ECALL COMMUNICATIONS TEST BENCH – STRUCTURE AND CONTENT OF FDS AND MDS MESSAGES

1 ECALL TEST BENCH

eCall communications test bench system simulates the operation of in-vehicle terminal, service centre and PSAP (Figure 1).

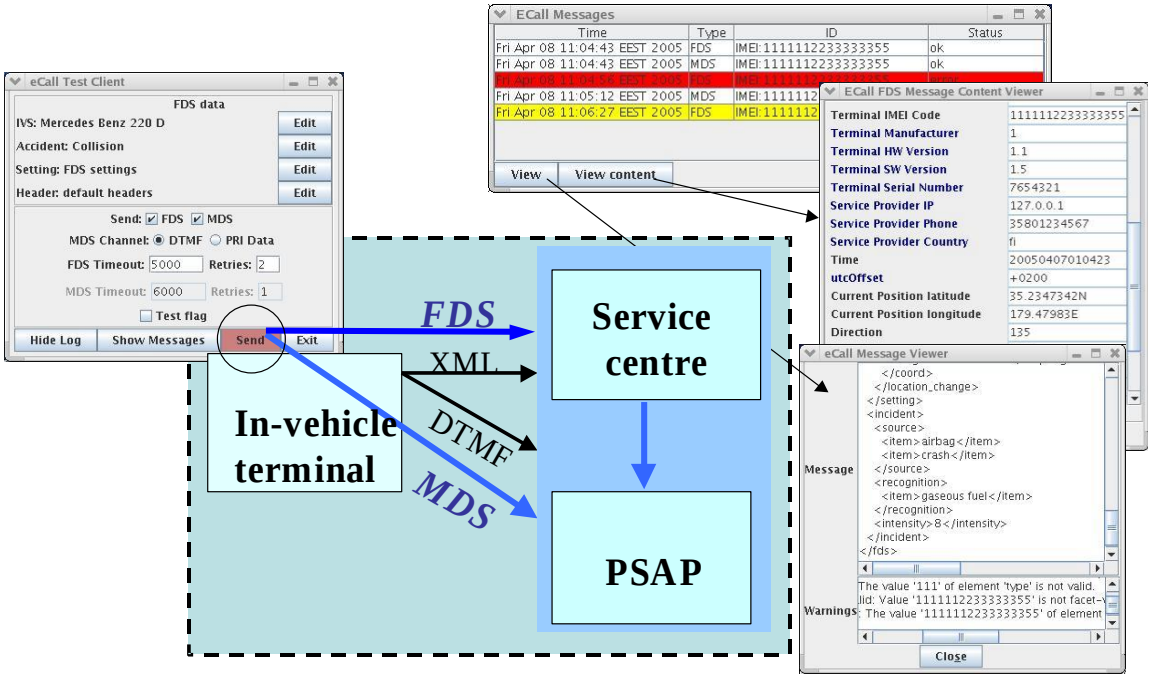

Figure 1. eCall communications test bench.

1.1 Communication framework

The data content and format of eCall messages is based on the eCall prestudy in Finland (within AINO programme) and previous specifications and drafts by E-MERGE (EU project), Telematics Forum (GTP 1.0) and ETSI (OCG EMTEL).

1.1.1 From in-vehicle terminal to PSAP

In-vehicle terminal sends message containing the minimum data set (MDS) to PSAP. The data transmission can be accomplished either using voice channel to send DTMF codes or via IP-network (e.g. GPRS) using the HTTP POST method.

The length of binary encoded MDS message is 19 bytes. Using voice channel (phone call) the message is coded as DTMF signals. One byte produces two DTMF marks, so the length of the message is 38 DTMF marks.

1.1.2 From in-vehicle terminal to service centre

In-vehicle terminal sends full data set (FDS) message to service centre. The message is in XML format and is transmitted over IP-network (GPRS) using the HTTP POST method. The service centre will check the validity of the structure and content of the message using XML schema [http://www.w3.org/XML/Schema].

1.1.3 From service centre to PSAP

Service centre forwards received valid FDS messages to PSAP. Further, the service centre can provide additional information or fill missing information. (For example, the centre may include a database containing information about the vehicle.) The additional information can be sent as separate messages (FDS+) following the original FDS message. All the messages are sent using HTTP POST.

2 THE STRUCTURE AND CONTENT OF THE MESSAGES

2.1 FDS message

FDS message includes the following information:

Content	XML description	Required (X)	Note
Status	`header/flags` element `private` attribute (`true` or *`false`*) `test` attribute (`true` or *`false`*)		
Message type	`header/type` element	X	Section 2.1.1
Version	`header/version` element	X	2.1.1
Message control	`header/control` element `buffered` attribute (`true` or *`false`*) `response` attribute (`true` or *`false`*)		
Privilege level	`header/privilege` element		
Vehicle type	`ivs/vehicle/type` element	X	2.1.2
Carco	`ivs/vehicle/cargo` element	X	2.1.2
Vehicle manufacturer	`ivs/vehicle/manufacturer` element		
Vehicle manufacturing year	`ivs/vehicle/model_year` element		
Vehicle identification number	`ivs/vehicle/vin` element		
Vehicle license number	`ivs/vehicle/license` element		
Vehicle colour	`ivs/vehicle/colour` element		
Vehicle model	`ivs/vehicle/model` element		
Terminal MSID code	`ivs/terminal/msid` element `(MSISDN | IMEI | IMSI)` element	X	2.1.2
Terminal manufacturer	`ivs/terminal/manufacturer` element		
Terminal HW version	`ivs/terminal/hardware` element		
Terminal SW version	`ivs/terminal/software` element		
Terminal serial number	`ivs/terminal/serial` element		
Service provider IP address	`ivs/service_provider/ip_address` element		
Palveluntarjoajan puh. no.	`ivs/service_provider/phone` element		
Service provider country	`ivs/service_provider/country` element		
Timestamp	`setting/time` element	X	2.1.3
Current location	`setting/current_location` element	X	2.1.3

Driving direction	`setting/direction` element	X	2.1.3
Velocity	`setting/velocity` element	X	2.1.3
Previous location	`setting/previous_location` element		
Position change	`setting/location_change` element		
Message source	`incident/source` element	X	2.1.4
Accident recognition	`incident/recognition` element		
Acciddent intensity	`incident/intensity` element		
Number of passagers	`incident/passengers` element		
Accident further data	`incident/data` element		
Other information	`info` element		

Table 1. The content of FDS message.

In the table, the required elements of the message in XML format are shown in italics and marked with (X) in the last column. The default values of attributes are shown in italics.

The structure and semantics of the message is described by an XML schema that is located in the following public address:

http://www.ecall.fi/schemas/fds_schema.xsd

The generic XML structure of FDS message is as follows:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<fds>
    <header>...</header>        (cf. Section 2.1.1)
    <ivs>...</ivs>              (Section 2.1.2)
    <setting>...</setting>      (Section 2.1.3)
    <incident>...</incident>    (Section 2.1.4)
    <info>Additional information</info>
</fds>
```

An example of FDS message:

http://www.ecall.fi/examples/fds_example.xml

2.1.1 FDS header

The type of the message is "FDS, which is coded in XML format ("type" element) as the number "11".

The current message version is "1.0", which is coded in XML representation as the number "0".

```
<header>
     <type>11</type>
     <version>0</version>
</header>
```

2.1.2 Vehicle and in-vehicle terminal information

Vehicle and in-vehicle terminal information are organized as subelements of the element "ivs".

Vehicle type is coded as a number between 0-7. (reserved)

Cargo is coded as a number between 0-255 (reserved)

Message identifier (MSID) is MSISDN, IMEI or IMSI (at least one should be given in the message). The values are represented as follows [3GPP TS 23.003 V6.5.0 (2004-12)]:

IMEI (International Mobile station Equipment Identities)

Representation: 15 digits (always)

TAC (Type Allocation Code): 8 digits

+ SNR (Serial Number): 6 digits

+ 1 digit

MSISDN (Mobile Station International ISDN Number)

Representation: in total at most 15 digits (default)

CC (Country Code):

+ NDC (National Destination Code):

+ SN (Subscriber Number):

IMSI (International Mobile Subscriber Identity)

Representation: in total at most 15 digits

MCC (Mobile Country Code): 3 digits

+ MNC (Mobile Network Code): 2-3 digits

+ MSIN (Mobile Subscriber Identification Number)

An example:

```
<ivs>
     <vehicle>
          <type>11</type>
          <cargo>0</cargo>
     </vehicle>
     <terminal>
          <msid>
               <msisdn>3580123456789</msisdn>
               <imei>012345768901234</imei>
          </msid>
          </terminal>
     </ivs>
```

2.1.3 Timestamp and in-vehicle movement/location information

Timestamp of current location is represented as follows [LIF MLP 3.0.0]:

year, month, day, hours, minutes, seconds

The attribute "utc_off" of the time element is optional. Its default value is utc_off=0.

All coordinates are represented in WGS-84 desimal format.

```
<setting>
     <time utc_off="+0200">20050613010423</time>
     <current_location>
          <coord>
               <latitude>60.123456N</latitude>
               <longitude>24.9876543E</longitude>
          </coord>
     </current_location>
     <direction>130</direction>
     <velocity>178.3</velocity>
</setting>
```

2.1.4 Incident information

Accident information include: message source and recognition, the number of passengers and other additional information. These are organised as subelements of the element "incident".

The values of message source ("source" element) belong to the following set: "manual", "rolled over", "airbag", "crash" and "moved"

```
<incident>
     <source>
          <item>rolled over</item>
          <item>airbag</item>
     </source>
     <passenger>4</passenger>
</incident>
```

2.2 MDS message

MDS message contains only the required data.

The message is encoded tightly to 19 bytes for sending in DTMF format within phone call.

Bytes	Content	Description	Note
1	Header	message type (5 bits) + version (3 bits)	Section 2.2.1
2	Status	source (5 bits) + vehicle type (3 bits)	2.2.2
3	Cargo	cargo type	2.2.2
4-10	Identifier	MSID (IMEI, IMSI or MSISDN)	2.2.3
11-13	Latitudi	WGS84 in degrees (signed -90 90)	2.2.4
14-16	Longitudi	WGS84 in degrees (signed, -180 180)	2.2.4
17	Velocity	km/h (0-254 and 255 when >= 255) (integer)	
18	Direction	degrees * 255 / 360 (rounded to nearest integer)	
19	Checksum	CRC-8	

Table 2. Encoding of MDS message into 19 bytes.

Using DTMF codes it is possible to send only numbers 0-9, letters A-D, and marks # and *. The conversion of binary data to DTMF is performed by translating the 19 bytes into hexadecimal numbers and replacing E and F with # and *, respectively.

(For example, the byte queue "243 14 6" is coded as hexadecimal numbers as follows:

"F3 0E 06" and after the replacement the resulting DTMF sequence "*30#06".)

2.2.1 MDS header

The type of the message is "MDS" which is coded as 5-bit binary number "01011".

The current version of the message is "1.0" which is coded as 3-bit binary number "000".

Thus, the header in binary format is: *01011000*.

2.2.2 Status and cargo

Message source is represented by five bits that is coded as follows:

Source	*Bit*
"manual"	1
"rolled over"	2
"airbag"	3
"crash"	4
"moved"	5

Vehicle type is coded as 3 bits, a number between 0-7. (reserved)

Cargo is coded as 1 byte, a number between 0-255. (reserved)

2.2.3 Coding of the MSID field of the message

The MSID field contains one of the following: IMSI, MSISDN or IMEI. Their values are represented as shown in Section 2.1.2.

Coding of the MSID identifier

The MSID identifier (number) will be translated into 7-byte binary number:

Type	*code*
IMEI	1
MSISDN	2
IMSI	3

Type code is included into the first byte (b0) using 5 bits (bit3-bit7).

Example:

Input: IMEI identifier "001234567890123".
=> in binary format:
00000000 00000001 00011111 01110001 11111011 00000100 11001011
where bytes b0 b1 b2 b3 b4 b5 and b6.

Then, MSID type is coded to the result (IMEI => 1):
=> **00001**000 00000001 00011111 01110001 11111011 00000100 11001011

2.2.4 Coding of Latitude and longitude values

Input:

- Latitude in WGS-84 decimal format (signed [-90, 90])
- Longitude in WGS-84 decimal format (signed [-180, 180]).

The values are coded as 24-bit binary numbers as follows:

Latitude: (latitude+90)*(2^24-1)/180 rounded to nearest integer converted to binary format.

Longitude: (longitude+180)*(2^24-1)/360 rounded to nearest integer converted to binary format.

Example:

latitude = 60.123456

(60.123456+90)*(2^24-1)/180 = 13992519 (rounded to nearest integer)
=> 11010101 10000010 01000111
where bytes b0 b1 and b2.

longitude = -24.123456

(-24.123456+180)*(2^24-1)/360 = 1124234
=> 00010001 00100111 10001010