GH3000 TCP-IP/UDP-IP DATA PROTOCOLS

DRAFT

(ver. 1.08)

(Confidential document)

This document describes protocol of data send via TCP/IP from GH3000 devices to the server.

TABLE OF CONTENTS

1. Communication with server

All the multibyte fields and values in this protocol use big-endian byte order, if not specified otherwise.

big-endian – of a computer, the most significant byte of a multibyte number is stored at a lower address than the least significant byte, that is, "big end" first

(http://en.wiktionary.org/wiki/big-endian)

little-endian – of a computer, storing most significant byte of a multibyte number is stored at a higher address than the least significant byte, that is, "little end" first

(http://en.wiktionary.org/wiki/little-endian)

First when device connects to server, module sends its IMEI. IMEI is sent in this way:

Length (2 Bytes)	**IMEI (CHAR data)**

First comes short identifying number of Bytes written and then goes IMEI as ASCII text (Bytes).

For example:

IMEI 123456789012345 will be sent as

→ 00 0F 31 32 33 34 35 36 37 38 39 30 31 32 33 34 35

After receiving IMEI, server should determine if it would accept data from this module. If yes, server will reply to module **0x01**, if not – replay **0x00**. Server must send answer – 1 Byte in HEX format.

After module received the positive answer, it starts to send first avl data packet. After server receives packet and parses it, server should report to module number of records received as integer (4 Bytes) in big-endian byte order. By the example below, server must send answer:

→ 00 00 00 04

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

2. TCP/IP Packet structure

Avl packet is used to encapsulate avl data and send it to server.

Preamble	Data Length	Data	CRC

Preamble Four zero Bytes (0x00)

Data length Number of Bytes in data field (Integer)

Data Any avl data array

CRC 16 bit CRC value of data (Integer). Polynomial **0xA001** (value is always 4 Bytes **00 00 XX XX** ; where **XX XX** is 16 bit CRC)

3. Data array structure

Codec ID	Number of Records	Record	…	Record	Number of Records

Codec ID Codec ID – **0x07** (1 Byte)

Number of Records Number of Records in Data array (1 Byte)

Record Information about one position point (coordinates, Altitude, Speed, etc.)

4. Packet and Data structure

Preamble	4B										
Data Length	4B										
Data	>=8 B	**Codec ID**	1B								
		Number of Records	1B								
		Record	>=5 B	**Priority + Timestamp**	4B	**Timestamp (from 2008.01.01 00:00:00)**	30 b	(0 ÷ 29)b			
						Priority	2b	(30 ÷ 31)b			
				Global Mask	1B	GPS Element	1b	0b			
						IO Element 1B	1b	1b			
						IO Element 2B	1b	2b			
						IO Element 4B	1b	3b			
						-	1b	4b			
						-	1b	5b			
						-	1b	6b			
						-	1b	7b			
				GPS Element	>=2 B	**Mask**	1B		Latitude, Longitude	1b	0b
									Altitude	1b	1b
									Angle	1b	2b
									Speed	1b	3b
									Satellites	1b	4b
									Cell ID	1b	5b
									Signal Quality	1b	6b
									Operator Code	1b	7b
						Latitude, Longitude	8B				
						Altitude	2B				
						Angle	1B				
						Speed	1B				
						Satellites	1B				
						Location area code, Cell ID	4B				
						Signal Quality	1B				
						Operator code	4B				
				IO Element 1B	>=3 B	**Quantity**	1B				
						ID	1B				
						Value	1B				
						…	…				
						Id	1B				
						Value	1B				
				IO Element 2B	>=4 B	**Quantity**	1B				
						ID	1B				
						Value	2B				
						…	…				
						…	…				
						Id	1B				
						Value	2B				
				IO Element 4B	>=6 B	**Quantity**	1B				
						ID	1B				
						Value	4B				
						…	…				
						Id	1B				
						Value	4B				
		Record	>=5 B								
		…	…								
		Record	>=5 B								
		Number of Records	1B								
CRC	4B										

5. Record array structure

Priority and Timestamp	Global Mask	Element	…	Element

Priority

Priority of Record (2 bits):
00 – Track point
01 – Periodic point
10 – Alarm point
11 – Reserved (not used)

Timestamp

Time in seconds from 2008.01.01 00:00:00 UTC (30 bits)

Global Mask

Show Elements in Record (1 Byte):
0b – GPS Element
1b – IO Element 1B
2b – IO Element 2B
3b – IO Element 4B

Element

Information about one point position (coordinates, Altitude, Speed, etc.)

6. GPS Element structure

Mask	GPS Element Data segments

Mask

Show Data Parameters in one GPS Element (1 Byte):
0b – Latitude and Longitude
1b – Altitude
2b – Angle
3b – Speed
4b – Satellites
5b – Local Area Code and Cell ID
6b – Signal Quality
7b – Operator Code

GPS Element Data segments

Latitude (4 Bytes, float)
Longitude (4 Bytes, float)
Altitude (2 Bytes, signed short)
Angle = value * 360 / 256 (1 Byte)
Speed (1 Byte)
Quantity of Satellites (1 Byte)
Local Area Code (2 Bytes)
Cell ID (2 Bytes)
GSM Signal Quality (1 Byte) – range [0 … 31]
Operator Code (4 Bytes)

For coordinates parsing, IEEE754 protocol is used:

http://www.h-schmidt.net/FloatApplet/IEEE754.html

7. IO Element structure

Quantity	IO Element Data	…	IO Element Data

Quantity	Quantity of IO Elements Data (1 Byte)
IO Element Data	Parameter ID (1 Byte) Parameter Value (1, 2 or 4 Bytes)

Parameter ID [decimal]	Parameter ID [HEX]	Parameter	Bytes	Description
1	1	Battery	1	Battery Level [%]
2	2	USB	1	USB connected or not
5	5	Live Time	4	Device work time after last reboot [sec]
20	14	HDOP	2	HDOP value
21	15	VDOP	2	VDOP value
22	16	PDOP	2	PDOP value
67	43	Battery voltage	2	Battery voltage, [mV]
220	DC	Time to FIX	4	GPS time to first FIX, [sec]
221	DD	Button ID	1	Pressed button, [0-4]
222	DE	Alarm activation	1	Alarm activation cause [none:0, button:1, SMS:2, AOC:3, ManDown: 5, Parking:6, Restore after reset:7]
240	F0	Movement	1	Movement [0/1 – No/Yes]
244	F4	Roaming	1	Roaming [0 – home, 1 - roaming]

Final DOP value is calculated from DOP value taken from GPRS packet and divided by 10. Example: **67** HEX = 103 / 10 = 10.3

8. Source code for CRC calculation

```
public static int getCrc16(byte[] buffer) {
    return getCrc16(buffer, 0, buffer.length,  0xA001, 0);
}

public synchronized static int getCrc16(byte[] buffer, int offset, int bufLen, int polynom, int preset) {
    preset &= 0xFFFF;
    polynom &= 0xFFFF;

    int crc = preset;
    for (int i = 0; i < bufLen; i++) {
        int data = buffer[i + offset] & 0xFF;
        crc ^= data;
        for (int j = 0; j < 8; j++) {
            if ((crc & 0x0001) != 0) {
                crc = (crc >> 1) ^ polynom;
            } else {
                crc = crc >> 1;
            }
        }
    }

    return crc & 0xFFFF;
}
```

9. Parsing example

Module connects to server and sends IMEI:

000F313233343536373839303132333435

Server accepts the module:

01

Module sends data packet:

070441bf9db00fff425adbd741ca6e1e009e1205070001030b160000601a02015e0200031400661500 0a160067010500000ce441bf9d920fff425adbb141ca6fc900a2b218070001030b160000601a02015e02000314006615000a160067010500000cc641bf9d740fff425adbee41ca739200b6c91e070001030b1f0000601a02015f02000314006615000a160066010500000ca841bf9cfc0fff425adba041ca70c100b93813070001030b1f0000601a02015f02000314002315000a160025010500000c3004

Server reports about successful data transfer (sends number of received records in HEX big-endian):

00 00 00 04

Device from the memory deletes sent coordinates and if all data were sent – device closes connection with the server. If there are more data in the memory, device starts to send next packet. If connection were not closed – device won't send IMEI number and will send data packet started from preamble.

9.1 Packet Data structure

Codec ID: 07

Number of Records: 04

Record No.1

41 bf 9d b0 0f ff 42 5a db d7 41 ca 6e 1e 00 9e 12 05 07 00 01 03 0b 16 00 00 60 1a 02 01 5e 02 00 03 14 00 66 15 00 0a 16 00 67 01 05 00 00 0c e4

Record No.2

41 bf 9d 92 0f ff 42 5a db b1 41 ca 6f c9 00 a2 b2 18 07 00 01 03 0b 16 00 00 60 1a 02 01 5e 02 00 03 14 00 66 15 00 0a 16 00 67 01 05 00 00 0c c6

Record No.3

41 bf 9d 74 0f ff 42 5a db ee 41 ca 73 92 00 b6 c9 1e 07 00 01 03 0b 1f 00 00 60 1a 02 01 5f 02 00 03 14 00 66 15 00 0a 16 00 66 01 05 00 00 0c a8

Record No.4

41 bf 9c fc 0f ff 42 5a db a0 41 ca 70 c1 00 b9 38 13 07 00 01 03 0b 1f 00 00 60 1a 02 01 5f 02 00 03 14 00 23 15 00 0a 16 00 25 01 05 00 00 0c 30
Number of Records: 04

9.2 Record Data parsing

41 bf 9d b0	Priority + Timestamp **41 bf 9d b0** [HEX] = **01000001101111111001110110110000** [BIN], there **01** – Priority (Periodical point) **00 0001 1011 1111 1001 1101 1011 0000** – Timestamp **1101111111001110110110000** = **29334960 sec** from 2007.01.01 00:00 **29334960 sec = 2007.12.06 12:36:00 UTC**
0f	Global Mask (**0F** – all Elements (GPS, IO 1B, IO 2B, IO 4B))
ff	GPS Element Mask (**FF** – all GPS Element Data Segments)
42 5a db d7	Latitude **N54.714687** (coordinates parsing by IEEE754 protocol)
41 ca 6e 1e	Longitude **E25.303768** (coordinates parsing by IEEE754 protocol)
00 9e	Altitude **158 m**
12	Angle **25 deg** (**12** HEX = 18 * 360 / 256 = 25.31 deg)
05	Speed **5 km/h**
07	Quantity of Satellites **7**
00 01 03 0b	Cell ID information – **LAC:0001 CI:030B**
16	Signal Quality **22** (range 0-31)
00 00 60 1a	Operator Code **24602** (Bite GSM, Lithuania)
02	1B IO Elements quantity: **2**
01	IO Element ID: **1 – Battery Level**
5e	Value: **94%**
02	IO Element ID: **2 – USB connection**
00	Value: **0 - USB not connected**
03	2B IO Elements quantity: **3**
14	IO Element ID: **20 – HDOP value**
00 66	Value: **10.2** (**66** HEX = 102 / 10 = 10.2)
15	IO Element ID: **21 – VDOP value**
00 0a	Value: 1 (0A HEX = 10 / 10 = 1)
16	IO Element ID: 22 - PDOP value
00 67	Value: 10.3 (67 HEX = 103 / 10 = 10.3)
01	4B IO Elements quantity: 1
05	IO Element ID: 5 - Life Time value
00 00 0c e4	Value: 3300 sec

10. Advanced parsing example

GH3000 data packets sends in HEX coding

Then module connects to server and sends IMEI

000f 33 35 32 38 34 38 30 32 30 30 37 39 33 31 31 – *IMEI 352848020079311*

Then goes server acceptance:

01 – *Server accepts the module*

After acceptance module sends data packet:

0000000000000045070244d4fae007df425ae4d341cab3fb009c9f0004170000601a02015d02010114005f44d4 fb1507df425ae4cc41cab3d20091ae0104160000601a02015d020101140062020000b63f

10.1 Data record parsing

00 00 00 00 - *Preamble*

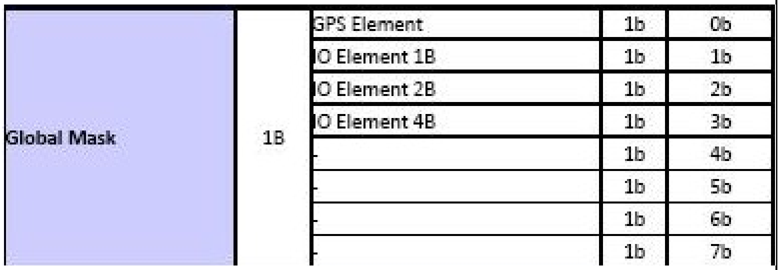
00 00 00 45 – *Data length:* ***69*** [DEC]

07 – *Codec ID*

02 – *Number of Records:* ***2*** [DEC]

44 d4 fa e0 – *Priority & Timestamp –* ***1000100110101001111101011100000*** [BIN] = 31 bit
Priority & Timestamp must be 32 bit lenght, so one bit is missing. To recover it we need to add a **0**.
0*1 000 1001 1010 1001 1111 0101 1100 000*
01 – *Priority (Periodical point*)
1001 1010 1001 1111 0101 1100 000 – *Timestamp:* ***81066720 sec*** [DEC] (from 2008.01.01)

07 – *Global Mask –* ***111*** [BIN]
Global Mask has 8 elements (bits) (4 present, 4 not). So to know which element is present in Global Mask we add missing bits as zeros and it will look like: ***00000111***.
According to Global Mask table it has *GPS Element, IO Element 1B* and *IO Element 2B.*



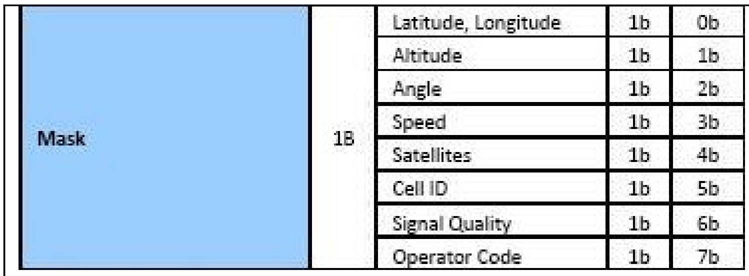
Global Mask	1B	GPS Element	1b	0b	**1**
		IO Element 1B	1b	1b	**1**
		IO Element 2B	1b	2b	**1**
		IO Element 4B	1b	3b	**0**
		-	1b	4b	**0**
		-	1b	5b	**0**
		-	1b	6b	**0**
		-	1b	7b	**0**

Global mask table

NOTE: Global mask and GPS Element Mask taken from Packet and Data structure table

df – *GPS Element Mask* - ***1101 1111*** [BIN]

GPS Element Mask has 8 elements (bits).
According to GPS Element Mask table it has *Latitude, Longitude, Altitude, Angle, Speed, Satellites, Signal Quality* and *Operator Code.*



Mask	1B	Latitude, Longitude	1b	0b	**1**
		Altitude	1b	1b	**1**
		Angle	1b	2b	**1**
		Speed	1b	3b	**1**
		Satellites	1b	4b	**1**
		Cell ID	1b	5b	**0**
		Signal Quality	1b	6b	**1**
		Operator Code	1b	7b	**1**

GPS Element Mask

42 5a e4 d3 – *Latitude* (Coordinate parsing by IEEE754 protocol)

41 ca b3 fb – *Longtitude* (Coordinate parsing by IEEE754 protocol)

00 9c – *Altitude:* ***156 m*** [DEC]

9f – *Angle:* ***223 deg*** (**9f** [HEX] = 159 * 360 / 256 = 223,59 deg)

00 – *Speed:* ***0 km/h*** [DEC]

04 – *Satellites:* ***4*** [DEC]

17 – *Signal Quality:* ***23*** [DEC]

00 00 60 1a – *Operator Code:* ***24602*** [DEC]

02 – *IO Element 1B quantity:* ***2*** [DEC]

01 – *IO Element ID:* ***1*** [DEC] **–** ***Battery Level***

5d – *Value:* ***93%*** [DEC]

02 – *IO Element ID:* ***2*** [DEC]- ***USB Connection***

01 – *Value:* ***1*** [DEC] – ***USB Connected***

01 – *IO Element 2B quantity:* ***1*** [DEC]

14 – *IO Element ID:* ***20*** [DEC] – ***HDOP value***

00 5f – *Value:* ***9,5*** (**5f** [HEX] = 95 / 10 = 9,5)

Left code (**44d4fb1507df425ae4cc41cab3d20091ae0104160000601a02015d020101140062**) is a second data record in sent data packet.
After the last record (in the end of Packet Data) goes *Number of Records* and *CRC:*

020000B68E

02 – *Number of Records:* ***2*** [DEC]

0000B68E - *CRC*

NOTE: IO Element 1B, IO Element 2B and IO Element 4B IDs shown in GH3000 Data IDs Table

11. Sending data over UDP/IP

11.1 UDP channel protocol

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

UDP datagram			
UDP channel packet x N	Packet length	2 bytes	Packet length (excluding this field) in big endian byte order
	Packet Id	2 bytes	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 acknowledgement
1	Data packet NOT requiring acknowledgement
2	Acknowledgement packet

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

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

11.2 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 acknowledgement packet. Since server should respond to the avl data packet, UDP channel acknowledgement 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 acknowledgement)

Module validates *Avl packet id* and *Number of accepted avl elements.* If server response with valid *Avl packet id* is not received within configured time-out, module can retry sending.

Example:

Module sends the data:

UDP channel header	***Avl packet header***	***Avl data array***
Len – 60, Id – 0x0000, Packet type – 01 (without ACK)	Avl packet id – 0x01, Imei – 352848020079311	CodecId – 07, NumberOfData – 2. (Encoded using continuous bit stream)
003C000001	01000F333532383438303230303739333131	0702...(data elements)...02

Server should respond with acknowledgment:

UDP channel header	***Avl packet acknowledgement***
Len – 5, Id – 0x0002, Packet type – 01 (without ACK)	Avl packet id – 0x01, NumberOfAcceptedData – 2
0005000201	0102

11.3 Example

Received Data:

```
003c00000102000f33353238343830323030373933313107044d54602011b425ae4ce4
1cab2de008c010444d5463e011b425ae4d741cab3be00970004444d54602011b425ae4
e941cab2bb0081000344d545d3011b425ae4ef41cab2ca007f000304

00050002010204
```

003c000001 – UDP channel header

00 3c – Len: 60

00 00 – ID

01 – Packet Type

02000f333532383438303230303739333131 – Avl packet header

02 – Avl packet ID

000f 333532383438303230303739333131 – IMEI 352848020079311

07 – Codec ID

04 – Number of records

44 d5 46 02 – Timestamp + Priority

01 – Global Mask (**01 –** Only GPS Elements)

1b – GPS Element Mask **(1b** – Latitude, Longitude, Altitude, Speed and Satellites**)**

42 5a e4 ce – Latitude

41 ca b2 de – Longitude

00 8c – Altitude

01 – Speed

04 – Satellites

At the end of the received data **04** means received number of records.

After received data goes server response:

00050002010204

0005000201 – UDP channel header

00 05 – Len: 5

00 02 – ID

01 – Packet type

0204 - Avl packet acknowledgement

02 – Avl packet ID

04 – Number of accepted data