FMXXXX Protocols

V2.10

Contents

1. FM1100, FM2100, FM2200, FM4100 AND FM4200 DATA PROTOCOL

1.1 AVL data array

Because the smallest information amount that can be written is one bit, there can be some bits left unused when result is byte array. Any unused bits should be left blank.

Codec ID	Number of Data	Data	Number of Data
1 Byte	*1 Byte*	*…*	*1 Byte*

Number of data – number of encoded data (number of records)
In FM4X00 and FM2X00 codec ID is 08

1.2 Data

AVL Data	…	AVL Data

AVL data – encoded data element

1.3 AVL Data

Timestamp	Priority	GPS Element	IO Element
8 Bytes	*1 Byte*	*15 Bytes*	*...*

Timestamp – difference, in milliseconds, between the current time and midnight, January 1, 1970 UTC.

1.4 Priority

0	Low
1	High
2	Panic
3	Security

1.5 GPS Element

Longitude	Latitude	Altitude	Angle	Satellites	Speed
4 Bytes	*4 Bytes*	*2 Bytes*	*2 Bytes*	*1 Byte*	*2 Bytes*

X	Longitude[1]
Y	Latitude[1]
Altitude	In meters above sea level[1]
Angle	In degrees, 0 is north, increasing clock-wise [1]
Satellites	Number of visible satellites[1]
Speed	Speed in km/h. 0x0000 if GPS data is invalid[1]

Longitude and latitude are integer values built from degrees, minutes, seconds and milliseconds by formula.

$$\left(d+\frac{m}{60}+\frac{s}{3600}+\frac{ms}{3600000}\right)*p$$

d	Degrees
m	Minutes
s	Seconds
ms	Milliseconds
p	Precision (10000000)

If longitude is in west or latitude in south, multiply result by –1. To determine if the coordinate is negative, convert it to binary format and check the very first bit. If it is 0, coordinate is positive, if it is 1, coordinate is negative. Example:

Received value: 20 9c ca 80
Converted to BIN: 00100000 10011100 11001010 10000000 first bit is 0, which means coordinate is positive
Convered to DEC: 547146368
For more information see two‘s compliment arithmetics.

1.6 IO element

Event IO ID	N of **Total** IO	N1 of **One Byte** IO	**1'st** IO ID	**1'st** IO Value	…	**N1'th** IO ID	**N1'th** IO Value	N2 of **Two Bytes**	**1'st** IO ID	**1'st** IO Value	…	**N2'th** IO ID	**N2'th** IO Value	N4 of **Four Bytes**	**1'st** IO ID	**1'st** IO Value	…	**N4'th** IO ID	**N4'th** IO Value	N8 of **Eight Bytes**	**1'st** IO ID	**1'st** IO Value	…	**N8'th** IO ID	**N8'th** IO Value
1 Byte	*1 Byte*	*1 Byte*	*1 Byte*	*1 Byte*		*1 Byte*	*1 Byte*	*1 Byte*	*1 Byte*	*2 Bytes*		*1 Byte*	*2 Bytes*	*1 Byte*	*1 Byte*	*4 Bytes*		*1 Byte*	*4 Bytes*	*1 Byte*	*1 Byte*	*8 Bytes*		*1 Byte*	*8 Bytes*

Event IO ID – if data is acquired on event – this field defines which IO property has changed and generated an event. If data cause is not event – the value is 0.

[1] If record is without valid coordinates – (there were no GPS fix in the moment of data acquisition) – Longitude, Latitude and Altitude values are last valid fix, and Angle, Satellites and Speed are 0.

N	total number of properties coming with record (N=N1+N2+N4+N8)
N1	number of properties, which length is 1 byte
N2	number of properties, which length is 2 bytes
N4	number of properties, which length is 4 bytes
N8	number of properties, which length is 8 bytes

1.7 Example

Received data:

```
080400000113fc208dff000f14f650209cca80006f00d6040004000403010115031603000
1460000015d0000000113fc17610b000f14ffe0209cc580006e00c0050001000403010115
0316010001460000015e0000000113fc284945000f150f00209cd20000950108040000000
4030101150016030001460000015d0000000113fc267c5b000f150a50209cccc000930068
0400000004030101150016030001460000015b0004
```

08 – Codec ID
04- Number of Data (4 records)

1’st record data

00000113fc208dff – Timestamp in milliseconds (1185345998335 → 1185345998,335 in Unix Timestamp = 25 Jul 2007 06:46:38 UTC)
00 – Priority

GPS Element

0f14f650 – Longitude 253032016 = 25,3032016º N
209cca80 – Latitude 547146368 = 54,7146368 º E
006f – Altitude 111 meters
00d6 – Angle 214º
04 – 4 Visible sattelites
0004 – 4 km/h speed

IO Element

00 – IO element ID of Event generated (in this case when 00 – data generated not on event)
04 – 4 IO elements in record
03 – 3 IO elements, which length is 1 Byte
01 – IO element ID = 01
01 – 1’st IO element’s value = 1
15 – IO element ID = 21
03 – 21’st IO element’s value = 3
16 – IO element ID = 22
03 – 22’nd IO element’s value = 3
00 – 0 IO elements, which value length is 2 Bytes
01 – 1 IO element, which value length is 4 Bytes
46 – IO element ID = 70

0000015d – 70’th IO element’s value = 349
00 – 0 IO elements, which value length is 8 Bytes

2’nd record data

```
     00000113fc17610b 00 0f14ffe0209cc580006e00c7050001
000403010115031601000146000000015e00
```

3’d record data

```
     00000113fc284945 00 0f150f00209cd200009501080400000
000403010115001603000146000000015d00
```

4’th record data

```
     00000113fc267c5b 00 0f150a50209cccc0000930068040000
000403010115001603000146000000015b00
```

04 – Number of Data (4 records)

2. SENDING DATA OVER TCP/IP

2.1 AVL data packet

AVL packet is used to encapsulate AVL data and send it to server.

Four zeros	Data length	Data	Crc

Four zeros	Four zero bytes (0x00)
Data length	Number of bytes in data field (Integer)
Data	Any AVL data array
CRC	16bit CRC value of data (Integer). Polynomial 0xA001.

2.2 Communication with server

First when module connects to server, module sends its IMEI. IMEI is sent the same way as encoding barcode. First comes short identifying number of bytes written and then goes IMEI as text (bytes).

For example IMEI 123456789012345 would be sent as 000F313233343536373839303132333435

After receiving IMEI, server should determine if it would accept data from this module. If yes server will reply to module 01 if not 00. Note that confirmation should be sent as binary packet.

Then module starts to send first AVL data packet. After server receives packet and parses it, server must report to module number of data received as integer (four bytes).

If sent data number and reported by server doesn't match module resends sent data.

Example:
Module connects to server and sends IMEI:
000F313233343536373839303132333435
Server accepts the module:
01
Module sends data packet:

AVL data packet header	***AVL data array***	***CRC***
Four zero bytes, 'AVL data array' length – 254	CodecId – 08, NumberOfData – 2. (Encoded using continuous bit stream. Last byte padded to align to byte boundary)	CRC of 'AVL data array'
00000000000000FE	0802...(data elements)...02	00008612

Server acknowledges data reception (2 data elements):
00000002

3. SENDING DATA OVER UDP/IP

3.1 UDP channel protocol

UDP channel is a transport layer protocol above UDP/IP to add reliability to plain UDP/IP using acknowledgment packets. The packet structure is as follows:

UDP datagram			
UDP channel packet x N	Packet length Packet Id	2 bytes 2 bytes	Packet length (excluding this field) in big endian byte order Packet id unique for this channel
	Packet Type	1 byte	Type of this packet
	Packet payload	m bytes	Data payload

Packet Type	
0	Data packet requiring acknowledgment
1	Data packet NOT requiring acknowledgment
2	Acknowledgment packet

Acknowledgment packet should have the same *packet id* as acknowledged data packet and empty data payload. Acknowledgement should be sent in binary format.

Acknowledgment packet		
Packet length	2 bytes	0x0003
Packet id	2 bytes	same as in acknowledged packet
Packet type	1 byte	0x02

3.2 Sending AVL data using UDP channel

AVL data are sent encapsulated in UDP channel packets (*Data payload* field).

AVL data encapsulated in UDP channel packet		
AVL packet id (1 byte)	Module IMEI	AVL data array

AVL packet id (1 byte) – id identifying this AVL packet
Module IMEI – IMEI of a sending module encoded the same as with TCP

AVL data array – array of encoded AVL data

Server response to AVL data packet	
AVL packet id (1 byte)	Number of accepted AVL elements (1 byte)

AVL packet id (1 byte) – id of received AVL data packet
Number of AVL data elements accepted (1 byte) – number of AVL data array entries from the beginning of array, which were accepted by the server.

Scenario:
Module sends UDP channel packet with encapsulated AVL data packet (*Packet* type=1 or 0). If packet type is 0, server should respond with valid UDP channel acknowledgment packet. Since server should respond to the AVL data packet, UDP channel acknowledgment is not necessary in this scenario, so *Packet type=1* is recommended.
Server sends UDP channel packet with encapsulated response (*Packet type=1* – this packet should not require acknowledgment)
Module validates *AVL packet id* and *Number of accepted AVL elements*. If server response with valid *AVL packet id* is not received within configured timeout, module can retry sending.

Example:
Module sends the data:

UDP channel header	***AVL packet header***	***AVL data array***
Len – 253, Id – 0xCAFE, Packet type – 01 (without ACK)	AVL packet id – 0xDD, IMEI – 1234567890123456	CodecId – 08, NumberOfData – 2. (Encoded using continuous bit stream)
00FDCAFE01	DD000F3133343536373839303132333435	0802…(data elements)…02

Server must respond with acknowledgment:

UDP channel header	***AVL packet acknowledgment***
Len – 5, Id – 0xABCD, Packet type – 01 (without ACK)	AVL packet id – 0xDD, NumberOfAcceptedData – 2
0005ABCD01	DD02

4. SENDING DATA USING SMS

AVL data or events can be sent encapsulated in binary SMS. TP-DCS field of these SMS should indicate that message contains 8-bit data (for example: TP-DCS can be 0x04).

SM data (TP-UD)	
AVL data array	*IMEI:* 8 bytes

AVL data array – array of encoded AVL data
IMEI – IMEI of sending module encoded as a big endian 8-byte long number.

5. 24 POSITION SMS DATA PROTOCOL

24-hour SMS is usually sent once every day and contains GPS data of last 24 hours. TP-DCS field of this SMS should indicate that message contains 8-bit data (i.e. TP-DCS can be 0x04).

Note, that 24 position data protocol is used only with subscribed SMS. Event SMS use standard AVL data protocol.

5.1 Encoding

To be able to compress 24 GPS data entries into one SMS (140 octets), the data is encoded extensively using bit fields. Data packet can be interpreted as a bit stream, where all bits are numbered as follows:

Byte 1	***Byte 2***	***Byte 3***	***Bytes 4-…***
Bits 0-7	Bits 8-15	Bits 16-24	Bits 25-…

Bits in a byte are numbered starting from least significant bit. A field of 25 bits would consist of bits 0 to 24 where 0 is the least significant bit and bit 24 – most significant bit.

5.2 Structure

SMS Data Structure			
	Size (bits)	Field	Description
	8	CodecId	CodecId = 4
	35	Timestamp	Time corresponding to the first (oldest) GPS data element, represented in seconds elapsed from 2000.01.01 00:00 EET.
	5	ElementCount	Number of GPS data elements.
ElementCount *		GPSDataElement	GPS data elements.
		Byte-aling padding	Padding bits to align to 8-bits boundary
	64	IMEI	IMEI of sending device as 8-byte long integer

The time of only the first GPS data element is specified in *Timestamp* field. Time corresponding to each further element can be computed as e*lementTime* = *Timestamp* + (1 hour * elementNumber).

GPSDataElement				
		Size (bits)	Field	Description
		1	ValidElement	ValidElement=1 – there is a valid GpdDataElement following, ValidElement=0 – no element at this position.

GPSDataElement				
ValidElement == 1		1	DifferentialCoords	Format of following data.
	DifferentialCoords == 1	14	LongitudeDiff	Difference from previous element's longitude. LongitudeDiff = prevLongitude – Longitude + 2^{13} – 1
		14	LatitudeDiff	Difference from previous element's latitude LatitudeDiff = prevLatitude – Latitude + 2^{13} – 1
	DifferentialCoords == 0	21	Longitude	Longitude= {(LongDegMult + 18 * 10^8) * (2^{21} – 1)} over {36*10^8}
		20	Latitude	Latitude=(LatDegMult + 9*10^8) * (2^{20} – 1) over {18*10^8}
		8	Speed	Speed in km/h.

Longitude	longitude field value of *GPSDataElement*
Latitude	latitude field value of *GPSDataElement*
LongDegMult	longitude in degrees multiplied by 10^7 (integer part)
LatDegMult	latitude in degrees multiplied by 10^7 (integer part)
prevLongitude	longitude field value of previous *GPSDataElemen*
prevLatitude	latitude field value of previous *GPSDataElement*

5.3 Decoding GPS position

When decoding GPS data with *DifferentialCoords*=1, *Latitude* and *Longitude* values can be computed as follows: *Longitude*=*prevLongitude* – *LongitudeDiff* + 2^{13} – 1, Latitude=*prevLatitude* – *LatitudeDiff* + 2^{13} – 1.

If there were no previous non-differential positions, differential coordinates should be computed assuming prevLongitude=prevLatitude=0.

When *Longitude* and *Latitude* values are known, longitude and latitude representation in degrees can be computed as follows:

$$LongDeg = \frac{Longitude*360}{2^{21}-1} - 180 \qquad LatDeg = \frac{Latitude*180}{2^{20}-1} - 90$$

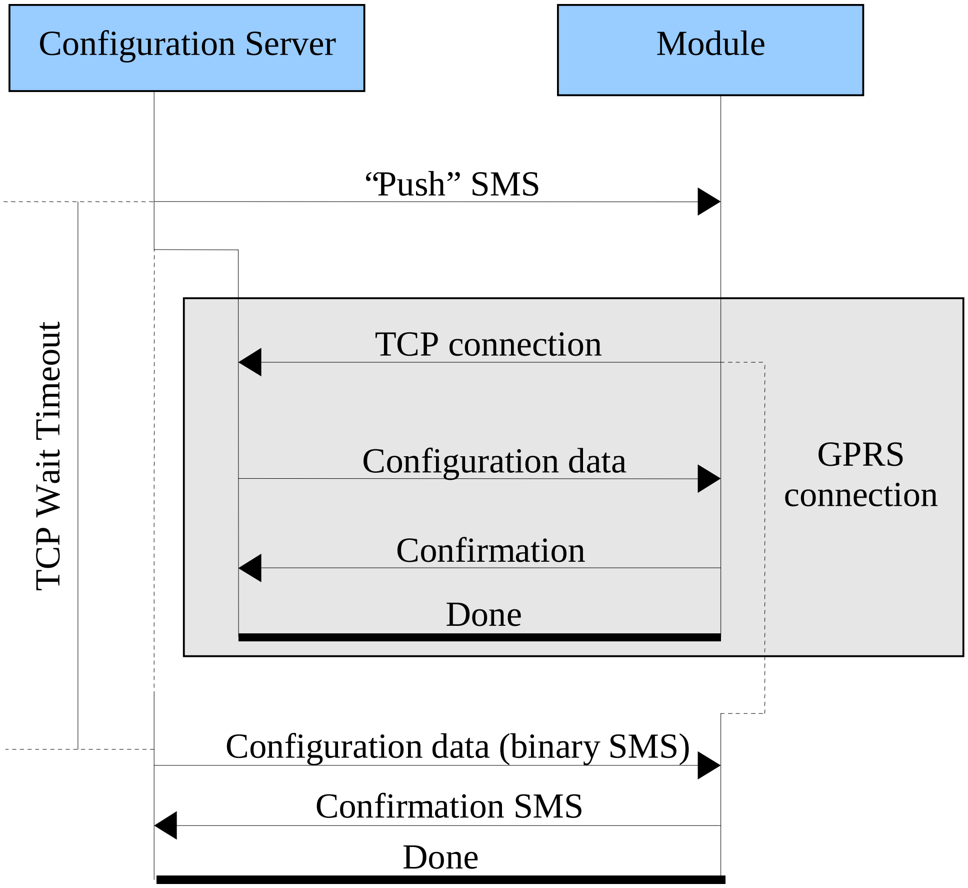
6. REMOTE CONFIGURATION

FM2X00 and FM4X00 share the same configuration protocol, but FM2X00 have only one profile (Profile No. 1). Sending configuration for more than one profile for FM2X00 might cause it to stop responding.

6.1 Configuration process

To initiate configuration process, configuration server sends binary initiation SMS ("Push" SMS) containing server host(ip address) and tcp port device should connect to and waits for TCP connection. Upon reception of "push" SMS, device tries to establish TCP connection to configuration server using GPRS. If TCP connection attempt succeeds, server sends out configuration data to device over established connection, device confirms configuration reception and configures itself. If device doesn‘t connect to server in TcpWaitTimeout time, server stops waiting for TCP connection, sends out configuration data using binary SMS, and waits for confirmation SMS from device. If confirmation SMS doesn‘t arrive in specified time, server assumes that configuration process failed.

Note: this is the preferred configuration procedure, but it is also possible to omit "Push" SMS and proceed directly to configuration via binary SMS.



6.2 Initiation SMS (“push” SMS)

“Push” SMS is sent to device to initiate configuration process. It contains authorization data, host and tcp port of configuration server (device should connect to this address to retrieve new configuration data). When sending “push” SMS, TP-Data-Coding-Scheme (TP-DCS) should be set to 0xF5 and TP-User-Data-Header-Indicator (TP-UDHI) should be 1.

“Push” SMS body (TP-UD)			
	Data(hex)	***Length***	***Description***
TP-UDH	060504	3 bytes	
	wdpPushPort	2 bytes	WDP Port listening for “push” SMS. Default: 0x07D1. BE byte order.
	0000	2 bytes	
TP-UD	LoginLength	1 byte	
	Login	LoginLength bytes	Device identifier (Can be set using FM4X00 or FM2X00 Configurator under “SMS” -> “Login”)
	PasswordLength	1 byte	
	Password	PasswordLength bytes	Device identifier (Can be set using FM4X00 or FM2X00 Configurator under “SMS” -> “password”)
	HostLength	1 byte	
	ServerHost	HostLength bytes	Configuration server host (ip address).
	ServerPort	2 bytes	Configuration server tcp port. BE byte order.
	APNLength	1 byte	Max 32 bytes
	APNAddress	APNLength bytes	APN name. If CHAP authentication is required – append ‘:c’, for PAP authentication – append ‘:p’.[2]
	GPRSLoginLength	1 byte	Max 30 bytes
	GPRSLogin	GPRSLoginLength bytes	CHAP user name (if exist)
	GPRSPasswordLength	1 byte	Max 30 bytes
	GPRSPassword	GPRSPasswordLength bytes	CHAP password (if exists)

[2] :c and :p should be counted into APNLenght bytes

6.3 Configuration packet format

Configuration data is sent to device encoded in configuration packet, the configuration packet format is the same whether configuring over GPRS or binary SMS.

Configuration packet			
	Data(hex)	***Length***	***Description***
	PacketLength	2 bytes	Packet length (this field is **not** counted). BE byte order.
	PacketId	1 byte	Packet id (can be freely chosen – used in confirmation response).
	ParamCount	2 bytes	Number of configuration parameters
Param Count times	ParamId	2 bytes	Configuration parameter id (BE byte order).
	ParamValueLength	2 bytes	Length of parameter value (BE byte order).
	ParamValue	ParamValueLength bytes	Parameter value (UTF-8 encoded string).
	…		

6.4 Configuring via TCP/IP connection

Upon reception of "push" SMS, device tries to establish a TCP connection to configuration server. If connection succeeds, configuration is done in following steps:

Device sends it's IMEI to server in following format:

IMEILength	2 bytes	Length of IMEI (BE byte order)
IMEI	IMEILength bytes	IMEI encoded in UTF-8

- Server sends configuration data:

ConfigurationPacket	PacketLength+2 bytes	Configuration data packet encoded as described in section 6.3

- When device receives valid configuration, it confirms configuration reception with following response:

PacketId	1 byte	Id of configuration packet received by the device
PacketLength	2 bytes	The *PacketLength* field of received configuration packet (BE byte order)

- Configuration done.

6.5 *Configuring using binary SMS*

6.5.1 Sending configuration data

Since one SMS can transfer at most 140 bytes, configuration data have to be split into multiple SMS. Each configuration SMS should have TP-Data-Coding-Scheme (TP-DCS) set to 0xF5 and TP-User-Data-Header-Indicator (TP-UDHI) set to 1.

Configuration data		
ConfigurationPacket	PacketLength+2 bytes	Configuration data packet encoded as described in section 6.3

Configuration data SMS			
	Data(hex)	***Length***	***Description***
TP-UDH	060504	3 bytes	
	wdpConfigPort	2 bytes	WDP Port listening for configuration data SMS. Default: 0x07D5. BE byte order.
	0000	2 bytes	
TP-UD	LoginLength	1 byte	
	Login	LoginLength bytes	Device identifier (Can be set using FM4X00 or FM2X00 Configurator SMS->Login)
	PasswordLength	1 byte	
	Password	PasswordLength bytes	Device password(Can be set using FM4X00 or FM2X00 Configurator SMS->Password)
	TransferId	1 byte	Id unique for all messages of single configuration.
	TotalParts	1 byte	Number of SMS used to transfer configuration.
	CurrentPart	1 byte	Current SMS sequence number in current transfer. Numbering starts from 0.
	ConfigurationData	140 – (12 + LoginLength + PasswordLength) bytes	Part of configuration data

6.5.2 Device‘s confirmation SMS

When device receives all configuration SMS, it assembles configuration data from parts. If received configuration packet is valid, device sends confirmation SMS back to the server and configures itself. TP-Data-Coding-Scheme (TP-DCS) of confirmation SMS is 0x04.

Confirmation SMS			
	Data(hex)	Length	Description
TP-UD	0xFF	1 byte	
	PacketId	1 byte	Id of configuration packet received by the device
	PacketLength	2 byte	The PacketLength field of received configuration packet (BE byte order)

6.6 *Example of configuration over TCP*

Push SMS (Server -> Device)

060504 07d1 0000 03 616161 03 626262 0b 3139322e3136382e312e31 aabb 08 696e7465726e65743a63 04 75736572 00

060504	
07d1	WdpPushPort – 0x07d1
0000	
03	Login length – 3
616161	Login – 'aaa'
03	Password length – 3
626262	Password – 'bbb'
0b	Host length – 11
3139322e3136382e312e31	Host – '192.168.1.1'
aabb	Port – 43707
0a	APN length – 10
696e7465726e65743a63	'internet:c'. APN('internet') with CHAP authentication (':c')
04	CHAP/PAP username length – 4
75736572	CHAP/PAP username – 'user'
01	CHAP/PAP password length – 1
61	CHAP/PAP password – 'a'

Device makes TCP connection to server (192.168.1.1:43707)

IMEI (Device -> Server)

000f 313233343536373839303132333435

000f	Length of IMEI – 15
313233343536373839303132333435	IMEI – ‘123456789012345’

Configuration packet (Server -> Device)

0092 8c 001b 03e8 0001 30 03f2 0001 31 03f3 0002 3230 03f4 0002 3130 03fc 0001 30 0406 0001 30 0407 0001 30 0408 0001 30 0409 0001 30 040ª 0001 30 0410 0001 30 0411 0001 30 0412 0001 30 0413 0001 30 0414 0001 30 041ª 0001 30 041b 0001 30 041c 0001 30 041d 0001 30 041e 0001 30 0424 0001 30 0425 0001 30 0426 0001 30 0427 0001 30 0428 0001 30 0cbd 000c 2b3337303434343434343434

0092	PacketLength – 146
8c	Packet id – 0x8c
001b	Param count – 27
03e8	Param id – 1000
0001	Param value length – 1
30	Param value – ‘0’
03f2	Param id – 1010
0001	Param value length – 1
31	Param value – ‘1’
03f3	Param id – 1011
0002	Param value length – 2
3230	Param value – ‘20’
03f4	Param id – 1012
0002	Param value length – 2
3031	Param value – ‘10’
03fc	Param id – 1020
0001	Param value length – 1
30	Param value – ‘0’
0406	Param id – 1030
0001	Param value length – 1
30	Param value – ‘0’
0407	Param id – 1031
0001	Param value length – 1

30	Param value – '0'
0408	Param id – 1032
0001	Param value length – 1
30	Param value – '0'
0409	Param id – 1033
0001	Param value length – 1
30	Param value – '0'
040a	Param id – 1034
0001	Param value length – 1
30	Param value – '0'
0410	Param id – 1040
0001	Param value length – 1
30	Param value – '1'
0411	Param id – 1041
0001	Param value length – 1
30	Param value – '0'
0412	Param id – 1042
0001	Param value length – 1
30	Param value – '0'
0413	Param id – 1043
0001	Param value length – 1
30	Param value – '0'
0414	Param id – 1044
0001	Param value length – 1
30	Param value – '0'
041a	Param id – 1050
0001	Param value length – 1
30	Param value – '0'
041b	Param id – 1051
0001	Param value length – 1
30	Param value – '0'
041c	Param id – 1052
0001	Param value length – 1
30	Param value – '0'

041d	Param id – 1053
0001	Param value length – 1
30	Param value – ‘0’
041e	Param id – 1054
0001	Param value length – 1
30	Param value – ‘0’
0424	Param id – 1060
0001	Param value length – 1
30	Param value – ‘0’
0424	Param id – 1061
0001	Param value length – 1
30	Param value – ‘0’
0425	Param id – 1062
0001	Param value length – 1
30	Param value – ‘0’
0426	Param id – 1063
0001	Param value length – 1
30	Param value – ‘0’
0426	Param id – 1064
0001	Param value length – 1
30	Param value – ‘0’
0427	Param id – 1065
0001	Param value length – 1
30	Param value – ‘0’
0cbd	Param id – 3261
000c	Param value length – 12
2b3337303434343434343434	ParamValue – ‘+37044444444’

Device response (Device -> Server)

8c0092

8c	Received packet id – 0x8c
0092	Packet length field of received configuration packet – 146

6.7 Example of configuration using binary SMS

Send configuration SMS 1 of 2 (Server -> Device)

```
    060504 07d5 0000 03 616161 03 626262 aa 02 00
00928c001b03e800013003f200013103f30002323003f40002313003fc0001300406000130040700013004080001300
409000130040a00013004100001300411000130041200013004130001300414000130041a000130041b000130041c0
00130041d000130041e000130042400013004250001300426000130
```

060504	
07d5	WdpConfigPort – 0x07D5
0000	
03	Login length – 3
616161	Login – 'aaa'
03	Password length – 3
626262	Password – 'bbb'
aa	TransferId – 0xaa
02	Total parts – 2
00	Current part – 0
00928c001b03e800013003f200013103f300023230 03f40002313003fc000130040600013004070001300 40800013004090001300040a00013004100001300 4110001300412000130041300013004140001300 4 1a000130041b000130041c000130041d000130041e 0001300424000130042500013004260001300	Part 1 of configuration data

Send configuration SMS 2 of 2 (Server -> Device)

```
        060504 07d5 0000 03 616161 03 626262 aa 02 01
    04270001300428000130 0cbd000c2b3337303434343434343434
```

060504	
07d5	WdpConfigPort – 0x07D5
0000	
03	Login length – 3
616161	Login – 'aaa'
03	Password length – 3
626262	Password – 'bbb'

aa	TransferId – 0xaa
02	Total parts – 2
01	Current part – 1
04270001300428000130 0cbd000c2b3337303434343434343434	Part 2 of configuration data

Device’s response SMS (Device -> server)

ff 8c 0092

ff	
8c	Received packet id – 0x8c
0092	PacketLength field of received configuration packet – 146

7. CHANGE LOG

Nr.	Date	New version number	Comments
1	080821	2.1	1.5.2; 1.5.3; 1.8.2. 1.8.6 corrected
2	081007	2.2	1.8; 1.9 chapters corrected
3	081023	2.3	2 chapter revised – CAN property explanation added.
4	081112	2.4	Parameter and property list moved to User Manual document. Updated remote configuration chapter.
5	090811	2.5	Included coordinate decoding sample, minor fixes in sample packet, FM2200 compatibility included.
6	091202	2.6	Minor formatting fixes.
7	100107	2.7	Shortened document name from "FM2100, FM2200, FM4100 and FM4200 Protocols" to "FMXXXX Protocols". Major formatting revision.
8	110113	2.8	Corrected GPS element description in page 4.
9.	111110	2.9	Corrected Binary SMS example in page 22.
10.	120224	2.10	Minor formatting fixes.