Communication Protocol for JC261/JC400

Version: V1.3.5

Release Date: Nov 28, 2025

Revision History

Date	Version	Description
2020-7-15	V1.0.0	Initial
2021-1-26	V1.1.0	- Add some alarm type for 95 packet - Add external mo dule transparent transmission packet 9B - Add 0E fuel sensor in 94 packet
2021-1-26	V1.2.0	Modify http upload protocol
2021-3-12	V1.2.1	Modify Alarm packet protocol
2021-3-16	V1.2.2	- Add some alert type - Add Mileage in location packet
2021-4-12	V1.2.3	Add namelist of playback video upload protocol
2021-5-29	V1.2.4	Modify 5.10 live video (multi live streaming)
2021-6-8	V1.2.5	Add Additional Location Data Packet 37
2021-7-15	V1.2.6	Modify some information
2021-8-5	V1.2.7	Add alert type for seat belt & door
2023-5-29	V1.2.8	- Add alert type for ADAS - Add 9C packet for QCVN31 - Add 02 illegal card type for DLT
2024-06-04	V1.2.9	Add camera channel info packet E0
2024-08-20	V1.3.0	- Add peripherals information packet F2 - Add Location packet response - Add some alert type
2024-10-28	V1.3.1	Add some alert type
2025-03-31	V1.3.2	- Added the file upload protocol. - Moved sections 5.8 "Image/Video Upload Protocol (Image Server)" and 5.9 "NameList of Playback Video" from Chapter 5 to Chapter 6.
2025-08-11	V1.3.3	Updated the description for the "Alert file" field in 0x95 packet.
2025-08-20	V1.3.4	Added alarm type 0x9E in 0x95 packet.
2025-11-28	V1.3.5	Added Packet 0x70.

Contents

1. Communication Protocol

This document defines instructions about interface protocol on application layer of car rearview mirror, location-based service platform and image service platform. Related interface protocol only applies in the interaction between the platform and the position terminal.

2. Terms and Definitions

Terms/Abbreviation	Definition in English
CMPP	China Mobile Peer to Peer
GPS	Global Positioning System
GSM	Global System for Mobile Communication
GPRS	General Packet Radio Service
TCP	Transport Control Protocol
LBS	Location Based Services
IMEI	International Mobile Equipment Identity
MCC	Mobile Country Code
MNC	Mobile Network Code
LAC	Location Area Code
Cell ID	Cell Tower ID
UDP	User Datagram Protocol
SOS	Save Our Ship/Save Our Souls
CRC	Cyclic Redundancy Check
NITZ	Network Identity and Time Zone,
GIS	Geographic Information System

3. Basic Rules

(1) If a GPRS connection is established successfully, the terminal will send a first login message packet to the server and, within five seconds, if the terminal receives a data packet responded by the server, the connection is considered to be a normal connection. The terminal will begin to send location information (i.e., GPS, LBS information package). A status information package will be sent by the terminal after three minutes to regularly confirm the connection.

(2) If the GPRS connection is established unsuccessfully, the terminal will not be able to send the first login message packet. The terminal will start scheduled reboot in twenty minutes if the GPRS connection failed for three times. Within twenty minutes, if the terminal successfully connected to the server and received the data packet from the server as the server's response to the login message packet sent by the terminal, the scheduled reboot will be off and the terminal will not reboot; otherwise, the terminal will reboot automatically in twenty minutes.

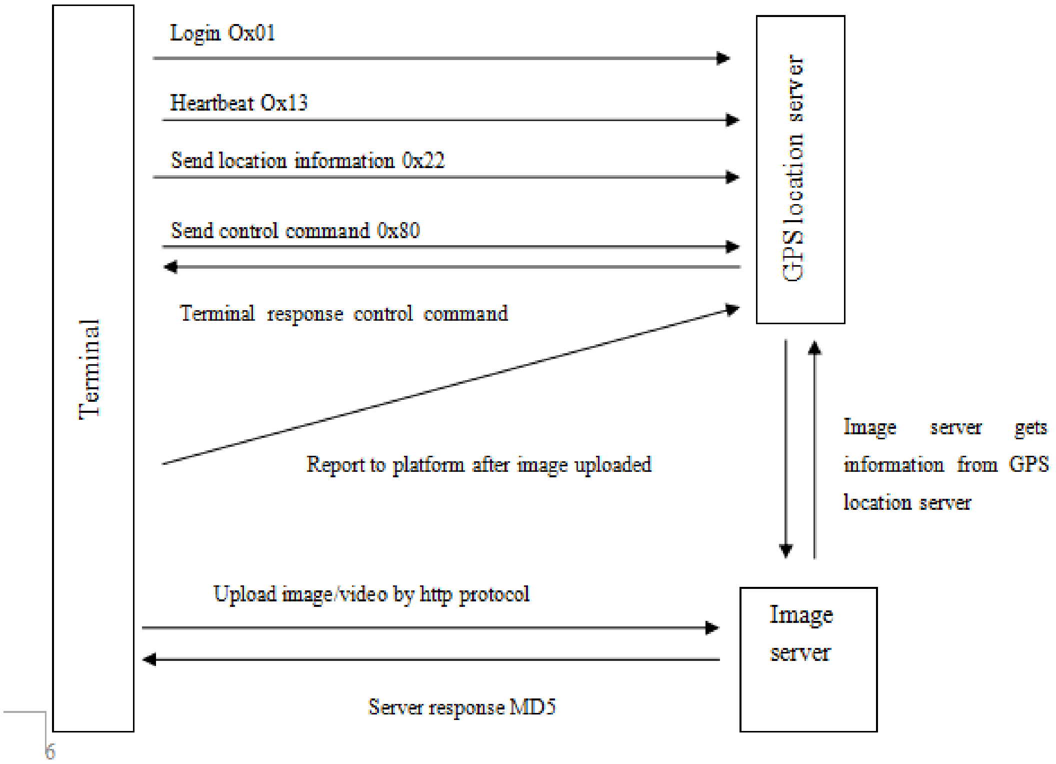
(3) After receiving the login message packet, the server will return a response data packet. If the terminal doesn't receive packet from the server within five seconds after sending the login message packet or the status information package, the current connection is regarded as an abnormal connection. The terminal will start a re-transmission function for GPS tracking data, which will cause the terminal to disconnect the current GPRS connection, rebuild a new GPRS connection and send a login message packet again.

(4) In case of the normal connection, the terminal will send a combined information package of GPS and LBS to the server after the GPS information is changed; and the server may set a default protocol for transmission by using commands.

(5) To ensure the effectiveness of the connection, the terminal will send status information to the server at regular intervals, and the server will return response data packets to confirm the connection.

(6) For the terminal which doesn't register an IMEI number, the server will reply the terminal with a login request response and heartbeat packet response, rather than directly disconnect the connection. (If the connection is directly disconnected or the server doesn't reply to the terminal, it will lead to a continuous reconnected by the terminal and the GPRS traffic will be consumed heavily.

Data Flow Diagram


Terminal
Login Ox01
Heartbeat Ox13
Send location information 0x22
Send control command 0x80
Terminal response control command
Report to platform after image uploaded
GPS location server
Image server gets information from GPS location server
Upload image/video by http protocol
Server response MD5
Image server
6

There are two ways to upload image and video:

(1) Upload by APP

Process:

Location server sends control command to terminal, asking for uploading image or video.

Terminal responses the command.

Terminal uploads image or video to image server through http protocol.

Image server returns result message and the MD5 information of image/video. If the result is failure, the terminal will upload image/video again.

Terminal reports IMEI, MD5 information of image/video and file type to location platform.

Terminal should send again if it receives the failure message returned by image server.

(2) Upload by terminal (Used for future extensions)

Process: Same as (3), (4), (5); (1) and (2) are omitted

NOTE: Currently, C2, C6 only support to upload by APP. Terminal upload automatically for future extension.

4. Data Packet Format

The communication is transferred asynchronously in bytes.

The total length of packets is (10+N) Bytes.

Format	Length(Byte)
Start Bit	2
Packet Length	1
Protocol Number	1
Information Content	N
Information Serial Number	2
Error Check	2
Stop Bit	2

4.1 Start Bit

Fixed value in HEX: 0x78 0x78

4.2 Packet Length

Length = Protocol Number + Information Content + Information Serial Number + Error Check, totally (5+N) Bytes, because the Information Content is in a variable length field.

4.3 Protocol Number

Type	Value
Login Message	0x01
Status information(Heartbeat)	0x13
Location Data Packet	0x22
Alarm Data Packet	0x95
Command information sent by the server to the terminal	0x80
Information Transmission Packet	0x94
External Module Transparent Transmission Packet	0x9B
Additional Location Data Packet	0x37
External Module Transparent Transmission Packet	0x9C
Camera Channel Information Packet	0XE0
Peripherals Sensor Data Packet	0xF2

4.4 Information Contents

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

4.5 Information Serial Number

The serial number of the first GPRS data (including status packet and data packet such as GPS, LBS) sent after booting is ‘1’, and the serial number of data sent later at each time will be automatically added ‘1’.

4.6 Error Checking

A check code may be used by the terminal or the server to distinguish whether the received information is wrong or not. To prevent errors occur during data transmission, error checking is added to against data inappropriate operation, so as to increase the security and efficiency of the system. The check code is generated by the CRC-ITU checking method.

The check codes of data in the structure of the protocol, from the Packet Length to the Information Serial Number (including "Packet Length" and "Information Serial Number") , are the value of CRC-ITU.

If CRC error occurs when the received information is calculated; the receiver will ignore and discard the data packet.

4.7 Stop Bit

Fixed value in HEX: 0x0D 0x0A

5. Details About Data Packet

The commonly used information packages sent by the terminal and those responded by the server will be interpreted separately.

5.1 Login Message Packet (0x01)

5.1.1 Login Packet Sent by Terminal to Server

The login message packet is used to be sent to the server with the terminal ID so as to confirm the established connection is normal or not.

Format		Length
Login Message Packet (18 Bytes)	Start Bit	2
	Packet Length	1
	Protocol Number	1
	Terminal ID	8
	Type Identification Code	2
	Time Zone Language	2
	Information Serial Number	2
	Error Check	2
	Stop Bit	2

Example: 78 78 11 01 07 52 53 36 78 90 02 42 70 00 32 01 00 05 12 79 0D 0A

Start Bit

For details see Data Packet Format section 4.1.

Packet Length

For details see Data Packet Format section 4.2.

Protocol Number

Protocol number: 0x01

Terminal ID

Example: if the IMEI is 123456789012345, then the terminal ID is 0x01 0x23 0x45 0x67 0x89 0x01 0x23 0x45.

Type Identification Code

Type identification code consumes two bytes, which shows the types of the terminals. The first three bits represent the type of the terminal, while the last bit represents the branch of the type.

Time Zone Language

One and a half bits **bit15—bit4**	15	Time zone expands 100 value	
	14		
	13		
	12		
	11		
	10		
	9		
	8		
	7		
	6		
	5		
	4		
Last nibble (bit 3-bit 0)	3	GMT. 0: Eastern; 1: Western	
	2	No definition	
	1	Language choice bit	1
	0	Language choice bit	0

Bit 3: Example: Extended bit: 0x32 0x00 means GMT+8

Calculation method: 8*100=800 converts to HEX: 0X0320

Extended bit: 0x4D 0xD8 means GMT-12:45

Calculation method: 12.45*100=1245 converts to HEX: 0x04 0xDD

Information Serial Number

For details see Data Packet Format section 4.5.

Error Check

For details see Data Packet Format section 4.6.

Stop Bit

For details see Data Packet Format section 4.7.

5.1.2 Server Responds

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

Format		Length
Login Message Packet (18 Bytes)	Start Bit	2
	Packet Length	1
	Protocol Number	1
	Information Serial Number	2
	Error Check	2
	Stop Bit	2

Example: 78 78 05 01 00 05 9F F8 0D 0A

5.2 Heartbeat Packet (0x13)

5.2.1 Heartbeat Packet Sent by Terminal to Server

Format			Length (Byte)
Information Content	Start Bit		2
	Packet Length		1
	Protocol Number		1
	Status Information	Terminal Information Content	1
		Voltage Level	1
		GSM Signal Strength	1
		Alarm/Extended Port Status	2
	Serial Number		2
	Error Check		2
	Stop Bit		2

Example: 78 78 0A 13 40 04 04 00 01 00 0F DC EE 0D 0A

Start Bit

For details see Data Packet Format section 4.1.

Packet Length

For details see Data Packet Format section 4.2.

Protocol Number

Protocol Number: 0x13

Terminal Information

1 byte is consumed defining for various status information of the terminal.

Bit		**Code Meaning**
BYTE	Bit7	Extended bit
	Bit6	1: GPS is positioning
		0: GPS does not positioning
	Bit3～Bit5	Extended bit
	Bit2	1: Charge On
		0: Charge Off
	Bit1	1: ACC high
		0: ACC low
	Bit0	Extended bit

For example: The binary encoding of 0x44 is 01000100

Voltage Level

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

- 0: No Power (shutdown)
- 1: Extremely Low Battery
- 2: Very Low Battery
- 3: Low Battery
- 4: Medium
- 5: High
- 6: Very High

GSM Signal Strength Levels

- 0x00: no signal;
- 0x01: extremely weak signal;
- 0x02: very weak signal;
- 0x03: good signal;
- 0x04: strong signal.

Language/Extended Port State

- 0x00 (former bit)
- 0x01 (latter bit)

Former bit: terminal extended port status

Latter bit: the current language used in the terminal

Former Bit	
Latter Bit	0x01 Chinese 0x02 English

Information Serial Number

For details see Data Packet Format section 4.5.

Error Check

For details see Data Packet Format section 4.6.

Stop Bit

For details see Data Packet Format section 4.7.

5.2.2 Server Responds

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

Format		**Length**
Information Content	Start Bit	2
	Packet Length	1
	Protocol Number	1
	Information Serial Number	2
	Error Check	2
	Stop Bit	2

Example: 78 78 05 13 00 0F 00 8F 0D 0A

5.3 Location Data Packet (0x22)

5.3.1 Location Packet Sent by Terminal to Server

Format			Length (Byte)
Information Content	Start Bit		2
	Packet Length		1
	Protocol Number		1
	GPS Information	Date Time (UTC format)	6
		Quantity of GPS information satellites	1
		Latitude	4
		Longitude	4
		Speed	1
		Course, Status	2
	LBS Information	MCC	2
		MNC	1
		LAC	2
		Cell ID	3
	ACC		1
	Data upload mode		1
	GPS real time upload		1
	Mileage		4
	Serial Number		2
	Error Check		2
	Stop Bit		2

Example: 78 78 26 22 16 02 08 09 2B 1E CF 02 6C 15 93 0C 39 8A 10 00 14 00 01 CC 0B 77 06 07 76 A4 01 0E 00 00 00 00 00 00 8C 6D 40 0D 0A

Start Bit

For details see Data Packet Format section 4.1

Packet Length

For details see Data Packet Format section 4.2.

Protocol Number

Protocol number 0x22

Date Time - UTC format

Format	Length(Byte)	Example
Year	1	0x0A
Month	1	0x03
Day	1	0x17
Hour	1	0x0F
Minute	1	0x32
Second	1	0x17

Example: 2010-03-23 15:50:23

Calculated as follows:

- 10 (Decimal) = 0A (Hexadecimal)
- 3 (Decimal) = 03 (Hexadecimal)
- 23 (Decimal) = 17 (Hexadecimal)
- 15 (Decimal) = 0F (Hexadecimal)
- 50 (Decimal) = 32 (Hexadecimal)
- 23 (Decimal) = 17 (Hexadecimal)

Then the value is: 0x0A 0x03 0x17 0x0F 0x32 0x17

Length of GPS information: quantity of positioning satellites

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 0xCB, it means the length of GPS information is 12 and the number of the positioning satellites is 11. (C = 12Bit Length, B = 11 satellites)

Latitude

4 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 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)X30000=40582974, then converted into a hexadecimal number

40582974(Decimal) = 26B3F3E (Hexadecimal)

At last the value is 0x02 0x6B 0x3F 0x3E.

Longitude

4 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 herein is same to the method mentioned in Latitude.

Speed

1 byte is consumed, defining the running Speed of GPS. The value ranges from 0x00 to 0xFF indicating a range from 0 to 225km/h.

e.g: 0x00 represents 0; 0x10 represents 16; 0xFF represents 255

Course & Status

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

Byte	**Bit**	**Meaning**
BYTE_1	Bit7	0
	Bit6	0
	Bit5	GPS real-time/differential positioning
	Bit4	GPS has been positioning or not
	Bit3	East Longitude, West Longitude
	Bit2	South Latitude, North Latitude
	Bit1	Course
	Bit0	
BYTE_2	Bit7	
	Bit6	
	Bit5	
	Bit4	
	Bit3	
	Bit2	
	Bit1	
	Bit0	

NOTE: The status information in the data packet is the status corresponding to the time bit recorded in the data packet.

For example: the value is 0x15 0x4C, the corresponding binary is 00010101 01001100,

BYTE_1 Bit 7	0
BYTE_1 Bit 6	0
BYTE_1 Bit 5	0 (Real-time GPS)
BYTE_1 Bit 4	1 (Located)
BYTE_1 Bit 3	0 (East longitude)
BYTE_1 Bit 2	1 (North latitude)
BYTE_1 Bit 1	0
BYTE_1 Bit 0	1
BYTE_1 Bit 7	0
BYTE_1 Bit 6	1
BYTE_1 Bit 5	0
BYTE_1 Bit 4	0
BYTE_1 Bit 3	1
BYTE_1 Bit 2	1
BYTE_1 Bit 1	0
BYTE_1 Bit 0	0

Heading of 332° (The binary 0101001100 translates to 332 in decimal)

which means real-time GPS tracking is on. The location is at north latitude, east longitude and the course is 332°

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.

MNC

Mobile Network Code (MNC)

Example: Chinese MNC is 0x00

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. (See GSM specification 03.03, 04.08 and 11.11).

Cell ID

The Value of Cell Tower ID (Cell ID) ranges from 0x000000 to 0xFFFFFF.

ACC

The stature of ACC: 00 means ACC OFF; 01 means ACC ON

GPS Data Upload Mode

- 0x00 upload by time interval
- 0x01 upload by distance
- 0x02 upload for corner correction
- 0x03 upload by ACC status change
- 0x04 Re-upload last valid locating point after shift from dynamic motion to static position.
- 0x05 Report last valid locating point after network reconnected.
- 0x06 Update the ephemeris to upload locating point.
- 0x07 Press to upload locating point
- 0x08 Power on to upload locating point
- 0x09 Power on and change locating point by GPS based station.
- 0x0E GPSDUP (upload at fixed time under static state or GPS not fixed)

GPS real-time upload

- 0x00 GPS real-time upload
- 0x01 GPS data re-upload

Mileage

Convert HEX into decimal. (Only available for devices with this function)

Information Serial Number

For details see Data Packet Format section 4.5

Error Check

For details see Data Packet Format section 4.6.

Stop Bit

For details see Data Packet Format section 4.7.

5.3.2 Server Responds

Format		Length
Information Content	Start Bit	2
	Packet Length	1
	Protocol Number	1
	Serial Number	2
	Error Check	2
	Stop Bit	2

Example: 78 78 05 22 01 23 2E 4B 0D 0A

5.4 Alarm Packet (0x95)

5.4.1 Alarm Data Packet Sent by Terminal to Server

Format	Length (Byte)		Length (Byte)	Description
Data	Start Bit		2	0x78 0x78
	Packet Length		1	From "Protocol number" to "Error check"
	Protocol Number		1	0x95
	Special marking		2	0xFF 0xFF
	Version		1	0x01
	Date Time		6	UTC format
	Location information	Satellite No.	1	The first character is GPS information length. The second character is positioning satellite number（converted to a decimal）
		Latitude	4	Convert to a decimal and divide 1800000
		Longitude	4	Convert to a decimal and divide 1800000
		Course, status	2	Convert to binary number of 16 bits and calculate by bits (see the following diagram)
		Speed	2	Convert to a decimal
	External battery voltage		2	Convert to a decimal and divide 10
	Alert information	Alert type	1	See below
		Alert value	2	Alert type = 0x87, it is the speed value
				Alert type = 0x8C - 0000 means level 1st, - 0002 means level 2nd
				Alert type = 0x6B

Format	Length (Byte)		Length (Byte)	Description
				- 0000 means disconnect, - 0002 means connected but communication abnormality
		Alert file	N	ASCII If facial recognition is successful (0xD4 alarm), the file name carries the driver ID (ID-NAME), as formatted: FILENAME:EVENT_353376110007413_00000000_2020_10_20_06_17_34_07.mp4;DRIVERID:869268BC67-abcd
	Information Serial Number		2	Serial number of data sent later at each time will be automatically added ‘1’.
	Error Check		2	Error check (From "Packet Length" to "Information Serial Number"), are values of CRC-ITU. CRC error occurs when the received information is calculated, the receiver will ignore and discard the data packet. (See Appendix 1)
	Stop Bit		2	0x0D 0x0A

Example: 78 78 93 95 FF FF 01 14 08 0B 06 11 24 DF 02 6C 19 58 0C 38 D1 DB 14 F3 00 00 00 8E 01 00 00 45 56 45 4E 54 5F 33 35 33 33 37 36 31 31 30 30 30 38 38 38 32 5F 30 30 30 30 30 30 30 30 5F 32 30 32 30 5F 30 38 5F 31 31 5F 31 34 5F 31 37 5F 33 36 5F 32 34 2E 6D 70 34 2C 45 56 45 4E 54 5F 33 35 33 33 37 36 31 31 30 30 30 38 38 38 32 5F 30 30 30 30 30 30 30 30 5F 32 30 32 30 5F 30 38 5F 31 31 5F 31 34 5F 31 37 5F 33 36 5F 32 33 2E 6D 70 34 00 0A DB DC 0D 0A

Start Bit

For details see Data Packet Format section 4.1.

Packet Length

For details see Data Packet Format section 4.2.

Protocol Number

Protocol number: 0x95

Date Time

Time of alarm data packet, UTC time

Alarm Type

	Alarm No.	Description
Alarm type	0x01	SOS
	0x0E	External battery low
	0x47	Fatigue driving
	0x50	Door closed
	0x51	Door opened
	0x53	Fuel stolen
	0x69	File uploading
	0x6B	DMS camera communication abnormality
	0x6C	Voice file download completed
	0x6D	Voice file download failed
	0x6F	SD card fault
	0x70	SD card mount
	0x73	Fuel data timeout
	0x76	Temperature data timeout
	0x80	Vibration
	0x83	Seat belt plugged (algorithm)
	0x84	Seat belt unplugged (algorithm)
	0x86	SD card unmount
	0x87	Over speed
	0x88	Power removal
	0x8A	Oil or electronics connected
	0x8B	Oil or electronics disconnected
	0x8C	DMS – Eye closed
	0x8F	DMS - Distraction
	0x90	Harsh acceleration
	0x91	Harsh deceleration
	0x92	Harsh turning
	0x93	Crash
	0x94	DMS - Missing face
	0x97	DMS - Phone calling
	0x9A	DMS - Smoking

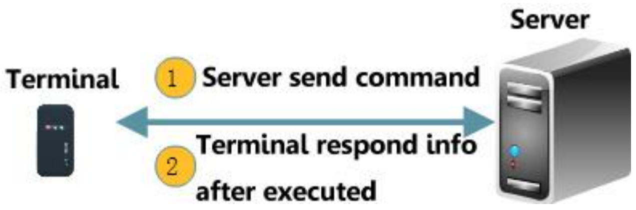
	Alarm No.	Description
	0x9E	Network status change notification
	0x9F	Illegal Driver
	0XA0	DMS - Yawn
	0XA1	DMS - Monitoring shelter
	0xA2	DMS – Calibration fail
	0xA3	DMS - Look down
	0xA4	SD card capacity is not enough
	0xA5	RFID reading
	0xA6	Seat belt plugged
	0xA7	Seat belt unplugged
	0xCC	Forward collision warning
	0xCD	Lane departure warning
	0xCE	Headway monitoring warning
	0xD4	Driver identification successful
	0xD5	Driver identification failed

The terminal needs to report the uploaded file name in the alarm packet

	Length	Description
Alarm value	2	When alarm type is 0x87, the data is the speed; 0x00 0x00 for other alarm
	N	Uploaded file name, ASCII format

Attention: 0x69 packet is judged whether file upload is successful, not for alarm.

5.5 Server Sends Data Packet to Terminal (0x80)



5.5.1 Control Command Sent by Server

Format	Length(Byte)
Start Bit	2

Format		Length(Byte)
Packet Length		1
Protocol Number		1
Information Detail	Command Length	1
	Server Zone Bit	4
	Command Content	M
Language		2
Info Serial Number		2
Error Check		2
Stop Bit		2

Example: 78 78 13 80 0B 00 00 00 00 56 45 52 53 49 4F 4E 00 02 59 D4 DA A7 0D 0A

Start Bit

See Detail at Data Packet Format 4.1

Packet Length

See Detail at Data Packet Format 4.2

Protocol Number

Terminal send Protocol Number by: 0x80

Command Length

Server Zone Bit + Command Length + Language

E.g: Byte Length as a unit, Ox0A, this command occupies 10 bytes

Server Flag Bit

For server detecting, terminal respond packet with the received original binary sample

Command Content

Express as string ASCII

Language

- Chinese: 0x00 0x01
- English: 0x00 0x02

Information Serial Number

See Detail at Data Packet Format 4.5

Error Check

See Detail at Data Packet Format 4.6

Stop Bit

See Detail at Data Packet Format 4.7

5.5.2 Terminal Return Data (Terminal Respond)

Format		Length(Byte)
Start Bit		2
Packet Length		2
Protocol Number		1
Information Detail	Server Flag Bit	1
	Content Code	4
	Content	M
Language		2
Info Serial Number		2
Error Check		2
Stop Bit		2

Example: 79 79 00 18 21 00 00 00 00 01 5B 54 49 4D 45 52 5D 20 4F 46 46 2C 31 30 00 2E 5D D7 0D 0A

Start Bit

Fixed Value 0x79 0x79

Packet Length

Occupy 2 bytes

Protocol Number

Use 0x21

Server Flag Bit

For server detecting, terminal respond packet with the received original binary sample

Content Code

- 0x01: ASCII
- 0x02: UTF16-BE

Content

The data needed to transmit

For transmitting image and video, this protocol is unavailable to upload image nor video, after terminal got command respond from server, terminal should link to image server to upload packet. This respond content fixed as ASCII code: picture (upload image) or video (upload video).

Information Serial Number

See Detail at Data Packet Format 4.5.

Error Check

See Detail at Data Packet Format 4.6.

Stop Bit

See Detail at Data Packet Format 4.7.

5.6 Data Transfer Common Packet (0x94)

Terminal send data packet to server

Format		Length (Byte)
Data packet (16 Bytes)	Start Bit	2
	Packet length	2
	Protocol number	1
	File Type	1
	Information Content	N

Format		Length (Byte)
	Serial Number	2
	Error Check	2
	Stop Bit	2

Example: 79 79 00 0B 94 0E 00 01 00 01 02 00 0A 24 4D 0D 0A

Start Bit

0x79 0x79

Packet Length

For details see Data Packet Format section 4.2.

Packet length=protocol number+ information type + information content +information serial number + error check

Protocol Number

For details see Data Packet Format section 4.3.

File Type

File type stands for the type of uploading file

……

- 07 DVR UUID
- 08 DVR report version
- 0A DVR report IMEI / IMSI / ICCID
- 0E Fuel Sensor

Information Content

- When type is 07, this bit transfer mirror terminal report UUID. By ASCII
- When type is 08, this bit transfer mirror terminal report VERSION. By ASCII
- When type is 0A, this bit transfer mirror terminal report IMEI / IMSI / ICCID. By hexadecimal

IMEI	8	E.g: IMEI is 123456789123456，the terminal ID would be: 0x01 0x23 0x45 0x67 0x89 0x12 0x34 0x56
IMSI	8	E.g: IMSI is 123456789123456，the terminal ID would be: 0x01 0x23 0x45 0x67 0x89 0x12 0x34 0x56

ICCID	10	E.g: ICCID is12345123456789123456， the terminal ID would be: 0X12 0x34 0x51 0x23 0x45 0x67 0x89 0x12 0x34 0x56

When the type is 0E

type	1	00: fuel data
Id	1	Fuel sensor’s ID
value	2	Hex, convert to decimal and divide 100
Unit	1	01 Height; 02 Percentage; 03 Voltage

5.7 External Module Transparent Transmission Packet (0x9B)

Format		**Length**	**Detailed**
Start bit		2	0x79 0x79
Packet length		2	Length = protocol number + message content + message serial number + error check
Protocol number		1	0x9B
Information	Module type coding	1	- 0x03 Temperature sensor - 0x05 DLT card reader
	Transparent data transmission	N	Transparent data transmission
Message serial number		2	After booting, the serial number will automatically increase by 1 each time the data is sent
Error checking		2	CRC-ITU value from "packet length" to" information sequence number". If the receiver has a CRC error in the calculation of the received information, it will be ignored and discarded (see Appendix A for the algorithm)
Stop bit		2	Fixed value, unified as 0x0D 0x0A

Example: 79 79 00 0D 9B 03 3C 03 01 00 FA C4 0D 00 08 24 81 0D 0A

When the module type is **0x03**

Data	n	Convert to ASCII Eg: HEX data 32 38 2E 33, 28.3 degree
Checksum	2	
Stop bit	2	

Or

Start bit	1	3C (fixed)
Length	1	Status+Data
Status	1	
Data	n	Hex convert to be Dec. & divided by 10 Eg: 00 FA, 25.0 degree
Checksum	1	BCC format
Stop bit	1	0D (fixed)

When the module type is **0x05**

Timestamp	6	UTC time, format is the same with location packet
Status	1	00: Logout; 01: Login; 02: Illegal card
Data	N	2520205E53554B534157414444454524534149544841524E244D4953535E5E3F3B363030373634333130303530303135373839313D3135303631393830303930393D3F2B202020202020202020202020203234202020202020202020202020203220202020202020202020202020202020202020202030303034353532202030303130302020202020202020202020202020202020202020202020203F0D0A

5.8 Live Video

Communication Protocol	RTMP
Push URL Demo	rtmp://domain or ip{: port}/live/{channel}/{imei}
Param Detail	{imei} is base on device imei {port} can be ignored {channel} 0 for road facing camera 1 for cabin facing camera pip for picture-in-picture streaming non channel for historical playback online
Communication Protocol	RTMP over HTTP
Push URL Demo	http://domain or ip{: port}/live/{channel}/{imei}.flv
Param Detail	{imei} is base on device imei {channel} 0 for road facing camera 1 for cabin facing camera pip for picture-in-picture streaming non channel for historical playback online

5.9 Additional Location Data Packet (0x37)

Terminal Sending Location Data Packet to Server

Format			Length (Byte)
Information Content	Start Bit		2
	Packet Length		1
	Protocol Number		1
	GPS Information	Date Time	6
		Quantity of GPS information satellites	1
		Latitude	4
		Longitude	4
		Speed	1
		Course, Status	2
	LBS Information	MCC	2
		MNC	1 or 2
		LAC	2
		Cell ID	3
	ACC		1
	Data upload mode		1
	GPS real time upload		1
	Mileage		4
	Extension module		N
	Serial Number		2
	Error Check		2
	Stop Bit		2

Example: 78 78 26 37 15 06 07 00 09 06 CF 02 6C 28 B5 0C 38 B9 BB 23 14 89 01 CC 01 25 4E 06 15 F8 01 1B 00 00 00 EA 95 22 3F 73 C5 0D 0A

Start Bit

For details see Data Packet Format section 4.1

Packet Length

For details see Data Packet Format section 4.2.

Protocol Number

Protocol number 0x37

Date Time

For details see Location Data Packet Format - "Date Time"

Length of GPS information: quantity of positioning satellites

For details see Location Data Packet Format - "Length of GPS information: quantity of positioning satellites"

Latitude

For details see Location Data Packet Format - "Latitude"

Longitude

For details see Location Data Packet Format - "Longitude"

Speed

For details see Location Data Packet Format - "Speed"

Course & Status

For details see Location Data Packet Format - "Course & Status"

MCC

For details see Location Data Packet Format - "MCC"

In order to be compatible with the situation that MNC occupies two bytes in some countries, the bit15 of MCC is used to distinguish the length of MNC

- When Bit15 of MCC is 1, the length of MNC is 2
- When Bit15 of MCC is 0, the length of MNC is 1
- The Bit15 of old device is 0; the Bit15 of the new device is 1

Bit15	**Bit 14 ~ Bit0**
1: MNC length is 2 bytes	

MNC

Mobile Network Code (MNC)

LAC

For details see Location Data Packet Format - "LAC"

Cell ID

For details see Location Data Packet Format - "Cell ID"

ACC

For details see Location Data Packet Format - "ACC"

GPS Data Upload Mode

- 0x00 upload by time interval
- 0x01 upload by distance
- 0x02 upload for corner correction
- 0x03 upload by ACC status change
- 0x04 Re-upload last valid locating point after shift from dynamic motion to static position.
- 0x05 Report last valid locating point after network reconnected.
- 0x06 Update the ephemeris to upload locating point.
- 0x07 Press to upload locating point
- 0x08 Power on to upload locating point
- 0x09 Power on and change locating point by GPS based station.
- 0X0A The device reports the last latitude and longitude after it is stationary, but the time is updated
- 0X0B WIFI parsing latitude and longitude upload package
- 0X0C LJDW (immediate positioning) instruction reporting
- 0X0D The device reports the last latitude and longitude after it is stationary
- 0X0E GPSDUP upload (upload at fixed time under static state or GPS not fixed)
- 0X0F Exit tracking mode
- 0x10 Open the door (container)
- 0x11 Close the door (container)
- 0x12 Demolition (container)
- 0x13 installation (container)
- 0x14 Entry fence (container)
- 0x15 Out of the fence (container)
- 0x16 Environment abnormal (container)
- 0x17 Switch to shipping mode (container)
- 0x18 Switch to land transport mode (container)
- 0x19 Switch to static mode (container)
- 0x1A low battery (container)
- 0x1B event triggers reporting (applicable to DVR devices)

GPS real-time upload

For details see Location Data Packet Format – "GPS real-time upload"

Mileage

For details see Location Data Packet Format – "Mileage"

Cannot exist with "Extension module" at the same time

Extension module

No available for DVR device

Information Serial Number

For details see Data Packet Format section 4.5

Error Check

For details see Data Packet Format section 4.6.

Stop Bit

For details see Data Packet Format section 4.7.

5.10 External Module Transparent Transmission Packet (0x9C)

Format		Length (Byte)	Detailed
Start bit		2	0x79 0x79
Packet length		2	Length = protocol number + message content + message serial number + error check
Protocol number		1	0x9B
Information	Module type coding	1	- 0x32 QCVN31 real-time data - 0x33 QCVN31 re-upload data
	Transparent data transmission	N	Transparent data transmission
Message serial number		2	After booting, the serial number will automatically increase by 1 each time the data is sent
Error checking		2	CRC-ITU value from "packet length" to "information sequence number". If the receiver has a CRC error in the calculation of the received information, it will be ignored and discarded (see Appendix A for the algorithm)
Stop bit		2	Fixed value, unified as 0x0D 0x0A

Example: 79 79 00 34 9C 32 24 47 53 48 54 2C 33 2C 33 30 2C 3C 31 30 3A 30 32 3A 35 38 2C 31 31 33 2C 39 34 33 31 35 2C 32 32 2C 35 37 36 35 31 2C 32 31 3E 2C 34 23 00 80 D7 3E 0D 0A

5.11 Camera Channel Information Packet (0xE0)

5.11.1 Camera info Packet Sent by Terminal to Server

Format		Length (Byte)	Detailed
Start bit		2	0x79 0x79
Packet length		2	Length = protocol number + message content + message serial number + error check
Protocol number		1	0xE0
Information	Built-in Camera	1	Total Camera quantity
		1	Camera’s number 1
		1	Status. 01 means available, 00 means unavailable
		1	Camera’s number N
		1	Status. 01 means available, 00 means unavailable
	Peripherals Camera	1	Total Camera quantity
		1	Camera’s number 1
		1	Status. 01 means available, 00 means unavailable
		1	Camera’s number N
		1	Status. 01 means available, 00 means unavailable
Message serial number		2	After booting, the serial number will automatically increase by 1 each time the data is sent
Error checking		2	CRC-ITU value from "packet length" to "information sequence number". If the receiver has a CRC error in the calculation of the received information, it will be ignored and discarded (see Appendix A for the algorithm)
Stop bit		2	Fixed value, unified as 0x0D 0x0A

Example: 78 78 0B E0 01 01 01 01 02 01 01 0C 0A AE 0D 0A

5.11.2 Server Responds

Same protocol with device sending. Or device will try one more attempt max.

5.12 Peripherals Information Packet (0xF2)

5.12.1 Peripherals Info Packet Sent by Terminal to Server

Format		Length (Byte)	Detailed
Start bit		2	0x79 0x79
Packet length		2	Length = protocol number + information content + message serial number + error check
Protocol number		1	0xF2
Information	Date and Time	4	UTC0 time in unix format, need to convert to be decimal, accurate to seconds.
	Peripherals ID	2	Peripherals ID
		1/2	The length is determined by the specific length of the peripherals ID as below
		n	Value
		…	…
Message serial number		2	After booting, the serial number will automatically increase by 1 each time the data is sent
Error checking		2	CRC-ITU value from "packet length" to "information sequence number". If the receiver has a CRC error in the calculation of the received information, it will be ignored and discarded (see Appendix A for the algorithm)
Stop bit		2	Fixed value, unified as 0x0D 0x0A

Example: 79 79 00 9F F2 65 11 67 A3 00 08 93 17 08 0F 09 15 08 02 25 20 20 5E 48 41 4D 55 4E 24 41 4E 55 43 48 41 24 4D 52 2E 5E 5E 3F 3B 36 30 30 37 36 34 31 31 32 30 36 30 30 31 31 32 30 32 38 3D 31 39 31 30 31 39 38 38 31 30 31 33 3D 3F 2B 20 20 20 20 20 20 20 20 20 20 20 20 20 32 36 30 30 20 20 20 20 20 20 20 20 20 20 20 20 31 20 20 20 20 20 20 20 20 20 20 20 20 35 31 30 30 31 36 32 38 20 20 31 30 37 30 30 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 3F 0D 0A 11 22 00 21 0D 0A

5.12.2 Server Responds

Format	Length (Byte)	Detailed
Start bit	2	0x79 0x79
Packet length	2	Length = protocol number + information content + message serial number + error check
Protocol number	1	0xF2

Format	Length (Byte)	Detailed
Message serial number	2	After booting, the serial number will automatically increase by 1 each time the data is sent
Error checking	2	CRC-ITU value from "packet length" to "information sequence number". If the receiver has a CRC error in the calculation of the received information, it will be ignored and discarded (see Appendix A for the algorithm)
Stop bit	2	Fixed value, unified as 0x0D 0x0A

Example: 78 78 05 F2 11 22 25 5C 0D 0A

5.12.3 Peripherals ID list

ID	Length	Date Type	Sensor Type	Description
0x0001	1	Int (2 bytes)	Capacitive Fuel Sensor	Divided by 100; Unit is V Eg: 00 06 = 0.06V
0x0002	1	Type (1 byte) Path (1 byte) Value (2 bytes) Unit (1 byte)	Multi Fuel Sensor	Type: 00 ultrasonic fuel sensor; 01 capacitive fuel sensor Path: default is 00, which refers to the fuel tank number Value: When Type=00, divided by 10 (eg: 00 06 = 0.6) When Type=01, divided by 100 (eg: 00 06 = 0.06) Unit: 01 Height, unit is cm 02 Percentage, unit is % 03 Voltage, unit is V 04 Volume, unit is Liters
0x0003	1	ID (1 byte) Value (2 bytes)	Multi ADC voltage	ID: the ADC interface number Value: divided by 100 (eg: 00 0B = 0.11), unit is V
0x0004	1	2bytes	Temperature Sensor	Divided by 10, unit is degree (eg: 01 1F = 28.7) Note: when bit32 is 1, need to negate the value (eg: FF 97 = -10.5)
0x0005	1	N bytes	RFID	ASCII convert to be character
0x0006	1	Int (2 bytes)	Pulse fuel sensor	Divided by 100, unit is V, range is 0~12V

<table>
<tr><th>ID</th><th>Length</th><th>Date Type</th><th>Sensor Type</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td></td><td>Eg: 00 06 = 0.06V</td></tr>
<tr><td>0x0007</td><td>2</td><td>2+8n bytes
Batch number (2 bytes)
1st UTC (4 bytes)
1st temperature (2bytes)
1st humidity (2 bytes)
…
nth UTC (4 bytes)
nth temperature (2 bytes)
nth humidity (2 bytes)</td><td>Multi temperature sensor</td><td>Temperature:
Divided by 10, unit is degree (eg: 01 1F = 28.7)
Note: when bit32 is 1, need to negate the value (eg: FF 97 = -10.5)

Humidity:
Divided by 10, unit is % (eg: 03 5B = 85.9)</td></tr>
<tr><td>0x0008</td><td>1</td><td>Timestamp (6 bytes)
Status (1 byte)
DLT data (n bytes)</td><td>DLT card reader</td><td>Timestamp: UTC format (eg: 17 08 0f 09 15 08 = 2023-08-15 9:21:08)
Status: 00 = Logout; 01 = Login; 02 = Illegal
DLT data: please follow the requirement of DLT of Thailand</td></tr>
<tr><td>0x0009</td><td>2</td><td>Peripherals ID (1 byte)
Data (n bytes)</td><td>Not defined</td><td></td></tr>
<tr><td>0x000B</td><td>1</td><td>Course (2 bytes)
Latitude (4 bytes)
Longitude (4 bytes)</td><td>GPS</td><td>Course:
<table>
<tr><td>Bit15</td><td rowspan="2">Extensions</td></tr>
<tr><td>Bit14</td></tr>
<tr><td>Bit13</td><td>GPS type: 0 = real-time, 1 = differential</td></tr>
<tr><td>Bit12</td><td>GPS: 0 = not fixed, 1 = fixed</td></tr>
<tr><td>Bit11</td><td>Longitude: 0 = East, 1= West</td></tr>
<tr><td>Bit10</td><td>Latitude: 0 = South, 1 = North</td></tr>
<tr><td>Bit9</td><td rowspan="10">Course</td></tr>
<tr><td>Bit8</td></tr>
<tr><td>Bit7</td></tr>
<tr><td>Bit6</td></tr>
<tr><td>Bit5</td></tr>
<tr><td>Bit4</td></tr>
<tr><td>Bit3</td></tr>
<tr><td>Bit2</td></tr>
<tr><td>Bit1</td></tr>
<tr><td>Bit0</td></tr>
</table>
Latitude: converted to be decimal and divided by 1800000</td></tr>
</table>

ID	Length	Date Type	Sensor Type	Description
				Longitude: converted to be decimal and divided by 1800000

Remark: Cyan highlighted are the ID which JC261 support

5.13 Extension Data Packet (0x70)

This packet transmits the extension module information.

5.13.1 Data Packet Sent by Terminal to Server

Table 5-1 Data Packet Structure

<table>
<tr><th colspan="2">Field</th><th>Length (Byte)</th><th>Description</th></tr>
<tr><td colspan="2">Start bit</td><td>2</td><td>0x79 0x79</td></tr>
<tr><td colspan="2">Packet length</td><td>2</td><td>Length = protocol number + message content + message serial number + error checking</td></tr>
<tr><td colspan="2">Protocol number</td><td>1</td><td>0x70</td></tr>
<tr><td rowspan="3">Information</td><td>Module 1 (0x33)</td><td>2</td><td>0x33 (GPS Information)</td></tr>
<tr><td>Module 1 Length</td><td>2</td><td>Length of the module content</td></tr>
<tr><td>Module 1 Content</td><td>n</td><td>[4] Timestamp: UTC time, format is the same with location packet
[1] Positioning satellite count: Convert to decimal
[2] Altitude: Convert to decimal; the MSB "1" in the binary represents a negative value; the last 7 bits convert to decimal to get the altitude
[4] Latitude: Convert to decimal and divide by 1,800,000
[4] Longitude: Convert to decimal and divide by 1,800,000
[1] Speed: Convert to decimal
[2] Course, status:
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>bit 15</td><td>ACC status bit (0: ACC OFF / 1: ACC ON)</td></tr>
<tr><td>bit 14</td><td>Whether ACC status bit is valid (0: Invalid / 1: Valid)</td></tr>
<tr><td>bit 13</td><td>0: Real-time positioning / 1: Differential positioning</td></tr>
<tr><td>bit 12</td><td>GNSS positioning status (0: No position fix / 1: Position fix acquired)</td></tr>
<tr><td>bit 11</td><td>0: East longitude / 1: West longitude</td></tr>
<tr><td>bit 10</td><td>0: South latitude / 1: North latitude</td></tr>
</table>
</td></tr>
</table>

Field		Length (Byte)	Description
			bits 09–00 Course
	Module 2 (0x27)	2	0x27 External battery voltage
	Module 2 Length	2	Length of the module content
	Module 2 Content	n	Voltage data (Convert to decimal and divide by 100 to get the value in volts.)
	Module 3 (0x35)	2	0x35 (Real-time or buffered GPS data upload)
	Module 3 Length	2	Length of the module content
	Module 3 Content	n	- 0x00 Real-time upload - 0x04 Buffered upload
Message serial number		2	After booting, the serial number will automatically increase by 1 each time the data is sent
Error checking		2	CRC-ITU value from "packet length" to "information sequence number". If the receiver has a CRC error in the calculation of the received information, it will be ignored and discarded (see Appendix A for the algorithm)
Stop bit		2	Fixed value, unified as 0x0D 0x0A

NOTE: Module numbers cannot be repeated within the same data packet.

Example data: 79 79 00 26 70 00 33 00 12 69 27 24 3C 0F 00 45 02 6C 15 E9 0C 39 8A 2B 00 14 A0 00 27 00 02 04 E0 00 35 00 01 00 0A D3 37 52 0D 0A

5.13.2 Server Response

If the response length is less than 255 bytes, the reply packet starting with 0x78 0x78 is used.

Table 5-2 Data Structure of Server Reply Packet

Field	Length (Byte)	Description
Start bit	2	0x78 0x78
Packet length	1	Length = protocol number + message content + message serial number + error checking
Protocol number	1	0x70
Information serial number	2	After booting, the serial number will automatically increase by 1 each time the data is sent
Error checking	2	CRC-ITU value from "packet length" to "information sequence number". If the receiver has a CRC error in the calculation of the received information, it will be ignored and discarded (see Appendix A for the algorithm)

Field	Length (Byte)	Description
Stop bit	2	Fixed value, unified as 0x0D 0x0A

6. HTTP Upload Protocols

6.1 Image/Video Upload Protocol (Image Server)

6.1.1 General

UTF-8 encoding default, and the data returned in JSON format

Context-Type: default is application/json charset=utf-8

Timestamp: UTC format, yyyy-MM-dd HH:mm:ss

Return data's attributes

Parameter	Types	Required	Description
code	string	Y	Result code
message	string	N	Code description
data	Object	N	Return data result

Transmission protocol

Data transmitted in JSON string format.

HTTP request Headers

Parameter	Types	Required	Description
Content - Type	String	Y	Fixed value "application/json; charset =UTF-8 "

Description of status code

Common status codes	
Code	Description
200	Request succeeded
400	Parameter error
500	System error

6.1.2 Agreement details

Upload interface

http://[IP address]/upload

HTTP request method

Support POST submission method. Since the uploaded file is different from other http request content, you must modify the Content-Type of the Header as follows:

Parameter	Types	Required	Description
Content-Type	String	Y	multipart/form-data

Request URL parameters

Parameter	Types	Required	Description
file	FilePart	Y	binary format
filename	String	Y	File name: must be unique, otherwise error happen during storage
timestamp	String	Y	Current time in milliseconds
sign	String	Y	Encrypted string, used to verify whether the request is legal. Encryption algorithm: MD5 encryption "filename + timestamp + secretKey" to be 32 bits lower letter, and then Base64 encryption of the encrypted string SecretKey: fixed value, jimidvr@123!443

Response parameter

Parameter	Types	Required	Description
code	string	Y	Return result code, 200 success
message	string	N	Status code description information
data	String	N	Return file name

Example:

```
{
"code":200,
"data":"img07-1588139792287.png",
"message":"the high success"
}
```

```
{
"code":400,
"message":"File content is empty"
}
```

6.2 NameList of Playback Video

Terminal upload the namelist of playback video to the specified URL when receive command.

6.2.1 Request

Http Method	POST
Data	Body / raw / json

Parameter

Parameter	Types	Required	Description
imei	String	Y	
fileNameList	String	Y	Video name list

Example

```
{"imei":"357730090564767","fileNameList":"2021_06_10_18_50_17_01.mp4,2021_06_10_18_51_20_01.mp4,2021_06_10_18_52_20_01.mp4"}
```

6.2.2 Response

Content-Type	Application/json ; charset=UTF-8

Your device data receiving service needs to respond correctly, the format is JSON

Parameter	Types	Required	Description
code	boolean	Y	Response result code, success =0,
msg	boolean	N	Result code description information
ok	boolean	Y	true

Example

```
{"code":0, "ok":true}
```

6.3 File Upload

6.3.1 General

UTF-8 encoding default, and the data returned in JSON format

Context-Type: default is application/json charset=utf-8

6.3.2 Agreement details

HTTP request method

Support POST submission method. Since the uploaded file is different from other http request content, you must modify the Content-Type of the Header as follows:

Basic Information:

Content-Type: multipart/form-data

Interface Request:

Method: POST

Endpoint: Depend on the data type

Request URL parameters

Parameter	Types	Required	Description
file	FilePart	Y	binary format
imei	String	Y	
instructionId	String	Y	The value of serverFlag in 0x80 packet
timestamp	String	Y	Current time in milliseconds
sign	String	Y	base64(md5(imei+instructionId+secretKey+timestamp)) SecretKey is fixed to be jimidvr@123!443 Note: Please sort all "parameters" in ASCII queue before encryption.

Response parameters

Parameter	Types	Required	Description
code	string	Y	Return result code, 200 success
message	string	N	Status code description information
data	String	Y	Return filename

Example:

```
{
"code":200,
"data":"img07-1588139792287.png",
"message":"the high success"
}
```

7. Appendix A

Code Fragment of the CRC-ITU Lookup Table Algorithm Implemented Based on C Language

Code fragment of the CRC-ITU lookup table algorithm implemented based on C language is as follow:

```
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
}
```