Contents

[OBD 终端在线升级通信协议 2](#_Toc325450420)

[一、数据协议暂时支持三种数据类型： 2](#_Toc325450421)

[二：消息类型的定义 3](#_Toc325450422)

[三：关于重复认证的需求： 3](#_Toc325450423)

[四：设备ERP需求 5](#_Toc325450424)

OBD 终端在线升级通信协议

## 一、数据协议暂时支持三种数据类型：

1：验证请求和数据请求，通过消息类型区分

2：验证通过，验证失败直接断开tcp连接即可，不需要处理

3：程序数据返回

通讯协议：TCP

数据验证格式

1：验证请求和数据请求

|  |  |
| --- | --- |
| **字节内容** | **字节数** |
| 数据头（填充为##） | 2 |
| CRC | 2 |
| 消息类型（兼容数据包号） | 2 |
| 硬件版本 | 1 |
| 软件版本 | 1 |
| IMEI(高7位) | 4 |
| IMEI（低八位） | 4 |

2：验证通过

|  |  |
| --- | --- |
| **字节内容** | **字节数** |
| 数据头（填充为**##**） | **2** |
| CRC | 2 |
| 消息类型（兼容数据包号） | **2** |
| 硬件版本 | 1 |
| 软件版本 | 1 |
| 数据包数量（分割后数据包的数量） | 2 |
| CRC\_APP(整个待升级固件的大小CRC) | 2 |
| SIZE\_APP(整个待升级固件的大小) | 4 |

验证失败，断开TCP

:3：程序数据返回

|  |  |
| --- | --- |
| **字节内容** | **字节数** |
| 数据头（填充为##） | 2 |
| CRC | 2 |
| 消息类型（兼容数据包号） | 2 |
| 硬件版本 | 1 |
| 软件版本 | 1 |
| 数据包内容CRC | 2 |
| 数据包长度 | 2 |
| 预留（填充为\*\*\*\*） | 4 |
| 附加程序数据 | 0-1024 |

注意：这是针对当前sim模块，附加数据体积为512字节（硬件限制）

## 二：消息类型的定义

消息类型为有符号短整型（16位）

0：认证请求信息

-1：验证通过信息类型（这里是否与认证请求一致？？）

1-5000：数据包索引请求信息（默认每次请求1KB，fireware大小不会超过256KB，所以预留最大容量是5000KB）

程序数据返回时小心类型保持不变。

预留扩展方法。

前六个字节不变化，剩余8个字节自由扩展

对于各种请求的返回信息，消息类型保持不变

Crc说明：

对于16个字节的数据头信息中，crc负责校验自crc后的12个字节的数据校验

程序数据返回包含两个crc校验信息，第一个对数据包头校验，第二个针对附加程序数据校验。

Crc算法：考虑到收到的数据crc校验一直失败，现在附加CRC16算法供参考，见协议最后

## 三：关于重复认证的需求：

1：考虑到中间经过移动服务器中转，下位机可能存在无法收到，造成服务器和终端状态不同步，crc校验失败了需要重新请求认证

2：汽车电源波动较大，很容易造成接收到的数据出现错误，crc校验失败了需要重新请求认证

移动网络

server

车载终端

通讯流程：

这里暂时用ascii代表整数，空格也是不存在的

-----> ####IMEI 1 2 终端硬件号为1，软件号2

<---- TCP断开（认证失败）

-----> ####IMEI 8 8 终端硬件号为8，软件号8

<---- #### 8 9 29 可升级硬件号为8，软件号9数据包个数为29

-----> ####1

<---- #### CRC 1 1000 data

-----> ####2

<---- #### CRC 2 1000 data

-----> ####3

<---- #### CRC 3 1000 data

。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。。

-----> ####29

<---- #### CRC 3 lengh data

over

设备通过校验后，可以从任何一个数据包开始取数据，如果当前数据连接断开，需要重新进行验证

CRC校验有两个，一个是当前的数据结构的整体校验，除去开始头和crc自身共6个字节

另一个是crc是如果携带了数据包，则为数据包内容的CRC的校验，便于数据完整性。

2：串口在线升级协议

因设备生产完成后，需要本地将设备程序正常程序写入，需要本地IAP升级功能支持，升级协议与网络IAP相同。希望能作出一个串口升级程序。

新添加：设备出厂需要统一的进入ERP流程，需要进行注册。本来想单独列出来，考虑到系统集成度及开发时间问题，现在添加到这里。

## 四：设备ERP需求

可以和串口升级坐在一起，但是建议使用不同的按键启用不同的功能。(设置默认值，按下某键手动设置)

1:：终端通过串口发送注册请求信息

####IMEI|HV|SV\*

2：server返回分配给设备的信息：

####DEV\_ID|IAP \_IP|IAP\_PORT|APP\_IP|APP\_PORT|BLUE\_NAME|BLUE\_PASSWORD|BLUE\_BAUD\*

其中DEV\_ID预留，建议作为设备号信息，用来取代imei信息

####AAAAAAAA|in.phonewingstech.com|9000|in.phonewingstech.com|9001|OBD\_1|0000|38400\*

3：终端返回server分配的信息，表示接收到信息，双方确保信息正确

####AAAAAAAA|in.phonewingstech.com|9000|in.phonewingstech.com|9001|OBD\_1|0000|38400\*

备注： ‘|’为分隔符 #表示为结束符

**附件：CRC16校验算法：**

uint16\_t calBufCrc(uint8\_t \*ptr, uint32\_t len) {

uint16\_t crc;

uint8\_t da;

const uint16\_t crc\_ta[256]={ /\* CRC余式表 \*/

0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,

0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,

0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,

0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,

0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,

0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,

0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,

0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,

0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,

0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,

0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12,

0xdbfd, 0xcbdc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,

0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,

0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,

0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,

0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,

0x9188, 0x81a9, 0xb1ca, 0xa1eb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,

0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,

0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,

0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,

0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,

0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,

0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,

0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,

0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,

0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3,

0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,

0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,

0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,

0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,

0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,

0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0

};

crc=0;

while(len--!=0) {

da=(uint8\_t) (crc/256); /\* 以8位二进制数的形式暂存CRC的高8位 \*/

crc<<=8; /\* 左移8位,相当于CRC的低8位乘以 \*/

crc^=crc\_ta[da^\*ptr]; /\* 高8位和当前字节相加后再查表求CRC ,再加上以前的CRC \*/

ptr++;

}

return(crc);

}