# 设备4G通信协议

**版本说明**

|  |  |  |
| --- | --- | --- |
| 版本 | 修改 | 时间 |
| V4 | 在车位锁协议基础上修改，增加其他设备 | 2017-06-28 |
| V5 | 增加显示控制协议 | 2017-07-11 |

**一，基本规定**

传输方式：TCP协议，数据都为JSON格式

服务器地址：域名（IP地址），端口

JSON字段：

|  |  |  |
| --- | --- | --- |
| type | char | 设备类型 |
| sn | string | 设备识别码 |
| cmd | char | 命令号 |
| version | char | 设备版本号 |
| nonce | string | 随机数 |
| sign | string | 签名方法，AES128，sn和nonce拼接，为32位 |
| result | char | 应答值 |
| status | char | 设备状态 |
| qrdata | string | 二维码数据 |
| face | char | 显示界面编号 0～100 （V5++） |

**二，协议格式**

1. 上传心跳（间隔30秒）

发送方：设备

{

"type":1, //设备类型，1=车位锁、2=班车

"sn": "123456789", //设备编号，字符串，9位

"cmd": 1, //命令号，0～255

"version" : 1, //固件版本编号，0～255

"nonce":"43276876", //随机数，字符串，8位

"sign":"xxxxxxxxx" //签名：对sn、cmd、nonce进行AES128加密签名

}

应答方：服务器

{

"type":1, //设备类型

"sn": "123456789", //设备编号

"cmd": 1, //命令号

"result" : 0, //应答值：0=成功

"nonce":"43276876", //随机数

"sign":"xxxxxxxxx" //签名

}

1. 遥控开

发送方：服务器

{

"type":1, //设备类型

"sn": "123456789", //设备编号

"cmd": 2, //命令号

"nonce":"43276876", //随机数

"sign":"xxxxxxxxx" //签名

}

应答方：设备

{

"type":1, //设备类型

"sn": "123456789", //设备编号

"cmd": 2, //命令号

"result" : 0, //应答值：0=成功，1=失败

"nonce":"43276876", //随机数

"sign":"xxxxxxxxx" //签名

}

1. 查询状态

发送方：服务器

{

"type":1, //设备类型

"sn": "123456789", //设备编号

"cmd": 3, //命令号

"nonce":"43276876", //随机数

"sign":"xxxxxxxxx" //签名

}

应答方：设备

{

"type":1, //设备类型

"sn": "123456789", //设备编号

"cmd": 3, //命令号

"result" : 0, //应答值：0=成功，1=失败

"status" : 0, //状态：0=未知，1=关，2=开（无车），3=开（有车）

"nonce":"43276876", //随机数

"sign":"xxxxxxxxx" //签名

}

1. 上传状态（变化时）

发送方：设备

{

"type":1, //设备类型

"sn": "123456789", //设备编号

"cmd": 4, //命令号

"status" : 1, //当前状态

"nonce":"43276876", //随机数

"sign":"xxxxxxxxx" //签名

}

应答方：服务器

{

"type":1, //设备类型

"sn": "123456789", //设备编号

"cmd": 4, //命令号

"result" : 0, //应答值：0=成功

"nonce":"43276876", //随机数

"sign":"xxxxxxxxx" //签名

}

1. 上传二维码数据（扫到时）

发送方：设备

{

"type":1, //设备类型

"sn": "123456789", //设备编号

"cmd": 5, //命令号

"qrdata" : "xxxxxxxxx" , //二维码数据：字符串

"nonce":"43276876", //随机数

"sign":"xxxxxxxxx" //签名

}

应答方：服务器

{

"type":1, //设备类型

"sn": "123456789", //设备编号

"cmd": 5, //命令号

"result" : 0, //应答值：0=成功

"nonce":"43276876", //随机数

"sign":"xxxxxxxxx" //签名

}

（6）控制屏显示 （V5++）

发送方：服务器

{

"type":1, //设备类型

"sn": "123456789", //设备编号

"cmd": 6, //命令号

"face": 1, //界面编号

"nonce":"43276876", //随机数

"sign":"xxxxxxxxx" //签名

}

应答方：设备

{

"type":1, //设备类型

"sn": "123456789", //设备编号

"cmd":6, //命令号

"result" : 0, //应答值：0=成功，1=失败

"nonce":"43276876", //随机数

"sign":"xxxxxxxxx" //签名

}



**（附件）**签名具体说明

1. 使用AES128算法，EBC模式，密钥（16字节）为

{0x59,0xDC,0x27,0x7F,0x08,0x3A,0x95,0x01,0xA4,0xBF,0x17,0x45,0xA5,0x69,0xE6,0x31}。

（2）使用sn和nonce数据拼接明文（16字节），sn取9字节，cmd取1字节，nonce取4字节，前面14字节计算CRC作为最后2字节。

（3）Sign为计算后的密文。

（4）例如：

{

"sn": "170627000",

"cmd": 1,

"nonce":"43276876",

"sign":"902D110A18FF69472C7820A4BAAA6092"

}

拼接sn&cmd&nonce（hex格式） = 3137303632373030300134333237

CRC-CCITT (XModem)计算（3137303632373030300134333237） = 7554

明文 = 31323334353637383902343332377554

说明：

SN取9字节，ASCII码转HEX码，如“12”=0x3132。

CMD取1字节，十进制转HEX码，如1=0x01。

NONCE取4字节，ASCII码转HEX码，如“12”=0x3132。

CRC占2字节，是前面14字节HEX数的CRC16结果值，高位在前。

密文 = CB6803E46DABC8D0A3FFFF2A295E8E7F