| | |
|---|---|
| **Type** | For customer |
| **Date** | 2017/03/13 |
| **Versi** | Ver : 01.04 |
| **Dep** | Engineering |

# Share Bike GPS+GPRS(AC)+BLE Smart Lock(Hardware)
# Air interface Protocol

Revision history:

| **Revisio** | **history:** | **Revision** | **history** |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# Content

**System overall framework and protocol description** ...................... 错误！未定义书签

1.1 Purpose of the documentation ...................................................错误！未定义书签。

1.2 Sharing Bicycle lock system black diagram..........................................................................3

1.3 Communication packet format ..........................................................................3

1.4 Data encryption process ..........................................................................3

1.5 Process for APP communication with bicycle lock ..........................................................4

1.6 UUID used for Bluetooth communication ..........................................................4

1.7Command Details and Example ..........................................................................4

1.7.1 Obtain unlocking KEY instruction (0x11) ..........................................................4

1.7.2 Unlock command (0x21) ..........................................................................5

1.7.3 Locking command (0x22) ..........................................................................6

1.7.4 Query lock state (0x31) ..........................................................................7

1.7.5 Obtain not uploaded data (0x51) ..........................................................................7

1.7.6 Clearing uploaded data in Car lock (0x52) ..........................................................8

2.0 TCP TCP Communication Protocol Description ..........................................................9

2.1 TCP Command List ..........................................................................9

2.1.1 Unlocking Command (L0) ..........................................................................9

2.1.2 Locking Command (L1) ..........................................................................11

2.1.3 Positioning Command (D0) ..........................................................................12

2.1.4 Signing in Command (Q0) ..........................................................................14

2.1.5 Heatbeat Command (H0) ..........................................................................14

2.1.6 Obtain information of Lock Command (S5) ..........................................................15

2.1.7 Ringing for finding a bike Command (S8) ..........................................................16

2.1.8 Firmware version Command (G0) ..........................................................................17

2.1.9 Alarming Command (W0) ..........................................................................18

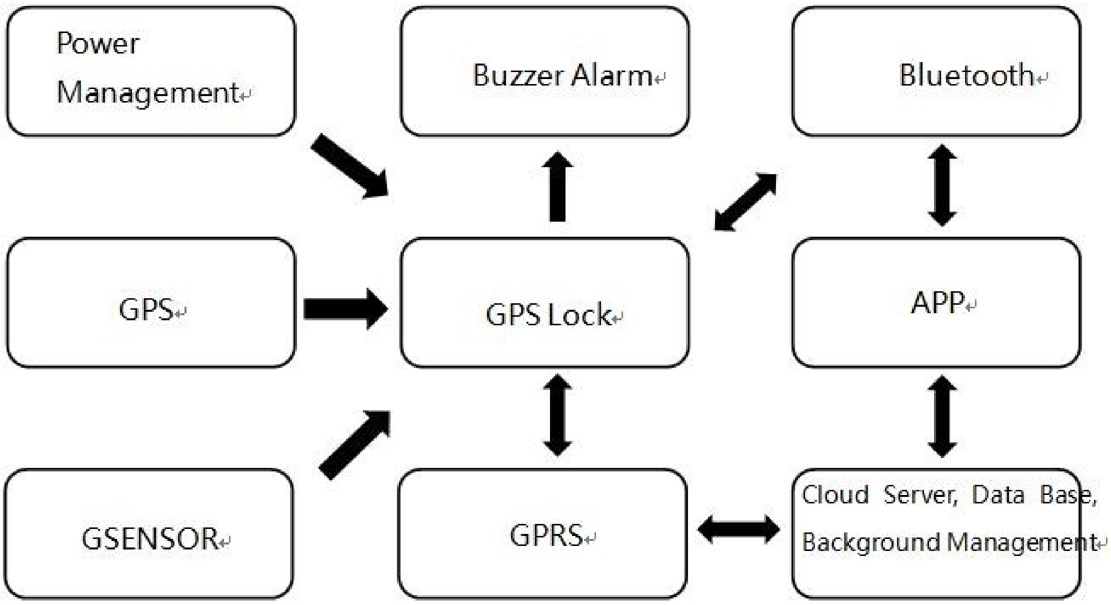
2.1.10 Obtain SIM card ICCID number Command (I0) ..........................................................18

# System overall framework and protocol description

## 1.1 Purpose

The document is a illustration about the communication protocol between smart bike lock and Server or APP designed and manufactured by Omni.

## 1.2 Items Functions Framework



## 1.3 Communication package format

| Byte | Item | Instructions |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: 0xFE |
| 1 | NUM | Random Number, is generated only when data sending, for encrypting |
| 2 | KEY | Communicate secret key, generated randomly by bike lock, APP receives the KEY by（0x11）command |
| 3 | CMD | Command byte |
| 4 | LEN | Data length |
| 5 | DATA | Data |
| 5+LEN | CRC | CRC16 checking value, the Data before CRC after being encrypted |
| 6+LEN | | |

## 1.4 Data encryption processing

Encryption consistent: random num, KEY.

Encryption processing:

1. Generate random NUM
2. Generate random variant NUM_1 = NUM + 0x32
3. Fill NUM_1 into the First byte of Data
4. Use NUM xor(^) after NUM， before CRC plaintext data, and fill the results

corresponding

5. Take the data before CRC to do CRC16 Checking, checking value files into CRC position.

For example -> unlock command(random NUM is 0x88 ; KEY is 0x66, uid is 0x01) **RED** part are CRC value.

| | |
|---|---|
| plaintext | 0xFE,0x88,0x66,0x21,0x04,0x00,0x00,0x00,0x01 |
| ciphertext | 0xFE,0xBA,0xEE,0xA9,0x8C,0x88,0x88,0x88,0x89,**0x5B,0xC2** |

## 1.5 Communications process between APP and bike lock

1.APP is connecting to bike lock by bluetooth.
2.APP is sending command（0x11）to bike lock for getting communication secret key.
3.Bike Lock returns to communication secret KEY, APP needs to save KEY for next communication
4.APP is building communication to bike lock.

**Note: KEY is only regenerated when APP is building communication to bike lock，communication is kept from now on.**

## 1.6 UUID used for Bluetooth communication

Service UUID :0783b03e-8535-b5a0-7140-a304d2495cb7

The characteristic under the service：

| characteristic UUID | Operation Type | Description |
|---|---|---|
| 0783b03e-8535-b5a0-7140-a304d2495cba | Write | Write instructions to the hardware |
| 0783b03e-8535-b5a0-7140-a304d2495cb8 | Notify | Hardware return information |

When register the notification, need to use the descriptor UUID under characteristic: 00002902-0000-1000-8000-00805f9b34fb

## 1.7 Command illustration &Example

### 1.7.1 Obtain unlock KEY Command (0x11)

#### 1.7.1.1 APP-> Bike Lock

App connected to the Bluetooth device, first to get to communicate with the Bluetooth device KEY

The device identification KEY in the data, is the only identification of the Bluetooth device. When connected to the Bluetooth device in 5 seconds without sending the command to obtain the communication KEY, or sending Error identification KEY for the device, the Bluetooth device will automatically disconnect with the app.

| Byte | Items | Instructions |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: 0xFE |
| 1 | NUM | Random Number |
| 2 | KEY | 0x00 |
| 3 | CMD | 0x11 |
| 4 | LEN | 0x08 |
| 5-12 | DATA | Device Identification KEY, 8 Bytes（Default data is ”yOTmK50z”） |
| 13 | CRC | CRC16 checking value, the Data before CRC after being encrypted (0-12) |
| 14 | | |

### 1.7.1.2 Bike lock ->APP

After receiving the communication KEY instruction, the device returns the KEY for communication in DATA

| Byte | Items | Instructions |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: 0xFE |
| 1 | NUM | Random Number |
| 2 | KEY | communication secret key |
| 3 | CMD | 0x11 |
| 4 | LEN | 0x01 |
| 5 | DATA | Secret KEY, KEY for communication |
| 6 | CRC | CRC16 checking value, the Data before CRC after being encrypted (0-5) |
| 7 | | |

## 1.7.2 Unlock command (0x21)

### 1.7.2.1 APP-> bike lock

When unlocking, the user ID(4 bytes) and time stamp(4 bytes) are sent in the data.

| Byte | Items | Instructions |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: 0xFE |
| 1 | NUM | Random Number |
| 2 | KEY | communication secret key |
| 3 | CMD | 0x21 |
| 4 | LEN | 0x08 |
| 5-8 | DATA | APP user' ID Number |
| 9-12 | DATA | Unlock time stamp(4 bytes), high in front |
| 13 | CRC | CRC16 checking value, the Data before CRC after being encrypted |
| 14 | | |

### 1.7.2.2 APP->bike lock

After the device performs an unlock action, it returns an unlocked state in DATA.

| Byte | Items | Instructions |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: 0xFE |
| 1 | NUM | Random Number |
| 2 | KEY | communication secret key |
| 3 | CMD | 0x21 |
| 4 | LEN | 0x05 |
| 5 | DATA | Return value 0: unlock succeed 1: unlock failed |
| 6-9 | DATA | Unlock time stamp(4 bytes), high in front |
| 10 | CRC | CRC16 checking value, the Data before CRC after being encrypted |
| 11 | | |

### 1.7.2.3 APP-> bike lock

APP receives unlock command sent from lock and then respond to lock.

| Byte | Item | Instructions |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: 0xFE |
| 1 | NUM | Random Number |
| 2 | KEY | communication secret key |
| 3 | CMD | 0x21 |
| 4 | LEN | 0x00 |
| 6 | CRC | CRC16 checking value, the Data before CRC after being encrypted |
| 7 | | |

## 1.7.3 lock command (0x22)

### 1.7.3.1 Bike lock ->APP

After the device is locked, it upload the upload status and ride time in DATA

| Byte | Item | Instructions |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: 0xFE |
| 1 | NUM | Random Number |
| 2 | KEY | communication secret key |
| 3 | CMD | 0x22 |
| 4 | LEN | 0x09 |
| 5 | DATA | 0: lock succeed 1: Lock failed |
| 6-9 | DATA | Unlock time stamp(4 bytes) |
| 10-13 | DATA | Riding time (4bytes)(Unit: Minute) |
| 14 | CRC | CRC16 checking value, the Data before CRC after being encrypted |
| 15 | | |

### 1.7.3.2 APP->Bike lock

App received a lock and reply, if not reply will turn out the old data.

| Byte | Item | Instructions |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: 0xFE |
| 1 | NUM | Random Number |

|   |   |   |
|---|---|---|
| 2 | KEY | communication secret key |
| 3 | CMD | 0x22 |
| 4 | LEN | 0x00 |
| 5 | CRC | CRC16 checking value, the Data before CRC after being encrypted |
| 6 |  |  |

## 1.7.4 Checking bike lock status (0x31)

### 1.7.4.1 APP-> Bike lock

Can obtain lock status like lock or unlock, battery power, whether there is old data, etc..

| Byte | Item | Instructions |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: 0xFE |
| 1 | NUM | Random Number |
| 2 | KEY | communication secret key |
| 3 | CMD | 0x31 |
| 4 | LEN | 0x00 |
| 5 | CRC | CRC16 checking value, the Data before CRC after being encrypted |
| 6 |  |  |

### 1.7.4.2 Bike lock ->APP

When the device receives the query status, it will return the current bike lock status, like lock/unlock, battery power, and whether there is old data.

| Byte | Item | Instructions |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: 0xFE |
| 1 | NUM | Random Number |
| 2 | KEY | communication secret key |
| 3 | CMD | 0x31 |
| 4 | LEN | 0x07 |
| 5 | DATA | bike lock status 0: unlock 1: lock |
| 6 |  | Battery volume e.g: 37（decimalism） represents 3.7V |
| 7 |  | Whether un-upload data 0: Yes 1: no |
| 8-11 |  | Time stamp 4 bytes |
| 12 | CRC | CRC16 checking value, the Data before CRC after being encrypted |
| 13 |  |  |

## 1.7.5 Obtain un-uploaded Data (0x51)

Note: This is the data which un-uploaded to APP or Server, including user ID and using time for payment.

### 1.7.5.1 APP->bike lock

When App is obtaining lock status, if there is old data detected, it will upload the old data to the server.

| Byte | Item | Instructions |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: 0xFE |
| 1 | NUM | Random Number |
| 2 | KEY | communication secret key |
| 3 | CMD | 0x51 |
| 4 | LEN | 0x00 |
| 5 | CRC | CRC16 checking value, the Data before CRC after being encrypted |
| 6 | | |

### 1.7.5.2 Bike lock->APP

The old data means data that was not uploaded to the server after the last ride, including unlock time stamp, ride time, user ID.

| Byte | Item | Instructions |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: 0xFE |
| 1 | NUM | Random Number |
| 2 | KEY | communication secret key |
| 3 | CMD | 0x51 |
| 4 | LEN | 0x012 |
| 5-8 | DATA | Time stamp used for user unlocking the lock |
| 9-12 | DATA | User riding time.　Unit: Minute |
| 13-16 | DATA | User ID |
| 17 | CRC | CRC16 checking value, the Data before CRC after being encrypted |
| 18 | | |

## 1.7.6 Clear the unuploaded data in the lock (0x52)

### 1.7.6.1 APP-> bike lock

After App upload old data to the server, it can clear the old data stored in the device.

| Byte | Item | Instructions |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: 0xFE |
| 1 | NUM | Random Number |
| 2 | KEY | communication secret key |
| 3 | CMD | 0x52 |
| 4 | LEN | 0x00 |
| 5 | CRC | CRC16 checking value, the Data before CRC after being encrypted |
| 6 | | |

### 1.7.6.2 Bike lock ->APP

The device returns the clear status of the old data in DATA

| Byte | Item | Instructions |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: 0xFE |

| | | |
|---|---|---|
| 1 | NUM | Random Number |
| 2 | KEY | communication secret key |
| 3 | CMD | 0x52 |
| 4 | LEN | 0x01 |
| 5 | DATA | Return value 0: unlock succeed 1: unlock failed |
| 6 | CRC | CRC16 checking value, the Data before CRC after being encrypted |
| 7 | | |

# 2.0 TCP Communication protocol description

## 2.1 TCP List of instructions

1 Unlock command (L0)
2 Lock command (L1)
3 Positioning command (D0)
4 Sign-in command (Q0)
5 Heartbeat command (H0)
6 Obtain lock status Command (S5)
7 Ringing command (S8)
8 Firmware query Command (G0)
8 Alarming Command (W0)
9 Obtain SIM card and ICCID number command (I0)

### 2.1.1 Unlock command (L0)

#### 2.1.1.1 Server –> Lock

| Item# | Item | Instruction |
|---|---|---|
| | HEX | Two bytes of reserved bits： 0xFF,0xFF |
| 0 | STX | Data header/frame header, fixed value: *CMDS |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command Code: L0 |
| 9 | SPACER | Spacer: , |
| 10 | VALU | Unlock(0) |
| 11 | SPACER | Spacer: , |
| 12 | VALU | User ID |

| | | |
|---|---|---|
| 13 | SPACER | Spacer: , |
| 14 | VALU | Unlock time stamp(Time stamp by second(s)，10 bytes) |
| 15 | END | Command end Code: #<LF> |

Example: *CMDS,OM,863725031194523,170619195455,L0,0,20,1497689816#<LF>

### 2.1.1.2 Lock->Server

| Item# | Item | Instruction |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: *CMDR |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command Code: L0 |
| 9 | SPACER | Spacer: , |
| 10 | VALU | Unlock (0) /Lock(1) |
| 11 | SPACER | Spacer: , |
| 12 | VALU | User ID |
| 13 | SPACER | Spacer: , |
| 14 | VALU | Unlock time stamp(Time stamp by second(s)，10 bytes) |
| 15 | END | Command end Code: #<LF> |

Example: *CMDR,OM,863725031194523,001497689816,L0,0,20,1497689816#<LF>

### 2.1.1.3 Server response

| Item# | Item | Instruction |
|---|---|---|
| | HEX | Two bytes of reserved bits： 0xFF,0xFF |
| 0 | STX | Data header/frame header, fixed value: *CMDS |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command code:Re |
| 9 | SPACER | Spacer: , |

| | | |
|---|---|---|
| 10 | VALU | The server responds with the corresponding Command code L0 |
| 11 | END | Command end Code: #<LF> |

Example: *CMDS,OM,863725031194523,001497689816,Re,L0#<LF>

## 2.1.2 Lock Command (L1)

### 2.1.2.1 Lock-> Server

| Item# | Item | Instruction |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: *CMDR |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command Code:L1 |
| 9 | SPACER | Spacer: , |
| 10 | VALU | User ID, 4 bytes, Here use the string form |
| 11 | SPACER | Spacer: , |
| 12 | VALU | Unlocking time stamp |
| 13 | SPACER | Spacer: , |
| 14 | VALU | Riding time, 4 bytes of data, here use the string form (unit: minutes) |
| 15 | END | Command end code:#<LF> |

Example: *CMDR,OM,863158022988725,000000000000,L1,1,1497689816,20#<LF>

### 2.1.2.2 Server response

| Item# | Item | Instruction |
|---|---|---|
| | HEX | Two bytes of reserved bits： 0xFF,0xFF |
| 0 | STX | Data header/frame header, fixed value: *CMDS |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command code:Re |

| | | |
|---|---|---|
| 9 | SPACER | Spacer: , |
| 10 | VALU | The server responds with the corresponding Command code L1 |
| 11 | END | Command end Code: #<LF> |

Example: *CMDS,OM,863725031194523,000000000000,Re,L1#<LF>

## 2.1.3 Positioning Command (D0)

### 2.1.3.1 Server -> Lock

| Item# | Item | Instruction |
|---|---|---|
| | HEX | Two bytes of reserved bits： 0xFF,0xFF |
| 0 | STX | Data header/frame header, fixed value: *CMDS |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command code:D0 |
| 9 | END | Command end Code: #<LF> |

Example: *CMDS,OM,863725031194523,000000000000,D0#<LF>

### 2.1.3.2 Lock -> Server

| Item# | Item | Instruction |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: *CMDR |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command Code:D0 |
| 9 | VALU | Reserve data |
| 10 | SPACER | Spacer: , |
| 11 | VALU | GPS Position data: |
| 12 | END | Command End Code:#<LF> |

Example:

*CMDR,863725031194523,000000000000,

D0,0,124458.00,A,2237.75314,N,11408.62621,E,0.066,,151216,,,A#<LF>

Instructions:

0,<1>,<2>,<3>,<4>,<5>,<6>,<7>,<8>,<9>,< 10>,<11>,<12>#

<1> UTC Time, hhmmss(Hour/miniute/second)format

<2> Positioning status, A = active positioning, V = invalid positioning

<3> Latitude ddmm.mmmm (degrees) format (the front 0 will also be transmitted)

<4> Latitude hemisphere N (northern hemisphere) or S (southern hemisphere)

<5> Longitude dddmm.mmmm (degrees) format (the front 0 will also be transmitted)

<6> Longitude hemisphere E (longitude) or W (west)

<7> Ground rate (000.0 to 999.9, the front 0 will also be transmitted)

<8> Ground heading (000.0 ~ 359.9 degrees, with true North as the reference, the previous 0 will also be transmitted)

<9> UTC date, ddmmyy (day / month) format

<10> Magnetic declination (000.0 ~ 180.0 degrees, the front 0 will also be transmitted)

<11> Magnetic declination direction, E (east) or W (west)

<12> Mode indication (A = automatic positioning, D = differential, E = estimation, N = data is invalid)

**Note: May emerge the below invalid positioning format:**

*CMDR,863725031194523,000000000000, D0,**0,033724.00,V,,,,,,,120517,,,N#**

*CMDR,863725031194523,000000000000,D0,**0,062102.362,V,,,,,0.06,277.72,120517,,,N#**

### 2.1.3.2 Server Response

| Item# | Item | Instruction |
|---|---|---|
| | HEX | Two bytes of reserved bits： 0xFF,0xFF |
| 0 | STX | Data header/frame header, fixed value: *CMDS |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command code:Re |
| 9 | SPACER | Spacer: , |
| 10 | VALU | The server responds with the corresponding Command code D0 |
| 11 | END | Command end Code: #<LF> |

Example: *CMDS,OM,863725031194523,000000000000,Re,D0#<LF>

## 2.1.4 Sign-in Command (Q0)

### 2.1.4.1 Lock->Server

| Item# | Item | Instruction |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: *CMDR |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command code:Q0 |
| 9 | SPACER | Spacer: , |
| 10 | VALU | Battery volume |
| 11 | END | Command End Code:#<LF> |

Example: *CMDR,OM,863725031194523,000000000000,Q0,410#<LF>

## 2.1.5 Heartbeat Command (H0)

### 2.1.5.1 Lock-> Server

| Item# | Item | Instruction |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: *CMDR |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command code:H0 |
| 9 | SPACER | Spacer: , |
| 10 | VALU | Lock status(0- Unlocked, 1-Locked) |
| 11 | SPACER | Spacer: , |
| 12 | VALU | Battery volume |
| 13 | SPACER | Spacer: , |
| 14 | VALU | GSM Signal value |
| 15 | END | Command End code:#<LF> |

Example: *CMDR,OM,863158022988725,161201150000,H0,1,400,24#<LF>

## 2.1.6 Obtain lock status Command (S5)

### 2.1.6.1 Server –> lock

| Item# | Item | Instruction |
|---|---|---|
| | HEX | Two bytes of reserved bits： 0xFF,0xFF |
| 0 | STX | Data header/frame header, fixed value: *CMDS |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Two bytes of reserved bits： 0xFF,0xFF |
| 8 | CMD | Command code:S5 |
| 9 | END | Command end code:#<LF> |

Example: *CMDS,OM,863725031194523,000000000000,S5#<LF>

### 2.1.6.2 Lock -> Server

| Item# | Item | Instruction |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: *CMDR |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command code:H0 |
| 9 | SPACER | Spacer: , |
| 10 | VALU | Battery volume |
| 11 | SPACER | Spacer: , |
| 12 | VALU | GSM Signal value |
| 13 | SPACER | Spacer: , |
| 14 | VALU | Reserve Parameter |
| 15 | SPACER | Spacer: , |
| 16 | VALU | Lock status(0- Unlocked, 1-Locked) |
| 17 | SPACER | Spacer: , |

| | | |
|---|---|---|
| 18 | VALU | Reserve Parameter |
| 19 | END | Command End code:#<LF> |

Example: *CMDR,XX,863158022988725,000000000000,S5,410,31,00,1,0#<LF>

## 2.1.7 Ringing for finding a bike Command (S8)

### 2.1.7.1 Server -> Lock

| Item# | Item | Instruction |
|---|---|---|
| | HEX | Two bytes of reserved bits： 0xFF,0xFF |
| 0 | STX | Data header/frame header, fixed value: *CMDS |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command code:S8 |
| 9 | SPACER | Spacer: , |
| 10 | VALU | Seconds of Ringing |
| 11 | SPACER | Spacer: ,, |
| 12 | VALU | Reserve Parameter |
| 13 | END | Command End code:#<LF> |

Example: *CMDS,OM,863725031194523,000000000000,S8,5,0#<LF>

### 2.1.7.2 Lock -> Server

| Item# | Item | Instruction |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: *CMDR |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command code:S8 |
| 9 | SPACER | Spacer: , |
| 10 | VALU | Seconds of Ringing |

| | | |
|---|---|---|
| 11 | SPACER | Spacer: ,, |
| 12 | VALU | Reserve Parameter |
| 13 | END | Command End code:#<LF> |

Example: *CMDR,XX,863158022988725,000000000000,S8,05,0#<LF>

## 2.1.8 Query Firmware Version Command (G0)

### 2.1.8.1 Server -> Lock

| Item# | Item | Instruction |
|---|---|---|
| | HEX | Two bytes of reserved bits： 0xFF,0xFF |
| 0 | STX | Data header/frame header, fixed value: *CMDS |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command code:G0 |
| 9 | END | Command End code:#<LF> |

Example: *CMDS,XX,863158022988725,000000000000,G0#<LF>

### 2.1.8.2 Lock -> Server

| Item# | Item | Instruction |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: *CMDR |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command code:G0 |
| 9 | SPACER | Spacer: , |
| 10 | VALU | Version info. |
| 11 | SPACER | Spacer: , |
| 12 | VALU | Compile time |
| 13 | END | Command End Code:#<LF> |

Example: *CMDR,XX,863158022988725,000000000000,G0,V1.0,May 17 2017#<LF>

## 2.1.9 Alarming Command (W0)

### 2.1.9.1 Server -> Lock

| Item# | Item | Instruction |
|---|---|---|
| | HEX | Two bytes of reserved bits： 0xFF,0xFF |
| 0 | STX | Data header/frame header, fixed value: *CMDS |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command code:W0 |
| 9 | END | Command End code:#<LF> |

Example: *CMDS,XX,863158022988725,000000000000,W0#<LF>

### 2.1.9.2 Lock -> Server

| Item# | Item | Instruction |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: *CMDR |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command code:W0 |
| 9 | SPACER | Spacer: , |
| 10 | VALU | Alarming status |
| 11 | END | Command End Code:#<LF> |

Example: *CMDR,XX,863158022988725,000000000000,W0,1#<LF>

## 2.1.10 Obtain SIM card ICCID Code command (I0)

### 2.1.10.1 Server -> Lock

| Item# | Item | Instruction |
|---|---|---|

| | | |
|---|---|---|
| | HEX | Two bytes of reserved bits： 0xFF,0xFF |
| 0 | STX | Data header/frame header, fixed value: *CMDS |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command code:I0 |
| 9 | END | Command End code:#<LF> |

Example: *CMDS,XX,863158022988725,000000000000,I0#<LF>

### 2.1.10.2 Lock -> Server

| Item# | Item | Instruction |
|---|---|---|
| 0 | STX | Data header/frame header, fixed value: *CMDR |
| 1 | SPACER | Spacer: , |
| 2 | CODE | Device Code |
| 3 | SPACER | Spacer: , |
| 4 | IMEI | Device(Lock) IMEI number |
| 5 | SPACER | Spacer: , |
| 6 | TIME | Local Time-Year/Month/Day/Hour/Min/Sec : YYMMDDHHMMSS (Reserve data), Currently filled with: 000000000000 |
| 7 | SPACER | Spacer: , |
| 8 | CMD | Command code:W0 |
| 9 | SPACER | Spacer: , |
| 10 | VALU | ICCID Code |
| 11 | END | Command End Code:#<LF> |

Example: *CMDR,XX,863158022988725,000000000000,I0,800602B8001050000281#<LF>

Note：

* This Protocol is created based on customers' product function requirements.

* Partial details of this Protocol may be adjusted based on real operating situation.