CEN/TC 278

Date: 2008-10

prEN 15722

CEN/TC 278

Secretariat: NEN

Road transport and traffic telematics — ESafety — ECall minimum set of data - Draft EN 081018

Strassenverkehrstelematik - E-Safety - Minimaler Datensatz für den elektronischen Notruf

Systèmes de transport intelligente — ESafety — ECall ensemble minimum de données (MSD)

ICS:

Descriptors:

Document type: European Standard
Document subtype:
Document stage: CEN Enquiry
Document language: E

D:\Standards\Standards-working docs\15722 eCall MSD\EN_15722_(E) DRAFT EN 081020.doc STD Version 2.1c2

Contents

Page

Foreword 3
Introduction 3
1 Scope 4
2 Conformance 4
3 Normative reference 4
4 Terms and definitions 5
5 Symbols and abbreviated terms 5
6 Requirements 6
6.1 Concepts and formats 6
6.1.1 MSD data concepts 6
6.1.2 Format definition of MSD data concepts 6
6.1.3 Sequence of MSD data concepts 7
6.1.4 Data presentation of MSD 7
6.2 Minimum set of data (MSD) 7
6.2.1 Order of bits and bytes 7
6.2.2 Contents of MSD 8
6.2.3 Contents of MSD 8
6.2.4 MSD acknowledgement and request 12
Annex A (normative) ASN.1 PER representation of MSD 14
Annex B (informative) ASN.1 Data representation PER and BER explained 18
B.1 Introduction to ASN.1 and the 'Packed Encoding Rules' 18
B.2 Basic encoding rules 19
B.3 Packed encoding rules 20

Foreword

This document prEN 15722 has been prepared by Technical Committee CEN/TC 278 “Intelligent Transport Systems”, the secretariat of which is held by NEN.

This document is currently submitted to the CEN Enquiry.

Introduction

The scale of death and injury on roads around the world needs to be fully comprehended to understand the need for "Emergency Call" (eCall). There are around 41,600 deaths and more than 1.7 million injured in 2005. Roads remain unsafe, and further efforts are needed. The pan-European in-vehicle emergency call, eCall, is estimated to have the potential to save up to 2.500 fatalities annually in EU-25 when fully deployed, and furthermore to reduce the severity of injuries, to bring significant savings to the society in healthcare and other costs and to reduce human suffering.

Emergency calls made from vehicles or mobile telephones using wireless technologies, can assist with the objectives of significantly reducing road deaths and injuries. but drivers often have poor (imprecise) location-awareness, especially on interurban roads or abroad. Additionally, in many situations a normal mobile phone may not be available for use, or the car occupants may not be in a position to call.

The situation is worse for those travelling abroad: For example, in EU there are over 100 million trips to another EU country per year (EU-15) -65 % people feel less protected while abroad and most do not know which number to call in an emergency (in some countries over 60%). Language problems are pertinent and prohibit proper communication.

Yet, in the most crucial cases, the victim(s) may not be able to call because they have been injured/trapped, do not know the local number to call, and in many cases, particularly in rural situations and late at night, there may be no witnesses who happen to have a mobile phone and a sense of community.

eCall, in the context of "Road Traffic and Transport Telematics" (otherwise known as "Intelligent Transport Systems" or "ITS") , can be described as a "user instigated or automatic system to provide notification to public safety answering points, by means of wireless communications, that a vehicle has crashed, and to provide coordinates and a defined minimum set of data ". This Technical Specification defines the "Minimum Set of Data" MSD to be transferred by such an in-vehicle eCall system in the event of a crash or emergency.

NOTE The communications media and means of transferring the eCall MSD are not defined in this European Standard.

1 Scope

This European Standard defines the standard data concepts that comprise the "Minimum Set of Data" to be transferred from a vehicle to a 'Public Safety Answering Point' (PSAP) in the event of a crash or emergency via an 'eCall' communication session.

NOTE 1 The communications media protocols and methods for the transmission of the eCall message are not specified in this Standard.

NOTE 2 Additional data concepts may also be transferred, and any such data concepts should be registered using a data registry as defined in ISO 24978 (Intelligent transport systems, Emergency and safety messages, Data registry)

2 Conformance

In order to claim conformance with this Technical Specification, communication shall be established using accepted wireless communication standards, and it shall be able to demonstrate that the minimum set of data (MSD) transferred together with any standardised optional data elements defined herein comply with the specifications of this Technical Specification, to the extent that such data is available from the vehicle.

3 Normative reference

This European Standard incorporates by dated or undated reference, provisions from other publications. These normative references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this European Standard only when incorporated in it by amendment or revision. For undated references the latest edition of the publication referred to applies (including amendments).

[Ref1]: ISO 24978, *Intelligent transport systems. Wireless communications, Emergency and safety messages data registry* (in ballot process)[1]

[Ref2]: ISO 6709, *Standard representation of latitude, longitude and altitude for geographic point locations*

[Ref3]: ISO/IEC/ITU IS/Rec 8824-1/ X.680/8824-1:2002 Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation Published 2002. Cor 1 2006

[Ref4]: ISO/IEC/ITU IS/Rec 8824-2 /X.681:2002 Information technology -- Abstract Syntax Notation One (ASN.1): Information object specification Published 2002. Amd1 2004

[Ref5]: ISO/IEC/ITU IS/Rec 8825-1/X.690 (2002) Information technology -- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER). Published 2002. Amd 1 2004 JTC1/SC6

[Ref6]: ISO/IEC/ITU IS/Rec 8825-2/X.691 (2002) Information technology -- ASN.1 encoding rules: Specification of Packed Encoding Rules (PER) Published 2002.Amd 1 2004. Cor 1 2006. Amd 2 2007. Cor 2 2006.

[Ref7]: EN xxxxxxx "Pan European eCall- Operating requirements"[1](under development)

[Ref8]: EN xxxxxxx "Third party support for eCall- Operating requirements"[1](under development)

4 Terms and definitions

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

4.1
eCall
Automatic or user generated system to provide notification and relevant coordinate information to public safety answering points, by means of cellular wireless networks and carries an defined "Minimum Set of Data" Standardised data, that there has been an incident that requires response from the emergency services, and establishes wherever possible an audio link to the vehicle.

5 Symbols and abbreviated terms

5.1
3G
third generation mobile cellular network system, defined by 3GPP standards

5.2
3GPP
third generation partnership protocol

5.3
BCD
binary coded decimal

5.4
BER
basic encoding rules (ASN.1)

5.5
CNG
compressed natural gas

5.6
ETSI
European telecommunications standards institute

5.7
EC
European Commission

5.8
EU
European Union

5.9
EU-27
27 countries that formed the European Union from 2007.

5.10
GSM
global standard mobile

5.11
GNSS
global navigation satellite system

5.12
ID
identity

5.13
IP
Internet protocol

5.14
LPG
liquid propane gas

5.15
M
mandatory

5.16
MSD
minimum set of data

5.17
O
optional

5.18
PER
packed encoding rules (ASN.1)

5.19
PSAP
public safety answering point

6 Requirements

NOTE he minimum set of data is important information to assist the provision of the most appropriate services to the crash or emergency site and to speed up the response. The minimum set of data makes it possible for the PSAP operator to respond to the eCall even without the voice connection.

6.1 Concepts and formats

6.1.1 MSD data concepts

The "Minimum Set of Data" shall be a direct, timely message to the PSAP operator receiving the emergency call.

6.1.2 Format definition of MSD data concepts

The definitions shown in this Standard are shown below in semantic representation. Data presentation shall be as determined in Clause 6.1.4.

The entries referring to “Byte Position” in the tables are therefore only arbitrary, and are listed to aid understanding of the semantic representation only. The real position of the element in the data-stream is defined by the ASN1 definition in Annex A.

NOTE The information elements in the minimum set of data have been selected on the basis of their relevance in an emergency rescue situation.

6.1.3 Sequence of MSD data concepts

The sequence of data presentation shall be as specified in Clause 6.2 below, presented as defined in Clause 6.1.4

6.1.4 Data presentation of MSD

The MSD shall be transmitted using one or more wireless communications media as defined in [Ref7] which defines one or more ETSI air interface Standards suitable for the transmission of eCall, and shall be presented in Abstract Syntax Notation, ASN.1 Packed encoding rules (PER unaligned) as defined in ISO 8825-2 using the ASN1 definitions defined in Annex A.

The MSD is also referred to in [Ref8].

NOTE In order to implement presentation in ASN.1 PER, readers are advised to also read Annex B " *ASN.1 Data Representation PER and BER explained* "; and also [Ref3], [Ref4], [Ref5], [Ref6].

6.2 Minimum set of data (MSD)

The following sub-clauses provide the definition of the minimum set of data that shall be sent from the vehicle in case of an emergency call.

6.2.1 Order of bits and bytes

The message shall be sent in the sequence defined within these sub-Clauses

The "Minimum Set of Data" (MSD) and the acknowledgment shall be transmitted by the network access device according to agreed European Standards. Figure 1 provides the order of the bits and bytes in the MSD frame.

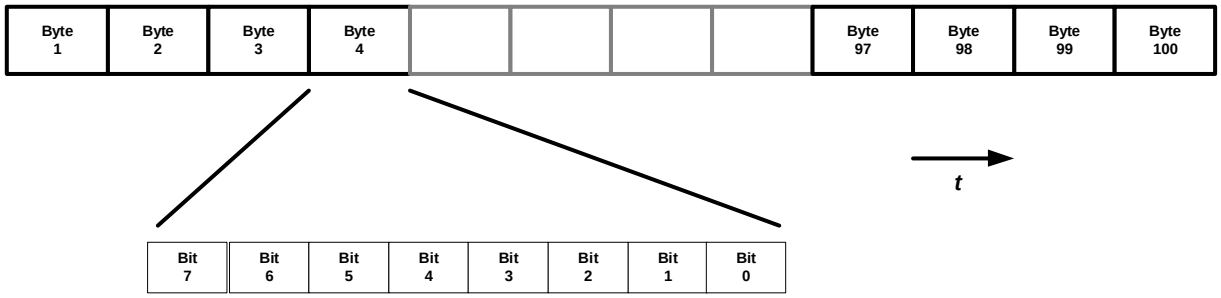

Figure 1 — Order of bits and bytes in MSD Frame

6.2.2 Contents of MSD

Table 1 provides a summary of the semantic contents of the MSD.

** The entries referring to “Byte Position” in the tables are only arbitrary, and are listed to aid understanding of the semantic representation only. The real position of the elements in the data stream is defined by the formal ASN1 definition in Annex A.

Table 1 — Contents/format of the MSD data concept

M – Mandatory data field
O – Optional data field, must be included even if no information is included.

6.2.3 Contents of MSD

Table 1 provides a summary of the semantic contents of the MSD.

** The entries referring to “Byte Position” in the tables are only arbitrary, and are listed to aid understanding of the semantic representation only. The real position of the elements in the data stream is defined by the formal ASN1 definition in Annex A.

Table 2 — Contents/format of the MSD data concept

M – Mandatory data field
O – Optional data field, must be included even if no information is included.

Block No.	Name	**Byte Position		Type	Unit		Description
		First Byte	Last Byte				
1	ID	1		Integer		M	MSD format version set to 1 to discriminate from later MSD formats
		2		Integer		M	Message identifier, starting with 1 and to be incremented with every MSD retransmission after the incident event
2	Control	3		Integer		M	Bit 7: 1 = Automatic activation 0 = Manual activation Bit 6: 1 = Test call 0 = Emergency Bit 5: 1 = Low confidence in position 0 = Position can be trusted Bit 4-0: vehicle type encoding: 00001= passenger vehicle (Class M1) 00010= buses and coaches (Class M2) 00011= buses and coaches (Class M3) 00100= light commercial vehicles (Class N1) 00101= heavy duty vehicles (Class N2) 00110= heavy duty vehicles (Class N3) 00111= motorcycles (Class L1e) 01000= motorcycles (Class L2e)

Block No.	Name	**Byte Position First Byte	Last Byte	Type	Unit		Description
							01001= motorcycles (Class L3e) 01010= motorcycles (Class L4e) 01011= motorcycles (Class L5e) 01100= motorcycles (Class L6e) 01101= motorcycles (Class L7e) NOTE: Vehicle definitions class M, N according directive 2007/46/EC; class L according directive 2002/24/EC NOTE: The bit 5 to be set if the position is not within the limits of +-150m with 95% confidence
3	Vehicle identification	4 *4* *7* *13*	20 *6* *12* *20*	String		M	VIN number according ISO 3779 *World Manufacturer Index (WMI)* *Vehicle Type Descriptor (VDS)* *Vehicle Identification Sequence (VIS)*
4	Vehicle propulsion storage type	21		Integer		M	This byte identifies the type of vehicle energy storage(s) present: 0 = indicates a type of storage not present 1 = indicates type of storage which is present All bits set to zero indicates an unknown type of energy storage Bit 7: unused Bit 6 : unused Bit 5 : 1= hydrogen storage Bit 4 : 1= electric energy storage (with more than 42v and 100Ah Bit 3 : 1= liquid propane gas (LPG) Bit 2 : 1= compressed natural gas (CNG) Bit 1 : 1= diesel tank present Bit 0 : 1= gasoline tank present NOTE: This information may be unreliable if there has been a change of vehicle propulsion type (e.g. from gasoline to CNG) NOTE: More than one bit may be set if there is more than one type of energy storage present
5	Time stamp	22	25	Integer	UTC sec	M	Timestamp of incident event As seconds elapsed since midnight January 1st, 1970 UTC.
6	Vehicle Location	26	29	Integer	milliarcsec	M	Position latitude (ISO 6709) Value range (-324000000 to 324000000) Maximum value Latitude = 90°00'00.000" = 90*60*60.000" = 324000.000"

Block No.	Name	**Byte Position First Byte	Last Byte	Type	Unit		Description
							= 324 000 000 Miliarcseconds = 0x134FD900 Minimum value Latitude = -90°00'00.000" = -90*60*60.000" = -324000.000" = -324 000 000 Miliarcseconds = 0xECB02700 *EXAMPLE 48°18'1.20" N = 48.3003333 lat* *= (48*3600)+(18*60)+1.20}" = 173881,200"* *which encodes to the following value:* *= 173881200d = 0x0A5D3770* If latitude is invalid or unknown, the value 0xFFFFFFFF shall be transmitted
		30	33	Integer	milliarcsec	M	Position longitude (ISO 6709) Value range (-648000000 to 648000000) Maximum value Longitude = 180°00'00.000" = 180*60*60.000" = 648000.000" = 648 000 000 Miliarcseconds = 0x269FB200 Minimum value Longitude = -180°00'00.000" = -180*60*60.000" = -648000.000" = -648 000 000 Miliarcseconds = 0xD9604E00 *EXAMPLE. 11°37'2.52" E = 11.6173666 long* *= (11*3600)+(37*60)+2.52}" = 41822.520"* *which encodes to the following value:* *= 41822520d = 0x027E2938* If longitude is invalid or unknown, the value 0xFFFFFFFF shall be used
	Vehicle direction	34	34	Integer	2°-Degree	M	Direction of travel in 2°-degrees steps from magnetic north (0–358, clockwise)
7	Recent Vehicle Location n-1	35	36	Integer	100 milliarcsec	O	Latitude Delta (+ for North and – for South) with respect to Current Vehicle position in Block 6. 1 Unit = 100 miliarcseconds, which is approximately 3m Coded value range -512..511) representing -51200 to +51100 miliarcseconds, or from 51,2''S to 51,1''N from the current position
		36	37	Integer	100 milliarcsec	O	Longitude Delta (+ for East and – for West) with respect to Current Vehicle position in Block 6.

Block No.	Name	**Byte Position First Byte	Last Byte	Type	Unit		Description
							Coded value range -512..511) representing -51200 to +51100 miliarcseconds, or from 51,2''W to 51,1''E from the current position
8	Recent Vehicle Location n-2	37	38	Integer	100 milliarcsec	O	Latitude Delta (+ for North and – for South) with respect to Recent Vehicle Location n-1 in Block 7 1 Unit = 100 miliarcseconds which is approximately 3m Coded value range -512..511) representing -51200 to +51100 miliarcseconds, or from 51,2''S to 51,1''N from the location represented by Recent Vehicle Location n-1
		39	40	Integer	100 milliarcsec	O	Longitude Delta (+ for East and – for West) with respect to Recent Vehicle Location n-1in Block 7, Coded value range -512..511) representing -51200 to +51100 miliarcseconds, or from 51,2''W to 51,1''E from the location represented by Recent Vehicle Location n-1
9	No. of passengers	41	41	Integer		O	Minimum known number of fastened seatbelts, to be set to 255 if no information is available NOTE This information is only relevant for automatic initiated eCalls and is indicative only as it may be not always be reliable in providing exact information about the number of passengers (e.g. because seatbelts may not be fastened by passengers or seatbelts may be fastened for other reasons)
10	Service provider	42	57	String	IPv6	O	Service provider IPv6 address in standard text representation, i.e. using up to eight groups of up to four characters “0-9”, or “a-f” to represent hexadecimal digits with each group separated by a colon (:) character. If one or more four-digit group(s) is 0000, the zeros may be omitted and replaced with two colons(::). Following this rule, any number of consecutive 0000 groups may be reduced to two colons, as long as there is only one double colon used in an address.

Block No.	Name	**Byte Position First Byte	Last Byte	Type	Unit		Description
							It is this (::) notation which allows typical addresses to be sent with much fewer bytes in the MSD. For example, “2001:0db8:85a3:08d3:1319:8a2e:0370:7334“ or “2001:0db8:0000:0000:0000:0000:1428:57ab” or “2001:0db8::1428:57ab” [equivalent to previous] or “::ffff:c000:280”
11	Format field	58	58	Integer		M	Format of the following optional additional data: 0 = No optional additional data* 1 = Binary data 2 = BCD 3 = XML encoded data 4 = ASN.1 BER defined data 5 = ASN.1 PER defined data 6 = ASCII encoded data 7 = (unallocated) * If the value for field 11 is 0 (no additional data) then the message may be truncated in accordance with block 14 below
12	Frame check field	59	62			M	CRC-32-Protection (ISO 3309) to secure mandatory MSD data from Byte 1 to 58**. MSB stored in Byte 59**.
13	Optional additional data	63	94	String	As Specified	O	Further 32 bytes of data encoded according to format field 11 to be used by service providers. Unused bytes shall be filled with blank characters (zeros). NOTE The format of such optional data concepts may be provided in a subsequent version of this Technical Specification or may be found in a data registry that is compliant to CEN ISO/TS 24978. Optional additional data shall be protected by an appropriate CRC.

NOTE Except where explicitly specified or determined in a reference standard, negative values are not allowed

6.2.4 MSD acknowledgement and request

Table 2 provides the content and format of an MSD acknowledgement and the request for MSD transmission by PSAPs operators.

Table 3 — MSD-Acknowledgement format data concept

M – Mandatory data field

Block No.	Name	Byte Position		Type	Unit		Description
1	ID	1		Integer		M	Format version set to 1 to discriminate from subsequent formats
		2		Integer		M	Message Identifier referring to the corresponding Message Identifier (Byte 7) of the received MSD
2	Status	3		Integer		M	= 0 positive acknowledgement = 1 error, repeat or initiate MSD transmission = 2 transaction ended, eCall can be terminated Further values currently unused and reserved for future use.
3	Frame check field	4	5			M	CRC-16-Protection (ITU X.25) to secure acknowledgment data from Byte 1 to 3. MSB stored in Byte 4.

Note: "Except where explicitly specified or determined in a reference standard, negative values are not allowed”

Annex A
(normative)

ASN.1 PER representation of MSD

```
MSDASN1Module

DEFINITIONS
AUTOMATIC TAGS ::=
BEGIN


ECallMessage ::=CHOICE {
 msd    MSDMessage,    -- Minimum Set Of Data uplink from vehicle
 msdAck MSDAckMessage, -- MSD acknowledge downlink to vehicle
 ...
-- By using ASN.1 definitions as above, a frame and delimiter are not required.
-- The ECallMessage structure above supports 2 message types (msd and msdack) but
allows extendibility
-- Thus further message types might be added later in future versions
}

-- The "uplink" msd message from the vehicle follows, using elements defined
below

MSDMessage ::= SEQUENCE {
msdStructure  MSDStructure,
frameCheckINTEGER(0 .. 4294967295),
-- 32 bit frame check of msdStructure
optionalAdditionalData UTF8String (SIZE(1..32)) OPTIONAL,
...             -- Extendible in future versions here e.g. to add extra optional
data
}


-- The main MSD structure follows, using elements described below

MSDStructure  ::= SEQUENCE {
 formatVersion                 INTEGER(0 .. 255),
 messageIdentifier             INTEGER(0 .. 255),
 control                 ControlType,
 vehicleIdentificationNumber  VIN,
 vehiclePropulsionStorageType VehiclePropulsionStorageType,
 timestamp       INTEGER(0 .. 4294967295),
 --Number of seconds since 1970:00:00:00
 vehicleLocation    VehicleLocation,
 vehicleDirection       INTEGER(0 .. 255),
 -- Clockwise heading in 2 degree units
 -- Coded values of 0..179
 -- represent headings from 0..358 deg
 -- Value 255 represents "unknown"
 -- other values are invalid
 recentVehicleLocationN1   VehicleLocationDelta OPTIONAL,
 recentVehicleLocationN2   VehicleLocationDelta OPTIONAL,
 numberOfPassengers     INTEGER(0 .. 255) OPTIONAL,
 serviceProvider    AddressIPV6 OPTIONAL,
```

```
 additionalDataFormatField  INTEGER(0 .. 255),
  ... -- Extendible in future versions here e.g. to add new data elements to MSD
}


-- The following basic elements are used in the main MSD structure MSDStructure

ControlType ::= SEQUENCE {
 activation          BOOLEAN, -- manualActivation (0) / automaticActivation (1),
 callType                BOOLEAN, -- emergency (0)/ testCall (1),
 positionconfidence      BOOLEAN, -- positionCanBeTrusted (0) /
lowConfidenceInPosition (1),
 vehicleType        VehicleType
}

VehicleType ::= ENUMERATED{
 passengerVehicleClassM1 (1),
 busesAndCoachesClassM2 (2),
 busesAndCoachesClassM3 (3),
 lightCommercialVehiclesClassN1 (4),
 heavyDutyVehiclesClassN2 (5),
 heavyDutyVehiclesClassN3 (6),
 motorcyclesClassL1e (7),
 motorcyclesClassL2e (8),
 motorcyclesClassL3e (9),
 motorcyclesClassL4e (10),
 motorcyclesClassL5e (11),
 motorcyclesClassL6e (12),
 motorcyclesClassL7e (13),
 ... -- Extendible in future versions here for new vehicle types
}
--NOTE: Vehicle definitions class M, N according directive 2007/46/EC;
--class L according directive 2002/24/EC

VIN ::= SEQUENCE {
 isowmi PrintableString (SIZE(3)) (FROM("A".."H"|"J".."N"|"P"|"R".."Z"|"0".."9")),
 -- World Manufacturer Index
 isovds PrintableString (SIZE(6)) (FROM("A".."H"|"J".."N"|"P"|"R".."Z"|"0".."9")),
 -- Vehicle Descriptor Section
 isovisModelyear PrintableString (SIZE(1))
(FROM("A".."H"|"J".."N"|"P"|"R".."Z"|"0".."9")),
 -- Modelyear in Vehicle Identifier Section
 isovisSeqPlant PrintableString (SIZE(7))
(FROM("A".."H"|"J".."N"|"P"|"R".."Z"|"0".."9"))
 -- plant code +  sequential number in Vehicle Identifier Section
}

VehiclePropulsionStorageType ::= SEQUENCE {
 gasolineTankPresent   BOOLEAN DEFAULT FALSE,
 dieselTankPresent     BOOLEAN DEFAULT FALSE,
 compressedNaturalGas  BOOLEAN DEFAULT FALSE,
 liquidPropaneGas      BOOLEAN DEFAULT FALSE,
 electricEnergyStorage BOOLEAN DEFAULT FALSE, --with more than 42v and 100Ah
 hydrogenStorage       BOOLEAN DEFAULT FALSE,
 ...  -- Extendible in future versions here for new fuel storage types
 -- Instead of defining "not used"
 -- Extendability Marker, for new version put a "," behind the "..."  and add new
elements
 -- Example: hydrogenStorage BOOLEAN DEFAULT FALSE,
 --          ...,
```

```
 --          xyzStorage BOOLEAN DEFAULT FALSE (here no "," if end of list)
}

VehicleLocation  ::= SEQUENCE {
 positionLatitude   INTEGER(-2147483648..2147483647),
 -- 32 bits (4 octets) allocated to make signed value handling easier
 -- Real latitude values in 1mas units
 -- from -324000000 to 324000000
 -- representing -90° to +90°
 -- 0xFFFFFFF means value not available
 positionLongitude  INTEGER(-2147483648..2147483647)
 -- 32 bits (4 octets) allocated to make signed value handling easier
 -- Real longitude values in 1mas units
 -- from -648000000 to 648000000
 -- representing -180° to +180°
 -- 0xFFFFFFF means value not available
}

VehicleLocationDelta ::= SEQUENCE {
 latitudeDelta      INTEGER (-512..511),
 -- 100 miliarcsecond resolution
 -- allows range 51.2''S to 51.1''N
 longitudeDelta     INTEGER (-512..511)
 -- 100 miliarcsecond resolution
 -- allows range 51.2''W to +51.1''E
}

AddressIPV6 ::= PrintableString (SIZE(0..39)) (FROM("a".."f"|":"|"0".."9"))


-- The "downlink" msdack message to the vehicle follows, using elements defined
below

MSDAckMessage ::= SEQUENCE {
 msdAckStructure MSDAckStructure,
 frameCheck          INTEGER(0 .. 65535)
 -- frame check of msdAckStructure
}

-- The main MSDACK structure follows

MSDAckStructure  ::= SEQUENCE {
formatVersion       INTEGER(0 .. 255),
messageIdentifer    INTEGER(0 .. 255),
msdackstatus            MsdAckStatus,
...
}

MsdAckStatus ::= ENUMERATED{

 positiveAck (0),

 repeatTransmissionRequest (1),

 transactionTerminateAllowed (2),

 ... -- Extendible in future versions here for new Ack status
}

END
```

```
-- Examples for the MSDAck and MSD structures follow
-- These examples do NOT form part of the formal ASN1 definition


msddownlinkexample MSDAckMessage ::= {
   msdAckStructure {
      formatVersion 1,
      messageIdentifer 1,
      status 1
      },
   frameCheck'5A62'H  --should be generated by CRC polynom
   }


msdexample MSDMessage ::= {
   msdStructure {
   formatVersion 1,
   messageIdentifier 1,
   control {
      activation   automaticActivation, --01110000=70
      callType  emergency,
      positionconfidence  positionCanBeTrusted,
      vehicleType  passengerVehicleClassM1
      },
   vehicleIdentificationNumber   "WMIVDSVDSYA123456",
   vehiclePropulsionStorageType {gasolineTankPresent TRUE}, --10000000=80
   timestamp 123456789,
   vehicleLocation {positionLatitude 173881200, positionLongitude 41822520},
   vehicleDirection 14,
   recentVehicleLocationN1 {latitudeDelta  10, longitudeDelta -10},
   recentVehicleLocationN2 {latitudeDelta  10, longitudeDelta -10},
   numberOfPassengers      2,
   serviceProvider         “::ffff:c000:280”,
   additionalDataFormatField            0,
   },
frameCheck            '5A62B0CD'H    --should    be    generated    by    CRC    polynom
}

END
```

Annex B
(informative)

ASN.1 Data representation PER and BER explained

B.1 Introduction to ASN.1 and the 'Packed Encoding Rules'

Source ;Public information from http://www.w3.org/Protocols/HTTP-NG/asn1.html
Prepared by Simon E Spero (ses@tipper.oit.unc.edu)

(Quick overview of ASN.1, BER, and PER adapted by Dave Raggett from an email message by Simon. S)

ASN.1 is a notation for describing data structures; it's very much like a type declaration in C or C++. Let's start with a C++ structure and create the appropriate ASN.1 Data structure. I'll use a simplified form of the the GET request from FHTTP to start with.

```
struct GetRequest {
int headerOnly; // flag: do we only want headers?
int lock; // flag: should we checkout the object?
string url; // the url to fetch
AcceptTypes* acceptTypes; // Optional list of accept types that only apply
// to this request
};

struct AcceptTypes {
List<bitset>* standardTypes; // list of bitmaps indicating which
// preference order for standard types
List<string>* otherTypes; // nonstandard types in preference order
};

GetRequest ::= [APPLICATION 0] IMPLICIT SEQUENCE {
headerOnly BOOLEAN,
lock BOOLEAN,
acceptTypes AcceptTypes OPTIONAL
url OCTET STRING,

}

AcceptTypes ::= [APPLICATION 1] IMPLICIT SEQUENCE {
standardTypes [0] IMPLICIT SEQUENCE OF BIT STRING
{html(0),plain-text(1),gif(2), jpeg(3)} OPTIONAL,

otherTypes [1] IMPLICIT SEQUENCE OF OCTET STRING OPTIONAL
}
For the encoding examples, we'll use the following example.

(GetRequest
:headerOnly TRUE
:lock FALSE
:acceptTypes (AcceptTypes
:standardTypes ((html)
(plain-text)))
:url "/ses/magic/moxen.html")
```

B.2 Basic encoding rules

The Basic Encoding Rules (BER) were the original rules for taking an ASN.1 data type, and turning it into a sequence of bits and bytes. BER uses a form of encoding commonly known as Tag-Length-Value. Each item is encoded as a tag, indicating what type it is, a length indicating the size of the object, and a value, which contains the actual contents of the object.

Tags are reasonably simple - in there simplest form they consist of a single byte; the highest two bits indicate the tag class (whether the tag is an official ISO tag, an application wide tag, a private tag, or a tag that only has meaning for this particular structure. The next bit is a flag to indicate whether the tagged object is a simple type, such as a number or a string, or a compound type made up from a bunch of other types. The remaining 5 bits are used to represent the tag number. If the tag is too big to fit in 5 bits, then these bits are set to all ones, and the tag number is encoded in the following bytes as a sequence of seven bit bytes. The high bit of these bytes is used as a flag to indicate whether there's more tag available.

Lengths are also quite simple. There are two ways of encoding lengths - the definite form, and the indefinite form.

For the definite form, if the length is less than 128, you just use a single byte, with the high bit set to zero. Otherwise the high bit is set to one, and the low seven bits set to the length of length. The length is then encoded in that many bytes.

The indefinite form is encoded by sending a length field with a length of length of 0 - i.e. [1000|0000]. The object is ended by sending two zero bytes.
Here's a test case encoded using the BER

```
0x60 -- [0110|0000], [APPLICATION 0, Compound] - GetRequest
0x80 -- [1000|0000], Indefinite length

0x01 -- [0000|0001], [BOOLEAN] GetRequest.headerOnly
0x01 -- [0000|0001], length 1
0x01 -- [0000|0001], value TRUE

0x01 -- [0000|0001], [BOOLEAN] GetRequest.lock
0x01 -- [0000|0001], length 1
0x00 -- [0000|0000] value FALSE

0x61 -- [0110|0001], [APPLICATION 1, Compound] - GetRequest.types
0x80 -- indefinite length

0xa0 -- [1010|0000], [CONTEXT 0, Compound] types.standardTypes
0x80 -- indefinite length

0x03 -- [0000|0011] [BIT STRING]
0x02 -- length 2
0x04 -- Four bits unused
0x80 -- [1000|0000] {html}

0x03 -- [0000|0011] [BIT STRING]
0x02 -- length 2
0x04 -- Four bits unused
0x40 -- [0100|0000] {plain-text}

0x00
0x00 -- End of list of standard Types
0x00
```

```
0x00 -- End of Accept Types

0x04 -- [0000|0100], [OCTET STRING] GetRequest.url
0x16 -- [0001|0110], length 22
[/ses/magic/moxen.html] -- value

0x00
0x00 -- End of get request

[50 bytes total], 22 url
```

B.3 Packed encoding rules

The packed encoding rules use a different style of encoding. Instead of using a generic style of encoding that encodes all types in a uniform way, the PER specialise the encoding based on the data type to generate much more compact representations.

Packed encoding rules have particular value for messages of known data structure- such as the MSD Inserted by EN 15722 editor)

PER only generates tags when they are needed to prevent ambiguity this only occurs when ASN.1's version of union is used (CHOICE). PER also only generates lengths when the size of an object can vary. Even then, PER tries to represent the lengths in the most compact form possible.

PER encodings are not always aligned on byte boundaries- if the 'aligned' variant of the rules is used, then strings *will* be aligned - otherwise the encoding is treated as a string of bits, allowing things like booleans and small integers to be squished together in one byte.

The presence of optional elements in a sequence is indicated by a list of single bit flags placed at the start of a sequence - if the bit is set, then the option is present.

Here's what our test case looks like in PER.

```
[1] -- flag bit indicates acceptTypes is present
[1] -- boolean, header only, TRUE
[0] -- boolean, lock, FALSE
[1] -- flag bit, indicates standardTypes is present
[0] -- flag bit, indicates otherTypes is absent
[000] -- pad bits to make length octet aligned
[00000010] -- length of 2, -- 2 standardType bit strings to follow
[1000] -- the first bit string, html is set
[0100] -- the second bit string, plain-text is set
[00010110] -- length 22; url is 22 octets long
/ses/magic/moxen.html -- value of url

[total size is 26, 22 url]
```

In this case, the PER are about twice as compact as the BER. If I had chosen a shorter URL, such as /, the difference would have been even greater - BER, 29; PER, 5, for a ratio of over 5-1.