GTK ON 4G PROTOCOL

GPS Tracker

Communication Protocol

V1.0

TABLE OF CONTENTS

REVISION HISTORY

Version	Date	Description	Author	Reviewer
V1.0.0	15/10/2024	Initial Release	Gustavo Rezende	Bernardo Polese

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 / GSM network. 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:

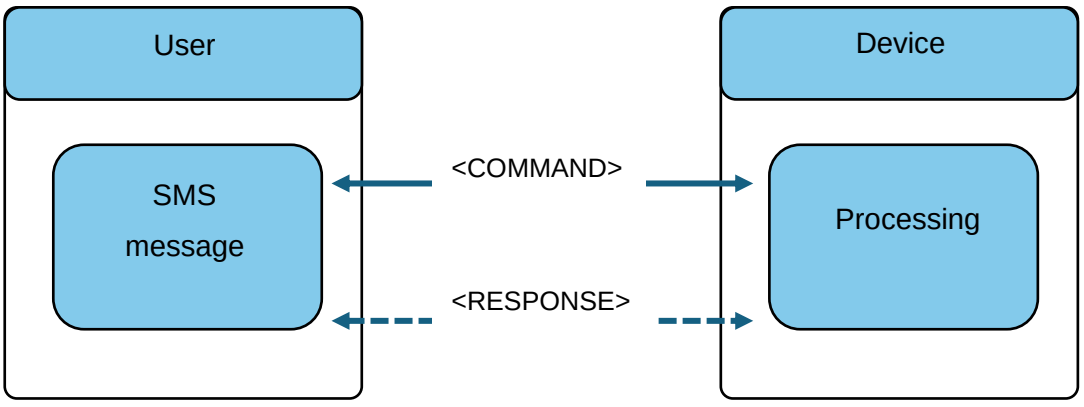
- Message over GSM (SMS)
- Data connection (GPRS)
- USB connection (USB)

All of them can be used to communicate with device.

4.2 SMS

The commands, described in this document, are available in text format. They can be sent in their raw format from 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)

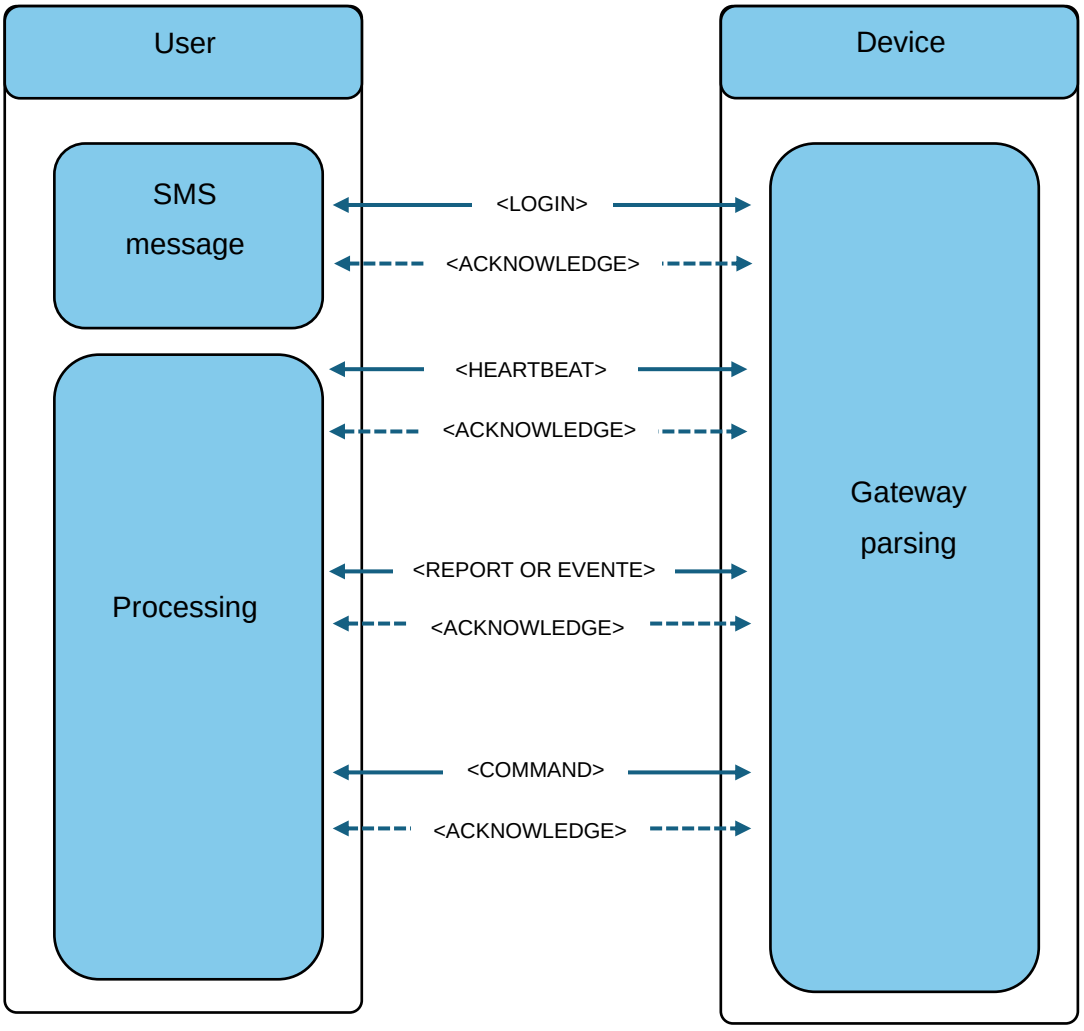
The protocol, described in this document, defines the data packages between server and device. It is based on IP connection over the device's modem. After an IP connection is established between server and device, the data packages are allowed to be exchanged between them.

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

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

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

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 received the acknowledge of login package from server.

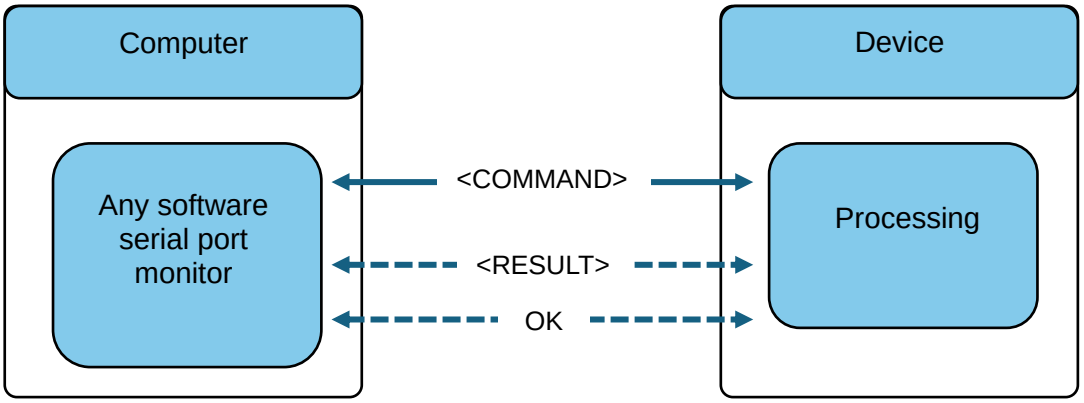
After the connection is established successfully, the heartbeat package will be sent periodically in a specific interval. The reason is to keep the connection and to detect the availability of connection.

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.

4.4 USB communication

The USB port support commands by serial port as workflow below:



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.

There are 3 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

The byte order of an integer can be Big Endian or Little Endian depending on the package.

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 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 times are encoded as a 6 Byte integer. All of them are represented in UTC time (GMT). In other words, they are the time in time zone 0. Below is a table showing how the time is represented:

Format	Length [Byte]	Example
Year	1	0x13
Month	1	0x01
Day	1	0x08
Hour	1	0x09
Minute	1	0x1E
Second	1	0x0A

Example: 2019-01-08 09:30:10

Calculated as follows:

- 19 (Decimal) = 13(Hexadecimal)
- 01 (Decimal) = 01(Hexadecimal)
- 08 (Decimal) = 08(Hexadecimal)
- 09 (Decimal) = 09(Hexadecimal)
- 30 (Decimal) = 1E(Hexadecimal)
- 10 (Decimal) = 0A(Hexadecimal)

Then the value is: **0x13 0x01 0x08 0x09 0x1E 0x0A**

5.4 Status Device

Occupy 1 byte, representing each piece of information on the device. Regarding 1 byte as 8bits, the lowest bit is 7, the highest is 0 (big-endian). In the process of transmitting data, the high one comes first and the low one follows. Each bit represents the detailed meaning as follows:

High bit Low bit

0	1	2	3	4	5	6	7

Note:

- ***Note: Status information refers to the status at a certain time***

For example: 0x4B converts to binary 01001011, which means fortified/ ACC ON/ not charged/ Cut-off alarm / GPS Fixed / DOUT On.

Bit	Description
0	0：Not fortified 1：Fortified
1	0：ACC Off 1：ACC On
2	0：Not charged 1：Charged
3 - 5	000：Normal 001：Vibration alarm 010：Cut-off alarm 011：Low-power alarm 100：SOS alarm
6	0：GPS not Fixed 1：GPS Fixed
7	0：DOUT Off 1：DOUT On

Note:

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

5.5 Course & Status

Two bytes are consumed, defining the running direction of GPS. The value ranges from 0° to 360° measured clockwise from north of 0°.

BYTE1	*Bit7*	Digital Input 2 Status
	Bit6	Digital Input 1 Status
	Bit5	GPS real-time/differential positioning
	Bit4	GPS having been positioning or not
	Bit3	East Longitude, West Longitude

	Bit2	South Latitude, North Latitude
	Bit1	Course
BYTE2	Bit0 Bit7	
	Bit6 Bit5 Bit4 Bit3 Bit2 Bit1 Bit0	

Note:

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

Four bytes are consumed, defining the latitude value of location data. The range of the value is 0 ~ 162000000, indicating a range of 0°~ 90°. The conversion method thereof is as follow: Converting the value of latitude and longitude output by GPS module into a decimal based on minute; multiplying the converted decimal by 30000; and converting the multiplied result into hexadecimal.

Example: 22º32.7658" = (22X60+32.7658) x 30000 = 40582974, then converted into a hexadecimal number 40582974 (Decimal) = 26B3F3E (Hexadecimal)
at last, the value is: 0x02 0x6B 0x3F 0x3E.

6.2 Longitude

Four bytes are consumed, defining the longitude value of location data. The range of the value is 0 ~ 324000000, indicating a range of 0° ~ 180°. The conversion method thereof is as follow: Converting the value of latitude and longitude output by GPS module into a decimal based on minute; multiplying the converted decimal by 30000; and converting the multiplied result into hexadecimal.

6.3 GPS Information

The field is 1 Byte displayed by two hex digits, wherein the first one is for the length of GPS information and the second one for the number of the satellites join in positioning.

Example: if the value is 0xCD, it means the length of GPS information is 12

and the number of the positioning satellites is 13. (C = 12Bit Length ，D = 13 satellites)

6.4 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

6.5 Battery Voltage Level

The range is 0~6 defining the Battery voltage is from low to high.

- 0：Lowest power and power off
- 1：Not enough power to dial a call or send messages.
- 2：Low power and alarm
- 3：Lower power but can work normally
- 3~6：Work in good condition

6.6 Battery Voltage

For example：External Voltage 4.1V，as：0x01 0x9A
For exemple：External Voltage 3.8V，as：0x01 0x7C

6.7 External Voltage

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

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 enter the tube, and they

will arrive 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 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: (10+N）Byte

Name	Byte	Description
Star bit	2	0x78 0x78
Package Length	1	Package Length is the total size of Sequence and Content
Protocol number	1	Package identifier
Information content	N	Package sequence number — Unsigned 16 bits integer
Information serial number	2	Package content
Error checking	2	CRC
End Bit	2	0xDA 0x0A

8.1 Start Bit

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

8.2 Package Length

Length = protocol number + Information content + Information serial number + error checking，（5+N）Byte in all，as the information Content is uncertain length data.

8.3 Protocol number

Refer to different “information content” and correspond to the protocol

number，**as** table:

Type	Value
Login Information	0x01
Location Data	0x17
Status Information (The heartbeat packets)	0x13
Strings Information	0x15
Alarm data	0x16
ICCID Information	0x94
Server send command to device	0x80

8.4 Information content

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

8.5 Information serial number

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).

8.6 Error Checking

Device or server can judge the accuracy of data received with 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 do nothing with such kind of 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 "Package Length' to "Information Serial Number" in the protocol (including "Package Length" and "Information Serial Number ").

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

8.7 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
Length	1	Package Length from protocol number to error checksum - Unsigned 8 bits integer
Protocol Number	1	Package identifier - 0x01
Device ID	8	Device IMEI 15 digits Example: if the IMEI is 123456789012345 The IMEI Contents is: 0x01 0x23 0x45 0x67 0x89 0x01 0x23 0x45
Information Serial Number	2	Package sequence number - Unsigned 16 bits integer
Error Check	2	error checksum
End Bit	2	0x0D 0x0A

9.1.2. Server Responds the Login information Packet.

Name	Byte	Description
Start Bit	2	0x78 0x78
Length	1	Package Length from protocol number to error checksum - Unsigned 8 bits integer
Protocol	1	Package identifier - 0x01

Number		
Information Serial Number	2	Package sequence number - Unsigned 16 bits integer
Error Check	2	error checksum
End Bit	2	0x0D 0x0A

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.

Example of login package and the response:

From device:

78780D01086508004401506900 3A53F20D0A

From server:

78780501003A568C0D0A

9.2. Location data Packet (0x17)

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
Packet Length		1	Package Length from protocol number to error checksum - Unsigned 8 bits integer
Protocol Number		1	Package identifier - 0x17
GPS	Date time	6	Time

information	Quantity of GPS satellites	1	GPS Information
	Latitude	4	0 ~ 90.0 degree: Unsigned 32 bits integer from 0 to 162000000
	Longitude	4	0 ~ 180.0 degree: Unsigned 32 bits integer from 0 to 324000000
	Speed	1	Unsigned 16 bits integer (in km/h)
	Course, Status	2	Course & Status
LBS information	MCC	2	Mobile Country Code - Unsigned 16 bits integer
	MNC	1	Mobile Network Code - Unsigned 8 bits integer
	LAC	2	Location Area Code - Unsigned 16 bits integer
	Cell ID	3	Cell ID with RNC - Unsigned 24 bits integer
Status information	Device Information	1	Status Device
	Battery Voltage Level	1	Battery Voltage Level
	GSM Signal Strength	1	GSM Signal Degree
	Battery Voltage	2	Battery Voltage
	External Vol	2	External Voltage
Mileage		4	Device mileage (in m) - Unsigned 32 bits integer. Range 0 ~1 999999000m
Hourmeter		4	Accumulated time (in seconds) that device identified ignition turned on - Unsigned 32 bits integer.

		Range:0 ~ 1299999999
Information Serial Number	2	Package sequence number - Unsigned 16 bits integer
Error Check	2	Error checksun
End Bit	2	0x0D 0x0A

9.2.1.1 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 ~ 0x03E7.

9.2.1.2 MNC

Mobile Network Code (MNC)
Example: Chinese MNC is 0x00.

9.2.1.3 LAC

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

5.3.2. Server Responds the Location Data Packet

Name	Byte	Description
Start Bit	2	0x78 0x78
Length	1	Package Length from protocol number to error checksum - Unsigned 8 bits integer
Protocol Number	1	Package identifier - 0x17

Information Serial Number	2	Package sequence number - Unsigned 16 bits integer
Error Check	2	error checksum
End Bit	2	0x0D 0x0A

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.

Example of login package and the response:

From device:

78782E17180217121736CD021EF55504B49FE30058DA0000000000000000460637018B05D10000094A000029EA002620540D0A

From server:

787805170026DF2D0D0A

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

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

9.4.1 Device Sending Status information Packet to Server

Name		Byte	Description
Start Bit		2	0x78 0x78
Packet Length		1	Package Length from protocol number to error checksum - Unsigned 8 bits integer
Protocol Number		1	Package identifier - 0x13
Status information	Device Information	1	Status Device
	Battery Voltage Level	1	Battery Voltage Level

	GSM Signal Strength	1	GSM Signal Degree
	External Voltage	1	External voltage (in V) - Unsigned 8 bits integer
	Language	1	Chinese：0x01 English：0x02
Information Serial Number		2	Package sequence number - Unsigned 16 bits integer
Error Check		2	Error checksun
End Bit		2	0x0D 0x0A

9.4.2. Server Responds the Status information Packet.

Name	Byte	Description
Start Bit	2	0x78 0x78
Length	1	Package Length from protocol number to error checksum - Unsigned 8 bits integer
Protocol Number	1	Package identifier - 0x17
Information Serial Number	2	Package sequence number - Unsigned 16 bits integer
Error Check	2	error checksum
End Bit	2	0x0D 0x0A

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.

Example of login package and the response:

From device:

78780A1346062B0F02020233D00D0A

From server:

787805130202E8DA0D0A

9.5. Alarm Packet (0x16)

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

9.5.1 Device Sending Alarm Packet to Server

Name		Byte	Description
Start Bit		2	0x78 0x78
Packet Length		1	Package Length from protocol number to error checksum - Unsigned 8 bits integer
Protocol Number		1	Package identifier — 0x16
GPS information	Date time	6	Time
	Quantity of GPS satellites	1	GPS Information
	Latitude	4	0 ~ 90.0 degree: Unsigned 32 bits integer from 0 to 162000000
	Longitude	4	0 ~ 180.0 degree: Unsigned 32 bits integer from 0 to 324000000
	Speed	1	Unsigned 16 bits integer (in km/h)
	Course, Status	2	Course & Status
LBS information	LBS	1	LBS - Unsigned 8 bits integer
	MCC	2	Mobile Country Code - Unsigned 16 bits integer
	MNC	1	Mobile Network Code - Unsigned 8 bits integer
	LAC	2	Location Area Code - Unsigned 16 bits integer
	Cell ID	3	Cell ID with RNC - Unsigned 24 bits

			integer
Status information	Device Information	1	Status Device
	Battery Voltage Level	1	Battery Voltage Level
	GSM Signal Strength	1	GSM Signal Degree
	Alarm Type	1	Alarm Type
	Language	1	Chinese：0x01 English：0x02
	Battery Voltage	2	Battery Voltage
	External Vol	2	External Voltage
Mileage		4	Device mileage (in m) - Unsigned 32 bits integer. Range 0 ~1 999999000m
Hourmeter		4	Accumulated time (in seconds) that device identified ignition turned on - Unsigned 32 bits integer. Range:0 ~ 1299999999
Information Serial Number		2	Package sequence number - Unsigned 16 bits integer
Error Check		2	Error checksun
End Bit		2	0x0D 0x0A

9.5.1.1 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

	0x06/0x08: Speed Alarm
	0x09: Move Alarm
	0X0C: External Battery Connected
	0x0E: Low Battery Alarm
	0x13: Disassemble Alarm
	0x14: ACC On Alarm
	0x15: ACC Off Alarm
	0x18: Reserved
	0x19: Digital Input 2
	0x0D: Sim Card Removed
	0x0F: Digital Output Activated
	0x26: Rapid acceleration alarm
	0x27: Rapid deceleration alarm
	0x28: Sharp turn alarm
	0x29: Collision alarm

9.4.2. Server Responds the Alarm Packet.

Name	Byte	Description
Start Bit	2	0x78 0x78
Length	1	Package Length from protocol number to error checksum - Unsigned 8 bits integer
Protocol Number	1	Package identifier - 0x16
Information Serial Number	2	Package sequence number - Unsigned 16 bits integer
Error Check	2	error checksum
End Bit	2	0x0D 0x0A

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.

Example of alarm package and the response:

From device:

78783116180217132E10C9021EF4EC04B49FF10019110900000000000000
0005006310202018B0000000009EC00003D40023070B10D0A

From server:

787805160230C3F60D0A

9.9. ICCID Packet (0x094)

9.9.1 Device Sending ICCID information Packet to Server

Name	Byte	Description
Start Bit	2	0x79 0x79
Length	1	Package Length from protocol number to error checksum - Unsigned 8 bits integer
Protocol Number	1	Package identifier - 0x94
FLAG	1	Fixed value: 0x0A
Device ID	8	Device IMEI 15 digits Example: if the IMEI is 123456789012345 The IMEI Contents is: 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 is: 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
Information Serial Number	2	Package sequence number - Unsigned 16 bits integer

Error Check	2	error checksum
End Bit	2	0x0D 0x0A

9.9.2. Server Responds Data Packet

Server does not need to Respond.

10. DATA PACKET SENT FROM SERVER TO DEVICE

10.1. Packet Sent by Server(0x80)

Name	Byte	Description
Start Bit	2	0x78 0x78
Packet Length	1	Package Length from protocol number to error checksum - Unsigned 8 bits integer
Protocol Number	1	Package identifier - 0x80
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 to 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.
Serial Number	2	Package sequence number - Unsigned 16 bits integer
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
Length	1	Package Length from protocol number to error checksum - Unsigned 8 bits integer
Protocol Number	1	Package identifier - 0x15
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 to 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
Information Serial Number	2	Package sequence number - Unsigned 16 bits integer
Error Check	2	error checksum
End Bit	2	0x0D 0x0A

Example of alarm package and the response:

Typed command:

CLEARD#

From device:

787811800B00000000434C4541524423000176D60D0A

From server:

78781B151300000000434C454152443D53756363657373210001000C2F5F0D0A

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
}
```