GTK LW+ 4G PROTOCOL

GPS Tracker

Communication Protocol

V1.2.0

TABLE OF CONTENTS

REVISION HISTORY

Version	Date	Description	Author	Reviewer
V1.0.0	04/12/2025	Initial Release	Gustavo Rezende	Bernardo Polese
V1.1.0	03/02/2026	• Add packets 0x9D and 0x9E • Correction of the PID of packets 0x9B and 0x9C • Correction of Size field of the ACK Packet for 2 Bytes	Gustavo Rezende	Lucas Oliveira
V1.2.0	19/03/2026	• Add field Bluetooth Beacon Battery Level and Bluetooth SOS Battery Level on 0x16 Alarm Packt • Add Alarm type: 0x1B: Bluetooth Beacon low battery alarm and 0x1B: Bluetooth SOS low battery alarm	Gustavo Rezende	Lucas Oliveira

1. COMMUNICATION PROTOCOL

This document defines the instructions for using the GPS vehicle tracker platform. The reference interface protocol outlined here only applies to data transfer between the platform and the server.

2. TERM / MEANINGS

Terms	Explanation
A-GPS	Assisted GPS
ACC	Accessory (Ignition of vehicle)
APN	Access Point Name
CAN	Controller Area Network
CID	Cell Tower Identifier
CRC	Cyclic Redundancy Check
DIN	Digital Input
GMT	Greenwich Mean Time
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communication
ICCID	Integrated Circuit Card Identifier
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
LAC	Location Area Code
LBS	Location Based Services
MCC	Mobile Country Code
MNC	Mobile Network Code
MNC	Mobile Network Code
NITZ	Network Identity and Time Zone
NTP	Network Time Protocol

RNC	Radio Network Controller
SMS	Short Message Service
OBD	On-Board Diagnostics
OTA	Over The Air
TCP	Transport Control Protocol
TFTP	Trivial File Transfer Protocol
UDP	User Datagram Protocol
UTC	Universal Coordinated Time
VIN	Vehicle Identification Number

3. DESCRIPTION

After startup, the device automatically turns “on” and registers with the LTE. After that, it will attempt to create an IP network connection. If such a connection is unavailable, it will still allow connection through SMS or the USB port. Configuration parameters are stored to flash memory and are automatically applied on device.

The commands can be executed on any available connection as these connections are not exclusive. Commands and responses have identical syntax.

Device has robust lockup protection provided by use of a dedicated hardware watchdog that cycles power and resets the system if a lockup is detected.

The device has more features as follows:

- Multiple location services (GNSS, LBS).
- Supporting A-GPS.
- Low power consumption.
- Using 3-axis accelerometer.
- Automatic time sync (GPS, NITZ, AGPS, NTP).

4. AVAILABLE INTERFACES

4.1 Interface

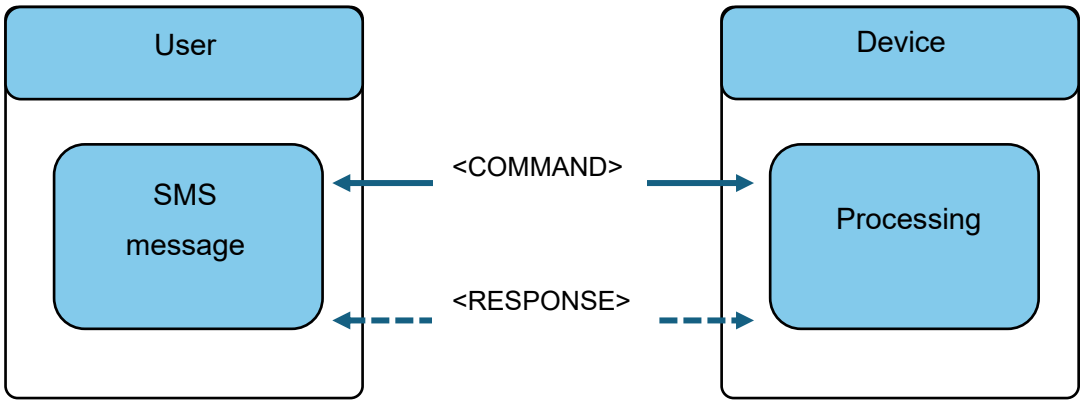
The device has three external interfaces:

- Message over GSM (SMS)
- Data connection (GPRS)

All of them can be used to communicate with devices.

4.2 SMS

The commands described in this document are available in text format. They can be sent in their raw format from the user to the device. After that, the results will be returned to user also in their raw format.
The workflow is as follows:



4.3 IP (network connection)

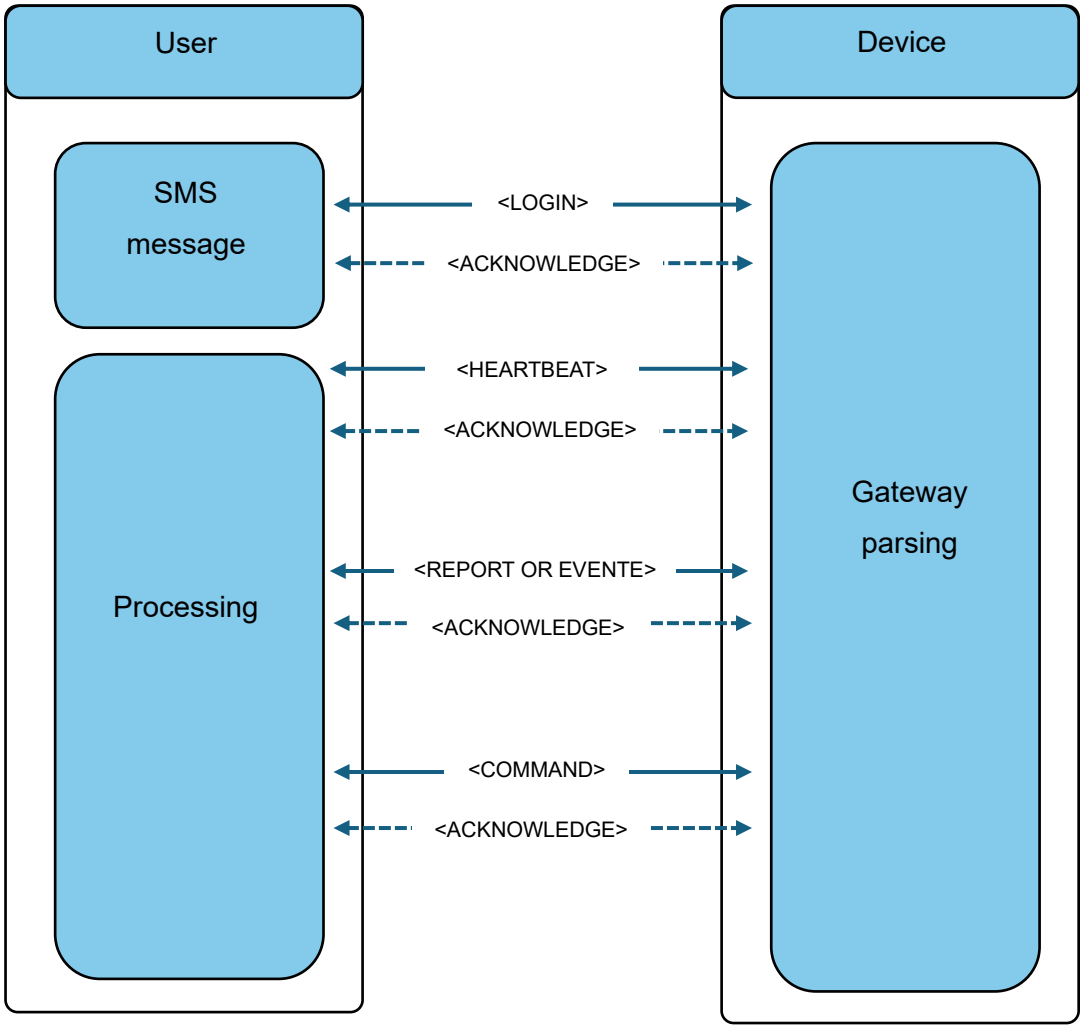
The protocol, described in this document, defines the data packages between server and device. It is based on IP connection over the device's modem. After an IP connection is established between server and device, the data packages are allowed to be exchanged between them.

There are two categories of data packages. One is that device reports something to server and server acknowledges it. Another is that server requests

something and device responses it. The format of data packages will be shown in the following chapters.

The commands described in this document have the specific package and are sent from server to device. The result is another specific package returned from the device to the server.

The whole workflow is as follows:



When device is powered up or session is disconnected (device reconnect), it attempts to establish a new connection to server. After being connected, device sends a login package to server at first.

All other packages will not be sent until device receives the acknowledge of login package from server.

After the connection is established successfully, the heartbeat package will be sent periodically in a specific interval. The reason is to keep the connection and to detect the availability of connection.

If all acknowledges of three consecutive heartbeat packages are not received, current session will be disconnected, and a new one will be established.

While the connection is valid, device will send packages according to different events. The primary packet is the location report which describes the location of device, and the other packets are alarms configured.

5. DATA TYPES

In this chapter, we discuss common data types used in protocol.

5.1 Integer

Integer is the most important data type in protocol. Most data are represented in an integer, e.g. the data length, the package type, the satellites number, etc.

The positive integers are represented in their binary value in unsigned format. And the negative integers are represented in the complement system in signed format.

Moreover, all float point numbers are transformed to integers. For example: temperature is written in normal format as “-6.5” Celsius degree. However, it can also be represented in “-65” in “0.1 Celsius degree”. In this form, we convert a float point number to an integer.

There are 6 integer types:

- unsigned 8 bits integer, from 0 to 255
- unsigned 16 bits integer, from 0 to 65535
- unsigned 32 bits integer, from 0 to 4294967295
- signed 8 bits integer, from -128 to 127
- signed 16 bits integer, from -32768 to 32767
- signed 32 bits integer, from -2147483648 to 2147483647

The order of an integer is Big Endian on all the packages.

5.2 String

All strings are coded in UTF-8.

Most strings in package have a limited length, e.g. password, name, etc. We use a fixed size space to contain them. If the size of space is more than the length of string, rest bytes will be zero. The length of string is never more than

the size of space.

Only a few strings have a variable length. When they appear in package, their length must be able to be calculated based on other data field. The byte order of a string is always from the first byte to the last byte.

5.3 Time

All time are coded as one unsigned 32 bits integer. All of them are represented in UTC (GMT) time. In another word, they are the time in time zone 0. Device uses Unix Timestamp reference.

The value of integer is the seconds from 1970/01/01 00:00:00.

For example, a decimal 1480209825 (hexadecimal 0x583A35A1) is 2016-11-27 01:23:45.

5.4 Status Device

Occupies 2 bytes, with each byte representing specific information from the device. Considering 2 bytes as a 16-bit value, the least significant bit (LSB) is bit 0, and the most significant bit (MSB) is bit 15 (big-endian format). During data transmission, the higher byte is transmitted first, followed by the lower byte.

Each bit within this structure represents a specific function or status, as described below:

Byte	Bit	Description
	15	0：Not fortified 1：Fortified
	14	0：ACC Off 1：ACC On
	13	0：Not charged 1：Charged
		000：Normal 001：Vibration alarm

BYTE 1 (MSB)	12 – 10	010：Cut-off alarm 011：Low-power alarm 100：SOS alarm
	9	0：GPS not Fixed 1：GPS Fixed
	8	0：DOUT 2 Off 1：DOUT 2 On
BYTE 2 (LSB)	7	0：DOUT 1 Off 1：DOUT 1 On
	6	0: DIN 2 OFF 1: DIN 2 ON
	5	0: DIN 1 OFF 1: DIN 1 ON
	4	Reserved for future used
	3	Reserved for future used
	2	Reserved for future used
	1	Reserved for future used
	0	Reserved for future used

Note:

- ***DOUT is the abbreviation of digital output port.***
- ***DIN is the abbreviation of digital input port.***

5.5 Device Position

Position is a compound data type. It includes all data related to location, e.g. latitude, longitude, altitude, GSM data, Input and Output, etc. In order to get minimum length, it is defined in variable size. It includes only valid data and eliminates invalid one. A mask is used to indicate which data is valid. The structure of a position is described as below:

Name	Bytes	Description
Time	4	The event time (UTC) when position data is collected
Latitude	4	-90.0 ~ 90.0 degree: Signed 32 bits integer from -162000000 to 162000000 (in 1/500”)
Longitude	4	-180.0 ~ 180.0 degree: Signed 32 bits integer from -324000000 to 324000000 (in 1/500”)
Altitude	2	Signed 16 bits integer from -32768 to 32767 (in meters)
Speed	1	Unsigned 16 bits integer (in km/h)
Course & Status	2	Course & Status
Satellites	1	The number of satellites
MCC	2	Mobile Country Code - Unsigned 16 bits integer
MNC	2	Mobile Network Code — Unsigned 16 bits integer
LAC	2	Location Area Code - Unsigned 16 bits integer
Cell ID	4	Cell ID with RNC — Unsigned 32 bits integer
GSM Signal	1	GSM Signal Degree

5.5.1 MCC

The country code to which a mobile user belongs, i.e., Mobile Country Code (MCC).

Example: Chinese MCC is 460 in decimal, or 0x01 0xCC in Hex (that is, a decimal value of 460 converting into a hexadecimal value, and 0 is added at the left side because the converted hexadecimal value is less than four digits). Herein the range is 0x0000 to 0x03E7.

5.5.2 MNC

Mobile Network Code (MNC)
Example: Chinese MNC is 0x0000.

5.5.3 LAC

Location Area Code (LAC) included in LAI consists of two bytes and is encoded in hexadecimal. The available range is 0x0001 to 0xFFFE, and the code group 0x0000 and 0xFFFF cannot be used.

5.5.4 GSM Signal Degree.

The GSM information range: 0 ~ 100; The stronger the number, the greater the GSM signal

- 0: no signal
- 100: signal is full

5.6 Course & Status

It occupies 2 bytes and is used to define the GPS operating direction, with values ranging from 0° to 360°, measured clockwise from true north (0°). Considering 2 bytes as a 16-bit value, the least significant bit (LSB) is bit 0 and the most significant bit (MSB) is bit 15 (big-endian format). During data transmission, the most significant byte is transmitted first, followed by the most significant byte. Each bit within this structure represents a specific function or status, as described below.

Byte	Bit	Description
BYTE 1 (MSB)	15	0
	14	0
	13	GPS real-time/differential positioning
	12	GPS having been positioning or not
	11	East Longitude, West Longitude
	10	South Latitude, North Latitude
	9	Course
	8	

BYTE 2 (LSB)	7	
	6	
	5	
	4	
	3	
	2	
	1	
	0	

Note:

- ***The status information in the data packet is the status corresponding to the time bit recorded in the data packet.***

6. PARSING DATA

For parsing the latitude and longitude we will use the Location package as sample. The method used for calculating the latitude and longitude is described below:

6.1 Latitude

The latitude value is represented using **four bytes**, encoded as a **signed INT32 in Big Endian format (ABCD)**. The decoded numerical range corresponds directly to geographic latitude in degrees.

Conversion Method:

1. Read the four-byte field as **INT32 – Big Endian (ABCD)**.
2. Divide the resulting integer by **1,800,000**.
3. The final value represents the latitude in **degrees with decimal precision** (deg. dddddd).

Example:

- Raw bytes: **FD 09 F9 B3**
- Convert to INT32 Big Endian → **–49,677,901**
- Divide by 1,800,000 → **–27.598834**
- Resulting latitude (deg. dddddd) → **–27.598834**

6.2 Longitude

The longitude value is represented using **four bytes**, encoded as a **signed INT32 in Big Endian format (ABCD)**. The decoded numerical range corresponds directly to geographic longitude in degrees.

Conversion Method:

1. Read the four-byte field as **INT32 – Big Endian (ABCD)**.
2. Divide the resulting integer by **1,800,000**.

3. The final value represents the longitude in **degrees with decimal precision** (deg. dddddd).

Example:

- Raw bytes: **FA C9 00 A2**
- Convert to INT32 (Big Endian) → **–87,490,398**
- Divide by 1,800,000 → **–48.605777**
- Resulting longitude (deg. dddddd) → **–48.605777**

6.3 Battery Voltage Level

Battery voltage values are represented with percentual precision.

For example：Battery voltage Level 100%，as：0x64
For exemple：Battery voltage Level 30%，as：0x1E

6.4 Battery Voltage

Battery voltage values are represented with 0.01V precision.

For example：Battery voltage 4.1V，as：0x01 0x9A
For exemple：Battery voltage 3.8V，as：0x01 0x7C

6.5 External Voltage

External voltage values are represented with 0.01V precision.

For example：External Voltage 30.00V，As：0x0B 0xB8
For example：External Voltage 11.85V，As：0x04 0xA1

6.6 ID Driver

ID Driver: 10 bytes. The first byte indicates the status of authorization:

Byte	ID
0	00: Unauthorized 01: Authorized card
1	Card number
2	
3	
4	
5	
6	
7	
8	
9	

6.6.1 Mobile Bluetooth or Beacon

For mobile Bluetooth or Beacon-type Bluetooth devices, the first 3 bytes are padded with 0s, and the last 6 bytes are the Bluetooth MAC address.

Bytes	ID
0	0x00
1	
2	
3	Mobile Bluetooth or Beacon
4	
5	
6	
7	
8	

Example:

MAC address is **11:22:33:44:55:66**

Card number: **0x00 0x00 0x00 0x11 0x22 0x33 0x44 0x55 0x66**.

6.6.2 Ibutton

For Ibutton devices, the first byte is padded with 0s, and the last 8 bytes are the Ibutton card number.

Bytes	ID
0	0x00
1	Ibutton number
2	
3	
4	
5	
6	
7	
8	

Example:

Ibutton card number is **1122334455667788**

Card number: **0x00 0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88.**

After a valid authorized card number is detected, an alarm "**Insert Driver ID**" is reported.

If no authorized card number is detected after the engine is turned off, an alarm "Remove Driver ID" is reported.

7. NETWORK PROTOCOL

7.1 TCP/IP

TCP is a transmission protocol based on IP network. It establishes a stream-like tube between device and server, and provides reliable, ordered, and error-checked delivery of a stream of octets. Any data enters the tube, and they will arrive at their destination with correct content in correct order. As a result, the packages, described in the following chapters, are injected into TCP session without any extra encapsulation.

Its disadvantage is also its stream-like feature. The delivered data may be split and recombined during transmitting (MTU of network). So, in order to recover the original package, the destination must detect the package head, specifically the length of package, then get the package body. As a result, the destination must save all partial packages. Only when a whole package is recognized, the destination can process it.

A TCP tube is as below:

…	Package 1	Package 2	…	Package N	…

8. PACKAGES

The communication is transferred asynchronously in bytes. In general, the structure of a package is described as below:

Data package length: (12+N）Byte

Name	Byte	Description
Star bit	2	0x78 0x78
PID	1	Package identifier
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number — Unsigned 16 bits integer
Device Model	1	Identifies the tracker model
Information content	N	Package sequence number — Unsigned 16 bits integer
Error checking	2	CRC
End Bit	2	0xDA 0x0A

8.1 Start Bit

Fixed value，hexadecimal number **0x78 0x78**

8.2 PID

Refer to different “information content” and correspond to the protocol number，**as** table:

Type	Value	Need Respond
Login Information	0x01	Yes
Location Data	0x12	Yes
Status Information (The heartbeat packets)	0x13	Yes
Strings Information	0x15	No
Alarm data	0x16	Yes
ICCID Information	0x94	No
Server send command to device	0x80	Yes
External module transparent transmission protocol - sent by server	0x9B	Yes
External module transparent transmission protocol - sent by modulo	0x9C	Yes
BLE Data Downlink – Sent by server	0x9D	Yes
BLE Data Uplink – Sent by device	0x9E	Yes

8.3 Size

Size = Sequence + Device Model + Information content + Error checking + End Bit，（7+N）Byte in all，as the information Content is uncertain data.

8.4 Sequence

After turning on the device, it will send the first item of GPRS data (including heartbeat package and GPS/LBS data package); the serial number of this item is “1”. After that, the serial number will be added on by 1 automatically at every sending process (including heartbeat package and GPS/LBS data package).

8.5 Device Model

Used to identify the tracker model.

Model	ID
GTK LW+ 4G	0x01
GTK LWs 4G	0x02
Reserved	0x04
Reserved	0x05

8.6 Information content

The specific contents are determined by the protocol numbers corresponding to different applications.

8.7 Error Checking

Device or server can judge the accuracy of data received with an identifying code. Sometimes, because of the electronic noise or other interference, data will be changed a little in the transit process. In this case, identifying code can make sure the core or associated core does nothing with such wrong data, which will strengthen the security and efficiency of system. This identifying code adopts CRC-ITU identifying method. The CRC-ITU value is from "PID" to "Information Content" in the protocol (including "PID" and "Information Content ").

If the receiver receives CRC wrong calculating information, then ignore it and discard this data package.

8.8 End bit

Fixed value by hexadecimal **0x0D 0x0A**

9. DATA PACKET SENT FROM DEVICE TO SERVER

9.1 Login information packet (0x01)

The login information packet is used to be sent to the server with the device ID to confirm the established connection is normal or not. Content for 8 BYTE, A total of 18 bytes.

9.1.1 Device Sending Login information Packet to Server

Name	Bytes	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier – 0x01
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Device ID	8	Device IMEI 15 digits Example: if the IMEI is 123456789012345 The IMEI Contents are: 0x01 0x23 0x45 0x67 0x89 0x01 0x23 0x45
Device Model	1	Identifies the tracker model
Firmware Version	2	Firmware version — Unsigned 16 bits integer (e.g. 0x0205: V2.05)
Protocol Version	2	Protocol version — Unsigned 16 bits integer (e.g. 0x0211: V2.1.1)
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

9.1.2. Server Responds the Login information Packet.

The response packet from the server to the device: the protocol number in the response packet is identical to the protocol number in the data packet sent by the device.

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier – 0x01
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Error Check	2	error checksum
End Bit	2	0x0D 0x0A

9.2. Location data Packet (0x12)

Location data package is the most important package. It transfers the position and other information of device to server. Its structure is:

9.2.1 Device Sending Location Data Packet to Server

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier - 0x12
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Device Model	1	Identifies the tracker model
Position	N	Device position
Device Information	2	Status Device
Battery Voltage Level	1	Battery Voltage Level
Battery Voltage	2	Battery Voltage
External Vol	2	External Voltage
Mileage	4	Device mileage (in m) - Unsigned 32 bits integer. Range 0 ~ 4.294.967.295 m
Hour meter	4	Accumulated time (in seconds) that device identified ignition turned on - Unsigned 32 bits integer. Range:0 ~ 4.294.967.295 s
ID Driver	10	Driver identification information via Bluetooth or Beacon.
Reserved	18	Reserved for future used
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

9.2.2. Server Responds to the Location Data Packet

The response packet from the server to the device: the protocol number in the response packet is identical to the protocol number in the data packet sent by the device.

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier – 0x12
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

9.3. Status information Packet (Heartbeat Packet) (0x13)

Status information packet is a data packet to maintain the connection between the device and the server.

9.3.1 Device Sending Status Information Packet to Server

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier - 0x13
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

9.3.2. Server Responds the Status Information Packet.

The response packet from the server to the device: the protocol number in the response packet is identical to the protocol number in the data packet sent by the device.

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier – 0x13
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

9.4. Alarm Packet (0x16)

An Alarm Packet will be sent to the server when a specific event occurs. Its structure is:

9.4.1 Device Sending Alarm Packet to Server

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier - 0x16
Size	2	Package size from next byte to end Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Device Model	1	Identifies the tracker model
Position	N	Device position
Alarm Type	1	Alarm type Unsigned 8 bits integer
Device Information	2	Status Device
Battery Voltage Level	1	Battery Voltage Level
Battery Voltage	2	Battery Voltage
External Vol	2	External Voltage
Mileage	4	Device mileage (in m) - Unsigned 32 bits integer. Range 0 ~ 4.294.967.295 m
Hour meter	4	Accumulated time (in seconds) that device identified ignition turned on - Unsigned 32 bits integer. Range:0 ~ 4.294.967.295 s
ID Driver	10	Driver identification information via Bluetooth or Beacon.
Bluetooth Beacon Battery Level	2	Battery Level
Bluetooth SOS Battery Level	2	Battery Level

Reserved	14	Reserved for future used
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

9.4.1.1 Alarm Type

The alarm type is listed as below:

Alarm	0x00: Normal
	0x01: SOS
	0x02: Power Cut Alarm
	0x03: Shock Alarm
	0x04: Fence In Alarm
	0x05: Fence Out Alarm
	0x06: Speed Alarm
	0x07: Digital Input 2
	0x08: Reserved for future used
	0x09: Move Alarm
	0x0A: External Battery Connected
	0x0B: Sim Card removed Alarm
	0X0C: Low Battery Alarm
	0X0D: Digital Output 1
	0x0E: Reserved for future used
	0x0F: Disassemble Alarm
	0x10: ACC On Alarm
	0x11: ACC Off Alarm
	0x12: Change ignition source
	0x13: Rapid acceleration alarm
	0x14: Rapid deceleration alarm
	0x15: Sharp turn alarm
	0x16: Insert Driver ID
	0x17: Remove Driver ID

	0x18: Unauthorized Driver ID
	0x19: The card reading fails because the card is removed
	0x1A: The card reading failed because of a data verification error
	0x1B: Bluetooth Beacon low battery alarm
	0x1C: Bluetooth SOS low battery alarm

9.4.2. Server Responds to the Alarm Packet.

The response packet from the server to the device: the protocol number in the response packet is identical to the protocol number in the data packet sent by the device.

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier - 0x16
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

9.5. ICCID Packet (0x094)

9.5.1 Device Sending ICCID information Packet to Server

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier – 0x94
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Device Model	1	Identifies the tracker model
FLAG	1	Fixed value: 0x0A
Device ID	8	Device IMEI 15 digits Example: if the IMEI is 123456789012345 The IMEI Contents are: 0x01 0x23 0x45 0x67 0x89 0x01 0x23 0x45
IMSI	8	Sim Card IMSI Number, 15 digits Example: if the IMSI is 123456789012345 The IMSI Contents are: 0x01 0x23 0x45 0x67 0x89 0x01 0x23 0x45
ICCID	10	Sim Card ICCID Number, 20 digits Example: if the ICCID is 01234567890123456789 The ICCID Contents is: 0x01 0x23 0x45 0x67 0x89 0x01 0x23 0x45 0x67 0x89
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

9.5.2. Server Responds Data Packet

Server does not need to Respond.

9.6 External module transparent transmission protocol (sent by server) (0x9B)

The server sends a data packet to the module. Its structure is:

9.6.1 Server Sending Transparent Transmission

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier - 0x9B
Size	2	Package size from next byte to end Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Device Model	1	Identifies the tracker model
Server Flag Bit	4	It is reserved for the identification of the server.
Transparent Data	N	Transparent Data
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

9.6.2. Device Responds to the Transparent Transmission Packet.

The response packet from the device to the server: the protocol number in the response packet is identical to the protocol number in the data packet sent by the device.

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier – 0x9B
Size	2	Package size from next byte to end — Unsigned 16 bits integer

Sequence	2	Package sequence number - Unsigned 16 bits integer
Server Flag Bit	4	It is reserved for the identification of the server.
Transparent Data	N	Transparent Data
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

9.7 External module transparent transmission protocol (sent by modulo) (0x9C)

The module sends a data packet to the server. Its structure is:

9.7.1 Device Sending Transparent transmission to Server

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier - 0x9C
Size	2	Package size from next byte to end Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Device Model	1	Identifies the tracker model
Position	N	Device position
Data Type	1	Used to identify the module type. Different module types have different meanings of transparent data transmission.
Transparent Data	N	Transparent Data
Device Information	2	Status Device
Battery Voltage	2	Battery Voltage
External Vol	2	External Voltage
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

9.7.1.1 Data Type

Used to identify the module type. Different module types have different meanings of transparent data transmission.

Type	Description
0x00	Transparent Data
0x01	Reserved for future used
0x10	Reserved for future used
0x11	Reserved for future used

9.7.2. Server Responds to the Transparent Transmission Packet.

The response packet from the server to the device: the protocol number in the response packet is identical to the protocol number in the data packet sent by the device.

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier – 0x9C
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Transparent Data	N	Transparent Data
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

9.8 BLE Data Downlink Protocol (sent by server) (0x9D)

This protocol defines structured bidirectional data communication between servers and external BLE accessories.

9.8.1 Server Sending Transparent Transmission

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier - 0x9D
Size	2	Package size from next byte to end Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Device Model	1	Identifies the tracker model
Server Flag Bit	4	It is reserved for the identification of the server.
BLE Protocol Version	1	BLE protocol version (Current: 0x01)
BLE Device Type	1	Identify accessory type
BLE Device ID	6	BLE MAC Address
Command ID	1	Identifies command type
Payload Length	2	Length of command payload
Payload	N	Command data
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

9.8.1.1 Command ID

Used to identify the module type. Different module types have different meanings of transparent data transmission.

Type	Description
0x00	Write Configuration
0x01	Read Configuration
0x02	Activate Output
0x03	Reset Device
0x04	OTA Chunk
0x05	OTA Commit
0x10	Custom Command

9.8.2. Server Responds the Transparent Transmission Packet.

The response packet from the server to the device: the protocol number in the response packet is identical to the protocol number in the data packet sent by the device.

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier – 0x9D
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Device Model	1	Identifies the tracker model
Length of Command	1	Server Flag Bit + Length of Command Content Example: measured in bytes, 0x0A means the content of command occupied ten bytes.
Server Flag Bit	4	It is reserved for the identification of the server.
Command Content	M	It is represented in ASC II of string, and the command content is compatible with text message command.
Reserved	2	Reserved. Fixed value: 0x0000
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

9.9 BLE Data Uplink Protocol (sent by modulo) (0x9E)

This protocol defines structured bidirectional data communication between external BLE accessories and server.

9.9.1 Device Sending Transparent transmission to Server

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier - 0x9E
Size	2	Package size from next byte to end Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Device Model	1	Identifies the tracker model
BLE Protocol Version	1	BLE protocol version (Current: 0x01)
BLE Device Type	1	Identify accessory type
BLE Device ID	6	BLE MAC Address (Big Endian)
BLE RSSI	1	Signal strength — Signed 8 bits integer (-127 to +20 dBm)
Timestamp	4	Unix timestamp (UTC) — Unsigned 32 bits integer
Payload Length	2	Length of BLE payload — Unsigned 16 bits integer
Payload	N	Structured or free data from accessory
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

9.9.1.1 BLE Device Type Definition

Used to identify the module type. Different module types have different meanings of transparent data transmission.

Type	Description
0x00	Temperature Sensor
0x01	Humidity Sensor
0x02	Pressure Sensor
0x03	Door Sensor
0x04	Beacon Identifier
0x05	Relay Actuator
0x06	Fuel Level Sensor
0x07	OBD-II BLE Adapter (ELM327 Compatible)
0x08	Pressure Sensor
0x09	Tire Pressure Monitoring (TPMS)
0x10	Custom Vendor Device
0xFF	Reserved for future used

9.9.1.2 BLE Device Type Definition

The payload format depends on BLE Device Type. All numeric values must follow Big Endian format.

Name	Byte	Description
Temperature Sensor	2*N	Signed 16 bits integer (0.1°C resolution). N represents the number of temperature group.
Humidity Sensor	2*N	Unsigned 16 bits integer (0.1% resolution). N represents the number of humidity channels.

Pressure Sensor	4	UINT32 Pressure (Pa)
Tire Pressure Monitoring (TPMS)	N	Pressure + Temperature
Beacon Identifier	1+N*35 （N <= 7）	The first byte is the number of Bluetooth beacons, followed by N Bluetooth beacon data; Bluetooth beacon data format: • 16 bytes: UUID • 6 bytes: MAC address • 8 bytes: Custom name (less than 8 bytes, filled with 0x00) • 1 byte: RSSI@1M (RSSI value calibrated at 1 meter distance) • 1 byte: Actual RSSI value • 2 bytes: Battery voltage, in mV. (0xFFFF indicates not supported) • 1 byte: Battery percentage. (0xFF means not supported)
Door Sensor	1	1 byte Status (0=Closed / 1=Open)
Fuel Level Sensor	2	UINT16 Level (0.1%)
Relay Actuator	1	1 byte Output State
OBD-II BLE Adapter (ELM327 Compatible)	N	OBD structured data
Custom Vendor Device	N	Vendor-defined

9.9.2. Server Responds the Transparent Transmission Packet.

The response packet from the server to the device: the protocol number in the response packet is identical to the protocol number in the data packet sent by the device.

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier – 0x9E
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Transparent Data	N	Transparent Data
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

10. DATA PACKET SENT FROM SERVER TO DEVICE

10.1. Packet Sent by Server(0x80)

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier – 0x80
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Device Model	1	Identifies the tracker model
Length of Command	1	Server Flag Bit + Length of Command Content Example: measured in bytes, 0x0A means the content of command occupied ten bytes.
Server Flag Bit	4	It is reserved for the identification of the server.
Command Content	M	It is represented in ASC II of string, and the command content is compatible with text message command.
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

10.2. Packet Replied by Device (0x15)

Name	Byte	Description
Start Bit	2	0x78 0x78
PID	1	Package identifier – 0x15
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number - Unsigned 16 bits integer
Device Model	1	Identifies the tracker model

Length of Command	1	Server Flag Bit + Length of Command Content Example: measured in bytes, 0x0A means the content of command occupied ten bytes.
Server Flag Bit	4	It is reserved for the identification of the server.
Command Content	M	It is represented in ASC II of string, and the command content is compatible with text message command.
Reserved	2	Reserved. Fixed value: 0x0000
Error Check	2	Error checksum
End Bit	2	0x0D 0x0A

11. CRC-ITU LOOKUP TABLE ALGORITHM CODE

```
static const U16 crctab16[] =
{
    0X0000, 0X1189, 0X2312, 0X329B, 0X4624, 0X57AD, 0X6536, 0X74BF,
    0X8C48, 0X9DC1, 0XAF5A, 0XBED3, 0XCA6C, 0XDBE5, 0XE97E,
    0XF8F7, 0X1081, 0X0108, 0X3393, 0X221A, 0X56A5, 0X472C, 0X75B7,
    0X643E, 0X9CC9, 0X8D40, 0XBFDB, 0XAE52, 0XDAED, 0XCB64,
    0XF9FF, 0XE876, 0X2102, 0X308B, 0X0210, 0X1399, 0X6726, 0X76AF,
    0X4434, 0X55BD, 0XAD4A, 0XBCC3, 0X8E58, 0X9FD1, 0XEB6E,
    0XFAE7, 0XC87C, 0XD9F5, 0X3183, 0X200A, 0X1291, 0X0318, 0X77A7,
    0X662E, 0X54B5, 0X453C, 0XBDCB, 0XAC42, 0X9ED9, 0X8F50,
    0XFBEF, 0XEA66, 0XD8FD, 0XC974, 0X4204, 0X538D, 0X6116,
    0X709F, 0X0420, 0X15A9, 0X2732, 0X36BB, 0XCE4C, 0XDFC5,
    0XED5E, 0XFCD7, 0X8868, 0X99E1, 0XAB7A, 0XBAF3, 0X5285,
    0X430C, 0X7197, 0X601E, 0X14A1, 0X0528, 0X37B3, 0X263A,
    0XDECD, 0XCF44, 0XFDDF, 0XEC56, 0X98E9, 0X8960, 0XBBFB,
    0XAA72, 0X6306, 0X728F, 0X4014, 0X519D, 0X2522, 0X34AB, 0X0630,
    0X17B9, 0XEF4E, 0XFEC7, 0XCC5C, 0XDDD5, 0XA96A, 0XB8E3,
    0X8A78, 0X9BF1, 0X7387, 0X620E, 0X5095, 0X411C, 0X35A3, 0X242A,
    0X16B1, 0X0738, 0XFFCF, 0XEE46, 0XDCDD, 0XCD54, 0XB9EB,
    0XA862, 0X9AF9, 0X8B70, 0X8408, 0X9581, 0XA71A, 0XB693,
    0XC22C, 0XD3A5, 0XE13E, 0XF0B7, 0X0840, 0X19C9, 0X2B52,
    0X3ADB, 0X4E64, 0X5FED, 0X6D76, 0X7CFF, 0X9489, 0X8500,
    0XB79B, 0XA612, 0XD2AD, 0XC324, 0XF1BF, 0XE036, 0X18C1,
    0X0948, 0X3BD3, 0X2A5A, 0X5EE5, 0X4F6C, 0X7DF7, 0X6C7E,
    0XA50A, 0XB483, 0X8618, 0X9791, 0XE32E, 0XF2A7, 0XC03C,
    0XD1B5, 0X2942, 0X38CB, 0X0A50, 0X1BD9, 0X6F66, 0X7EEF,
    0X4C74, 0X5DFD, 0XB58B, 0XA402, 0X9699, 0X8710, 0XF3AF,
    0XE226, 0XD0BD, 0XC134, 0X39C3, 0X284A, 0X1AD1, 0X0B58,
    0X7FE7, 0X6E6E, 0X5CF5, 0X4D7C, 0XC60C, 0XD785, 0XE51E,
    0XF497, 0X8028, 0X91A1, 0XA33A, 0XB2B3, 0X4A44, 0X5BCD,
    0X6956, 0X78DF, 0X0C60, 0X1DE9, 0X2F72, 0X3EFB, 0XD68D,
```

```
        0XC704,  0XF59F,  0XE416,  0X90A9,  0X8120,  0XB3BB,  0XA232,
        0X5AC5,  0X4B4C,  0X79D7,  0X685E,  0X1CE1,  0X0D68,  0X3FF3,
        0X2E7A,  0XE70E,  0XF687,  0XC41C,  0XD595,  0XA12A,  0XB0A3,
        0X8238,  0X93B1,  0X6B46,  0X7ACF,  0X4854,  0X59DD,  0X2D62,
        0X3CEB,  0X0E70,  0X1FF9,  0XF78F,  0XE606,  0XD49D,  0XC514,
        0XB1AB,  0XA022,  0X92B9,  0X8330,  0X7BC7,  0X6A4E,  0X58D5,
        0X495C, 0X3DE3, 0X2C6A, 0X1EF1, 0X0F78,
};

// Calculate the 16-bit CRC of data with predetermined length.
U16 GetCrc16(const U8* pData, int nLength)
{
  U16 fcs = 0xffff;  // initialization
  while(nLength>0)
{
    fcs = (fcs >> 8) ^ crctab16[(fcs ^ *pData) & 0xff].
    nLength--;
    pData++;
  }
  return ~fcs; // negated
}
```