrn.ITrn
SourceURI (host part):	user defined
Dataset:	ECSP_CONF_REQUEST
Reply timeout:	3,0 s

The TRDP message is defined as shown in Figure E.5.

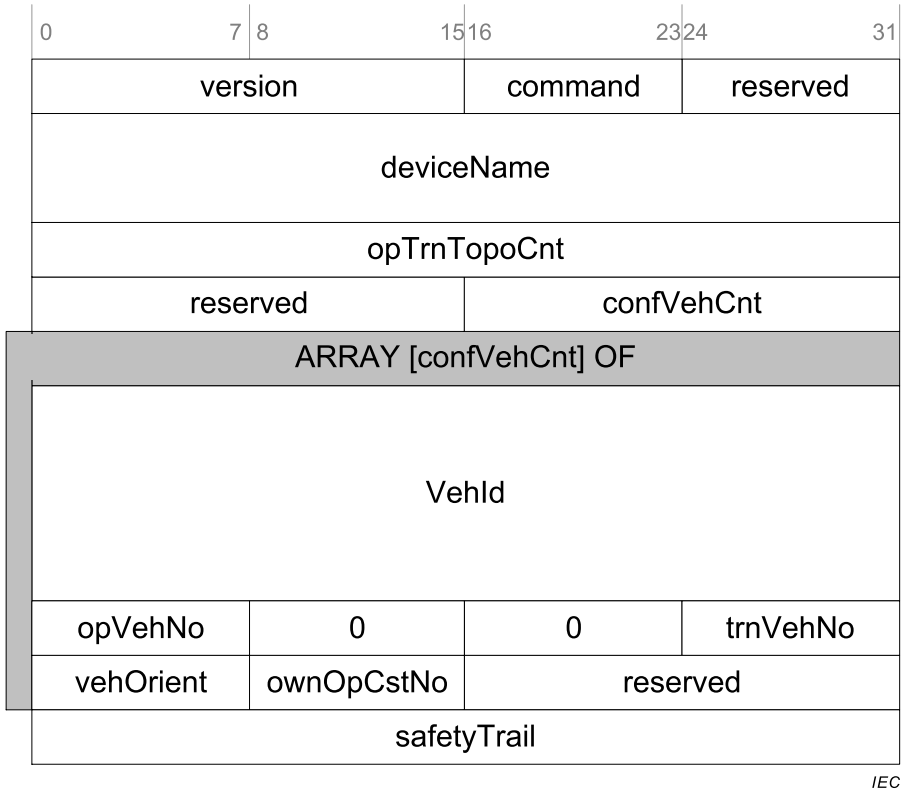


Figure E.5 – ECSP confirm/correction request data

The payload of the TRDP message is defined as:

```
ECSP_CONF_REQUEST::= RECORD
  {
  version          VERSION       -- telegram version information
                                    main version = 1
                                    subversion = 0
  command          UINT8         -- confirmation order
                                    1 = confirmation/correction request
                                    2 = un-confirmation request
  reserved01       UINT8         -- reserved (= 0)
  deviceName       LABEL         -- function device of ECSC which sends
                                    the telegram
  opTrnTopoCnt     UINT32        -- operational train topocounter value
                                    of the operational train directory the
                                    correction is based on
  reserved02       UINT16        -- reserved (= 0)
  confVehCnt       UINT16        -- number of confirmed vehicles in the
                                    train (1..63).
                                    if set to 0: no correction
  confVehList ARRAY [confVehCnt] OF OP_VEHICLE
                                  -- ordered list of
                                    confirmed vehicles in the train,
                                    starting with vehicle at train head,
                                    see chapter 5.3.3.2.10.
                                    Parameters 'isLead' and 'leadDir'
                                    to be set to 0.
  safetyTrail      ETBCTRL_VDP   -- ETB-VDP trailer
                                    completely set to 0 = SDTv2 not used
   }
```

TRDP MD Reply telegram parameters:

MsgType:	‘Mp’
Protocol:	UDP
ComId:	123
SMI:	123 (only needed if SDTv2 is used)
UserDataVersion	‘0100’H (only needed if SDTv2 is used)
Dataset:	ECSP_CONF_REPLY

The TRDP message is defined as shown in Figure E.6.

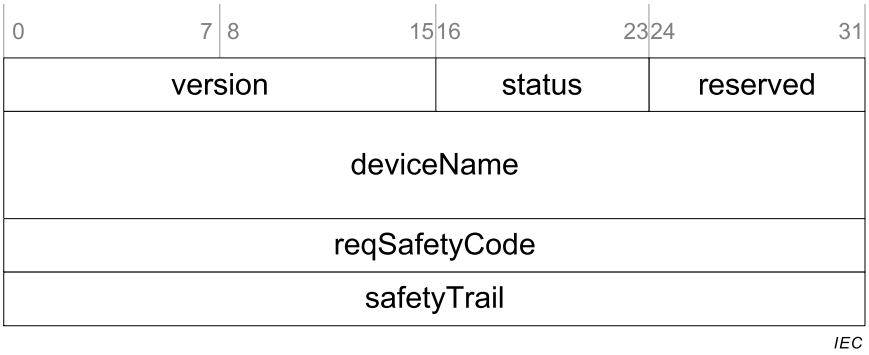


Figure E.6 – ECSP confirm/correction reply data

The payload of the TRDP message is defined as:

```
ECSP_CONF_REPLY::= RECORD
  {
  version          VERSION       -- telegram version information
                                    main version = 1
                                    subversion = 0
  status           UINT8         -- status of storing correction info
                                    0 = correctly stored
                                    1 = not stored
  reserved01       UINT8         -- reserved (= 0)
  deviceName       LABEL         -- function device of ECSC which sends
                                    the telegram
  reqSafetyCode    UINT32        -- SC-32 value of the request message
  safetyTrail      ETBCTRL_VDP   -- ETB-VDP trailer
                                    completely set to 0 = SDTv2 not used
  }
```

E.4 TTDB manager interface

E.4.1 General

The TTDB manager interface defines the TRDP telegrams, which are used to retrieve data from the TTDB. A TRDP-PD telegram is used to inform about the actual operational train directory status, a notification message is sent after each change of the operational train directory. In addition, a set of TRDP-MD telegrams can be used to retrieve data from the TTDB as it is shown in Figure E.7.

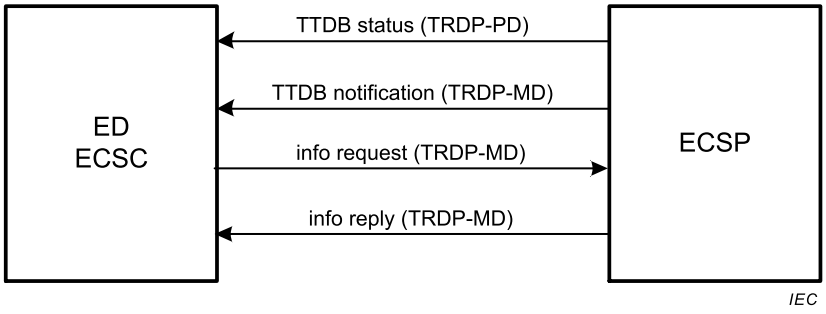


Figure E.7 – TTDB manager interface telegrams

E.4.2 TTDB status information

PD push telegram with operational train status information, pushed from TTDB manager (or a proxy of thereof) to all devices of the consist.

TRDP PD telegram parameters:

MsgType:	‘Pd’
ComId:	100
SMI:	100 (only needed if SDTv2 is used)
UserDataVersion:	‘0100’H (only needed if SDTv2 is used)
DestinationURI (host part):	grpAll.aVeh.lCst.lClTrn.lTrn
Dataset:	TTDB_OP_TRAIN_DIRECTORY_STATUS_INFO
Cycle time	(1,0 ± 0,1) s
Timeout:	5,0 s

The payload of the TRDP telegram is defined as:

```
TTDB_OP_TRAIN_DIRECTORY_STATUS_INFO::= RECORD
  {
  state             OP_TRAIN_DIRECTORY_STATE
                                -- operational train directory state
  etbTopoCnt        UINT32      -- ETB topography counter value of the ETB
                                   the sender of this telegram is
                                   connected to (e.g. ETB0)
```

```
  ownOpCstNo       UINT8          -- own operational consist number
                                     value range: 1..63
  ownTrnCstNo      UINT8          -- own train consist number
                                     value range: 1..63
  reserved02       UINT16         -- reserved for future use (= 0)
  safetyTrail      ETBCTRL_VDP    -- ETB-VDP trailer
                                     completely set to 0 = SDTv2 not used
  }
```

E.4.3 TTDB notification

MD notification message with dynamic train information, pushed from ECSP to all devices of the consist after each TCN inauguration.

TRDP MD Notification telegram parameters:

MsgType:	‘Mn’
Protocol:	UDP
ComId:	101
SourceURI (user part):	user defined
DestinationURI (user part):	void
DestinationURI (host part):	grpAll.aVeh.lCst.lClTrn.lTrn
SourceURI (host part):	user defined
Dataset:	TTDB_OP_TRAIN_DIRECTORY_INFO

The payload of the TRDP message is defined as:

```
TTDB_OP_TRAIN_DIRECTORY_INFO::= RECORD
  {
  opTrnDir         OP_TRAIN_DIRECTORY
                                  -- operational train directory
  }
```

E.4.4 TTDB information – train directory

TRDP-MD telegram to retrieve train directory information of a given ETB.

TRDP MD Request telegram parameters:

MsgType:	‘Mr’
Protocol:	UDP
ComId:	102
SourceURI (user part):	user defined

DestinationURI (user part): void

DestinationURI (host part): devECSP.anyVeh.ICst.ICITrn.ITrn

SourceURI (host part): user defined

Dataset: TTDB_TRAIN_DIRECTORY_INFO_REQUEST

Reply timeout: 3,0 s

The payload of the TRDP message is defined as:

```
TTDB_TRAIN_DIRECTORY_INFO_REQUEST::= RECORD
  {
  etbId       UINT8              -- identification of the ETB the train
                                    directory information is requested for
                                    0 = ETB0 (operational network)
                                    1 = ETB1 (multimedia network)
                                    2 = ETB2 (other network)
                                    3 = ETB3 (other network)
  }
```

TRDP MD Reply telegram parameters:

MsgType: ‘Mp’

ComId: 103

Dataset: TTDB_TRAIN_DIRECTORY_INFO_REPLY

The payload of the TRDP message is defined as:

```
TTDB_TRAIN_DIRECTORY_INFO_REPLY::= RECORD
  {
  trnDir      TRAIN_DIRECTORY    -- dynamic train info
  }
```

E.4.5 TTDB information – static consist information

TRDP-MD telegram to retrieve static consist information of a given consist. Because of the possible large quantity of this information it is recommended to use TRDP-MD on TCP for this transmission.

TRDP MD Request message telegram parameters:

MsgType: ‘Mr’

Protocol: UDP

ComId: 104

SourceURI (user part): user defined

DestinationURI (user part): void

DestinationURI (host part): devECSP.anyVeh.ICst

SourceURI (host part): user defined

Dataset: TTDB_STATIC_CONSIST_INFO_REQUEST

Reply timeout: 3,0 s

The payload of the TRDP message is defined as:

```
TTDB_STATIC_CONSIST_INFO_REQUEST::= RECORD
  {
  cstUUID     UINT8[16]          -- consist unique identification
  }
```

TRDP MD Reply message telegram parameters:

MsgType: 'Mp'

ComId: 105

Dataset: TTDB_STATIC_CONSIST_INFO_REPLY

The payload of the TRDP message is defined as:

```
TTDB_STATIC_CONSIST_INFO_REPLY::= RECORD
  {
  cstInfo     CONSIST_INFO       -- consist information
  }
```

E.4.6 TTDB information – train network directory information

TRDP-MD telegram to retrieve train network directory information of a given ETB.

NOTE It is also possible to read the train network directory directly from the local ETBN.

TRDP MD Request message telegram parameters:

MsgType: 'Mr'

Protocol: UDP

ComId: 106

SourceURI (user part): user defined

DestinationURI (user part): void

DestinationURI (host part): devECSP.anyVeh.ICst

SourceURI (host part): user defined

Dataset: TTDB_TRAIN_NETWORK_DIRECTORY_INFO_REQUEST

Reply timeout: 3,0 s

The payload of the TRDP message is defined as:

```
TTDB_TRAIN_NETWORK_DIRECTORY_INFO_REQUEST::= RECORD
  {
  etbId       UINT8              -- identification of the ETB the train
                                    network directory information is
                                    requested for
                                    0 = ETB0 (operational network)
                                    1 = ETB1 (multimedia network)
                                    2 = ETB2 (other network)
                                    3 = ETB3 (other network)
  }
```

TRDP MD Reply message telegram parameters:

MsgType: ‘Mp’

ComId: 107

Dataset: TTDB_TRAIN_NETWORK_DIRECTORY_INFO_REPLY

The payload of the TRDP message is defined as:

```
TTDB_TRAIN_NETWORK_DIRECTORY_INFO_REPLY::= RECORD
  {
  trnNetDir   TRAIN_NETWORK_DIRECTORY     -- dynamic ETB info
  }
```

E.4.7 Operational train directory information

TRDP-MD telegram to retrieve operational train directory information of a given ETB.

TRDP MD Request message telegram parameters:

MsgType: ‘Mr’

Protocol: UDP

ComId: 108

SourceURI (user part): user defined

DestinationURI (user part): void

DestinationURI (host part): devECSP.anyVeh.ICst

SourceURI (host part): user defined

Dataset: void

Reply timeout: 3,0 s

TRDP MD Reply message telegram parameters:

MsgType: 'Mp'

ComId: 109

Dataset: TTDB_OP_TRAIN_DIRECTORY_INFO_REPLY

The payload of the TRDP message is defined as:

```
TTDB_OP_TRAIN_DIRECTORY_INFO_REPLY::= RECORD
  {
  opTrnDir          OP_TRAIN_DIRECTORY      -- dynamic train info

  }
```

E.4.8 Read TTDB

TRDP-MD telegram to read the train information of the TTDB.

TRDP MD Request message telegram parameters:

MsgType: 'Mr'

Protocol: UDP

ComId: 110

SourceURI (user part): user defined

DestinationURI (user part): void

DestinationURI (host part): devECSP.anyVeh.ICst

SourceURI (host part): user defined

Dataset: TTDB_READ_COMPLETE_REQUEST

Reply timeout: 3,0 s

The payload of the TRDP message is defined as:

```
TTDB_READ_COMPLETE_REQUEST::= RECORD
  {
  etbId       UINT8            --identification of the ETB the TTDB
                                 information is requested for
                                 0 = ETB0 (operational network)
                                 1 = ETB1 (multimedia network)
                                 2 = ETB2 (other network)
                                 3 = ETB3 (other network)
  }
```

TRDP MD Reply message telegram parameters:

MsgType:	'Mp'
ComId:	111
Dataset:	TTDB_READ_COMPLETE_REPLY

The payload of the TRDP message is defined as:

```
TTDB_READ_COMPLETE_REPLY::= RECORD
  {
  state        OP_TRAIN_DIRECTORY_STATE
                                    -- operational train directory state
  opTrnDir     OP_TRAIN_DIRECTORY      -- operational train directory
  trnDir       TRAIN_DIRECTORY         -- train directory
  trnNetDir    TRAIN_NETWORK_DIRECTORY -- train network directory
  }
```

E.5 DNS server interface

E.5.1 DNS standard interface

The DNS server standard interface is specified in RFC 1034 and RFC 1035.

E.5.2 DNS TCN interface

E.5.2.1 General

Instead of using the DNS standard interface, the DNS TCN interface can be used which is more efficient when multiple TCN-URIs need to be resolved.

TCN-URI address resolution is requested by sending a DNS resolving request message. The TCN-DNS will resolve the TCN-URIs contained within the DNS resolving request message and will return a list with related IP addresses.

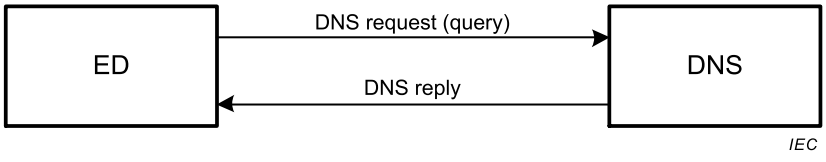


Figure E.8 – TCN-URI resolving

E.5.2.2 TCN_URI Record

The host part of a TCN-URI is defined in accordance to 5.4.4 as a string of the following form:

```
fctdevLabel.vehLabel.cstLabel.cltrnLabel.trnLabel
```

Hence the maximal length of host part character string is (5 × 15) + 4 +'/0' = 80 character (ASCII NUL terminated string).

For the DNS TCN interface a simplified resource record, called TCN_URI record is defined:

```
TCN_URI::= RECORD
  {
  tcnUriStr         CHAR[80]       -- Host part of TCN-URI, terminated with
                                      ASCII character NUL
  resolvState       INT16          -- request message: reserved (= 0)
                                      reply message:
                                      0 = OK
                                      -1 = unknown TCN-URI
  tcnUriIpAddr      UINT32         -- IP address related to query
                                      unicast address (source and destination
                                      scope, address range) or
                                      MC group address (destination scope)
                                      value = 0 if 'resolvState' < 0
  tcnUriIpAddr2     UINT32         -- 2nd IP address in case the query
                                      produces an IP address range
                                      (see 5.4.4.6.2).
                                      Defines the higher IP address of the
                                      range.
                                      To be set to 0 if not resolved to an
                                      IP address range
  }
```

This record is filled in the DNS query (DNS resolving request message) with the TCN-URI string and will be complemented with the related IP address by the DNS server (resolver) in the reply (DNS resolving reply message).

E.5.2.3 DNS resolving request message

The request for resolving TCN-URI is defined as:

TRDP MD Request message telegram parameters:

MsgType:	'Mr'
Protocol:	UDP
ComId:	140
SMI:	140 (only needed if SDTv2 is used)
UserDataVersion:	'0100'H (only needed if SDTv2 is used)
SourceURI (user part):	user defined
DestinationURI (user part):	Void
DestinationURI (host part):	devDNS.anyVeh.ICst.ICITrn.ITrn
SourceURI (host part):	user defined
Dataset:	DNS_REQUEST
Reply timeout:	3,0 s

The TRDP message is defined as shown in Figure E.9.

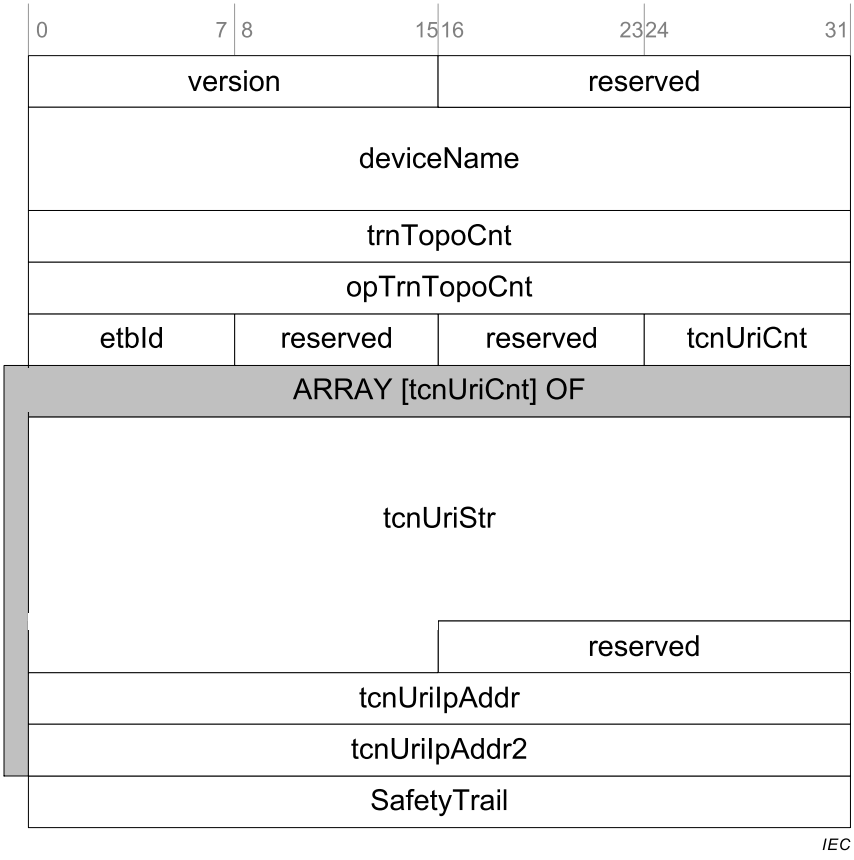


Figure E.9 – DNS resolving request data

The payload of the TRDP message is defined as:

```
DNS_REQUEST::= RECORD
  {
  version          VERSION       -- telegram version information
                                    mainVersion = 1
                                    subVersion = 0
  reserved01       INT16         -- reserved (= 0)
  deviceName       LABEL         -- function device of ED which sends
                                    the telegram
  trnTopoCnt       UINT32        -- train topography counter
  opTrnTopoCnt     UINT32        -- operational train topography counter
                                    needed for TCN-URIs related to the
                                    operational train view
                                    = 0 if not used
  etbId            UINT8         -- identification of the related ETB
                                    0 = ETB0 (operational network)
                                    1 = ETB1 (multimedia network)
                                    2 = ETB2 (other network)
                                    3 = ETB3 (other network)
                                    255 = don't care (for access to
                                           local DNS server)
  reserved02       UINT8         -- reserved (= 0)
  reserved03       UINT8         -- reserved (= 0)
  tcnUriCnt        UINT8         -- number of TCN-URIs to be resolved
                                    value range: 0 .. 255
```

```
  tcnUriList       ARRAY[tcnUriCnt] OF TCN_URI
                                    -- List of TCN_URI records
  safetyTrail      ETBCTRL_VDP     -- ETB-VDP trailer
                                       completely set to 0 = SDTv2 not used
  }
```

TRDP MD Reply telegram parameters:

MsgType:	‘Mp’
ComId:	141
SMI:	141 (only needed if SDTv2 is used)
UserDataVersion	‘0100’H (only needed if SDTv2 is used)
Dataset:	DNS_REPLY

The TRDP message is defined as shown in Figure E.10.

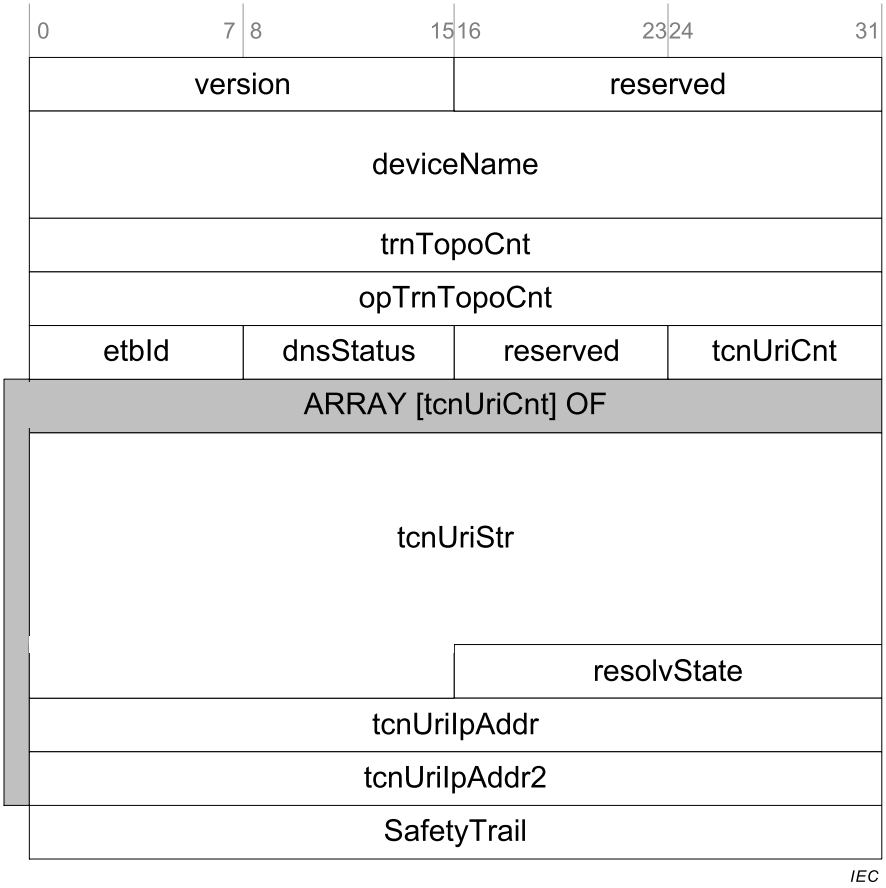


Figure E.10 – DNS resolving reply data

The payload of the TRDP message is defined as:

```
DNS_REPLY::= RECORD
  {
  version          VERSION       -- telegram version information
```

```
                                    mainVersion = 1
                                    subVersion = 0
reserved01          UINT16          -- reserved (= 0)
deviceName          LABEL           -- function device of DNS which sends
                                       the telegram
trnTopoCnt          UINT32          -- train topography counter
opTrnTopoCnt        UINT32          -- operational train topography counter
                                       = 0 if not known
etbId               UINT8           -- identification of the related ETB
                                       0 = ETB0 (operational network)
                                       1 = ETB1 (multimedia network)
                                       2 = ETB2 (other network)
                                       3 = ETB3 (other network)
replyStatus         INT8            -- status of reply:
                                       0 = OK
                                       -1 = DNS Server not ready
                                       -2 = Inauguration in progress
reserved02          UINT8           -- reserved (= 0)
tcnUriCnt           UINT8           -- number of resolved TCN-URIs
                                       value range: 0 .. 255
tcnUriList          ARRAY[tcnUriCnt] OF TCN_URI
                                    -- List of TCN_URI records
safetyTrail         ETBCTRL_VDP     -- ETB-VDP trailer
                                       completely set to 0 = SDTv2 not used
}
```

E.6 ETBN control interface

E.6.1 General

The ETBN control interface defines the TRDP telegrams which are used to control the ETBN and to read status information from the ETBN. Additionally it provides telegrams for reading information from the ETBN as it is shown in Figure E.11.

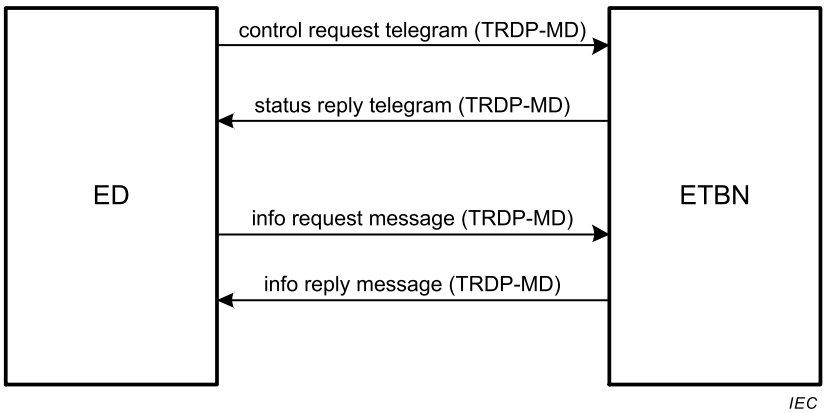


Figure E.11 – ETBN control interface telegrams

E.6.2 ETBN control and status data

TRDP MD Request telegram parameters:

MsgType:	‘Mr’
Protocol	UDP
ComId:	130
SMI:	130 (only needed if SDTv2 is used)
UserDataVersion:	‘0100’H (only needed if SDTv2 is used)
SourceURI (user part)	void
DestinationURI (user part)	void
DestinationURI (host part):	a specific ETBN in the consist
Dataset:	ETBN_CTRL
Reply timeout:	3,0 s

The TRDP message is defined as shown in Figure E.12.

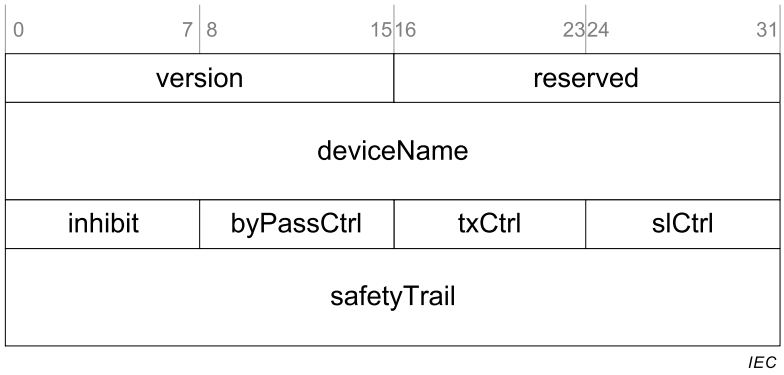


Figure E.12 – ETBN control request data

The payload of the TRDP message is defined as:

```
ETBN_CTRL::= RECORD
  {
  version          VERSION       -- telegram version information
                                    mainVersion = 1
                                    subVersion = 0
  reserved01       UINT16        -- reserved (= 0)
  deviceName       LABEL         -- function device of ED which sends
                                    the telegram
  inhibit          UINT8         -- ETBN inhibit
                                    0 = no action (keep old state)
                                    1 = no inhibit request
                                    2 = inhibit request
  byPassCtrl       UINT8         -- ETBN bypass control
```

```
                                    0 = no action (keep old state)
                                    1 = no bypass
                                    2 = activate bypass
  txCtrl            UINT8         -- ETBN transmission control
                                    0 = no action (keep old state)
                                    1 = activate sending on ETB (default)
                                    2 = stop sending on ETB
  slCtrl            UINT8         -- sleep mode control (option)
                                    0 = no action (keep old state)
                                    1 = deactivate sleep mode
                                    2 = activate sleep mode (line
                                        activity sensing)
  safetyTrail       ETBCTRL_VDP   -- ETB-VDP trailer
                                    completely set to 0 = SDTv2 not used
  }
```

TRDP MD Reply telegram parameters:

MsgType:	‘Mp’
ComId:	131
SMI:	131 (only needed if SDTv2 is used)
UserDataVersion:	‘0100’H (only needed if SDTv2 is used)
Dataset:	ETBN_STATUS

The TRDP message is defined as shown in Figure E.13.

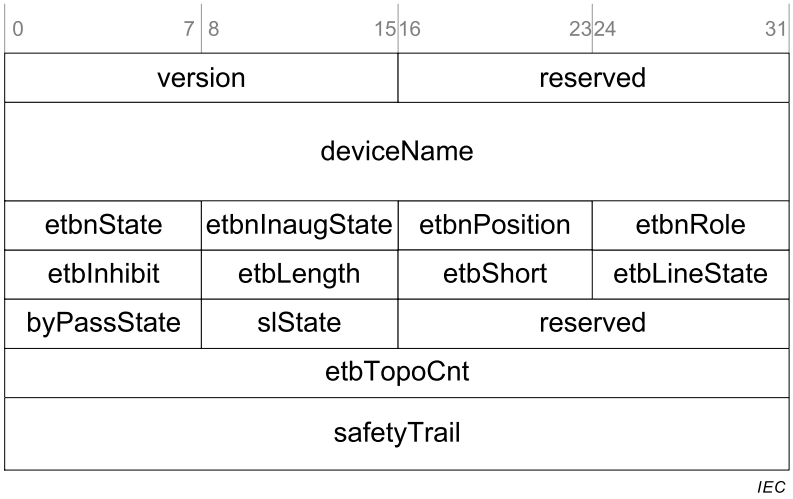


Figure E.13 – ETBN status reply data

The payload of the TRDP message is defined as:

```
ETBN_STATUS::= RECORD
  {
  version          VERSION       -- telegram version information
                                    version = 1
                                    release = 0
  reserved01       UINT16        -- reserved (= 0)
  deviceName       LABEL         -- function name of ETBN
  etbnState        UINT8         -- state indication of the (active) ETBN
                                    0 = ETBN not operational(initial value)
                                    1 = ETBN in operation
  etbnInaugState   UINT8         -- ETBN inauguration state as defined in
                                    IEC 61375-2-5
                                    0 = init
                                    1 = not inaugurated
                                    2 = inaugurated
                                    3 = ready for inauguration
  etbnPosition     UINT8         -- position of the ETBN
                                    0 = unknown (default)
                                    1 = single node
                                    2 = middle node
                                    3 = end node TCN direction 1
                                    4 = end node TCN direction 2
  etbnRole         UINT8         -- ETBN node role as defined in
                                    IEC 61375-2-5
                                    0 = undefined
                                    1 = master (redundancy leader)
                                    2 = backup (redundancy follower)
                                    3 = not redundant
  etbInhibit       UINT8         -- inauguration inhibit indication
                                    0 = n/a (default)
                                    1 = inhibit not requested on ETB
                                    2 = inhibit set on local ETBN
                                    3 = inhibit set on remote ETBN
                                    4 = inhibit set on local and remote
                                        ETBN
  etbLength        UINT8         -- indicates train lengthening in case
                                    train inauguration is inhibit
                                    0 = no lengthening (default)
                                    1 = lengthening detected
  etbShort         UINT8         -- indicates train shortening in case
                                    train inauguration is inhibit
                                    0 = no shortening (default)
                                    1 = shortening detected
  etbLineState     BITSET8       -- indication of ETB line status
                                    (FALSE == not trusted, TRUE == trusted)
                                    bit0 = line A ETBN direction 1
                                    bit1 = line B ETBN direction 1
                                    bit2 = line C ETBN direction 1
                                    bit3 = line D ETBN direction 1
                                    bit4 = line A ETBN direction 2
                                    bit5 = line B ETBN direction 2
                                    bit6 = line C ETBN direction 2
                                    bit7 = line D ETBN direction 2
  byPassState      UINT8         -- state of bypass function
                                    0 = bypass disabled
```

```
                                    1 = bypass enabled
  slState          UINT8          -- sleep mode state (option)
                                    0 = no sleep mode
                                    1 = sleep mode active (line activity
                                        sensing)
  reserved02       UINT16         -- reserved (= 0)
  etbTopoCnt       UINT32         -- ETB topography counter
  safetyTrail      ETBCTRL_VDP    -- ETB-VDP trailer
                                    completely set to 0 = SDTv2 not used
  }
```

NOTE As defined in IEC 61375-2-5 ETB lines ‘C’ and ‘D’ are optional.

E.6.3 ETBN train network directory

TRDP-MD telegram to retrieve train network directory information from ETBN.

TRDP MD Request message telegram parameters:

MsgType:	‘Mr’
Protocol	UDP
ComId:	132
SMI:	132 (only needed if SDTv2 is used)
UserDataVersion:	‘0100’H (only needed if SDTv2 is used)
SourceURI (user part):	user defined
DestinationURI (user part):	void
DestinationURI (host part):	a specific ETBN in the consist
SourceURI (host part):	user defined
Dataset:	void
Reply timeout:	3,0 s

TRDP MD Reply message telegram parameters:

MsgType:	‘Mp’
ComId:	133
SMI	133 (only needed if SDTv2 is used)
UserDataVersion:	‘0100’H (only needed if SDTv2 is used)
Dataset:	ETBN_TRAIN_NETWORK_DIRECTORY_INFO_REPLY

The payload of the TRDP message is defined as:

```
ETBN_TRAIN_NETWORK_DIRECTORY_INFO_REPLY::= RECORD
  {
  trnNetDir    TRAIN_NETWORK_DIRECTORY      -- dynamic train info

  }
```

Annex F
(normative)

Communication profile conformance test guideline

F.1 General

This annex provides a methodological framework for conformance and interoperability test of consists connected on ETB. The conformance test checks the conformity to the protocols specified in this part of IEC 61375 by stimulating the consist interface on ETB level as shown in Figure F.1. Conformance to the communication profile is one of the building blocks necessary to achieve interoperability between consists on ETB level.

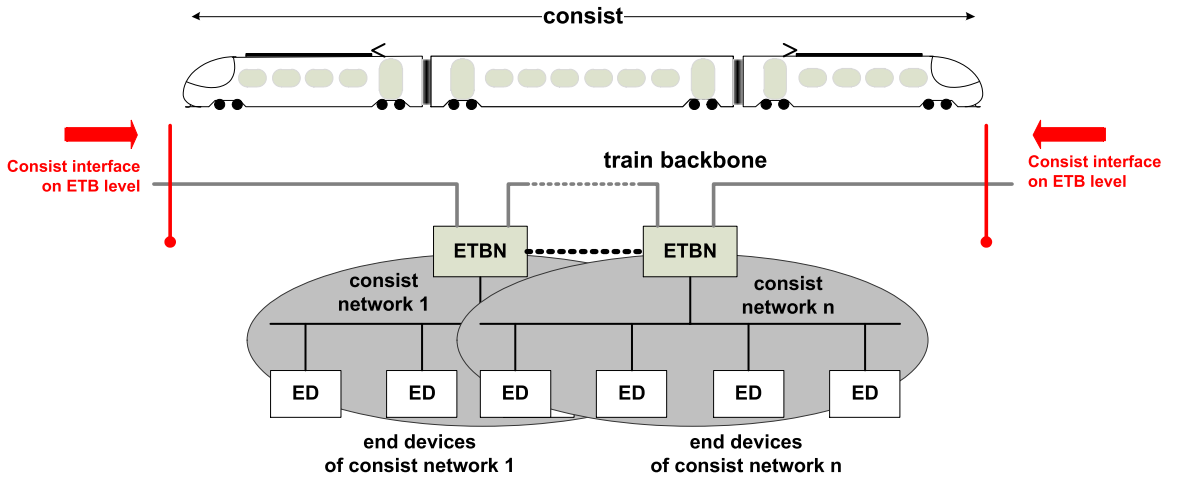


Figure F.1 – Consist interface on ETB level

More precisely, the conformity test objective is to test:

- Conformity with respect to the services of the communication profile as defined within this part of the standard.
- Conformity to TRDP as a protocol used on ETB for data exchange between abstract functions.

NOTE 1 The difference between “conformance test” and “interoperability test” is that with the conformance test a UuT is tested against the protocol specification, while for interoperability testing two devices implementing the protocol are connected in order to test whether these devices are able to properly exchange and interprete data. Hence, a properly performed conformance test will supersede an interoperability test, while an interoperability test alone will not be sufficient to prove conformity to a protocol.

NOTE 2 It is not in the scope of this standard to provide a guideline of testing conformity of end devices to functions or services specified in this part of the standard, e.g. for testing TRDP conformity of end devices.

F.2 Scope of conformance test

A complete conformance test of consists connected on ETB encompasses both communication functions as defined in this part of the standard as well as functions defined in IEC 61375-2-4 (application profile). There is however an overlapping with respect to end-to-end communication protocols like TRDP and SDTv2, which can specifically only be tested in conjunction with the application functions using them (Figure F.2). Performing a generic conformance test for those protocols without involvement of the specific application can therefore never provide complete conformance test coverage.

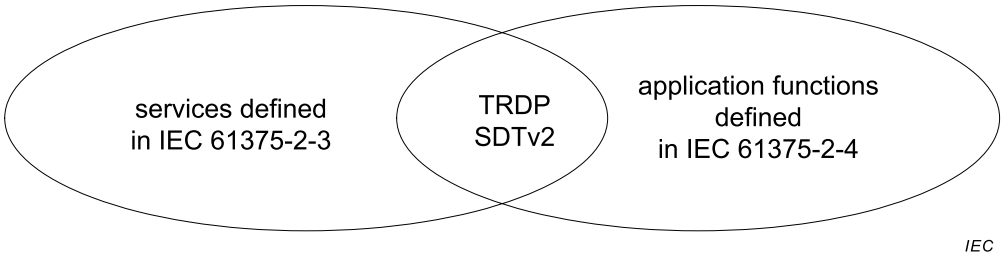


Figure F.2 – Scope of conformance test

Consequently, performing a generic conformance test for those protocols as specified in this part of the standard will be a necessary, but not sufficient condition for a complete conformance test.

F.3 Conformance test overview

Subject of conformance testing are:

- ETB control service, as defined in Clause 6
- TCN-DNS service, as defined in 5.5
- Train info service, as defined in 5.8
- TRDP, as defined in Annex A
- SDTv2, as defined in Annex B

F.4 Test laboratory

F.4.1 General

Conformance tests shall only be executed by a laboratory operating in accordance to ISO/IEC 17025.

An accreditation of the test laboratory executed by an independent accreditation body that operates in accordance with ISO/IEC 17011 is recommended.

F.4.2 Tasks

The test laboratory shall take responsibility for the following activities:

– Establish a process for TCN communication profile conformance testing.
– Provide a conformance test specification.
– Provide a test bed for conformance testing.
– Provide instructions for UuT conformance test preparation.
– Execute conformance test sessions on agreement between test laboratory and the UuT provider.
– Prepare a detailed conformance test report.
– In case of a positive test result: provide a certificate stating conformity of the UuT to this part of the standard.

F.5 Guideline for writing conformance test specifications

F.5.1 Overview of the main components

The main components of a test specification include:

- an identification of the base standard(s) to which the test specification applies;
- an identification of the relevant Protocol Implementation Conformance Statement (PICS) proforma which describes standards requirements and the supplier's support for these requirements, e.g. with respect to supported options;
- a specification of the abstract test architecture;
- a Protocol Implementation eXtra Information for Testing (PIXIT) proforma which provides additional information regarding parameters and values or their ranges;
- a test suite structure identifying each test that is to be defined and a grouping of the tests into a structure which is appropriate to the particular UuT;
- a test purpose for each identified test. A test purpose consists of:
  - a unique test purpose identifier;
  - a status specifying whether the test purpose is mandatory or optional according to the associated PICS;
  - references (usually by (sub)section number and paragraph number) to the requirements in the base standard(s) covered by the test purpose;
  - a description of the purpose of the test in plain language;
- for each test a test description which comprises:
  - a unique test identifier;
  - a brief description of the objective of the test, making reference to the test purpose included;
  - references to the requirements in the base standard(s) covered by the test (taken directly from the associated test purpose);
  - references to the individual PICS and PIXIT items covered by the test;
  - pre-conditions describing the static configuration of the test scenario;
  - the test procedure in a series of steps written in plain or structured language. This should include any preamble to establish the UuT into a known initial condition prior to the test and postamble to return it to a specific quiescent state after the completion of the test;
  - for each step, a description of the results which constitute a PASS condition.

As their name implies, guidelines are only for guidance and the actual process followed should use and adapt whichever of these guidelines are most applicable in each particular situation. In some cases this may mean the application of all aspects.

F.5.2 Protocol Implementation Conformance Statement (PICS)

The purpose of a PICS is to identify those standardized functions which an UuT shall support, those which are optional and those which are conditional on the presence of other functions. Although not strictly part of the TCN communication profile test suite, the PICS helps to provide a means for selection of the suite of tests which will subsequently be developed.

In addition, the PICS can be used as a proforma by a TCN communication profile implementation supplier in identifying which functions an UuT will support when interoperating with similar equipment from other suppliers.

The information in a PICS is usually presented in tabular form as recommended in ISO/IEC 9646-7.

The PICS can be considered as a set of “"switches”" which specify the capability of supporting the requirement in base standards to be tested. It is possible that with different choices in a PICS proforma, several different set of TPs will be necessary.

F.5.3 Abstract test architecture

An abstract test architecture provides a general framework within specific test arrangements which must fit in order to perform the specified suite of tests. It describes the entities involved with the protocol being tested and their means of communication, for example, the definition and placement of UuT, test components and test drivers.

Abstract test architectures can be depicted through diagrams or equivalent tables. An example abstract test architecture is provided in Clause F.6.

F.5.4 Protocol Implementation eXtra Information for Testing (PIXIT)

A Protocol Implementation eXtra Information for Testing (PIXIT) proforma identifies which PICS items are to be tested and which parameters to be instantiated for the TSS&TP being developed.

The production of a PIXIT Proforma is specified in ISO/IEC 9646-6. A supplier, providing an UuT for conformance testing, is required to complete a PIXIT proforma, which contains additional questions that need to be answered in order to turn on/off the "switches” and identify means of testing for a particular UuT.

F.5.5 Test suite structure

There is no hard and fast rule that can be used to determine how a test suite should be divided up into test groups other than to say that there should be a logical basis to the choice of groups. In many cases, the division will be rather arbitrary and based on the preferences of the author(s). However, the following categorizations should be considered when identifying appropriate test groups within a test suite structure:

- Abstract test configuration: A test group for each valid configuration specified. For example:
  - UuT as end node in varying train compositions;
  - UuT as intermediate node in varying train compositions.
- Functionality: A test group for each of the major functions (and sub-functions, when necessary) supported. For example:
  - CSTINFO telegram exchange;
  - Computation of TTDB;
  - ETBCTRL packet exchange;
  - ETB control – function leading;
  - ETB control – function sleep mode;
  - ETB control – function confirmation/correction;
  - TCN-DNS service.
- Success, inclusive or failure: test groups for normal behavior, inclusive behavior and exceptional behavior.

F.6 Abstract test architecture (option)

F.6.1 General

The following test setups are examples only, but it is recommended to follow these examples in real implementations because they define a standard way of TCN communication profile conformance testing.

F.6.2 Test architecture with one ETB

A principal abstract architecture is shown in Figure F.3. Here, the UuT is connected to the tester only via one ETB and a power connector, which allows to switch the UuT on/off. This principal abstract architecture is suited for an automated test execution.

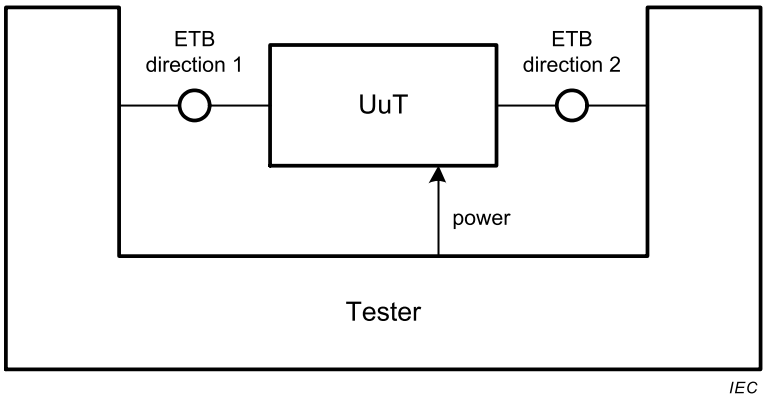


Figure F.3 – Abstract test architecture (1 ETB)

F.6.3 Test architecture for multiple ETB

The test architecture described in F.6.2 can be extended for supporting multiple ETBs, as in the example of Figure F.4 shown for two ETBs.

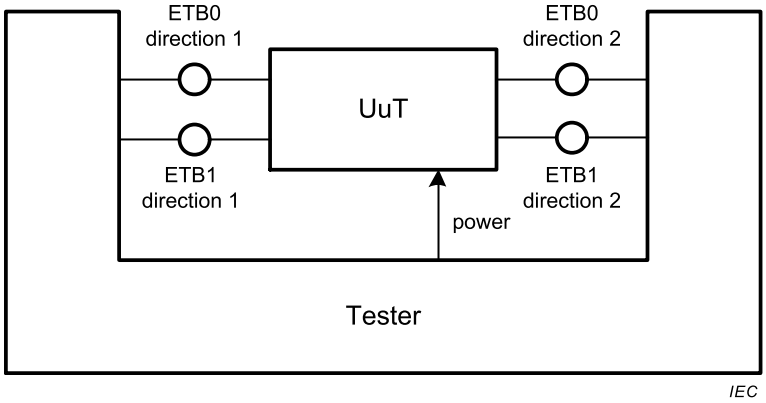


Figure F.4 – Abstract test architecture (2 ETBs)

F.6.4 Set-up for automatic test

An automatic test requires that the UuT is instrumented with a test function which receives and executes test stimuli generated by the tester. The principal architecture of an instrumented UuT is shown in Figure F.5. The UuT contains the ECSP, the TCN-DNS server and the ETBN for connection to the ETB. In addition, it provides a test function which allows to emulate application functions and the ECSC to an extend which is required for the test. The test control implements the interface to the tester. Via this interface, test control commands containing test stimuli are received and, in reply, test status reports are returned.

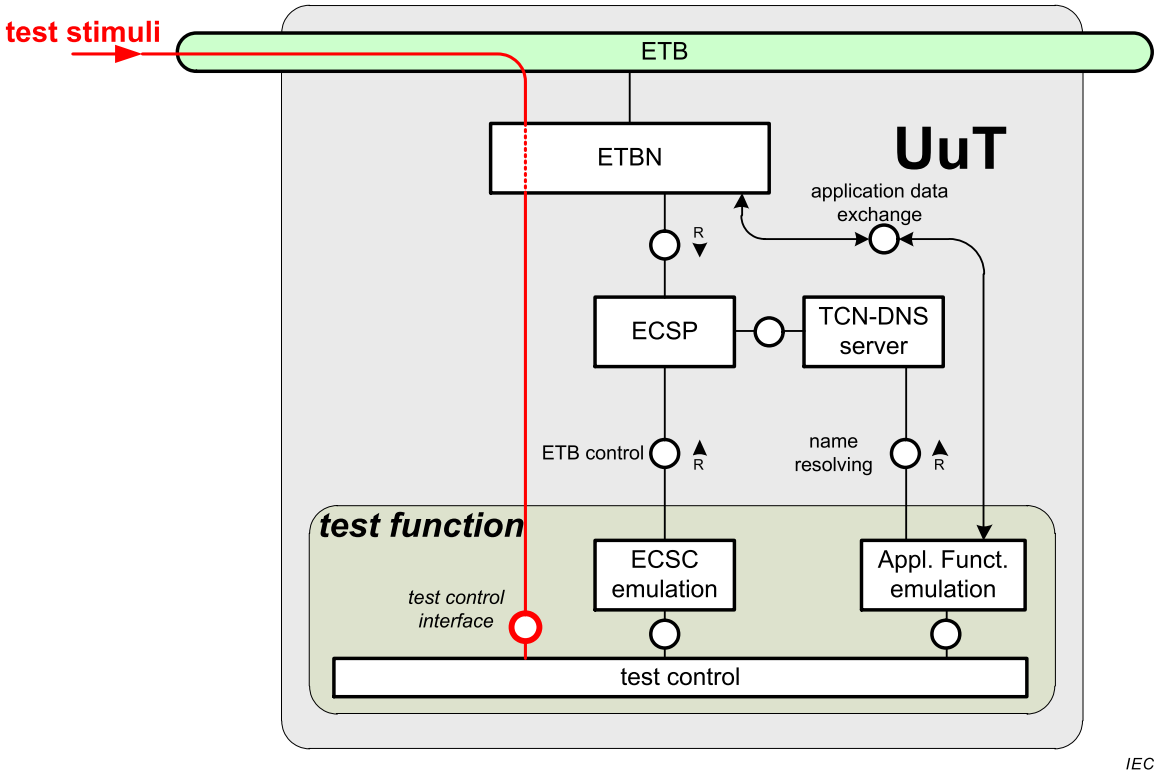


Figure F.5 – Unit under test abstract architecture

In closed trains supporting multiple consists multiple instances of the test functions may be implemented.

F.7 Test of conformity to the common ETB framework

F.7.1 General

The objective of this clause is to give guidance for testing conformity of communication devices (end devices or network devices) to the common ETB framework as defined in Clause 5. Dependant on the device type some functions and definitions of the common ETB framework may have to be considered or not. This will be explicitely mentioned in the following subclauses.

F.7.2 Test of CSTINFO telegram

A functional test of the CSTINFO telegram exchange is only affecting ECSP devices and is implicitely done with the test of the ETB control services as defined in Clause F.8.

F.7.3 Test of TTDB

A test of the TTDB is only affecting ECSP devices and is implicitely done with the test of the ETB control services as defined in Clause F.8.

Additional tests may be executed for testing the consist local access to the TTDB, as for instance defined within Annex E.

F.7.4 Test of service addressing and TCN-DNS server

Service addressing uses abstract functional addresses based on TCN-URI schemes as defined in 5.4. However, most protocols used for train wide communication (including TRDP) are solely based on IP addressing. The process of resolving TCN-URI to IP addresses is a consist

internal function, which has no direct impact on the interoperability on ETB level. Therefore it is in the resonsibility of the consist manufacturer to perform related tests. Only in cases where the TCN-URI is part of the user data (e.g. used within the payload of TRDP-PD or TRDP-MD telegrams), ETB interoperability will be affected. For those cases a conformity test of TCN-URI resolution shall be executed, which may be restricted to use cases defined on application level, e.g. defined by IEC 61375-2-4 or IEC 62580.

The communication test control interface defined in Clause F.8 provides a possibility to test the TCN-URI address resolution.

The support and test of TCN-URI shall be stated in the PICS.

F.7.5 Test of data exchange

F.7.5.1 TRDP protocol compliance

TRDP (Annex A) defines a generic end-to-end protocol with implementations on all end devices which are involved in process data and message data communication over ETB. TRDP protocol compliance shall be proven by a dedicated TRDP conformance test.

It shall be within the scope of the test laboratory offering a TRDP conformance test to define the necessary test setup and the test cases to be executed. The test set-up may deviate from the test architecture defined in Clause F.6.

NOTE Most often a simple computer is used for tester to which the UuT is directly connected.

F.7.5.2 SDTv2 protocol compliance

SDTv2 (Annex B) defines a generic end-to-end protocol with implementations on all end devices which are involved in safe process data communication over ETB. SDTv2 protocol compliance shall be proven by a dedicated SDTv2 conformance test.

It shall be within the scope of the test laboratory offering a SDTv2 conformance test to define the necessary test setup and the test cases to be executed. The test set-up may deviate from the test architecture defined in Clause F.6.

NOTE Most often a simple computer is used for tester and the UuT is directly connected to it.

F.7.5.3 End-to-end data communication

The objectives of an end-to-end data communication test are:

- Test the proper setup of the IP routers within the ETBN.
- Test the proper configuration of the ECN for correctly passing telegrams to the intended destination.
- Test the proper configuration of the ED (test application) for correctly accepting telegrams intended for the test application.
- Test the proper data marshalling by the destination.
- Test the TCN-URI address resolution of the UuT.

The end-to-end data communication test does not have the objective to test compliance to TRDP and SDTv2 protocols.

Testing end-to-end data communication within a train has to deal with different aspects:

- Transmission protocol, which can be any IP based protocol (e.g. TRDP, streaming protocols, etc.).
- Addressing (IP addresses, or if TCN-URI is used, also the resolving process).

- Train composition changes (inaugurations).
- Transversal communication, e.g. communication between devices connected to the operational network and devices connected to the multimedia network.

The communication profile conformance test shall include tests for generic communication szenarios considering these aspects.

These tests are necessary, but not sufficient. They shall be supplemented by tests on application level, see Clause F.2.

The echo function defined in Clause F.9 provides a possibility to test generic communication patterns.

F.7.6 Test of service discovery

Service discovery as defined in 5.7 relies on the availability of service information within the CSTINFO telegrams. Service information content of CSTINFO is specified in IEC 61375-2-4, therefore a test of service discovery will be in the scope of IEC 61375-2-4.

F.7.7 Test of train info service

The train info service as defined in 5.8 is a consist local function and has no effect on ETB interoperability. Therefore it shall be in the responsibility of the consist manufacturer to execute those tests. The execution of those tests shall be stated in the PICS.

F.8 ETB Control Service conformity test

F.8.1 General

The objective of this clause is to give guidance for testing conformity to the ETB control services as specified in Clause 6. As those services are related to consist and train operation, a stronger focus is set on the test environment in order to generate a common standard for conformance testing. This includes the adoption of the test setup as defined in Clause F.6 and the definition of a common test control interface.

The definition of a test control interface aims to define a standard way of conformance test execution. It intends to avoid a situation that each test laboratory defines its own interface, making it difficult to change the test laboratory once this would be necessary.

In addition, a common set of test cases shall be defined.

NOTE A new standard part will be proposed for the definition of the conformance test cases.

The test control interface is defined by a set of TRDP telegrams which are exchanged between the Tester and the UuT's test function over ETB. Besides this interface, there is no other interface required.

F.8.2 Test control interface for the test of ETB control services

F.8.2.1 Test scope

The test of communication services aims to check the following functions:

- Proper detection of (different) train topologies.
- Handling of leading vehicle requests.
- Handling of confirmation and correction.
- Handling of sleep request.

F.8.2.2 Test control telegram

The Test Control Telegram is cyclically sent point-to-point from the tester to the UuT (ECSC emulation) (Figure F.6). Upon reception of a command from the tester, the UuT (ECSC emulation) shall forward this command to its ECSP for execution.

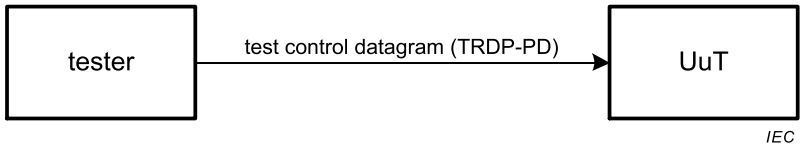


Figure F.6 – Conformance test control telegram

TRDP PD telegram parameters:

Message type:	'Pd'
ComId:	80
Destination:	IP address of UuT device executing the test function (unicast)
Dataset:	CONFTEST_CTRL
Cycle time:	(1,0 ± 0,1) s
Timeout:	5,0 s

If the receiving UuT detects a timeout, the UuT shall reset waiting for the start of a new test run.

The TRDP telegram is defined as shown in Figure F.7.

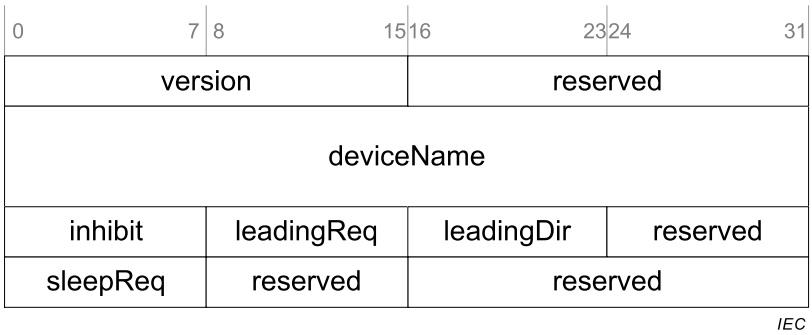


Figure F.7 – Conformance test control telegram data

The payload of the TRDP process data is defined as follows:

```
CONFTEST_CTRL::= RECORD
  {
  version          VERSION       -- telegram version information
                                    main version = 1
                                    subversion = 0
  reserved01       UINT16        -- reserved (= 0)
```

```
  deviceName       LABEL         -- function device of tester which sends
                                    the telegram
  inhibit          UINT8         -- inauguration inhibit
                                    0 = no inhibit request
                                    1 = inhibit request
  leadingReq       UINT8         -- leading request
                                    0 = no leading request
                                    1 = leading request
  leadingDir       UINT8         -- leading direction
                                    0 = no leading request
                                    1 = leading request direction 1
                                    2 = leading request direction 2
  reserved02       UINT8         -- reserved (= 0)
  sleepReq         UINT8         -- sleep request
                                    0 = no sleep request
                                    1 = sleep request
  reserved03       UINT8         -- reserved (= 0)
  reserved04       UINT16        -- reserved (= 0)
  }
```

F.8.2.3 Test status telegram

The Test Control Data Telegram is cyclically sent point-to-point from the UuT to the tester (Figure F.8).

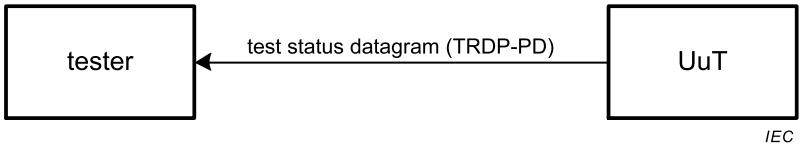


Figure F.8 – Conformance test status telegram

TRDP PD telegram parameters:

Message type:	‘Pd’
ComId:	81
Destination:	test function of the tester
Dataset:	CONFTEST_STATUS
Cycle time:	(1,0 ± 0,1) s
Timeout:	5,0 s

The TRDP telegram is defined as shown in Figure F.9.

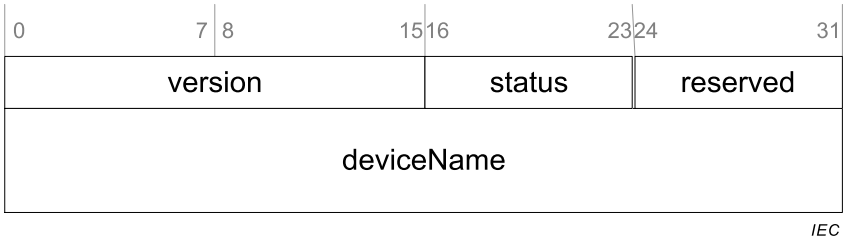


Figure F.9 – Conformance test status telegram data

The payload of the TRDP process data message is defined as follows:

```
CONFTEST_STATUS::= RECORD
  {
  version          VERSION       -- telegram version information
                                    main version = 1
                                    subversion = 0
  status           UINT8         -- status of the UuT
                                    0 = ready
                                    > 1 = error detected (UuT specific)
  reserved01       UINT8         -- reserved for future use (= 0)
  deviceName       LABEL         -- function device of UuT which sends
                                    the telegram
  }
```

F.8.2.4 Confirmation request

F.8.2.4.1 General

This message is sent by the tester to request (un-)confirmation and to inform the UuT about corrections rules. This is done by sending a corrected sequence of vehicles. Vehicle gaps are identified by parameter ‘trnVehNo’ set to 0, see Figure F.10.

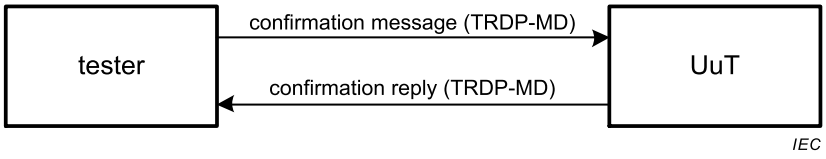


Figure F.10 – (Un-)confirmation request

F.8.2.4.2 Request message

TRDP MD request message parameters:

MsgType:	‘Mr’
Protocol:	UDP
ComId:	82
SourceURI (user part):	“ComProfTester”
DestinationURI (user part):	“ComProfTestAppl”

DestinationURI (host part):	test function of the tester
SourceURI (host part):	user defined
Dataset:	CONFTEST_CONF_REQUEST
ReplyTimeOut:	Defined by tester. Default value: 2,5 s

The TRDP message is defined as shown in Figure F.11

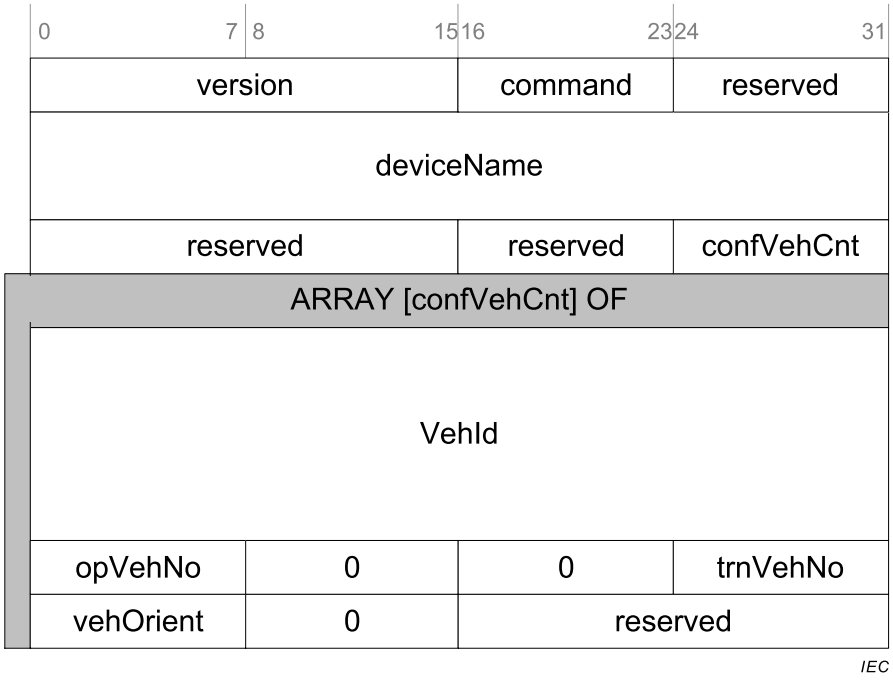


Figure F.11 – Conformance test confirmation/correction request data

The payload of the TRDP request message is defined as follows:

```
CONFTEST_CONF_REQUEST::= RECORD
  {
  version          VERSION       -- telegram version information
                                    main version = 1
                                    subversion = 0
  command          UINT8         -- confirmation order
                                    0 = no command (message to be ignored)
                                    1 = confirmation/correction request
                                    2 = un-confirmation request
  reserved01       UINT8         -- reserved (= 0)
  deviceName       LABEL         -- device name of tester which sends
                                    the telegram
  reserved02       UINT16        -- reserved (= 0)
  reserved03       UINT8         -- reserved (= 0)
  confVehCnt       UINT8         -- number of confirmed vehicles in the
                                    train (1..63).
                                    if set to 0: no correction
  confVehList      ARRAY [confVehCnt] OF OP_VEHICLE
                                 -- ordered list of
```

```
                                    confirmed vehicles in the train,
                                    starting with vehicle at train head,
                                    see chapter 5.3.3.2.10.
                                    Parameters 'isLead', 'leadDir'
                                    and 'ownOpCstNo' to be set to 0.
  }
```

F.8.2.4.3 Reply message

TRDP MD reply message parameters:

ComId:	83
Dataset:	CONFTEST_CONF_REPLY

The TRDP message is defined as shown in Figure F.12.

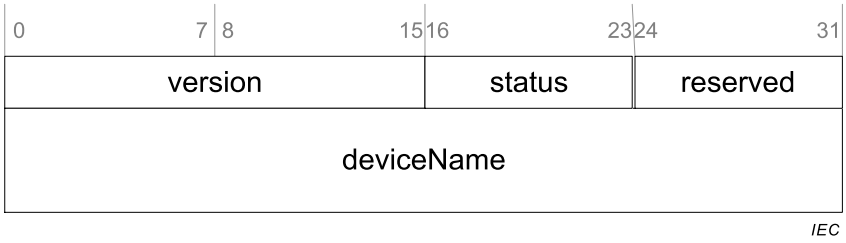


Figure F.12 – Conformance test confirmation/correction reply data

The payload of the TRDP reply message is defined as follows:

```
CONFTEST_CONF_REPLY::= RECORD
  {
  version          VERSION       -- telegram version information
                                    main version = 1
                                    subversion = 0
  status           UINT8         -- status of storing correction info
                                    0 = correctly stored
                                    1 = not stored
  reserved01       UINT8         -- reserved (= 0)
  deviceName       LABEL         -- function device of ECSC which sends
                                    the telegram
  }
```

F.8.2.5 Operational train directory request

The operational train directory request is sent as a TRDP-MD request expecting a TRDP-MD reply containing a copy of the operational train directory as computed by the UuT (Figure F.13). This request provides a possibility to check the correct computation of the TTDB.

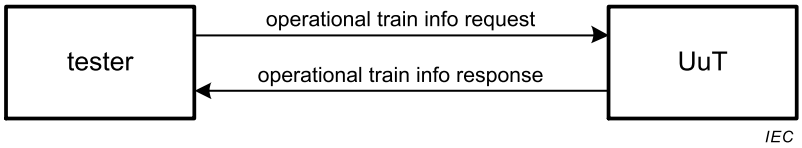


Figure F.13 – Conformance test operational train directory request

TRDP MD request message parameters:

Message type:	‘Mr’
Protocol:	UDP
ComId:	84
SourceURI (user part):	“ComProfTestAppl”
DestinationURI (user part):	“ComProfTester”
DestinationURI (host part):	test function of the UuT
SourceURI (host part):	user defined
Dataset:	CONFTEST_OPTRAIN_REQUEST
ReplyTimeOut:	Defined by tester. Default value: 2,5 s

The TRDP message is defined as shown in Figure F.14.

Figure F.14 – Conformance test operational train directory request data

The payload of the TRDP request message is defined as follows:

```
CONFTEST_OPTRAIN_REQUEST::= RECORD
  {
  version          UINT16        -- telegram version information
                                    main version = 1
                                    subversion = 0
  command          UINT8         -- command to UuT
                                    0 = no command (message to be ignored)
                                    3 = respond operational train directory
  reserved01       UINT8         -- reserved (= 0)
  deviceName       LABEL         -- function device of tester which sends
                                    the telegram
  }
```

TRDP MD reply message parameters:

ComId:	85
Dataset:	CONFTEST_OPTRAIN_REPLY

The TRDP message is defined as shown in Figure F.15.

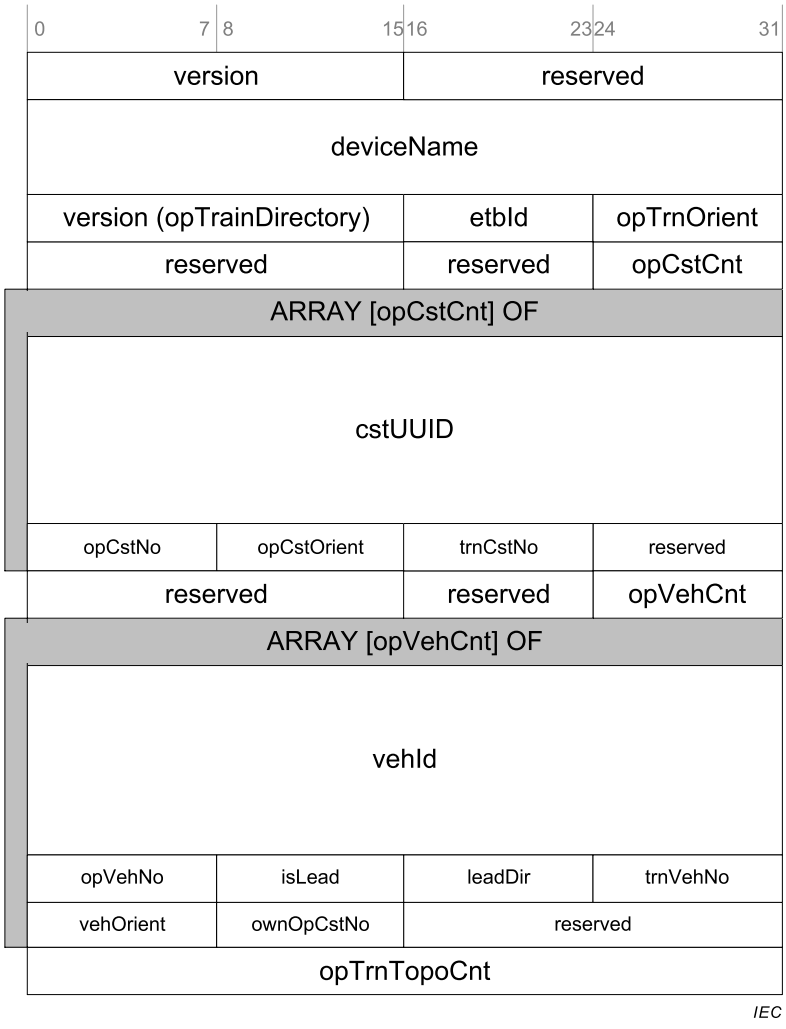


Figure F.15 – Conformance test operational train directory reply data

The payload of the TRDP reply message is defined as follows:

```
CONFTEST_OPTRAIN_REPLY::= RECORD
  {
  version          VERSION        -- telegram version information
                                     main version = 1
                                     subversion = 0
  reserved01       UINT16         -- reserved (= 0)
  deviceName       LABEL          -- function device of tester which sends
                                     the telegram
  opTrnDir         OP_TRAIN_DIRECTORY
                                  -- operational train directory as
                                     defined in 5.3.3.2.13
  }
```

F.9 Echo function

F.9.1 General

The echo function allows a simple communication test between sources and sinks connected to TCN. The echo function is divided into two echo communication patterns:

- Echo test, as specified in F.9.2.
- Reverse-Echo test, as specified in F.9.3.

F.9.2 TRDP echo test

The tester sends an echo request message to the UuT and expects a reply with echoing the user data payload of the request as shown in Figure F.16.

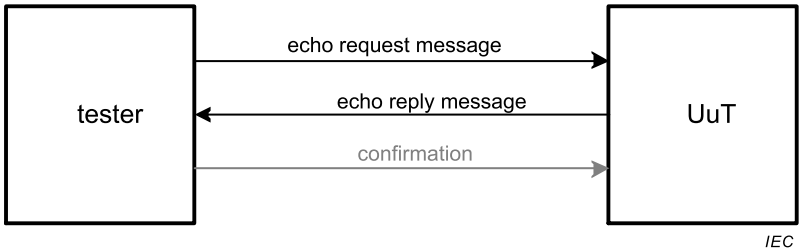


Figure F.16 – Echo test

TRDP MD request message parameters:

MsgType:	‘Mr’
Protocol:	UDP or TCP
ComId:	86
NoOfRepliers:	0
SourceURI (user part):	“ComProfTester”or UuT defined (Reverse-Echo test)
DestinationURI (user part):	“ComProfTestAppl” or “ComProfTester” (Reverse-Echo test)
DestinationURI (host part):	-
SourceURI (host part):	user defined
DestinationIpAddress:	IP unicast address of ED (application function emulation), computed from the fctId (see 5.3.3.2.4) or computed from the Reverse-Echo notification message. IP multicast address of the multcast group the ED (application function emulation) belongs to, computed from the fctId (see 5.3.3.2.4) or computed from the Reverse-Echo notification message.
Dataset:	Defined by tester or UuT (Reverse-Echo test)

ReplyTimeOut:	Defined by tester or UuT (Reverse-Echo test). Default value: 2,5 s

The tester can run multiple message transfers in parallel.

TRDP MD reply message parameters:

MsgType:	‘Mp’ or ‘Mq’
ComId:	87
SMI:	-
UserDataVersion	-
Dataset:	Taken from echo request

It is an UuT option to implement an echo reply with or without confirmation.

F.9.3 Reverse-Echo test

The tester sends a reverse echo notification message to the UuT triggering an Echo test initiated by the UuT as shown in Figure F.17.

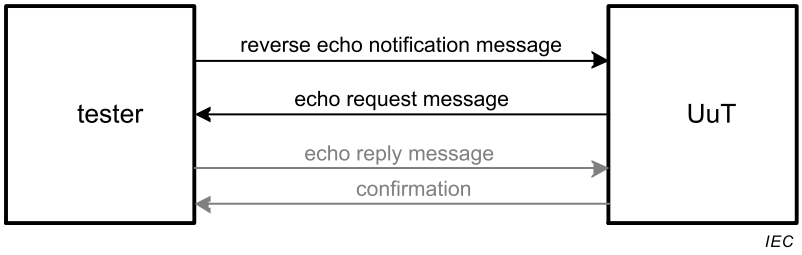


Figure F.17 – Reverse-Echo test

TRDP MD notification message parameters:

MsgType:	‘Mn’
Protocol:	UDP
ComId:	88
SourceURI (user part):	“ComProfTester”
DestinationURI (user part):	“ComProfTestAppl”
DestinationURI (host part):	–
SourceURI (host part):	user defined
DestinationIpAddress:	IP unicast address of ED (application function emulation), computed from the fctId (see 5.3.3.2.4)
Dataset:	CONFTEST_MDTRIG_MD

The TRDP message is defined as shown in Figure F.18.

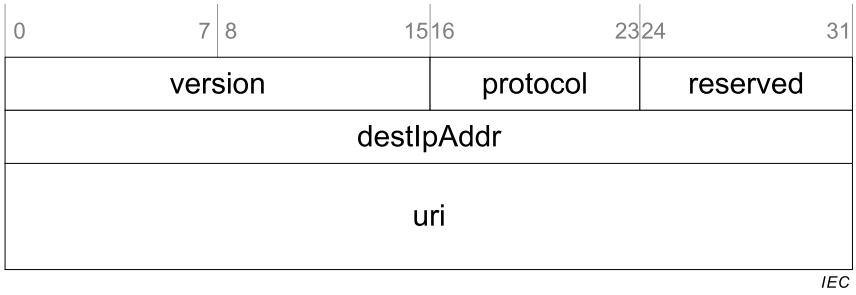


Figure F.18 – Conformance test message data telegram data

The payload of the TRDP notification message is defined as follows:

```
CONFTEST_MDTRIG_MD::= RECORD
  {
  version          VERSION       -- telegram version information
                                    mainVersion = 1
                                    subVersion = 0
  protocol         UINT8         -- protocol for echo test
                                    0 = UDP
                                    1 = TCP
  reserved01       UINT8         -- reserved (= 0)
  destIpAddr       UINT32        -- The destination IP address the UuT
                                    shall send the request to
                                    0 = not used
  uri              CHAR[128]     -- TCN-URI (host part), the UuT shall
                                    send the notification or request to
                                    0 = not used
  }
```

It is an option to choose either an IP address (parameter ‘destIpAddr’) or an URI (Parameter ‘uri’) as destination of the echo request message.

Upon reception of a reverse echo notification message, the UuT shall initiate an echo test as defined in F.9.2 latest after a time of 2,0 s.

F.10 Statement of conformity

Complete conformity to this part of IEC 61375 is demonstrated when all mandatory conformance tests have been successfully executed.

Table F.1 provides a list of all mandatory and optional conformance tests which have been discussed more deeply in the above subclauses.

Table F.1 – Conformance testing summary

Function/Service	Affected device	Manda-tory(M) Optio-nal(O)	Conformance test specification
Common ETB Framework – Service addressing	DNS server (e.g. one per consist network)	M	General test based on Reverse-Echo test. Application specific test in the scope of IEC 61375-2-4
Common ETB Framework – TRDP	End Devices	M	Dedicated specification, e.g. provided by a test laboratory or a user organization
Common ETB Framework – SDTv2	End Devices	O	
Common ETB Framework – Service discovery	ECSP, End Devices	O	In the scope of IEC 61375-2-4
Common ETB Framework – Train info service	ECSP, End Devices	O	Not in the scope of this part of IEC 61375
Common ETB Framework – TTDB	Consist (ECSP)	M	Conformance test interface specified in this part of IEC 61375 Conformance test cases to be generally specified (e.g. in new standard part)
Common ETB Framework – CSTINFO telegram	Consist (ECSP)	M	
ETB control service	Consist (ECSP)	M	
End-to-end data communication	End Devices, ETBN (router), ECN	M	

Annex G
(informative)

SNMP Management Information Base (MIB)

G.1 General

This annex defines SNMP MIBs for the following data groups:

- TTDB-MIB for train topology database operational train directory, as defined in 5.3.
- TRDP-MIB for the TRDP protocol statistic data, as defined in Annex D.

NOTE IEC 61375-2-5 defines an MIB for the train topology discovery protocol (TTDP).

The OID tree is defined in accordance to ISO/IEC 9834-1.

G.2 TTDB-MIB

The TTDB-MIB should be implemented on each ECSP. The MIB is defined as follows:

```
--
          TTDB-MIB DEFINITIONS::= BEGIN
--
-- This MIB defines objects of the Train Topology Database (TTDB) which
-- is specified in standard IEC 61375-2-3.
--
--
-- The objects defined in this MIB are located under:
--
--              iso(1).std(0)
--                     |
--                stdx61375(61375)
--                     |
--                IEC 61375p2(2)
--                     |
--                     |_________________________
--                     |           |            |
--                     |           |            |
--                  (trdp(1))    ttdb(3)    (ttdp(5))
--
--
--
-- trdp: train realtime data protocol, MIB defined in IEC 61375-2-3
-- ttdb: train topology database, this MIB
-- ttdp: train topology discovery protocol, MIB defined in IEC 61375-2-5
--


          IMPORTS
               OBJECT-GROUP, MODULE-COMPLIANCE
                    FROM SNMPv2-CONF
               Unsigned32, OBJECT-TYPE, MODULE-IDENTITY
                    FROM SNMPv2-SMI
               TEXTUAL-CONVENTION
                    FROM SNMPv2-TC;


-- ************************************************************************
-- Root OID
-- ************************************************************************

          -- 1.0.61375.2
          IEC 61375p2 MODULE-IDENTITY
               LAST-UPDATED "201405220000Z"       -- May 22, 2014 00:00 GMT
               ORGANIZATION
```

```
                    "IEC"
               CONTACT-INFO
                    "International Electrotechnical Commission
                    IEC Central Office
                    3, rue de Varembe
                    P.O. Box 131
                    CH - 1211 GENEVA 20
                    Switzerland
                    Phone: +41 22 919 02 11
                    Fax: +41 22 919 03 00
                    email: info@iec.ch"
               DESCRIPTION
                    "This MIB module defines the Network Management interfaces
                    for the Train Topology Database (TTDB) defined by the
                    IEC standard 61375-2-3.
                    This definition specifies a pure monitoring variant of a
SNMP entity."
               REVISION "201405220000Z"
               DESCRIPTION "First Release"
               ::= { stdx61375 2 }

--
-- Textual conventions
--


          TtdbOrient::= TEXTUAL-CONVENTION
               STATUS current
               DESCRIPTION
                    "Represents orientation of a vehicle or a Consist"
               SYNTAX INTEGER
                    {
                    direct(1),
                    inverse(2),
                    undefined(3)
                    }

          TtdbValidity::= TEXTUAL-CONVENTION
               STATUS current
               DESCRIPTION
                    "defines validity of the operational train directory"
               SYNTAX INTEGER
                    {
                    invalid(1),
                    valid(2)
                    shared(3)
                    }

          TtdbConfirmation::= TEXTUAL-CONVENTION
               STATUS current
               DESCRIPTION
                    "defines  confirmation  status  of  the  operational  train
directory"
               SYNTAX INTEGER
                    {
                    unconfirmed(1),
                    confirmed(2)
                    }

--
-- Node definitions
--

          -- 1.0
          std OBJECT IDENTIFIER::= { iso 0 }


          -- 1.0.61375
          stdx61375 OBJECT IDENTIFIER::= { std 61375 }



-- ****************************************************************
-- Train Topology Database (TTDB)
```

```
-- ********************************************************************
          -- 1.0.61375.2.3
          ttdb OBJECT IDENTIFIER::= { IEC 61375p2 3 }


-- ********************************************************************
-- objects groups of TTDB object identifiers
-- ********************************************************************
          -- 1.0.61375.2.3.1
          ttdbObjects OBJECT IDENTIFIER::= { ttdb 1 }


-- General information
          -- 1.0.61375.2.3.1.1
          ttdbGenInfo OBJECT IDENTIFIER::= { ttdbObjects 1 }


          -- 1.0.61375.2.3.1.1.1
          ttdbEtbId OBJECT-TYPE
               SYNTAX Unsigned32 (1..4)
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "ETB Identifier"
               ::= { ttdbGenInfo 1 }

          -- 1.0.61375.2.3.1.1.2
          ttdbValidityState OBJECT-TYPE
               SYNTAX TtdbValidity
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Validity of operational train directory"
               ::= { ttdbGenInfo 2 }

          -- 1.0.61375.2.3.1.1.3
          ttdbConfirmationState OBJECT-TYPE
               SYNTAX TtdbConfirmation
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Confirmation state of operational train directory"
               ::= { ttdbGenInfo 3 }

          -- 1.0.61375.2.3.1.1.4
          ttdbTrainId OBJECT-TYPE
               SYNTAX OCTET STRING (SIZE (16))
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Train Identifier"
               ::= { ttdbGenInfo 4 }

          -- 1.0.61375.2.3.1.1.5
          ttdbOpTrnTopoCnt OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "operational train topocounter"
               ::= { ttdbGenInfo 5 }



-- Operational vehicle list

          -- 1.0.61375.2.3.1.2
          ttdbOpVehList OBJECT IDENTIFIER::= { ttdbObjects 2 }

          -- 1.0.61375.2.3.1.2.1
          ttdbOpVehCnt OBJECT-TYPE
               SYNTAX Unsigned32 (1..63)
               MAX-ACCESS read-only
               STATUS current
```

```
     DESCRIPTION
          "Number of vehicles in train"
     ::= { ttdbOpVehList 1 }


-- 1.0.61375.2.3.1.2.2
ttdbOpVehTable OBJECT-TYPE
     SYNTAX SEQUENCE OF TtdbOpVehEntry
     MAX-ACCESS not-accessible
     STATUS current
     DESCRIPTION
          "Operational vehicle table"
     ::= { ttdbOpVehList 2 }

-- 1.0.61375.2.3.1.2.2.1
ttdbOpVehEntry OBJECT-TYPE
     SYNTAX TtdbOpVehEntry
     MAX-ACCESS not-accessible
     STATUS current
     DESCRIPTION
          "An antry in the operational vehicle table"
     INDEX { ttdbOpVehIdx }
     ::= { ttdbOpVehTable 1 }

TtdbOpVehEntry::=
     SEQUENCE {
           ttdbOpVehIdx       Unsigned32,
           ttdbOpVehId        OCTET STRING,
           ttdbOpVehNo        Unsigned32,
           ttdbOpVehIsLead    INTEGER,
           ttdbOpVehLeadDir   INTEGER,
           ttdbOpVehOrient    TtdbOrient }

-- 1.0.61375.2.3.1.2.2.1.1
ttdbOpVehIdx OBJECT-TYPE
     SYNTAX Unsigned32 (1..63)
     MAX-ACCESS not-accessible
     STATUS current
     DESCRIPTION
          "Operational vehicle table index"
     ::= { ttdbOpVehEntry 1 }

-- 1.0.61375.2.3.1.2.2.1.2
ttdbOpVehId OBJECT-TYPE
     SYNTAX OCTET STRING (SIZE(0..16))
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Operational vehicle identifier (UIC vehicle number)"
     ::= { ttdbOpVehEntry 2 }

-- 1.0.61375.2.3.1.2.2.1.3
ttdbOpVehNo OBJECT-TYPE
     SYNTAX Unsigned32 (1..63)
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Operational vehicle sequence number"
     ::= { ttdbOpVehEntry 3 }

-- 1.0.61375.2.3.1.2.2.1.4
ttdbOpVehIsLead OBJECT-TYPE
     SYNTAX INTEGER {notLeading(1),leading(2)}
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Operational vehicle leading property"
     ::= { ttdbOpVehEntry 4 }

-- 1.0.61375.2.3.1.2.2.1.5
ttdbOpVehLeadDir OBJECT-TYPE
     SYNTAX INTEGER {dir1(1),dir2(2)}
     MAX-ACCESS read-only
     STATUS current
```

```
               DESCRIPTION
                    "Operational vehicle leading direction"
               ::= { ttdbOpVehEntry 5 }

          -- 1.0.61375.2.3.1.2.2.1.6
          ttdbOpVehOrient OBJECT-TYPE
               SYNTAX TtdbOrient
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Operational vehicle orientation"
               ::= { ttdbOpVehEntry 6 }



-- ************************************************************************
-- conformance statements
-- ************************************************************************
          -- 1.0.61375.2.3.2
          ttdbConformance OBJECT IDENTIFIER::= { ttdb 2 }


          -- 1.0.61375.2.3.2.2
          ttdbBasicGroup OBJECT-GROUP
              OBJECTS { ttdbEtbId,
                        ttdbValidityState,
                         ttdbConfirmationState,
                         ttdbTrainId,
                         ttdbOpVehCnt,
                         ttdbOpTrnTopoCnt}
               STATUS current
               DESCRIPTION
                    "basic group of TTDB parameters"
             ::= { ttdbConformance 2 }


          -- 1.0.61375.2.3.2.3
          ttdbOpVehListGroup OBJECT-GROUP
              OBJECTS { ttdbOpVehId,
                        ttdbOpVehNo,
                         ttdbOpVehIsLead,
                         ttdbOpVehLeadDir,
                         ttdbOpVehOrient}
               STATUS current
               DESCRIPTION
                    "operational vehicle list parameters for TTDB monitoring"
             ::= { ttdbConformance 3 }

          -- 1.0.61375.2.3.2.4
          ttdbBasicCompliance MODULE-COMPLIANCE
               STATUS current
               DESCRIPTION
                    "basic implementation requirements for TTDB monitoring"
               MODULE -- this module
               MANDATORY-GROUPS {ttdbBasicGroup,ttdbOpVehListGroup}
             ::= { ttdbConformance 4 }


END
```

G.3 TRDP-MIB

The TRDP-MIB should be implemented on each end device which is running TRDP for train wide communication. The MIB is defined as follows:

```
          TTDB-MIB DEFINITIONS::= BEGIN
--
--
-- This MIB defines objects of the Train Realtime Data Protocol (TRDP) which
-- is specified in standard IEC 61375-2-3.
--
-- The objects defined in this MIB are located under:
--
--             iso(1).std(0)
--                    |
--               stdx61375(61375)
--                    |
--               IEC 61375p2(2)
--                    |
--                    |_________________________
--                    |          |              |
--                    |          |              |
--                  trdp(1)   (ttdb(3))   (ttdp(5))
--
--
--
-- trdp: train realtime data protocol, this MIB
-- ttdb: train topology database, MIB defined in IEC 61375-2-3
-- ttdp: train topology discovery protocol, MIB defined in IEC 61375-2-5
--
--


          IMPORTS
               OBJECT-GROUP, MODULE-COMPLIANCE
                    FROM SNMPv2-CONF
               IpAddress, Unsigned32, OBJECT-TYPE,    MODULE-IDENTITY

                    FROM SNMPv2-SMI     ;


-- ************************************************************************
-- Root OID
-- ************************************************************************

          -- 1.0.61375.2
          IEC 61375p2 MODULE-IDENTITY
               LAST-UPDATED "201405220000Z"       -- May 22, 2014 00:00 GMT
               ORGANIZATION
                    "IEC"
               CONTACT-INFO
                    "International Electrotechnical Commission
                    IEC Central Office
                    3, rue de Varembe
                    P.O. Box 131
                    CH - 1211 GENEVA 20
                    Switzerland
                    Phone: +41 22 919 02 11
                    Fax: +41 22 919 03 00
                    email: info@iec.ch"
               DESCRIPTION
                    "This MIB module defines the Network Management interfaces
                    for the Train Topology Database (TTDB) defined by the
                    IEC standard 61375-2-3.
                    This definition specifies a pure monitoring variant of a
SNMP entity."
               REVISION "201405220000Z"
               DESCRIPTION "First Release"
               ::= { stdx61375 2 }
```

```
--
-- Node definitions
--

          -- 1.0
          std OBJECT IDENTIFIER::= { iso 0 }


          -- 1.0.61375
          stdx61375 OBJECT IDENTIFIER::= { std 61375 }



-- ************************************************************************
-- Train Realtime Data Protocol (TRDP)
-- ************************************************************************
          -- 1.0.61375.2.1
          trdp OBJECT IDENTIFIER::= { IEC 61375p2 1 }


-- ************************************************************************
-- objects groups of TTDB object identifiers
-- ************************************************************************
          -- 1.0.61375.2.1.1
          trdpObjects OBJECT IDENTIFIER::= { trdp 1 }


-- General information
          -- 1.0.61375.2.1.1.1
          trdpGenInfo OBJECT IDENTIFIER::= { trdpObjects 1 }

          -- 1.0.61375.2.1.1.1.1
          trdpGenVers OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "TRDP version"
               ::= { trdpGenInfo 1 }

          -- 1.0.61375.2.1.1.1.2
          trdpGenUpTime OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "time in seconds since last initialization"
               ::= { trdpGenInfo 2 }

          -- 1.0.61375.2.1.1.1.3
          trdpGenStatTime OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "time in seconds since last reset of statistics"
               ::= { trdpGenInfo 3 }

          -- 1.0.61375.2.1.1.1.4
          trdpGenHostName OBJECT-TYPE
               SYNTAX OCTET STRING (SIZE(0..16))
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Own host name"
               ::= { trdpGenInfo 4 }

          -- 1.0.61375.2.1.1.1.5
          trdpGenLeadName OBJECT-TYPE
               SYNTAX OCTET STRING (SIZE(0..16))
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Leader host name"
```

```
          ::= { trdpGenInfo 5 }

        -- 1.0.61375.2.1.1.1.6
        trdpGenOwnIp OBJECT-TYPE
          SYNTAX IpAddress
          MAX-ACCESS read-only
          STATUS current
          DESCRIPTION
               "Own IP address"
          ::= { trdpGenInfo 6 }

        -- 1.0.61375.2.1.1.1.7
        trdpGenLeadIp OBJECT-TYPE
          SYNTAX IpAddress
          MAX-ACCESS read-only
          STATUS current
          DESCRIPTION
               "Leader IP address"
          ::= { trdpGenInfo 7 }

        -- 1.0.61375.2.1.1.1.8
        trdpGenProcPrio OBJECT-TYPE
          SYNTAX Unsigned32
          MAX-ACCESS read-only
          STATUS current
          DESCRIPTION
               "priority of TRDP process"
          ::= { trdpGenInfo 8 }

        -- 1.0.61375.2.1.1.1.9
        trdpGenProcCycle OBJECT-TYPE
          SYNTAX Unsigned32
          MAX-ACCESS read-only
          STATUS current
          DESCRIPTION
               "Cycle time of TRDP process in microseconds"
          ::= { trdpGenInfo 9 }

        -- 1.0.61375.2.1.1.1.10
        trdpGenNumJoin OBJECT-TYPE
          SYNTAX Unsigned32
          MAX-ACCESS read-only
          STATUS current
          DESCRIPTION
               "Number of IGMP joins"
          ::= { trdpGenInfo 10 }

        -- 1.0.61375.2.1.1.1.11
        trdpGenNumRed OBJECT-TYPE
          SYNTAX Unsigned32
          MAX-ACCESS read-only
          STATUS current
          DESCRIPTION
               "Number of redundancy groups"
          ::= { trdpGenInfo 11 }



-- Memory statistics
        -- 1.0.61375.2.1.1.2
        trdpMemStat OBJECT IDENTIFIER::= { trdpObjects 2 }

        -- 1.0.61375.2.1.1.2.1
        trdpMemTotal OBJECT-TYPE
          SYNTAX Unsigned32
          MAX-ACCESS read-only
          STATUS current
          DESCRIPTION
               "Total memory size"
          ::= { trdpMemStat 1 }

        -- 1.0.61375.2.1.1.2.2
        trdpMemFree OBJECT-TYPE
          SYNTAX Unsigned32
```

```
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                 "Free memory"
            ::= { trdpMemStat 2 }

        -- 1.0.61375.2.1.1.2.3
        trdpMemMinFree OBJECT-TYPE
            SYNTAX Unsigned32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                 "Minimal free memory in statistics interval"
            ::= { trdpMemStat 3 }

        -- 1.0.61375.2.1.1.2.4
        trdpMemAllocBlocks OBJECT-TYPE
            SYNTAX Unsigned32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                 "Number of allocated blocks"
            ::= { trdpMemStat 4 }

        -- 1.0.61375.2.1.1.2.5
        trdpMemAllocErr OBJECT-TYPE
            SYNTAX Unsigned32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                 "Number of allocation errors"
            ::= { trdpMemStat 5 }

        -- 1.0.61375.2.1.1.2.6
        trdpMemFreeErr OBJECT-TYPE
            SYNTAX Unsigned32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                 "Number of freeing errors"
            ::= { trdpMemStat 6 }



-- Process data statistics
        -- 1.0.61375.2.1.1.3
        trdpPdStat OBJECT IDENTIFIER::= { trdpObjects 3 }


        -- 1.0.61375.2.1.1.3.1
        trdpPdDefQos OBJECT-TYPE
            SYNTAX Unsigned32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                 "Default QoS for PD"
            ::= { trdpPdStat 1 }

        -- 1.0.61375.2.1.1.3.2
        trdpPdDefTtl OBJECT-TYPE
            SYNTAX Unsigned32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                 "Default TTL for PD"
            ::= { trdpPdStat 2 }

        -- 1.0.61375.2.1.1.3.3
        trdpPdDefTo OBJECT-TYPE
            SYNTAX Unsigned32
            MAX-ACCESS read-only
            STATUS current
            DESCRIPTION
                 "Default timeout for PD"
```

```
     ::= { trdpPdStat 3 }

-- 1.0.61375.2.1.1.3.4
trdpPdNumSubs OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Number of subscribed ComIds"
     ::= { trdpPdStat 4 }

-- 1.0.61375.2.1.1.3.5
trdpPdNumPub OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Number of published ComIds"
     ::= { trdpPdStat 5 }

-- 1.0.61375.2.1.1.3.6
trdpPdNumRcv OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "number of received PD packets"
     ::= { trdpPdStat 6 }

-- 1.0.61375.2.1.1.3.7
trdpPdNumCrcErr OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Number of received PD packets with CRC error"
     ::= { trdpPdStat 7 }

-- 1.0.61375.2.1.1.3.8
trdpPdNumProtErr OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Number of received PD packets with protocol error"
     ::= { trdpPdStat 8 }

-- 1.0.61375.2.1.1.3.9
trdpPdNumTopoErr OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Number of received PD packets with wrong
          topo counter value"
     ::= { trdpPdStat 9 }

-- 1.0.61375.2.1.1.3.10
trdpPdNumNoSubs OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Number of received PD packets without subscription"
     ::= { trdpPdStat 10 }

-- 1.0.61375.2.1.1.3.11
trdpPdNumNoPub OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Number of received PD packets without publisher"
     ::= { trdpPdStat 11 }
```

```
          -- 1.0.61375.2.1.1.3.12
          trdpPdNumTo OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Number of PD timeouts"
               ::= { trdpPdStat 12 }

          -- 1.0.61375.2.1.1.3.13
          trdpPdNumSend OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Number of sent PD packets"
               ::= { trdpPdStat 13 }

-- Message data statistics (UDP)
          -- 1.0.61375.2.1.1.4
          trdpMduStat OBJECT IDENTIFIER::= { trdpObjects 4 }

          -- 1.0.61375.2.1.1.4.1
          trdpMduDefQos OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Default QoS for MD"
               ::= { trdpMduStat 1 }

          -- 1.0.61375.2.1.1.4.2
          trdpMduDefTtl OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "default TTL for MD"
               ::= { trdpMduStat 2 }

          -- 1.0.61375.2.1.1.4.3
          trdpMduDefReplyTo OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Default reply timeout for MD"
               ::= { trdpMduStat 3 }

          -- 1.0.61375.2.1.1.4.4
          trdpMduDefConfTo OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Default confirm timeout for MD"
               ::= { trdpMduStat 4 }

          -- 1.0.61375.2.1.1.4.5
          trdpMduNumList OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Number of Listeners"
               ::= { trdpMduStat 5 }

          -- 1.0.61375.2.1.1.4.6
          trdpMduNumRcv OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
```

```
                    "Number of received MD packets"
               ::= { trdpMduStat 6 }

          -- 1.0.61375.2.1.1.4.7
          trdpMduNumCrcErr OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Number of received MD packets with CRC error"
               ::= { trdpMduStat 7 }

          -- 1.0.61375.2.1.1.4.8
          trdpMduNumProtErr OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Number of received MD packets with protocol
                     error"
               ::= { trdpMduStat 8 }

          -- 1.0.61375.2.1.1.4.9
          trdpMduNumTopoErr OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Number of received MD packets with wrong
                    topo counter value"
               ::= { trdpMduStat 9 }

          -- 1.0.61375.2.1.1.4.10
          trdpMduNumNoList OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Number of received MD packets without
                    Listener"
               ::= { trdpMduStat 10 }

          -- 1.0.61375.2.1.1.4.11
          trdpMduNumReplyTo OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Number of MD reply timeouts"
               ::= { trdpMduStat 11 }

          -- 1.0.61375.2.1.1.4.12
          trdpMduNumConfTo OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "Number of MD confirm timeouts"
               ::= { trdpMduStat 12 }

          -- 1.0.61375.2.1.1.4.13
          trdpMduNumSend OBJECT-TYPE
               SYNTAX Unsigned32
               MAX-ACCESS read-only
               STATUS current
               DESCRIPTION
                    "number of sent MD packets"
               ::= { trdpMduStat 13 }

-- Message data statistics (TCP)
          -- 1.0.61375.2.1.1.5
          trdpMdtStat OBJECT IDENTIFIER::= { trdpObjects 5 }

          -- 1.0.61375.2.1.1.5.1
```

```
trdpMdtDefQos OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Default QoS for MD"
     ::= { trdpMdtStat 1 }

-- 1.0.61375.2.1.1.5.2
trdpMdtDefTtl OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "default TTL for MD"
     ::= { trdpMdtStat 2 }

-- 1.0.61375.2.1.1.5.3
trdpMdtDefReplyTo OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Default reply timeout for MD"
     ::= { trdpMdtStat 3 }

-- 1.0.61375.2.1.1.5.4
trdpMdtDefConfTo OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Default confirm timeout for MD"
     ::= { trdpMdtStat 4 }

-- 1.0.61375.2.1.1.5.5
trdpMdtNumList OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Number of Listeners"
     ::= { trdpMdtStat 5 }

-- 1.0.61375.2.1.1.5.6
trdpMdtNumRcv OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Number of received MD packets"
     ::= { trdpMdtStat 6 }

-- 1.0.61375.2.1.1.5.7
trdpMdtNumCrcErr OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Number of received MD packets with CRC error"
     ::= { trdpMdtStat 7 }

-- 1.0.61375.2.1.1.5.8
trdpMdtNumProtErr OBJECT-TYPE
     SYNTAX Unsigned32
     MAX-ACCESS read-only
     STATUS current
     DESCRIPTION
          "Number of received MD packets with protocol
           error"
     ::= { trdpMdtStat 8 }

-- 1.0.61375.2.1.1.5.9
trdpMdtNumTopoErr OBJECT-TYPE
```

```
                SYNTAX Unsigned32
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                     "Number of received MD packets with wrong
                     topo counter value"
                ::= { trdpMdtStat 9 }

           -- 1.0.61375.2.1.1.5.10
           trdpMdtNumNoList OBJECT-TYPE
                SYNTAX Unsigned32
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                     "Number of received MD packets without
                     Listener"
                ::= { trdpMdtStat 10 }

           -- 1.0.61375.2.1.1.5.11
           trdpMdtNumReplyTo OBJECT-TYPE
                SYNTAX Unsigned32
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                     "Number of MD reply timeouts"
                ::= { trdpMdtStat 11 }

           -- 1.0.61375.2.1.1.5.12
           trdpMdtNumConfTo OBJECT-TYPE
                SYNTAX Unsigned32
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                     "Number of MD confirm timeouts"
                ::= { trdpMdtStat 12 }

           -- 1.0.61375.2.1.1.5.13
           trdpMdtNumSend OBJECT-TYPE
                SYNTAX Unsigned32
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                     "number of sent MD packets"
                ::= { trdpMdtStat 13 }

-- Redundancy statistics
           -- 1.0.61375.2.1.1.6
           trdpRedStat OBJECT IDENTIFIER::= { trdpObjects 6 }

           -- 1.0.61375.2.1.1.6.1
           trdpRedId OBJECT-TYPE
                SYNTAX Unsigned32
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                     "Redundancy Id"
                ::= { trdpRedStat 1 }

           -- 1.0.61375.2.1.1.6.2
           trdpRedState OBJECT-TYPE
                SYNTAX Unsigned32
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                     "State: leader/follower"
                ::= { trdpRedStat 2 }



-- ************************************************************************
-- conformance statements
-- ************************************************************************
           -- 1.0.61375.2.1.2
           trdpConformance OBJECT IDENTIFIER::= { trdp 2 }
```

```
-- 1.0.61375.2.1.2.2
trdpGenGroup OBJECT-GROUP
    OBJECTS { trdpGenVers,
                trdpGenUpTime,
                trdpGenStatTime,
                trdpGenHostName,
                trdpGenLeadName,
                trdpGenOwnIp,
                trdpGenLeadIp,
                trdpGenProcPrio,
                trdpGenProcCycle,
                trdpGenNumJoin,
                trdpGenNumRed}
     STATUS current
     DESCRIPTION
          "General statistics and configuration data"
   ::= { trdpConformance 2 }

-- 1.0.61375.2.1.2.3
trdpMemGroup OBJECT-GROUP
    OBJECTS { trdpMemTotal,
                trdpMemFree,
                trdpMemMinFree,
                trdpMemAllocBlocks,
                trdpMemAllocErr,
                trdpMemFreeErr,
                trdpMemFreeErr}
     STATUS current
     DESCRIPTION
          "Memory statistics and configuration data"
   ::= { trdpConformance 3 }

-- 1.0.61375.2.1.2.4
trdpPdGroup OBJECT-GROUP
    OBJECTS { trdpPdDefQos,
                trdpPdDefTtl,
                trdpPdDefTo,
                trdpPdNumSubs,
                trdpPdNumPub,
                trdpPdNumRcv,
                trdpPdNumCrcErr,
                trdpPdNumProtErr,
                trdpPdNumTopoErr,
                trdpPdNumNoSubs,
                trdpPdNumNoPub,
                trdpPdNumTo,
                trdpPdNumSend}
     STATUS current
     DESCRIPTION
          "PD statistis and configuration data"
   ::= { trdpConformance 4 }

-- 1.0.61375.2.1.2.5
trdpMduGroup OBJECT-GROUP
    OBJECTS { trdpMduDefQos,
                trdpMduDefTtl,
                trdpMduDefReplyTo,
                trdpMduDefConfTo,
                trdpMduNumList,
                trdpMduNumRcv,
                trdpMduNumCrcErr,
                trdpMduNumProtErr,
                trdpMduNumTopoErr,
                trdpMduNumReplyTo,
                trdpMduNumNoList,
                trdpMduNumConfTo,
                trdpMduNumSend}
     STATUS current
     DESCRIPTION
          "MD statistis and configuration data"
   ::= { trdpConformance 5 }
```

```
        -- 1.0.61375.2.1.2.6
        trdpMdtGroup OBJECT-GROUP
            OBJECTS { trdpMdtDefQos,
                      trdpMdtDefTtl,
                      trdpMdtDefReplyTo,
                      trdpMdtDefConfTo,
                      trdpMdtNumList,
                      trdpMdtNumRcv,
                      trdpMdtNumCrcErr,
                      trdpMdtNumProtErr,
                      trdpMdtNumTopoErr,
                      trdpMdtNumReplyTo,
                      trdpMdtNumNoList,
                      trdpMdtNumConfTo,
                      trdpMdtNumSend}
             STATUS current
             DESCRIPTION
                  "MD statistis and configuration data"
           ::= { trdpConformance 6 }

        -- 1.0.61375.2.1.2.7
        trdpRedGroup OBJECT-GROUP
            OBJECTS { trdpRedId,
                      trdpRedState }
             STATUS current
             DESCRIPTION
                  "General statistis and configuration data"
           ::= { trdpConformance 7 }


        -- 1.0.61375.2.1.2.8
        trdpBasicCompliance MODULE-COMPLIANCE
             STATUS current
             DESCRIPTION
                  "basic implementation requirements for TRDP monitoring"
             MODULE -- this module
             MANDATORY-GROUPS {trdpGenGroup,
                               trdpMemGroup,
                               trdpPdGroup,
                               trdpMduGroup,
                               trdpMdtGroup,
                               trdpRedGroup }
           ::= { trdpConformance 8 }


END
```

Bibliography

IEC 60571, *Railway applications – Electronic equipment used on rolling stock*

IEC 61131-3:2013*, Programmable controllers – Part 3: Programming languages*

IEC 61375-3-4, *Electronic railway equipment – Train Communication Network (TCN) – Part 3-4: Ethernet Consist Network (ECN)*

IEC 61508-1:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 1: General requirements*

IEC 61784-3-3:2010*, Industrial communication networks – Profiles – Part 3-3: Functional safety fieldbuses – Additional specifications for CPF 3*

IEC 62580 (all parts), *Electronic railway equipment – Onboard multimedia and telematic subsystems for railways*

ISO/IEC 7498-1, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*

ISO/IEC 9834-1:2012*, Information technology – Procedures for the operation of object identifier registration authorities: General procedures and top arcs of the international object identifier tree*

EN 15380-4, *Railway applications – Classification system for railway vehicles – Part 4: Function groups*

UIC Leaflet 550, *Power supply installations for passenger stock, 11th edition, April 2005*

UIC Leaflet 550-1, *Electrical switch cabinets on passenger stock, 1st edition of 1.1.90*

UIC Leaflet 556, *Information transmission in the train (train bus), 5th edition 2010*

IETF Requests For Comments:
RFC 768, *User Datagram Protocol*
RFC 791*, Internet Protocol*
RFC 793, *Transmission Control Protocol*
RFC 956, *Algorithms for synchronizing network clocks*
RFC 1034, *Domain names – concepts and facilities*
RFC 1035, *Domain names – implementation and specification*
RFC 2326, *Real Time Streaming Protocol (RTSP)*
RFC 2616, *Hypertext Transfer Protocol – HTTP/1.1.*
RFC 2663, *IP Network Address Translator (NAT) Terminology and Considerations*
RFC 2766, *Network Address Translation – Protocol Translation (NAT-PT)*
RFC 3022, *Traditional IP Network Address Translator (Traditional NAT)*
RFC 3261, *SIP: Session Initiation Protocol*
RFC 3550, *RTP: A Transport Protocol for Real-Time Applications*
RFC 3605, *Real Time Control Protocol (RTCP) attribute in Session Description Protocol (SDP)*
RFC 3711, *The Secure Real-time Transport Protocol (SRTP)*
RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*
RFC 4122, *A Universally Unique IDentifier (UUID) URN Namespace*
RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2.*

———————

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

3, rue de Varembé
PO Box 131
CH-1211 Geneva 20
Switzerland

Tel: + 41 22 919 02 11
Fax: + 41 22 919 03 00
info@iec.ch
www.iec.ch