TCP Interface protocol

Gofunow Scooter IoT Device

Interface protocol

Revision history record:

Version	Date	Modification description	Writer
V1.0.0		Original version	

Content

1. TCP communication protocol description

1.1 Commands format

Note: Instructions are in the form of strings, each separated by ‘,’, and each instruction ends with a newline (‘\n’).

When the server sends an instruction, it needs to add 0xFFFF to the instruction header, HEX form.

<1> <2> <3> <4> <5> <6> <7>

0xFFFF*SCOS,OM,123456789123456,XX,DDD#<Wrap>

item	content	Description
1	0xFFFF	Two bytes reserved, HEX form
2	*SCOS	Command header server->Lock use *SCOS, lock->Server use *SCOR
3	OM	Vendor code
4	123456789123456	Lock unique ID number, use the IMEI number of the lock communication module (15 numbers)
5	XX	Instruction type (see list of instructions)
6	DDD	Contents items carried by the instruction, there may be multiple items here, separated by ‘,’
7	#<Wrap>	End of instruction

1.2 Commands list

Serial number	command	Command description
1	Q0	Check-in command, the lock will be sent first after each connection to the server, (including reconnection after disconnection)
2	H0	Heartbeat command, the lock is sent every 4 minutes to maintain the connection
3	R0	Unlocking/Lock operation request command
4	L0	Unlock the command, send the R0 command before sending this command
5	L1	Lock the scooter command, send the R0 command before sending this command

6	S5	IOT device setting instructions
7	S6	Get scooter information
8	S7	Scooter setting instruction 1
9	S4	Scooter setting instruction 2
10	W0	Alarm command
11	V0	Beep playback command
12	D0	Get positioning instructions, single time
13	D1	Positioning tracking instruction
14	G0	Get the firmware version
15	E0	Upload controller fault code
16	U0	Detect upgrade/boot upgrade
17	U1	Get upgrade data
18	U2	Notification of upgrade successfully
19	K0	Setting/Getting BLE 8-byte communication KEY

1.3 Instruction details

1.3.1 Q0（Singing in Command）

	<1><2> <3>
lock->server	*SCOR,OM,123456789123456,Q0,412,80,28#<LF>
1	IOT current voltage 412->4.12V
2	Scooter current power 80->80%
3	Current network signal value，the value is 2-32. The larger the value，the better the signal
server->lock	No response

1.3.2 H0（Heatbeat）

	<1><2> <3><4><5>
lock->server	*SCOR,OM,123456789123456,H0,0,412,28,80,0#<LF>
1	Scooter status 0->unlocked state，1->locked state
2	IOT current voltage 412->4.12V
3	Current network signal value，the value is 2-32. The larger the value，the better the signal
4	Scooter current power 80->80%
5	Scooter charging status 0->Uncharged，1->Charging
server->lock	No response

1.3.3 R0（Unlocking/Lock operation request command）

	<1><2><3> <4>
server->lock	*SCOS,OM,123456789123456,R0,0,20,1234,1497689816#<LF>
1	Requested operation 0->Unlock operation 1->Lock operation 2->RFID card unlock operation 3->RFID card lock car operation 6-> unlocks again and does not reset the ride time
2	KEY effective time Unit: Second (The lock will randomly generate KEY when receiving this command)
3	User ID
4	Timestamp/operation sequence number
	<1><2> <3> <4>
lock->server	*SCOR,OM,123456789123456,R0,0,55,1234,1497689816#<LF>
1	Requested operation 0->Unlock operation 1->Lock operation
2	The operation KEY is randomly generated by the lock，and the value is 0-255. This KEY is required when the server sends the unlock/lock operation.
3	User ID (sent with the server)
4	Timestamp/operation sequence number (sent by the server)

1.3.4 L0（Unlock the command, send the R0 command before sending this command）

	<1> <2> <3>
Server->lock	*SCOS, OM, 123456789123456, L0, 55, 1234, 1497689816#<LF>
1	Operation KEY, obtained by R0 instruction
2	User ID
3	Timestamp/operation sequence number
	<1> <2> <3>
Lock->server	*SCOR, OM, 123456789123456, L0, 0, 1234, 1497689816#<LF>
1	Status Return 0->Success 1->Failure 2->KEY Error or Failure
2	User ID (sent with the server)
3	Timestamp/operation sequence number (sent by the server)
Server->lock	*SCOS, OM, 123456789123456, L0#<LF> (Increase server response, make sure the server has received the unlock and return)

1.3.5 L1（Lock the car command, send the R0 command before sending this command）

	<1>
Server->lock	*SCOS, OM, 123456789123456, L1, 55#<LF>
1	Operation KEY, obtained by R0 instruction
	<1><2> <3> <4>
Lock->server	*SCOR, OM, 123456789123456, L1, 0, 1234, 1497689816, 3#<LF> Response
1	Status Return 0->Success 1->Failure 2->KEY Error or Failure
2	User ID (same unlock command)
3	Timestamp/operation sequence number (same unlock command)
4	Cycling time
Server->lock	*SCOS, OM, 123456789123456, L1#<LF>(Increase server response, make sure the server has received the unlock and return)

1.3.6 S5（IOT device settings）

Note: The following settings are saved after power down.

	<1><2><3><4>
Server->lock	*SCOR, OM, 123456789123456, S5, 3, 1, 240, 10#<LF>
1	Accelerometer sensitivity 0:invalid (Don’t set) 1:low 2:Middle 3:High (Defaults:

	2:中）
2	Unlock status upload scooter information（S6） 0:invalid（Don’ t set） 1:shut down 2:Open （Defaults：1:shut down）
3	Heartbeat upload interval 0:invalid（Don’ t set） 0: Invalid (not set) Unit: Second (default 240S)
4	Scooter information (S6) upload interval in unlocked state 0: Invalid (not set) Unit: Second （default: 10S）
<1><2><3><4>	
Lock->server	*SCOR,OM,123456789123456,S5,3,1,240,10#<LF>
1	Accelerometer sensitivity 0:invalid（Don’ t set） 1:low 2:Middle 3:High
2	Unlock status upload scooter information（S6） 0:invalid（Don’ t set） 1:shut down 2:open
3	Heartbeat upload interval 0:invalid（Don’ t set） Unit: Second (default 240S)
4	Scooter information (S6) upload interval in unlocked state 0:invalid（Don’ t set） Unit: Second （default: 10S）

1.3.7 S6（Obtain scooter information）

Server->lock	*SCOS,OM,123456789123456,S6#<LF>
<1><2><3><4><5> <6><7><8>	
lock->server	*SCOR,OM,123456789123456,S6,80,3,221,0,372,372,0,28#<LF>
1	The current power of the scooter 80->80%
2	Current mode of scooter 1:low speed 2:medium speed 3:high speed
3	Current speed 22->22km/h
4	Scooter charging status 0->Not charged，1->Charging
5	Battery 1 Voltage unit：0.1V 372->37.2V
6	Battery 2 Voltage unit：0.1V 372->37.2V（When the scooter has no a battery 2, it is 0 here.）
7	Scooter status 0->unlocked， 1->locked
8	Current network signal value, the value from 2-32. The larger value the better signal.
9	Single riding mileage unit: 10m 100->1000m

1.3.8 S7（Scooter setting instruction 1）

Note: The following setting information is not saved after power-off, and the default value is restored after restarting or unlocking.

<1><2><3><4>	
server->lock	*SCOS,OM,123456789123456,S7,0,3,0,0#<LF>
1	Headlight switch 0:invalid（Don’ t set） 1:shut down 2:open （Defaults：1:shut down）
2	Mode setting 0:invalid（Don’ t set） 1:shut down 2:Medium speed 3:high speed（Defaults: 2: shut down）
3	Throttle response 0:invalid（Don’ t set） 1:shut down 2:open （Defaults: 1:shut down）
4	Taillights flashing 0:invalid（Don’ t set） 1:shut down 2:open （Defaults: 1:shut down）
<1><2><3><4>	

lock->server	*SCOR,OM,123456789123456,S7,0,3,0,0 #<LF>
1	Headlight switch 0:invalid（Don’ t set） 1:shut down 2:open
2	Mode setting 0:invalid（Don’ t set） 1:low speed 2:medium speed 3:high speed
3	Throttle response 0:invalid（Don’ t set） 1:shut down 2:open
4	Taillights flashing 0:invalid（Don’ t set） 1:shut down 2:open

1.3.9 S4（Scooter setting instructions 2）

Note: The following setting information is saved after power down.

<1><2><3><4><5><6><7><8>	
server->lock	*SCOS,OM,123456789123456, S7,0,0,0,0,0,15,20,25#<LF>
1	Inch speed display 0:invalid（Don’ t set） 1:shut down 2:open （Defaults: 1:shut down）
2	Cruise control setting 0:invalid （Don’ t set） 1:shut down 2:open （Defaults: 1:shut down）
3	Startup mode setting 0:invalid（Don’ t set） 1:Non-zero start 2:Zero start（Defaults: 1:Non-zero start）
4	Button switching speed mode 0:invalid（Don’ t set） 1:shut down 2:open （Defaults: 2:open）
5	Key switch headlight 0:invalid（Don’ t set） 1:shut down 2:open （Defaults:2:open）
6	Low speed mode speed limit value 0:invalid（Don’ t set） Range：6-25km/h （Defaults:15km/h）
7	Medium speed mode speed limit value 0:invalid（Don’ t set） Range：6-25km/h （Defaults:20km/h）
8	High speed mode speed limit value 0:invalid（Don’ t set） Range：6-25km/h （Defaults:25km/h）
<1><2><3><4><5><6><7><8>	
lock->server	*SCOR,OM,123456789123456,S7,0,0,0,0,0,15,20,25#<LF>
1	Inch speed display 0:invalid（Don’ t set） 1:shut down 2:open
2	Cruise control setting 0:invalid（Don’ t set） 1:shut down 2:open
3	Startup mode setting 0:invalid（Don’ t set） 1:Non-zero start 2:Zero start
4	Button switching speed mode 0:invalid（Don’ t set） 1:shut down 2:open
5	Key switch headlight 0:invalid（Don’ t set） 1:shut down 2:open
6	Low speed mode speed limit value 0:invalid（Don’ t set） Range：6-25km/h
7	Medium speed mode speed limit value 0:invalid（Don’ t set） Range：6-25km/h
8	High speed mode speed limit value 0:invalid（Don’ t set） Range：6-25km/h

1.3.10 W0（Alarm commands）

	<1>
lock->server	*SCOR,OM,123456789123456,W0,1#<LF>
1	Alarm content 1:Illegal movement alarm 2:Falling alarm 3: illegal removed alarm
	4:Low power alarm 6, Lifted up alarm 7, illegal demolition alarm(connection restoration)
server->lock	*SCOS,OM,123456789123456,W0#<LF>

1.3.11 V0（Beep playback commands）

	<1>
server->lock	*SCOS,OM,123456789123456,V0,1#<LF>
1	Play content 1: Hold 2: Find a scooter alert 80: Turn off voice 81: Turn on voice
	<1>
lock->server	*SCOR,OM,123456789123456,V0,1#<LF>
1	Play content 1: Hold 2: Find a scooter alert 80: Turn off voice 81: Turn on voice

1.3.12 D0（Get positioning instructions, single time）

Note:Invalid format may occur like *SCOR,OM,123456789123456,D0,0,033724.00,V,,,,,,,120517,,,N#

server->lock	*SCOS,OM,123456789123456,D0#<LF>
	<1> <2> <3> <4> <5> <6> <7><8> <9> <10> <11><12><13>
lock->server	*SCOR,OM,123456789123456,D0,0,124458.00,A,2237.7514,N,11408.6214,E,6,0.21,151216,10,M,A#<LF>
1	0: obtain positioning and upload identifier 1: Position tracking &upload positioning identifier
2	UTC time, hhmmss (hours, minutes and seconds) format
3	Positioning status, A=effective positioning, V=invalid positioning
4	Latitude ddmm.mmmm (degrees) format (the previous 0 will also be transmitted)
5	Latitude hemisphere N (northern hemisphere) or S (southern hemisphere)
6	Longitude dddmm.mmmm (degrees) format (the previous 0 will also be transmitted)
7	Longitude hemisphere E (East) or W (Western)
8	Number of satellites searched
9	HDOP (positioning accuracy)
10	UTC date, ddmmyy (day and month) format
11	Altitude
12	Height unit M: meter
13	Mode indication (A= autonomous positioning, D=differential, E=estimate, N=invalid data)

Note: the obtained longitude and latitude coordinates are in degree scale format, and the coordinates converted into degree format are the coordinates of WGS84 coordinate system.This coordinate can be used directly on maps using the WGS84 coordinate system, such as Google maps.

Latitude conversion algorithm: lat=dd+mm.mmm/60. In terms of N or S, where N is positive and S is negative.

Longitude conversion algorithm: lng = dd +mm. mmmm/60.

That is, the original latitude coordinates obtained from the above table example are 2237.7514.

Converted to WGS84 coordinates, lat =22+37.7514/60=22.62919 . N is northern hemisphere, so it is positive.

Converted to WGS84 coordinates, lng =114+08.6214/60=114.14369, E is east longitude and therefore positive.

Note : WGS84 coordinates cannot be directly used on domestic maps such as gaode map or baidu map, and need to be converted to the coordinates used on the corresponding map. Please refer to the corresponding map API document for conversion algorithm.

1.3.13 D1（Positioning tracking commands）

	<1>
server->lock	*SCOS,OM,123456789123456,D1,60#<LF>
1	Upload positioning interval Unit: second, when this value is 0, turn off tracking
	<1>
lock->server	* SCOR,OM,123456789123456,D1,60#<LF>
1	Upload positioning interval Unit: seconds, when this value is 0, close tracking

1.3.14 G0（Get the firmware version）

server->lock	*SCOS,OM,123456789123456,G0#<LF>
	<1> <2> <3> <4>
lock->server	*SCOR,OM,123456789123456,G0,110,Jul 4 2018,1101,0#<LF>
1	Iot device software version 110->V1.1.0
2	Iot device software compilation date
3	Scooter controller software version 1101->0x1101 (represents PCB version number 1, firmware version number V1.0.1)
4	Reserved

1.3.15 E0（Upload controller fault code）

	<1>
lock->server	*SCOR,OM,123456789123456,E0,1#<LF>
1	error code
server->lock	*SCOS,OM,123456789123456,E0#<LF>

1.3.16 U0（Detect upgrade/boot upgrade）

	<1><2> <3>
lock->server	*SCOR,OM,123456789123456,U0,110,8A,1101#<LF>
1	Iot device identification code (fixed 8A)
2	Iot device hardware version 110->V1.1.0
3	Scooter controller hardware version 1101->V1.0.1
	<1> <2> <3> <4> <5>
server->lock	*SCOS,OM,123456789123456,U0,220,12345,32434,8A,C7qn#<LF>
1	Total number of upgrade data packages (packaged by 1-128 bytes per packet)
2	Total length of upgrade data (unit: Byte)
3	Upgrade data total CRC16 check value, in decimal form
4	Device ID corresponding to the upgrade file (8A->Omni Iot device 20-> controller)
5	Upgrade key (C7qn)

Note: When the server determines that there is no need to upgrade, it does not need to respond to the U0 command. The service can also send U0 to initiate the upgrade when the lock does not upload U0.

1.3.17 U1（Get upgrade data）

	<1><2>
lock->server	*SCOR,OM,123456789123456,U1,100,8A#<LF>
1	How many packages to get the upgrade file (starting at 0)
2	Obtain the device identification code corresponding to the upgrade file (85->Iot device CT-> scooter controller)
	<1> <2> <3> <4>
server->lock	*SCOS,OM,123456789123456,U1,100,128,1234,DATA#<LF>
1	How many packages to get the upgrade file (starting at 0)
2	Data length per packet Range: 1-128Byte
3	CRC16 check value for each package upgrade data
4	Upgrade data (the corresponding data intercepted in the upgrade file, non-string form)

1.3.18 U2（Successfull notification of upgrade）

	<1><2>
lock->server	*SCOR,OM,123456789123456,U2, 8A,0#<LF>
1	Upgrade device ID (8A->Iot device 20->controller)
2	Upgrade result 0->success 1->fail
server->lock	No response

1.3.19 K0（Set / Get BLE 8-byte communication KEY）

	<1> <2>
server->lock	*SCOS,OM,123456789123456,K0,1,OmniW4GX#<LF>
1	Set or read the ID 0->Read 1->Set (When 0, the second item is empty, ’, ’Reserved)
2	BLE 8-byte communication KEY
	<1>
lock->server	*SCOR,OM,123456789123456,K0,OmniW4GX#<LF>
1	BLE 8-byte communication KEY

1.3.20 S1（Event notification command）

	<1>
server->lock	*SCOS,OM,123456789123456,S1,X#<LF>
1	Event code
	<1>
lock->server	*SCOR,OM,123456789123456,S1,X#<LF>
1	Event code

Note :event code list

Serial number	Event code	Event description
1	1	IOT turn off
2	2	IOT restart
3	10	The scooter is reserved
4	11	Cancel scooter reservation
5	12	Mark scooter fault
6	13	Cancel fault flag
7	16	Marked scooter lost
8	17	Cancel the lost scooter tag

1.3.21 L5（Unlock external devices）

	<1>
Server ->lock	*SCOS,OM,123456789123456,L5,1#<LF>
1	Operation 1-> Unlock battery lock 2-> Unlock wheel lock 3-> Unlock cable lock 17-> lock Battery lock 18-> lock Wheel lock 19-> lock Cable lock 33-> Get battery lock status 34-> Get wheel lock status 35-> Get cable lock status
	<1><2>
lock->server	*SCOR,OM,123456789123456,L5,1 ,0#<LF>
1	Operation 1-> Unlock battery lock 2-> Unlock wheel lock 3-> Unlock cable lock 17-> lock Battery lock 18-> lock Wheel lock 19-> lock Cable lock 33-> Get battery lock status 34-> Get wheel lock status 35-> Get cable lock status
2	Unlock result 0->success 1->failure 2-> communication timeout with equipment 16-> lock state 17-> unlock state

1.3.22 Z0（Get controller custom data）

	<1>
Server->lock	*SCOS,OM,123456789123456,Z0,1 #<LF>
1	1->Take the type of data (Type customization)
	<1><2> <3>
lock->server	*SCOR,OM,123456789123456,Z0,1,5,A0B1C2D3E4F5#<LF>
1	Take the type of data
2	Data length obtained
3	String form hexadecimal data

1.4 Extended instruction

1.4.1 I0（Get the SIM card ICCID number）

Server->lock	*SCOS,OM,123456789123456,I0#<LF>
	<1>
lock->server	*SCOR,OM,123456789123456,I0,123456789AB123456789#<LF>
1	SIM card ICCID (Generally 20 numbers)

1.4.2 M0（Get IOT Bluetooth MAC Address）

server->lock	*SCOS,OM,123456789123456,M0#<LF>
	<1>
lock->server	*SCOR,OM,123456789123456,M0,12:34:56:78:90:AB#<LF>
1	MAC adress

2. BLE communication protocol description

2.1 Instruction format

Byte	Item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	Data length (length of DATA)
3	RAND	Random number, generated by the data sender, used to encrypt
4	KEY_once	The communication key is randomly generated by the lock
5	CMD	Command word
6	DATA	data
6+LEN	CRC	Encrypted CRC8 check value before CRC

Note: greater than 2Byte data high byte before

2.2 Data encryption process

Encrypted composition: random number, KEY。

Encryption process:

1、Generate random number RAND

2、Generate random number variants RAND_1 = RAND + 0x32

3、Fill RAND_1 into the third byte of the data.

4、After using RAND to XOR (^) RAND, the plaintext data before the CRC and the result corresponding backfill

5、Perform CRC8 check on the data before CRC, and fill in the check value to the CRC position.

Bluetooth encryption, decryption reference Appendix I.

2.3 APP and IoT communication process

APP establishes Bluetooth connection with IoT

1. APP sends a (0x01) command to the IoT to obtain the communication key KEY

2. The IoT returns the communication key KEY, and the APP needs to save the secret key for subsequent communication.

3. APP communicates with the IoT

Note: The key KEY is re-acquired only when the APP establishes a Bluetooth connection with the IoT, and the communication remains unchanged after that.

2.4 UUID used by the lock

Service UUID :6e400001-b5a3-f393-e0a9-e50e24dcca9e

Characteristic under the service

characteristic UUID	Operation type	Description
6e400002-b5a3-f393-e0a9-e50e24dcca9e	Write	Write commands to the hardware
6e400003-b5a3-f393-e0a9-e50e24dcca9e	Notify	Information returned by the hardware

Note: When you register for notifications, you need to use the UUID of the character under the character: 00002902-0000-1000-8000-00805f9b34fb， IOS does not need to.

2.5 Lock command details and examples

2.5.1 Verify device KEY to get communication KEY command (0x01)

APP->lock

When the App is connected to the Bluetooth device, the device KEY is first verified by the 0x01 command to obtain the KEY for communication with the Bluetooth device. The device KEY of each IoT is different, and the user can also define the device KEY by himself. When the connected Bluetooth device does not send the 0x01 command within 5 seconds, or the authentication pairing password is incorrect, the Bluetooth device will automatically disconnect from the app.

Byte	Item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x08
3	RAND	random number
4	KEY	0x00
5	CMD	0x01
6-13	DATA	Device KEY, 8 bytes (yOTmK50z)
14	CRC	Encrypted CRC8 check value before CRC

If the user is connected to Bluetooth, the device KEY is [0x4F6D6E6957344758].
Therefore DATA[0]=0x4F, DATA[1]=0X6D, DATA[2]=0x6E,DATA[3]=0x69
DATA[4]=0x57,DATA[5]=0X34,DATA[6]=0x47,DATA[7]=0x58

lock->APP

After receiving the communication KEY command, the IoT returns the KEY for communication in DATA. This obtained communication KEY is valid for this connection.

Byte	Item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x02
3	RAND	random number 0x00
4	KEY	Communication key 0x00 Populate the current KEY
5	CMD	0x01
6	DATA	Verification ID: (1: Success, 0: Failure)
7	DATA	Communication key (KEY) for current communication KEY
8	CRC	Encrypted CRC8 check value before CRC

2.5.2 Instruction error prompt instruction (0x10)

IoT->APP

After the user connects to the device, if the KEY of the communication is not obtained through the 0x01 command, the user directly sends an instruction such as 0x12, 0x13 to communicate with the device. Then the device prompts the user with the instruction. The user has acquired the communication KEY, but when sending other instructions, the communication KEY used is incorrect, and the instruction is also prompted.

Byte	Item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x01
3	RAND	random number
4	KEY	Communication key 0x00 or current KEY
5	CMD	0x10
6	DATA	Error message: 1: CRC authentication error 2: The communication KEY was not obtained 3: The communication KEY has been obtained, but the communication KEY is wrong
7	CRC	CRC8 check value after data encryption before CRC

2.5.3 Unlock command (0x05)

APP->lock

Byte	Item	description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x0A
3	RAND	random number
4	KEY	Communication key
5	CMD	0x05
6	DATA	0x01 (Control instruction)
7-10	DATA	User ID
11-14	DATA	Operation timestamp
15	DATA	Status 0-> Normal unlock 0xA0-> Do not reset riding time when unlock
16	CRC	CRC8 check value after data encryption before CRC

lock-> APP

Byte	Item	description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x05
3	RAND	random number
4	KEY	Communication key
5	CMD	0x05
6	DATA	Control result (1: Success 2: Fail / Timeout)
7-10	DATA	Operation timestamp
11	CRC	Encrypted CRC8 check value before CRC

APP->lock

After the APP receives the operation status command issued by the lock (1: success, 2: failure/timeout), the vehicle locks the lock. If it does not reply, the lock is uploaded through the network (L0 command).

Byte	item	description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x01
3	RAND	random number
4	KEY	Communication key
5	CMD	0x05
6	DATA	0x02 (Reply instruction)
7	CRC	Encrypted CRC8 check value before CRC

2.5.4 Lock command (0x15)

APP->lock

Byte	Item	description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x01
3	RAND	random number
4	KEY	Communication key
5	CMD	0x15
6	DATA	0x01 (Control instruction)
7	CRC	CRC The encrypted data of the previous CRC8 check value

lock->APP

Byte	Item	description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x09
3	RAND	random number
4	KEY	Communication key
5	CMD	0x15
6	DATA	Control result (1: Success 2: Fail / Timeout)
7-10	DATA	Unlock timestamp [unlock sequence]
11-14	DATA	Unlocking time
15	CRC	Encrypted CRC8 check value before CRC

APP->lock

After the APP receives the parking state command issued by the IoT (1: the operation is successful, 2 the operation timeout), the IoT is restored, and if it does not reply, the IoT uploads the operation state through the network (L1 command).

Byte	Item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x01

3	RAND	Randam number
4	KEY	Communication key
5	CMD	0x15
6	DATA	0x01 (Reply instruction)
7	CRC	Encrypted CRC8 check value before CRC

2.5.5 Query lock information (0x31)

APP->lock

You can get information about the lock switch status, battery level, and old data.

Byte	Item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x01
3	RAND	Randam number
4	KEY	Communication key
5	CMD	0x31
6	DATA	0x01 (Control instruction)
7	CRC	Encrypted CRC8 check value (0-5) before CRC

lock->APP

After the device receives the query status, it will return the current switch status, battery power, and old data in DATA.。

Byte	Item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x07
3	RAND	Random number
4	KEY	Communication key
5	CMD	0x31
6	DATA	Battery voltage (mV) H
7		Battery voltage (mV) L
8		Device status 1
9		0(Reserved)
10	DATA	Major version number
11	DATA	Minor version number
12	DATA	Version revisions
13	CRC	Encrypted CRC8 check value before CRC

Device status 1[0-7] BIT bit identification details.

BTT bit	Bit details	Value status	Value status (0)
0	unlock	Unlocked state	invalid
1	Lock	Locked state	invalid
2	No content	---	--
3	No content	--	--
4	No content	--	--
5	No content	--	--
6	Old usage	Unlocked state	invalid
7	No content	Locked state	invalid

2.5.6 Get the last usage data (0x51)

Note: This data is the return data of the lock that has passed the BLE or failed to upload via TCP after receiving the lock command, including the user ID and the time of the vehicle, for the settlement fee.

APP->lock

When the App obtains the lock information, if it finds that the old data is used, the old data can be obtained and uploaded to the server.

Byte	Item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x01
3	RAND	Random number
4	KEY	Communication key
5	CMD	0x51
6	DATA	0x01 (Control instruction)
7	CRC	Encrypted CRC8 check value before CRC

lock->APP

Used data, that is, data that was not uploaded to the server after the last lockout, including timestamp, duration, user ID.

Byte	Item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x0C
2	RAND	Random number
3	KEY	Communication key
4	CMD	0x51
5-8	DATA	Unlock timestamp
9-12	DATA	usage time Unit: second
13-16	DATA	User ID
17	CRC	Encrypted CRC8 check value before CRC

2.5.7 Clear usage data in the lock (0x52)

APP->lock

After the app uploads the used data to the server, it can erase the used data stored in the device.

Byte	Item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x01
3	RAND	random number
4	KEY	Communication key
5	CMD	0x52
6	DATA	0x01 (Control instruction)
7	CRC	Encrypted CRC8 check value before CRC

lock->APP

Byte	Item	Description
0-1	STX	According to the header/frame header Fixed value: 0xA3A4
2	LEN	0x01
3	RAND	random number
4	KEY	Communication key
5	CMD	0x52
5	DATA	Return value: 0: Success 1: Failure
6	CRC	Encrypted CRC8 check value before CRC

2.5.8 Get scooter information (0x60)

APP->lock

Byte	Item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x01
3	RAND	random number
4	KEY	Communication key
5	CMD	0x60
6	DATA	0x01 (Control instruction)
7	CRC	Encrypted CRC8 check value before CRC

lock->APP

Byte	Item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x08
3	RAND	random number
4	KEY	Communication key
5	CMD	0x60
6	DATA	The current power of the scooter is 80->80%
7		Current mode of scooter 1: low speed 2: medium speed 3:High speed
8-9		Current speed 221->22.1km/h
10-11		Single riding mileage, Unit 10m
12-13		Estimated remaining mileage, unit 10m
14	CRC	Encrypted CRC8 check value before CRC

2.5.9 Set scooter (0x61)

APP->lock

Byte	Item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x04
3	RAND	random number
4	KEY	Communication key
5	CMD	0x61
6	DATA	Headlight switch 0: Invalid (not set) 1: Off 2: On
7		Mode setting 0: Invalid (not set) 1: Low speed 2: Medium 3: High
8		Throttle response 0: Invalid (not set) 1: Off 2: On
9		Taillight flashing 0: Invalid (not set) 1: Off 2: On
10	CRC	Encrypted CRC8 check value before CRC

Lock->APP

Byte	Item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x01
3	RAND	random number
4	KEY	Communication key
5	CMD	0x61
6	DATA	Return value: 0: Success 1:failure
7	CRC	Encrypted CRC8 check value before CRC

2.5.11 Scooter setting 2 (0x62)

Note: The following setting information can be set to save the original value after power off

APP-> IOT

Byte	Item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x06
3	RAND	random number
4	KEY	Communication key
5	CMD	0x62
6	DATA	0: The following settings are not saved, and the original settings are restored after the switch is locked 1: Save the following settings, and maintain the settings after power off
7		Fixed speed cruise setting 0: invalid (not set) 1: off 2: on
8		Start mode setting 0: invalid (not set) 1: non-zero start 2: zero start
9		Low speed mode speed limit value 0: invalid (not set) Range: 6-25km / h
10		Medium speed mode speed limit value 0: invalid (not set) Range: 6-25km / h
11		High speed mode speed limit value 0: invalid (not set) Range: 6-25km / h
12	CRC	CRC8 check value after data encryption before CRC

IOT->APP

byte	item	Description
0-1	STX	Data header / frame header Fixed value: 0xA3A4
2	LEN	0x01
3	RAND	random number
4	KEY	Communication key
5	CMD	0x62
6	DATA	Return value: 0: success 1: failure
7	CRC	CRC8 check value after data encryption before CRC

2.5.12 Unlock external devices (0x81)

APP->lock

byte	item	Description
0-1	STX	Data header/frame header Fixed value: 0xA3A4
2	LEN	0x01
3	RAND	random number
4	KEY	Communication key
5	CMD	0x81
6	DATA	Operation 0x01-> Battery lock unlock 0x02-> Wheel lock unlock 0x03-> Cable lock unlock 0x11-> Battery lock lock 0x12-> Wheel lock lock 0x13-> Cable lock lock 0x21-> Get battery lock status 0x22-> Get wheel lock status 0x23-> Get cable lock status
7	CRC	Encrypted CRC8 check value before CRC

lock->APP

byte	Item	Description
0-1	STX	Encrypted CRC8 check value before CRC
2	LEN	0x02
3	RAND	random number
4	KEY	Communication key
5	CMD	0x81
6	DATA	Operation 0x01-> Battery lock unlock 0x02-> Wheel lock unlock 0x03-> Cable lock unlock 0x11-> Battery lock lock 0x12-> Wheel lock lock 0x13-> Cable lock lock 0x21-> Get battery lock status 0x22-> Get wheel lock status 0x23-> Get cable lock status
7		0x00-> success 0x01-> failure 0x02-> communication timeout with the device 0x10-> locked state 0x11-> unlocked state
8	CRC	Encrypted CRC8 check value before CRC

Appendix I: Bluetooth encryption， Decryption process

1. Encryption: Take the operation KEY and 0x01 instructions to the lock as an example. 4F6D6E6957344758

item	index	hex(original)	hex(+0x32)	hex(xor 34)	calc CRC
STX	0	A3	A3	A3	A3
STX	1	A4	A4	A4	A4
len	2	08	08	08	08
rand	3	1E	50	50	50
key	4	00	0	1E	1E
cmd	5	01	01	1F	1F
data	6	4F	4F	51	51
data	7	6D	6D	73	73
data	8	6E	6E	70	70
data	9	69	69	77	77
data	10	57	57	49	49
data	11	34	34	2A	2A
data	12	47	47	59	59
data	13	58	58	46	46
crc	14				01

2. Decryption: Take the lock to the app to return the KEY as an example.

item	index	hex	step1	step2	Step3	Step4	Step5

stx	0	**A3**	**A3**		A3	A3	A3
stx	1	**A4**	**A4**		A4	A4	A4
len	2	**2**		**2**	2	2	2
rand	3	**B0**			B0	B0	7E(B0-32)
key	4	**27**			27	27	59(27^7E)
cmd	5	**7F**			7F	7F	01(7F^7E)
data (f)	7	**7F**			7F	7F	01(7F^7E)
data (key)	8	**27**			27	27	59(27^7E)
crc	9	**1A**			1A	1A	
?	8	**0(IF HAVE)**					
?	9	**A(IF HAVE)**					
?	10	**B(IF HAVE)**					
?	11	**C(IF HAVE)**					
?	12	**D(IF HAVE)**					
?	13	**E(IF HAVE)**					

Step1:Find the subscript address of A3 A4

Step2:Find the location of len

Step3:According to the starting position and length of the instruction, get the instruction

Step4:Detect CRC value

Step6::Parse the instruction to get the required value.

Appendix II　Bluetooth broadcast data

Manufacturer　data shows

Eg：

0xFFFFD713315DDBF68500290037

Data	Description
FFFF	ID
D713315DDBF6	MAC address: D7:13:31:5D:DB:F6
85	Device type low byte
00	Device type High type
29	IOT operating voltage 0x29->41->4.1V
00	Lock status 0->Off lock status 1->Unlock status
37	Scooter power 0x37->55->55%

AppendixⅢ:

CRC8 calculation code（C code as an example）

```
unsigned char CRC8Table[]={
  0, 94, 188, 226, 97, 63, 221, 131, 194, 156, 126, 32, 163, 253, 31, 65,
  157, 195, 33, 127, 252, 162, 64, 30, 95, 1, 227, 189, 62, 96, 130, 220,
  35, 125, 159, 193, 66, 28, 254, 160, 225, 191, 93, 3, 128, 222, 60, 98,
  190, 224, 2, 92, 223, 129, 99, 61, 124, 34, 192, 158, 29, 67, 161, 255,
  70, 24, 250, 164, 39, 121, 155, 197, 132, 218, 56, 102, 229, 187, 89, 7,
  219, 133, 103, 57, 186, 228, 6, 88, 25, 71, 165, 251, 120, 38, 196, 154,
  101, 59, 217, 135, 4, 90, 184, 230, 167, 249, 27, 69, 198, 152, 122, 36,
  248, 166, 68, 26, 153, 199, 37, 123, 58, 100, 134, 216, 91, 5, 231, 185,
  140, 210, 48, 110, 237, 179, 81, 15, 78, 16, 242, 172, 47, 113, 147, 205,
  17, 79, 173, 243, 112, 46, 204, 146, 211, 141, 111, 49, 178, 236, 14, 80,
  175, 241, 19, 77, 206, 144, 114, 44, 109, 51, 209, 143, 12, 82, 176, 238,
  50, 108, 142, 208, 83, 13, 239, 177, 240, 174, 76, 18, 145, 207, 45, 115,
  202, 148, 118, 40, 171, 245, 23, 73, 8, 86, 180, 234, 105, 55, 213, 139,
  87, 9, 235, 181, 54, 104, 138, 212, 149, 203, 41, 119, 244, 170, 72, 22,
  233, 183, 85, 11, 136, 214, 52, 106, 43, 117, 151, 201, 74, 20, 246, 168,
  116, 42, 200, 150, 21, 75, 169, 247, 182, 232, 10, 84, 215, 137, 107, 53};
unsigned char CRC8_Table(unsigned char *pucFrame, char usLen)
{
    unsigned char crc8 = 0;
            while(usLen--)
                crc8 = CRC8Table[crc8^*(pucFrame++)];
    return(crc8);
}
```

Appendix Ⅳ:

CRC16 calculation code（C code as an example）

```
unsigned char CRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
};
```

```
 unsigned char CRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07,
      0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F,
      0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08,
      0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A, 0x1E,
      0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4, 0xD5,
      0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3, 0x11,
      0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2,
      0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C,
      0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B,
      0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29, 0xEB,
      0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED, 0xEC,
      0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26, 0x22,
      0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61,
      0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5,
      0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E,
      0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68, 0x78,
      0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E, 0x7F,
      0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5, 0x77,
      0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70,
      0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96,
      0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C, 0x5D,
      0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B, 0x99,
      0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B, 0x8A,
      0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C, 0x44,
      0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43,
      0x83, 0x41, 0x81, 0x80, 0x40
};

 unsigned int CRC16( unsigned char * pucFrame, unsigned int usLen)
{
        unsigned char  ucCRCHi = 0xFF;
        unsigned char  ucCRCLo = 0xFF;
        unsigned int iIndex=0x0000;

        while(usLen--)
        {
                iIndex = ucCRCLo ^ *(pucFrame++); ucCRCLo =
                ucCRCHi ^ CRCHi[iIndex]; ucCRCHi = CRCLo[iIndex];
                                        }
              return (unsigned int)(ucCRCHi << 8 | ucCRCLo);
}
```