Por um mundo
mais conectado

Av. Luiz Paulo Franco, 603
Belvedere, Belo Horizonte - MG

iTR120 Protocol Standard

REVISION HISTORY

Version	Date	Description	Author	Reviewer
V1.0.0	15/07/2021	Initial Release	Everton Silva Rosa	Jairo Abreu
V2.0.0	06/08/2021	Add new samples of packages	Everton Silva Rosa	Bernardo Polese
V3.0.0	18/11/2021	Fixed some issues on document	Everton Silva Rosa	Bernardo Polese
V4.0.0	04/02/2022	Fixed some issues on document	Everton Silva Rosa	Bernardo Polese
V5.0.0	02/05/2022	Fixed some issues on document	Everton Silva Rosa	Bernardo Polese
V6.0.0	26/05/2022	Described new features	Everton Silva Rosa	Bernardo Polese
V6.0.1	06/09/2022	The RFU size has been adjusted in the PID: 0x12	Maria Helena Demetrio de Sousa	Bernardo Polese
V7.0.0	08/03/2023	Fixed some issues Add idletime package and configuration	Gustavo Fontes	Bernardo Polese
V7.0.1	14/07/2023	Add fence and simdetect package and configuration	Lucas Roberto de Oliveira	Bernardo Polese
V7.0.2	01/11/2024	Add VIRTUALIGN Fixed some issues on document	Gustavo Rezende	Bernardo Polese

SUMMARY

SCOPE

This document describes the communication protocol between server and device. Also describes the commands/responses supported by device.

The interface between server and device is GPRS and the interface between user and device can be by SMS or desktop software.

TERMS

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

Terms	Explanation
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

DESCRIPTION

After startup, the device automatically turns "on" and registers with the 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:

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

1 AVAILABLE INTERFACES

INTERFACE

The device has three external interfaces:

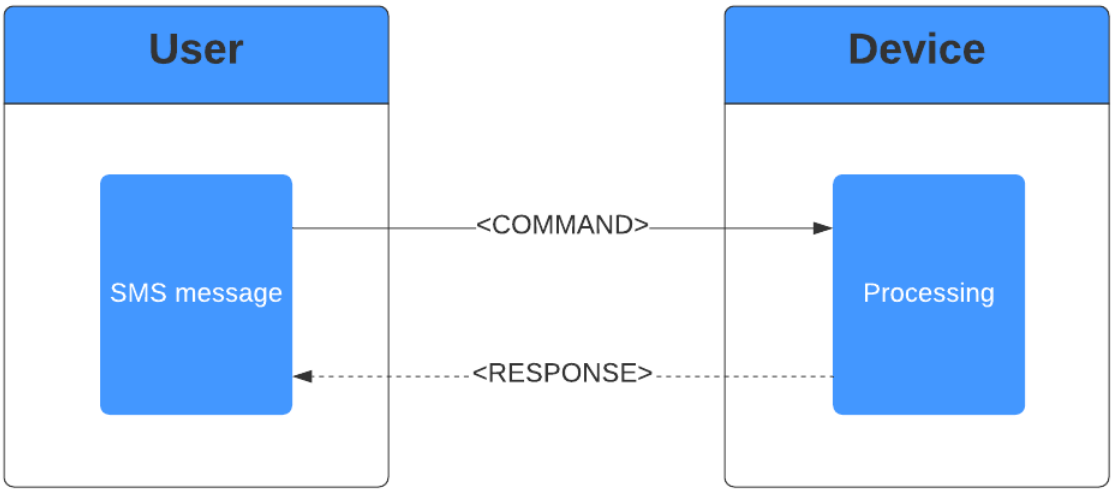
- Message over GSM (SMS)
- Data connection (GPRS)
- USB connection (USB)

All of them can be used to communicate with device.

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 touser also in their raw format.

The workflow is as follows:



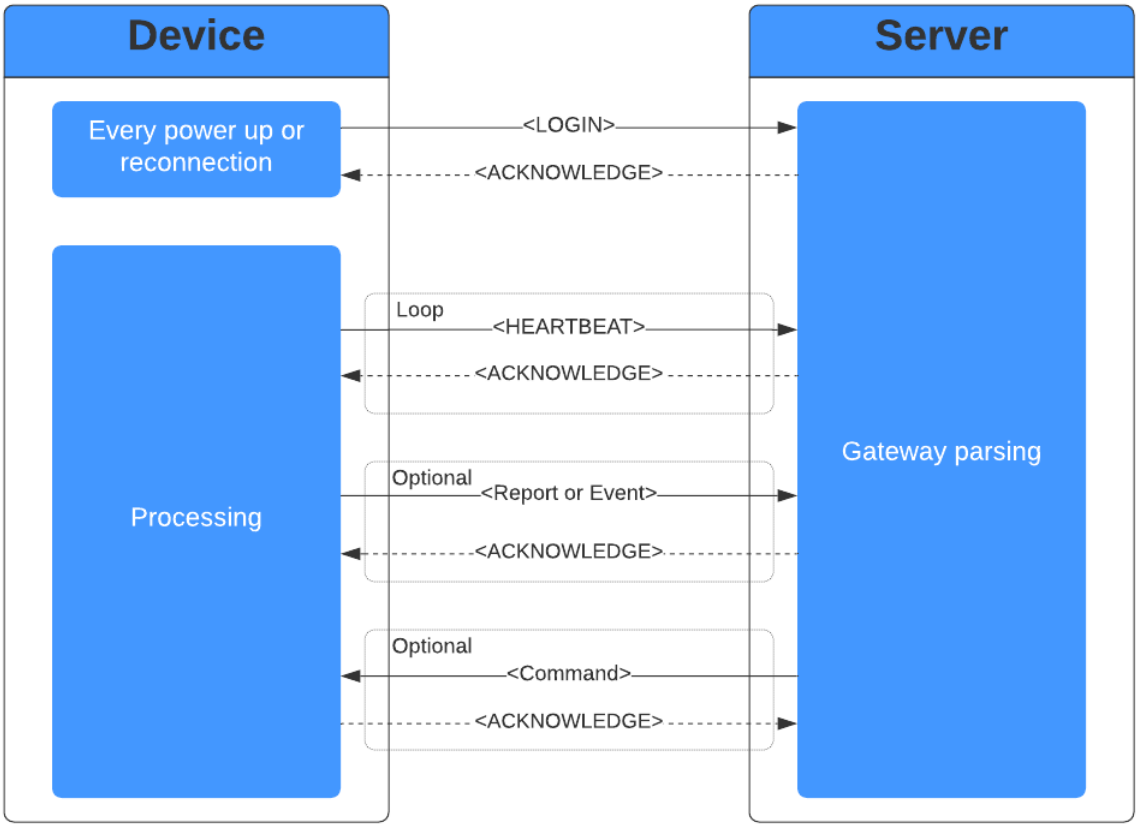
IP (network connection)

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.

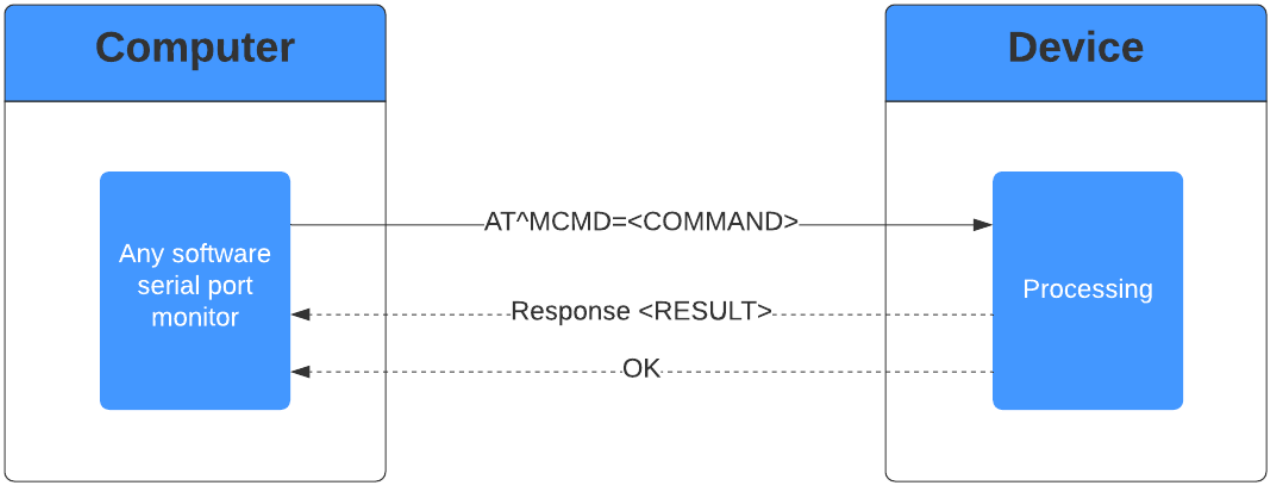
After the connection is established successfully, the heartbeat package will be sent periodically in a specific interval. The reason is to keep the connection and to detect the availability of connection.

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.

USB communication

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




```
AT^MCMD=<COMMAND>
<RESULT>OK
```

Example of USB command and reply.

```
Command: AT^MCMD=PARAM?
Response:
AT^MCMD=PARAM?
IMEI:355322092077477
APN:smart.m2m.vivo.com.br
SERVER:"tcp://107.170.21.173:8256"
COLLECT:60,1000,30,60,1
LANG:EN
GMT:W3.00
GPS:0
OK
```

2 DATA TYPES

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

INTEGER

Integer is the most important data type in protocol. Most data are represented in aninteger, 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

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 mustbe able to be calculated based on other data field. They will be discussed in detail later in Chapter 5 PACKAGES.

The byte order of a string is always from the first byte to the last byte.

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.

STATUS

An unsigned 16-bit integer in little-endian format is used to represent the status of the device. The definition of each bit is described below:

Bit	Description
0	1: GPS is fixed 0: GPS is not fixed
1	1: Device is designed for car 0: Device is not designed for car
2	1: Car engine is on (only valid when bit 1 is 1) 0: Car engine is off

3	1: Accelerometer is supported0: No accelerometer
4	1: The motion-warning is activated (only valid when bit 3 is 1) 0: The motion-warning is deactivated
5	1: Output control is supported 0: Not have output control
6	0: Output is on (only valid when bit 5 is 1) 1: Output is off
7	1: External charging is supported0: No external charging
8	1: Device is charging (only valid when bit 7 is 1)0: Device is not charging

Bit	Description
9	1: Device is active (only valid when bit 3 is 1)0: Device is stationary
10	1: GPS module is running 0: GPS module is not running
11	Not used
12	Not used
13	Not used
14	Not used
15	1: DIN0 is high level (only valid when DIN0 is supported)0: DIN0 is low level

Note:

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

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 are valid. The structure of a position is described as below:

Name	Bytes	Description
Time	4	The event time (UTC) when position data is collected
Mask	1	The mask to indicate which data are valid bit0: GPS data (always activated at firmware version 2.1.3) bit1: BSID 0 bit2: BSID 1 bit3: BSID 2 bit4: BSID 3 bit 5~7: not used
GPS Data	-	**Data are related to GPS (valid only if BIT0 of mask is 1)**
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	2	Unsigned 16 bits integer (in km/h)
Course	2	Unsigned 16 bits integer from 0 to 360 (in degrees)
Satellites	1	The number of satellites
BSID0	-	**Data are related to Cell Info (valid only if BIT1 of *mask* is 1)**
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
CID	4	Cell ID with RNC — Unsigned 32 bits integer
RxLev	1	Cell signal level — Unsigned 8 bits integer (0: -110dB1:-109dB 2:-108dB … 110: 0dB)
BSID1	-	**The following data are related to the 1st neighbor base station and valid only if BIT2 of mask is 1**
LAC	2	Same as definition in BSID0
CI	4	Same as definition in BSID0
RxLev	1	Same as definition in BSID0
BSID2	-	**The following data are related to the 2st neighbor base station and valid only if BIT3 of mask is 1**
LAC	2	Same as definition in BSID0
CI	4	Same as definition in BSID0
RxLev	1	Same as definition in BSID0
BSID3	-	**The following data are related to the 3st neighbor base station and valid only if BIT4 of mask is 1**
LAC	2	Same as definition in BSID0
CI	4	Same as definition in BSID0
RxLev	1	Same as definition in BSID0

Parsing the latitude and longitude data

For parsing the latitude and longitude we will use the Location package as sample.

The method used for calculate the latitude and longitude is described below:

Sample of Location Package (PID:0x12)

2828120047000e610d37f403fd09f9b3fac900a2000b000000000802d4000604180000cee
b1a046f116f0495000000000202155115510000000000000000000000000000000000c6
1fc000

Latitude
fd09f9b3
→ Convert to INT32 - Big Endian (ABCD) = -49677901
→ Divide the value converted to 1800000 = -27.598834
→The value of Latitude in format degree and decimal degree [deg.dddddd] is -27.598834
→ -27.598834

Longitude
fac900a2
→ Convert to INT32 - Big Endian (ABCD) = -87490398
→ Divide the value converted to 1800000 = -48.605777
→ The value of Longitude in format degree and decimal degree [deg.dddddd] is -48.605777
→ -48.605777

The latitude and longitude were included in all packages (location and alarm) on firmware version 2.1.3 or later versions.

3 TCP/IP

The protocol between device and server is based on IP network. It can be based on either TCP or UDP. This chapter describes the difference between using TCP and using UDP.

TCP PROTOCOL

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 arrived 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	...

UDP PROTOCOL

With UDP, device can send messages, in this case referred to as datagrams, to other hosts on an IP network. Prior communications are not required in order to set up transmission channels or data paths. It has no handshaking dialogues, and thus exposes the device to any unreliability of the underlying network: there is no guarantee of delivery, ordering, or duplicate protection. As a result, an extra encapsulation must be applied on all transmitting packages.

A UDP header will be added to the transmitting data. It includes datagram size, datagram checksum and device IMEI. The destination must check the integrity of datagram using them. Each datagram is allowed to contain multiple packages, but its total size is limited up to 1200 bytes.A UDP datagram is as below:

UDP Header	Package 1	Package 2	...	Package N

The structure of a UDP header is described as below:

Name	Bytes	Description

Mark	2	‘EL’
Size	2	Datagram size from next byte to end
Checksum	2	Datagram checksum (see note 1) from next byte to end
IMEI	8	Device IMEI

4 PACKAGES

GENERAL

In general, the structure of a package is described as below:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier
Size	2	Package size from next byte to end — Unsigned 16 bitsinteger
Sequence	2	Package sequence number — Unsigned 16 bits integer
Content	N	Package content

> Note:
>
> 1. *Package size* is the total size of *Sequence* and *Content*.
> 2. *Package sequence number* starts from 1 after device is rebooted, and is increased by 1 for every package.
> 3. All response packages must have same *Package identifier* and *Package sequence number* as the request package has.

The *package identifier* are listed as below:

PID	Direction	Respond?	Type
0x01	UP	YES	Login package

0x03	UP	YES	Heartbeat package
0x12	UP	YES (UDP)	Location package
0x14	UP	YES	Warning package
0x15	UP	YES	Report package
0x1B	UP	YES	Param-Set Package
0x80	DOWN	Y	Instruction package

> Note:
>
> 1. *UP* means that this package is originated by device. It will be sent to server and server should process and respond it.
> 2. *DOWN* means that this package is originated by server. It will be sent to device and device should process and respond it.
> 3. The PID of an *UP* package is less than 0x7F, and the PID of a *DOWN* package is equal to or greater than 0x80.
> 4. If using TCP, server needs to respond the location package(0x12).

LOGIN PACKAGE – PID: 0x01

If using TCP, login package will be sent to server immediately after every session is established. If using UDP, it will be sent only once after the device is rebooted. The server must respond it, or all other packages will not be sent.

Its structure is:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier — 0x01
Size	2	Package size from next byte to end — Unsigned 16 bitsinteger
Sequence	2	Package sequence number — Unsigned 16 bits integer
IMEI	8	Device UIN (IMEI)
Language	1	Device language: 0x01 — English;

Timezone	1	Device timezone — Signed 8 Bits integer (in 15 mins)
Sys Ver	2	System version — Unsigned 16 bits integer (e.g. 0x0205:V2.0.5)
App Ver	2	Application version — Unsigned 16 bits integer (e.g. 0x0211: V2.1.1)
PS Ver	2	Param-set (see note 1) version — Unsigned 16 bits integer(e.g. 0x0001: V1)
PS OSize	2	Param-set original size — Unsigned 16 bits integer
PS CSize	2	Param-set compressed size (see note 2) — Unsigned 16 bitsinteger
PS Sum16	2	Param-set checksum (see note 3) — Unsigned 16 bits integer

The response from server is:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier — 0x01
Size	2	Package size from next byte to end — Unsigned 16 bitsinteger
Sequence	2	Package sequence number — Unsigned 16 bits integer
Time	4	Current time (UTC) in the server
Version	2	Protocol version (see note 1) — 0x01: default
PS Action	1	Param-set action mask (see note 2)

Note:

1. Protocol version is the version of the protocol supported by server. If it is different from the protocol version in device, device will generateand transmit only the compatible packages to server.
2. When server receives the login package from device, it can check the information of Param-set to determine how to operates the Param-set. Then 2 optional actions may be taken and both of them can be taken at the same time:
   bit0: 1 — Tell device to upload the Param-set immediately. 0 — Do not upload it now.

bit1: 1 — Tell device to upload the Param-set if changed in the future.
0 — Do not upload it in the future.

Example of login package and the response:

```
From device:
28280100180005035254407167747100200205020500010432000088BD

From server:
2828010009000559OBD477000103
```

HEARTBEAT PACKAGE – PID: 0x03

Heartbeat package only appears in TCP. It is used to keep the session active. In common situation, if nothing are sent via the pathway (GPRS) in a few minutes, the pathway may be recycled by network provider. So heartbeat package must be sent to keep the sessionactive when it is nearly due.

If device has sent some heartbeat packages and not received any response, it will cut the corrupt connection and attempt to establish a new one.

UDP is not a stream-like protocol, so heartbeat package is not necessary for it.

Its structure is:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier — 0x03
Size	2	Package size from next byte to end — Unsigned 16 bitsinteger
Sequence	2	Package sequence number — Unsigned 16 bits integer
Status	2	Device status, see Section 3.5 STATUS

The response from server is:

Name	Bytes	Description
Mark	2	0x28 0x28

PID	1	Package identifier — 0x03
Size	2	Package size from next byte to end — Unsigned 16 bitsinteger
Sequence	2	Package sequence number — Unsigned 16 bits integer

Example of heartbeat package and the response:

```
From device: 282803000400070188
From server: 28280300020007
```

LOCATION PACKAGE – PID: 0x12

Location package is the most important package. It transfers the position and other information of device to server. If using TCP/UDP the server must respond it.

Its structure is:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier — 0x12
Size	2	Package size from next byte to end — Unsigned 16 bitsinteger
Sequence	2	Package sequence number — Unsigned 16 bits integer
Position	N	Device position
Status	2	Device status
Battery	2	Battery voltage (in mV) — Unsigned 16 bits integer
AIN0	2	AIN0 value (in cV) — Unsigned 16 bits integer
AIN1	2	AIN1 value (in cV) — Unsigned 16 bits integer
Mileage	4	Device mileage (in m) — Unsigned 32 bits integer
GSM Cntr	2	GSM counter from last *GSM* command (in min) —Unsigned 16 bits integer

GPS Cntr	2	GPS counter from last *GPS* command (in min) — Unsigned16 bits integer
RFU	16	Reserved to future used
Hourmeter	4	Accumulated time (in seconds) that device identified ignition turned on — Unsigned 32 bits integer
RFU	2	Reserved to future used

> Note:
>
> 1. *AIN* is the abbreviation of analog input port.
> 2. *Mileage* is accumulated only when GPS is fixed.
> 3. *GSM Counter* is used to recognize which phase GSM module is in. The phases are described in Section 6.3.11 GSM.
> 4. *GPS Counter* is used to recognize which phase GPS module is in. The phases are described in Section 6.3.12 GPS.
> 5. Not all fields are valid in a device, so please ignore non-existent or unnecessary data. For example: sensor data are valid only if relevant sensors exist. If a datum is invalid, its value will be zero.

The response from server is:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier — 0x12
Size	2	Package size from next byte to end — Unsigned 16 bitsinteger
Sequence	2	Package sequence number — Unsigned 16 bits integer

Example of location package and the response:

```
From device:
282812004716815cf817b503fd09f682fac900bb000a000000000802d4001012d50000114733046f118105650
00000007a24f36cf36c0713046000000000000000000000000000317c390000

From server: 28281200021681
```

WARNING PACKAGE – PID: 0x14

A warning package will be sent to server when a specific warning occurs.

Its structure is:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier — 0x14
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number — Unsigned 16 bits integer
Location	N	Device position
Warning	1	Warning type — Unsigned 8 bits integer
Status	2	Device status
Mileage	4	Device mileage (in m) — Unsigned 32 bits integer
Hourmeter	4	Accumulated time (in seconds) that device identified ignition turned on — Unsigned 32 bits integer
Battery	2	Battery voltage (in mV) — Unsigned 16 bits integer
AIN0	2	AIN0 value (in cV) — Unsigned 16 bits integer
AIN1	2	AIN1 value (in cV) — Unsigned 16 bits integer

The *warning type* is listed as below:

Value	Description
0x02	SOS (Panic)
0x01	External power cut-off
0x24	External power reconnected
0x87	External battery low voltage

0x05	Shift warning
0x08	GPS antenna open-circuit (only for the device with external GPS antenna)
0x09	GPS antenna short-circuit (only for the device with external GPS antenna)
0x81	Under-speed warning
0x82	Over-speed warning
0x83	In-to-fence warning
0x84	Out-of-fence warning
0x04	Activity warning
0x85	Shock warning
0x80	Over-speed within fence warning

The response from server is:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier — 0x14
Size	2	Package size from next byte to end — Unsigned 16 bitsinteger
Sequence	2	Package sequence number — Unsigned 16 bits integer
Content	N	Warning content — String

> Note:
>
> 1. *Warning content* has variable length. Its length is the size of the body.
> 2. If *warning content* is not empty, it will be sent as a message to all managers registered in device.

Example of warning package and the response:

```
From device:
2828140032000A5df0f85e03fd09f61ffac900a3000e000000000802d40005060c000028963187046b00000
072000433bc1155030c0000

From server:

2828140002000A
```

REPORT PACKAGE – PID: 0x15

A report package will be sent to server when a specific event occurs.

Its structure is:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier — 0x15
Size	2	Package size from next byte to end — Unsigned 16 bitsinteger
Sequence	2	Package sequence number — Unsigned 16 bits integer
Location	N	Device position
Report	1	Report type — Unsigned 8 bits integer
Status	2	Device status
Mileage	4	Device mileage (in m) — Unsigned 32 bits integer
Hourmeter	4	Accumulated time (in seconds) that device identified ignition turned on — Unsigned 32 bits integer
Battery	2	Battery voltage (in mV) — Unsigned 16 bits integer
AIN0	2	AIN0 value (in cV) — Unsigned 16 bits integer
AIN1	2	AIN1 value (in cV) — Unsigned 16 bits integer

The *report type* is listed as below:

Value	Description
0x01	Ignition on
0x02	Ignition off
0x03	DIN changed

The response from server is:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier — 0x15
Size	2	Package size from next byte to end — Unsigned 16 bitsinteger
Sequence	2	Package sequence number — Unsigned 16 bits integer

Example of report package and the response:

```
From Device:
282815003200345defaf8903fd09f63bfac900ad0005000000000702d40005060c000028963b01046f000000
7000031fa1115505670000

From server: 28281500020034
```

IDLETIME PACKAGE – PID: 0x3B

An idle time package will be sent to server when the ignition is ON, and the vehicle remains stationary for the configurated time.

Its structure is:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier — 0x3B
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number — Unsigned 16 bits integer
Location	N	Device position
Status	2	Device status
Battery	2	Battery voltage (in mV) — Unsigned 16 bits integer
AIN0	2	AIN0 value (in cV) — Unsigned 16 bits integer
AIN1	2	AIN1 value (in cV) — Unsigned 16 bits integer
IdleTime	2	Time set for idletime

The response from server is:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier — 0x3B
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number — Unsigned 16 bits integer

Example of IdleTime package and the response:

```
From Device:
28283B002B001D63E5387003FD09FF75FAC8A87C000000000360502D4000604180000CFC83E04
7F11A205080000001E

From server: 28283B0002001D
```

5 COMMANDS

All command keywords are case insensitive, but their parameters should be case sensitive. For example, "STATUS?" is the same as "status?", but "APN,CMNET#" is different of APN,cmnet#.

In general, end with '#' indicates it is an executive command, and end with '?' indicates it is a query command.

For example, "APN,cmnet#" is used to modify APN setting, and "APN?" is used to query current APN setting.

INSTRUCTION PACKAGE – PID: 0x80

Sometimes, server need request device to change a setting or to do something else. Inorder to do that, an instruction package will be sent to device.

Its structure is:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier — 0x80
Size	2	Package size from next byte to end — Unsigned 16 bitsinteger
Sequence	2	Package sequence number — Unsigned 16 bits integer
Type	1	Instruction type (see note 1) — Unsigned 8 bits integer
UID	4	Instruction UID (see note 2) — Unsigned 32 bits integer
Content	N	Instruction content — String

The *instruction type* is:

- 0x01: Indicate that *instruction content* is a device command
- Other: Reserved

The *instruction UID* is a unique number recognized by server. Device must respond same

UID in response package.

The response from device is:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier — 0x80
Size	2	Package size from next byte to end — Unsigned 16 bitsinteger
Sequence	2	Package sequence number — Unsigned 16 bits integer
Type	1	Instruction type — Unsigned 8 bits integer
UID	4	Instruction UID — Unsigned 32 bits integer
Result	N	Instruction result — String

Example of instruction package and the response:

```
From server (Command VERSION?):
282880000F000101131313137665727369 6F6E3F

From Device (Response of VERSION command):
28288000905788014C754C75494D45493A333532353434303731363737343731 0A494D53493A39343630
303137333139383137363834 0A49434349443A383938363031313438343132303231333033383 20A5359
5354454D3A4D373630315F56322E302E350A56455253494F4E3A4D584150505F56322E302E350A425549
4C443A4D617920203520323031372030393A32313A3038
```

SECURITY COMMANDS

LOGIN

If a password has been set in device, device is protected. If user wishes to send a command to a protected device via short message, it is necessary for him to log in device. After he logs in, he acquires the privilege to execute a command on device. At last, user can log out from device by specific command, or will log out automatically if idle more than 15 minutes.

If user has been registered as a manager, he need not log in to send a command.

```
LOGIN,[PASSWORD]#
> Login OK
> Login Error
```

LOGOUT

This command requests to log out from device immediately.

```
LOGOUT#
> Logout OK
> Logout Error
```

PASSWORD

This command requests to change the password in device.

```
PASSWORD,[OLD PASSWORD],[NEW PASSWORD]#
> SET PASSWORD OK
> SET PASSWORD Error
```

ACTION COMMANDS

UPGRADE

This command requests to upgrade the firmware in device.

```
UPGRADE,[REMOTE],[DOMAIN],[PORT]#
> Upgrade OK
> Upgrade Error
> Upgrade Error: Device busy
> Upgrade Error: Transfer fail
> Upgrade Error: Disk full
> Upgrade Error: File wrong

UPGRADE?
> TRANSFER:[STATE],[BLOCKS]
```

Name	Description
[REMOTE]	The remote file name

[DOMAIN]	The address of upgrade server (domain name or IP)
[PORT]	The port of upgrade server
[STATE]	The transferring state: IDLE, INITIAL, ACTIVE
[BLOCKS]	The transferred blocks

RESET

This command requests to reboot device.

```
RESET#
> Reset OK
> Reset Error
```

SHUTDOWN

This command requests to shutdown device. It is valid only if user remove the main power voltage of device before send this command.

```
SHUTDOWN#
> Shutdown OK
> Shutdown Error
```

FACTORY

This command requests to restore all settings to default in factory.

```
FACTORY#
> Factory OK
> Factory Error
```

PAGING

This command requests to upload a location package (0x12) now.

```
PAGING#
> Paging OK
> Paging Error
```

CLEAR

This command requests to clear all pending packages. Pay attention because after device receives this command, it will not discharge positions accumulated in the memory.

```
CLEAR#
> Clear OK
> Clear Error
```

RELAY

This command controlling the output 1 of device.

Normally used to activated/deactivated a relay.

```
  > Relay disable Error
  > Relay enable OK
  > Relay enable Error
  > Relay enable Delayed: Undetermined movement
  > Relay enable Delayed: Too fast (>20km/h)
  > Relay enable Delayed: Moving
  > Relay enable Error: Undetermined movement
  > Relay enable Error: Too fast (>20km/h)
  > Relay enable Error: Moving

RELAY?
  > RELAY:[STATE]
```

Name	Description
[PARAMETER]	0: Disable relay; 1: Enable relay; 2: safely immobilization[1]; 3: very safely immobilization[2]
[STATE]	ON/OFF

> Note:
> 1. The vehicle is safe only when the speed is lower than 20 km/h if GPS fixed, or it is stationary if GPS is not fixed.
> 2. The vehicle is very safely only when the speed is lower than 5 km/h if GPS is fixed

withing ACC = 0

SETTING COMMANDS

UIN / IMEI

This command requests the UIN/IMEI of device.

```
IMEI?
IMEI:[IMEI]RELAY,[PARAMETER]#
> Relay disable OK
```

Name	Description
[IMEI]	The IMEI of device

LANG

This command requests to change the language of device.

```
LANG,[LID]#
> SET LANG OK
> SET LANG Error

LANG?
> LANG:[LNAME]
```

Name	Description
[LID]	The language ID — 0: English
[LNAME]	The language name — EN

GMT

This command requests to change the time zone of device.

```
GMT,[E/W],[HOUR],[MINUTE],[DST]#
> SET GMT OK
> SET GMT Error

GMT?
GMT:[E/W][TZ],[DST]
```

Name	Description
[E/W]	Which globe — E: East W: West
[HOUR]	The hour part of time difference — -12 ~ 12
[MINUTE]	The minute part of time difference — 0, 15, 30, 45
[DST]	The day saving time (in hours) — 0, 1, 2
[TZ]	The time difference

HBT

This command requests to change the heartbeat timer. This parameter defines the idle time before device originates a heartbeat package in TCP session.

```
HBT,[HBT]#
> SET HBT OK
> SET HBT Error

HBT?
> HBT:[HBT]
```

Name	Description
[HBT]	The heartbeat timer (in minutes) — Float point number

DELAY

This command requests to change the answer timer. This parameter defines the delaytime before device answers an incoming call.

```
DELAY,[DELAY]#
> SET DELAY OK
> SET DELAY Error

DELAY?
> DELAY:[DELAY]
```

Name	Description
[DELAY]	The answer timer (in seconds) — Integer

APN

This command requests to change the APN of GPRS network.

```
APN,[APN],[USERNAME],[PASSWORD]#
> SET APN OK
> SET APN Error

APN?
> APN:[APN],[USERNAME],[PASSWORD]
```

Name	Description
[APN]	The APN to GPRS service
[USERNAME]	The username for GPRS service
[PASSWORD]	The password for GPRS service

SERVER

This command requests to change the address of location server.

```
SERVER,[SERVER]#
> SET SERVER OK
> SET SERVER Error

SERVER?
> SERVER:[SERVER]([IP])
```

Name	Description
[SERVER]	The URI of location server
[IP]	The IP address of location server, e.g. 202.128.0.32

Note:

1. If using TCP, the URI should be as "**tcp://www.domain.com:12345**".
2. If using UDP, the URI should be as "**udp://www.domain.com:54321**".
3. [IP] is valid only when device has connected to location server.

COLLECT

This command requests to change the parameters of location collection. All parameters define how to collect location package and how many location packages to be cached before they are sent to server. We have 3 strategies to collect location package.

The first strategy is based on time. The location packages are generated in specific interval. There are 2 intervals: [INTERVAL] and [ACTIVE]. [INTERVAL] defines the regular interval. [ACTIVE] defines the time interval when device is active/moving.

The second strategy is based on position. After device moves a specific distance, alocation package will be generated. [DISTANCE] defines the gap.

The third strategy is based on course. When device turns more than a specific angle, a location package will be generated. [TURN] defines the angle.

Every strategy can be omitted if its parameter is set to 0. If multiply strategies are set, a location packages will be generated when any one is met.

```
COLLECT,[INTERVAL],[DISTANCE],[TURN],[ACTIVE],[QUANTITY]#
> SET COLLECT OK
> SET COLLECT Error

COLLECT?
> COLLECT:[INTERVAL],[DISTANCE],[TURN],[ACTIVE],[QUANTITY]
```

Name	Description
[INTERVAL]	The time interval (in seconds) 0, 300~65535

[DISTANCE]	The running distance (in meters) 0, 500~65535
[TURN]	The turning angle (in degrees) 0, 30~180
[ACTIVE]	The time interval when device is moving/active (in seconds) 0, 10~65535
[QUANTITY]	The number of cached location packages before they are sent 1~10

MILEAGE

This command requests to initialize the mileage in device. After the mileage is initialized, it will be increased automatically when GPS is fixed.

```
MILEAGE,[MILEAGE]#
> SET MILEAGE OK
> SET MILEAGE Error

MILEAGE?
> MILEAGE:[MILEAGE](km)
```

Name	Description
[MILEAGE]	The mileage (in km)

MOTION

This command requests to enable/disable motion warning and set its parameter. After motion warning is enabled, any vibration will trigger a warning.

```
> SET MOTION OK
> SET MOTION Error
MOTION?
> MOTION:[SENSE],[DELAY]
```

Name	Description
[SENSE]	The sensitivity: 0: Disable warning; 1 ~ 9: Enable warning. 1 is the most sensitive, and 9 is the least sensitive.

[DELAY]	The delay time before a warning is emitted (in seconds).

SPEED

This command requests to enable/disable speed warning and set its parameter. After speed warning is enabled, any speed not in range will trigger a warning.

```
SPEED,[LOW],[HIGH],[OVER]#
> SET SPEED OK
> SET SPEED Error

SPEED?
> SPEED:[LOW],[HIGH],[OVER]
  MOTION,[SENSE],[DELAY]#
```

Name	Description
[LOW]	The low limit of the speed (in km/h) 0~255
[HIGH]	The high limit of the speed (in km/h) 0~255
[OVER]	The speed threshold (in km/h) over which the device will drive the relay 0~255

> Note:
>
> 1. When [LOW] is 0, the under-speed warning is disabled.
> 2. When [HIGH] is 0, the over-speed warning is disabled.
> 3. When [OVER] is 0, the speed-relay feature is disabled.

Example	Description
SPEED#	Disable speed warning
SPEED,30,0#	Enable under-speed warning when speed is less than 30km/h
SPEED,0,100#	Enable over-speed warning when speed is more than 100km/h
SPEED,30,100#	Enable both under-speed warning and over-speed warning
SPEED,0,0,80#	Drive the relay when the speed is over 80 km/h

FENCE

This command requests to add/remove/modify one or more fences in device. Up to 8 fences can be added into device. Each fence can be round or rectangle. And it can also be out-type, in-type or bidirectional. If it is a out-type, the outside of fence is banned. If device leaves it, an out-of-fence warning will be triggered. If it is a in-type, the inside of fence is banned. If device enters it, an in-to-fence warning will be triggered. If it is bidirectional, any action to cross the border will trigger a warning.

```
FENCE,[INDEX],[FLAG],[LNG0],[LAT0],[RADIUS],[OVER]#
FENCE,[INDEX],[FLAG],[LNG1],[LAT1],[LNG2],[LAT2]#
> SET FENCE OK
> SET FENCE Error

FENCE,[INDEX]?
> FENCE[INDEX]:[FLAG],[LNG0],[LAT0],[RADIUS],[OVER]#
> FENCE[INDEX]:[FLAG],[LNG1],[LAT1],[LNG2],[LAT2]
> GET FENCE Error
```

Name	Description
[INDEX]	The index of fence — Integer, 0 - 8
[FLAG]	The type and shape of fence — String, each char represents an attribution
[LNG0],[LAT0]	The longitude and latitude of the center of round fence
[RADIUS]	The radius of round fence (in meters)
[LNG1],[LAT1]	The longitude and latitude of the left-top corner of rectangle fence
[LNG2],[LAT2]	The longitude and latitude of the right-bottom corner of rectangle fence
[OVER]	The speed threshold (in km/h) over which the device will drive the relay

Note:

1. When [INDEX] is 0, it means a command to all fences.
When [LNG0] and [LAT0] are empty, we use last fixed position.
2. When [OVER] is 0, the speed-relay feature is disabled.

[FLAG]:

- N/A: Fence is disabled
- O: Out-type fence
- I: In-type fence
- C: Bidirectional fence
- R: Round fence
- S: Rectangle fence

Example	Description
FENCE,1,OR,,,500#	Setup fence 1 (out-type, round) round last fixed position
FENCE,1,IR,,,500#	Setup fence 1 (in-type, round) round last fixed position
FENCE,1,CR,,,500#	Setup fence 1 (bidirectional, round) round last fixed position
FENCE,1,OR,113.5,22.5,500#	Setup fence 1 (out-type, round) round specific position
FENCE,1,IR,113.5,22.5,500#	Setup fence 1 (in-type, round) round specific position
FENCE,1,CR,113.5,22.5,500#	Setup fence 1 (bidirectional, round) round specific position
FENCE,1,OS,113.2,22.2,113.8,22.8#	Setup fence 1 (out-type, rectangle)
FENCE,1,IS,113.2,22.2,113.8,22.8#	Setup fence 1 (in-type, rectangle)
FENCE,1,CS,113.2,22.2,113.8,22.8#	Setup fence 1 (bidirectional, rectangle)
FENCE,1,CR,113.5,22.5,500,80#	Setup fence 1 (bidirectional, round) round specific position and drive the relay when the speed is over 80 km/h
FENCE,1#	Remove the first fence
FENCE,0#	Remove all fences
FENCE,1?	Return the first fence
FENCE,0?	Return all fences

SLEEP

This command enable/disable the sleep mode. Device will enter in sleep mode after turned off ignition (ACC OFF) and motion sensor is static (stopped) during all timer to enter in sleep mode.

Device will go out of sleep mode after any condition described happen: Ignition turned on (ACC ON), Input 1 turned ON, Power cut, motion sensor detect moving.

```
SLEEP,[status],[sleep_time],[timer_enter_sleep]#
> SET SLEEP OK
> SET SLEEP Error

SLEEP?
> SLEEP:[status],[sleep_time],[timer_enter_sleep]
```

Name	Description
[status]	Parameter to enable or disable the sleep mode. Possible values: 0 (disable) or 1 (enable)
[sleep_time]	Transmission time used when device enter in sleep mode. Possible values: 0, 300~43200 seconds
[timer_enter_sleep]	Timer to device enter in sleep mode. Possible values: 180~3600 seconds

> Note:
>
> 1. When [status] is 0, the sleep mode is disabled.
> 2. When [status] is 1, the sleep mode is enable.

Example	Description
SLEEP,0,0,0#	Disable sleep mode
SLEEP,1,600,180#	Enable sleep mode after 180 seconds and transmission time will be each 600 seconds

STORED

This command configures the discharge mode of stored packages. As default, the device works using growing discharge without the current packages generated in real time.

The configuration allows device using growing discharge including the current packages generated in real time.

```
STORED,[mode]#

> SET STORED OK
> SET STORED Error

STORED?
> STORED:[mode]
```

Name	Description
[mode]	Possible values: 0 - means discharge packages without the current packages generated in real time; 1 - means discharge packages including the current packages generated in real time.

Example	Description
STORED,0#	Device will discharge packages without the current packages generated in real time
STORED,1#	Device will discharge packages including the current packages generated in real time.

INRELAY

This command configures logic of output control trigger. Device allows two modes to control of output.

- Output using standard logic, where relay,1# activate the output and relay,0# deactivated the output.

- Output logic is inverted, where relay,1# deactivated the output and relay,0# activated the output.

```
INRELAY,[mode]#

SET INRELAY OK
> SET INRELAY Error

INRELAY?
> INRELAY:[mode]
```

Name	Description	
[mode]	Possible values: 0 means standard logic, where relay,1# activate the output and relay,0# deactivated the output. 1 means logic is inverted, where relay,1# deactivated the output and relay,0# activated the output.	

Example	Description
INRELAY,0#	Standard logic.
INRELAY,1#	Logic is inverted.

SIMDETECT

This command has the function of identifying the absence of sim card in the equipment and after changing the output logic.

```
SIMDETECT,[mode]#

SET SIMDETECT OK
> SET SIMDETECT Error

INRELAY?
> SIMDETECT:[mode]
```

Name	Description

mode	Possible values: • 0 means standard logic, where the SIM DETECT function does not change the output. • 1 means detection is on and changes the output.

Example	Description
SIMDETECT,0#	Function Disabled.
SIMDETECT,1#	Function activated.

SHIFT

This command requests to enable/disable a shift fence in device. Shift fence is an automatic fence. It becomes valid whenever ignition is OFF, and returns invalid when ignition is ON. When ignition is OFF and car moves out of it, a shift warning will be triggered.

In order to make it to work, ignition line must be connected correctly.

```
SHIFT,[RADIUS]#
> SET SHIFT OK
> SET SHIFT Error

SHIFT?
> SHIFT:[RADIUS]
```

Name	Description
[RADIUS]	The radius of shift fence (in meters) 0: Shift fence is disabled, 50~1000 shift fence is enabled

> Note:
>
> 1. Normally, the radius should be more than 50m, or a wrong warning may be triggered because of random drifting position.

IDLETIME

This command enable/disable an idle time alert. This happen when the ACC is On, and the vehicle remains without movement for a configured time.

```
IDLETIME,[time],[speed],[accelerometer],[output]#

SET IDLETIME OK
> SET IDLETIME Error

IDLETIME?

> IDLETIME,[time],[speed],[accelerometer],[output]
```

Name	Description
[time]	Possible values: 0 ~ 65536 0 means standard logic and disable this function. Time in seconds that determines an idle event.
[speed]	Possible values: 0 ~ 65536 This value is used in km/h. Is a reference speed, equal or less than this is consider idle. Note: if [accelerometer] is set to 1, a vibration is considered movement, even if the vehicle has a velocity equal to 0 km/h;
[accelerometer]	Possible values: 0 – 1 0 disables the accelerometer verification. 1 enables the accelerometer verification.
[Output]	Possible values: 0 – 1 0 disables the Relay action when idle time event is generated. 1 enables the Relay action, if an idle time event is generated the Relay turns ON.

Example	Description

IDLETIME,30,5,0,1	Device will send a package and active Relay if the vehicle speed is less than 5km/h for 30 seconds.
IDLETIME,40,8,1,0	Device will send a package if the vehicle speed is less than 8km/h for 40 seconds and there is no vibration. Relay will not be triggered.

VIRTUAL IGNITION

This command enables or disables ignition mode:

```
VIRTUALIGN, [stopped detection], [movement detection], [mode]#

SET VIRTUALIGN OK
> SET VIRTUALIGN Error

VIRTUALIGN?
> VIRTUALIGN:[stopped detection], [movement detection], [mode]
```

Name	Description
mode	Possible values: 1 to 1800 seconds • 0 means standard logic, where the VIRTUALIGN function does not enable. • 1 means virtual ignition is on.
stopped detection	Possible values: Minimum time required for the accelerometer to not detect movement and consider that the vehicle's ignition is off.
movement detection	Possible values: 1 to 1800 seconds Minimum time required for the accelerometer to detect movement and consider that the vehicle's ignition is on.

Example	Description
VIRTUALIGN,120,10,0#	Function Disabled.
VIRTUALIGN,120,10,1#	Function activated.

QUERY COMMANDS

VERSION

This command requests to return the firmware version in device.

```
VERSION#
VERSION?
> IMEI:[IMEI] IMSI:[IMSI]
  ICCID:[ICCID]
  SYSTEM:[SYS VERSION]
  VERSION:[APP VERSION]
  BUILD:[BUILT TIME]
```

Name	Description
[IMEI]	The IMEI of device
[IMSI]	The IMSI of SIM
[ICCID]	The ICCID of SIM
[SYS VERSION]	The operating system version
[APP VERSION]	The application version
[BUILT TIME]	The built time of firmware

> Note:
>
> 1. The firmware in device contains two parts. One is the operating system, and another is application. The application is upgradable over the air.

Example	Result
VERSION?	IMEI:354188046036385 IMSI:9460016668297433 ICCID:89860116851009444751 SYSTEM:M6130_V2.0.5 VERSION:MXAPP_V2.0.5 BUILD:Apr 26 2017 17:35:13

PARAM

This command requests to return major parameters in device.

```
PARAM#
PARAM?
> IMEI:[IMEI]
  APN:[APN]
  SERVER:[SERVER]
  COLLECT:[TIMERS]
  LANG:[LNAME]
  GMT:[GMT]
  SAVING:[SAVING]
```

Name	Description
[IMEI]	The IMEI of device
[APN]	The APN
[SERVER]	The URI of location server
[TIMERS]	The timers of location collection
[LNAME]	The language name
[GMT]	The time zone
[SAVING]	The work mode of GPS

Example	Result
PARAM?	IMEI:354188046036385 APN:cmnet SERVER:” tcp://lb.iter.sc:8256” COLLECT:60,0,0,0,1 LANG:EN GMT:E12.00 SAVING:0

STATUS

This command requests to return current status in device.

```
STATUS#
STATUS?
> BATTERY:[BATTERY]
  GPRS:[GPRS]
  GSM:[GSM],[SIGNAL]
  GPS:[GPS],[SATELLITE]
  ACC:[ACC]
  RELAY:[RELAY]
  POWER:[POWER]
  MS:[MS]
```

Name	Description
[BATTERY]	The battery percentage — 0% ~ 100%
[GPRS]	The GPRS status — CLOSED, FAILED, SUCCESS
[GSM]	The GSM status — CLOSED, NONE, HIGH, MED, LOW
[SIGNAL]	The signal level — 0: -110dB 1: -109dB 2: -108dB 50: -60dB 70: -40dB
[GPS]	The GPS status — CLOSED, FIXED, UNFIXED
[SATELLITE]	The satellites number
[ACC]	The ACC status — ON, OFF
[RELAY]	The relay status — ON, OFF
[POWER]	The external power status — OK, NC
[MS]	The motion sensor model

> Note:
>
> 1. Not all data will be shown in result. For example, [ACC] will be lacked if device is not designed for vehicle.

Example	Result
STATUS?	BATTERY:100% GPRS:CLOSED GSM:MED,22 GPS:FIXED,6 ACC:ON RELAY:OFF MS:LIS3DH

WHERE

This command requests to return the recent coordinate of device.

```
WHERE#
WHERE?
> Lat:[LATITUDE]
  Lon:[LONGITUDE]
  Course:[COURSE]
  Speed:[SPEED]
  DateTime:[DATETIME]
```

Name	Description
[LATITUDE]	The latitude (in degrees)
[LONGITUDE]	The longitude (in degrees)
[COURSE]	The moving course
[SPEED]	The moving speed (in km/h)
[DATETIME]	The date and time

Example	Result
WHERE?	Lat:N22.55552 Lon:E113.94014 Course:0.0 Speed:0.2km/h DateTime:2017-05-02 22:19:14

URL

This command requests to return the google URL of device.

```
URL#URL?
> [URL]
```

Name	Description
[URL]	The google URL and other information

Example	Result
URL#	http://maps.google.com/?q=22.555525,113.940147 <0.0km/h 0.0> <2017-05-02 22:20:34>IMEI:354188046036385

PARAM-SET PACKAGE — PID: 0x1B

Param-set package will be sent to server when server requests param-set or param-set is changed. The data of param-set may becompressed and split. Server can not parse it until all blocks are received and merged.

Its structure is:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier — 0x1B
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number — Unsigned 16 bits integer

PS Ver	2	Param-set (see note 1) version — Unsigned 16 bits integer (e.g. 0x0001: V1)
PS OSize	2	Param-set original size — Unsigned 16 bits integer
PS CSize	2	Param-set compressed size (see note 2) — Unsigned 16 bits integer
PS Sum16	2	Param-set checksum (see note 3) — Unsigned 16 bits integer
Offset	2	Uploading offset — Unsigned 16 bits integer
Data	N	Uploading data

Note:

1. *Param-set* is the set of all parameters, which isdescribed in Appendix A.3 PARAM-SET.

The response from server is:

Name	Bytes	Description
Mark	2	0x28 0x28
PID	1	Package identifier — 0x1B
Size	2	Package size from next byte to end — Unsigned 16 bits integer
Sequence	2	Package sequence number — Unsigned 16 bits integer
Next	1	Indicate whether next block is sent — 0: Do not send 1: Send

Example of param-set package and the response:

```
From device:
28281B009E0005000104320092 66DF000008053FC0A20341303EFE8110D414404C0680185610CEF3A23C8C18154
005AB64300BD0AAA845755C0CE331CF0C1B036478B843D0EA288988320B42D068956405053C11A4588FA38803FD
599EC6EF4B7383D0FC3FB7333919EA637F3D8EFB1D79F9D27B8D7782191146AE344DC0766F01599EE898BBE5ED32
17444DBECA0AB4BADA4B08224A48F235D59759EDEB2A24EE9C20

From server: 28281B0003000500
```