# 前言

## 编写目的

统一公司协议规范，提高开发和沟通效率，降低开发维护成本。

## 缩略语

表 1. 缩略语中英文对照表

|  |  |  |
| --- | --- | --- |
| 缩写 | 英文解释 | 中文解释 |
|  |  |  |

# 使用说明

## 协议选型

对于数据量比较大的或者对功耗有着极高要求的设备，建议使用二进制协议，相反，建议使用json协议。

## 支持的数据场景

目前我们的产品主要涉及：

1. GPS/北斗位置信息：经纬度，速度，航向
2. 温湿度：温度，湿度
3. 气压：气压传感器数据
4. 基站信息：基站定位需要的信息
5. Wi-Fi数据：Wi-Fi网络定位需要的信息
6. 时间同步：从服务器同步时间
7. 下行指令：服务器主动下发配置命令参数
8. 下行文本：服务器主动下发文本通知
9. 扩展协议：对于某些需要特定接口的设备，比如头羊的上报羊群接口，可以在默认协议下增加新的扩展接口。

## 扩展性和兼容性

我们尽量把通用的接口标准化，一旦开发完成，新的设备上报类似数据时，直接采用这些接口，这样服务器就不用重新开发，直接复用标准的接口就可以。

对于通用接口不能支持的业务场景，可以定制新的接口，设计时尽量考虑通用和复用。

## 二进制协议优势

1. 更小传输体积和内存消耗：相比较文本和json协议，所有的内容采用二进制表示，尽量压缩传输的体积，减小传输需要的GPRS功耗。例如，设备id “12345678”作为文本字段是9个字节，但是作为二进制内容可以用0x12345678 四个字节表示，相比较原来的内容，容量压缩一半。
2. 无需额外格式符：二进制通过结构体内存空间地址编码和解码，相比较文本协议，没有文本分隔符或者格式符号，表达效率更高。
3. 高效编解码：如果服务器使用c/c++开发，设备端和服务端可以通过共用头文件的方式，无缝编解码，字节将结构转成内存字节，或者将内存字节转为结构，基本没有解析成本。对于java这样的高级语言，虽然没有原生的结构体和字节码支持，但也有针对字节内存转码的开源组件，基本也是开箱即用：

[JavaStruct](http://code.google.com/p/javastruct/wiki/HowToUseJavaStruct)：

<http://code.google.com/p/javastruct/wiki/HowToUseJavaStruct>

这个实现比较轻量级，POPO的方式，还不支持联合体，但是联合体的问题，在编程中一般是可以避开的。

[Javolution](http://javolution.org/)：

<http://javolution.org/>

这个类库比较庞大，对结构体和联合体都有支持。

## 二进制协议劣势

二进制协议在具备一定的优势同时，也具有一定的劣势：

1. 可读性不好：由于所有的数据都是二进制编码，不能很方便的被人脑所识别，可读型不如文本协议或者json协议，增加调试阶段的难度，但一旦调试完成，即无额外影响。
2. 对于java等语言支持不是很好，但是有开源的第三方可以使用，可以直接对接内存，转化成需要的结构对象。

# 设计基本原则

## 原则简述

1. 所有的协议都必须要回复，通过SN确认每一个包的唯一性。
2. 所有的设备上线都会发送登录数据包，服务器可以根据此此协议做一些存留消息推送，或者长连接管理更新之类的操作。
3. 长连接的架构中，增加心跳协议，在没有数据时，也要定期发送心跳，心跳数据包尽量小，心跳的间隔决定了断链重建的时效性。
4. 请求消息使用奇数ID，回复使用偶数ID（请求ID自加1）。
5. 协议只是定义了通讯的数据交互，并未规定发送的方向，同一种协议，既可以由设备发向服务器，也可以由服务器发向设备，完全根据数据需要选择。
6. 除了极度苛刻的场景，尽量不要使用位操作定义。
7. 协议ID使用双字节：主类型加子类型，子类型是默认协议号是01，如果此默认协议不能够支持的场景，需要扩这，就可以增加新的子类型。大类型是可以连续的，子类型遵循奇偶的约定。
8. 凡是浮点数类型的，建议直接用字符串表示，不要转换成二进制。
9. 尽量不要用联合，可以通过增加成不同的子类型协议来实现。

# 协议格式

说明：文档中的时间，对应硬件的时间，是UTC时间。

## 请求数据包格式

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字段 | 起始符 | 校验码 | 包长度 | 协议类型 | 序列号SN | 日期时间 | 终端id | 内容 | 结束符 |
| 长度  (字节) | 2 | 2 | 2 | 2 | 4 | 6 | 8 | N | 2 |

1. 起始符

固定值 0xFFFF

1. 校验码

校验范围: 包长度+协议类型+序列号+时间日期+终端ID+ID校验+内容。校验码放在前段，可以不用长度计算偏移，直接索引位置。默认crc16校验

1. 包长度

整个包大小（起始+校验+长度+协议类型+序列号+日期+终端ID +内容+结束符）.除了内容，是28字节

1. 协议类型

由主类型加子类型组成，主类型是顺序编号的。子类型请求是基数，回复是偶数数。

子类型协议，设备发起的编号 0x01-7F. 服务器发起的0x81-0xff以后。

|  |  |  |  |
| --- | --- | --- | --- |
| 协议类型 | 主类型 | 子类型 | 说明 |
| 注册包 | 0x01 | 0x01 | 第一登录平台的发送一次。 |
| 登录包 | 0x02 | 0x01 | 每次上电都需要发送 |
| 心跳包 | 0x03 | 0x01 | 内容为空 |
| GPS数据 | 0x04 | 0x01 | GPS基础接口，大牲畜使用此接口 |
| 0x03 | 设备使用的羊群位置上报接口 |
|  |  |
| 时间同步 | 0x05 | 0x01 | 服务器需要返回当前的UTC时间，年月日时分秒 |
| 请求配置 | 0x06 | 0x01 | 查询服务器是否有时间 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. 序列号 SN

从硬件系统上电开机，第一个包0，以后所有消息依次累加。如果是服务器发送的消息，服务器也要有自己的流水号，每个设备掉电生命周期内所有的消息的sn都是不一样的。

1. 日期时间

当前数据的GPS时间，年月日时分秒：0x120109110A0C，就表示 18年1月9号17点10分12秒。

1. 终端id : 8 byte，建议使用产品类型+批次+流水号，或者用mac，或者用imei。比较好的做法是使用前者，但是需要量产时额外来写入这个编号。将16字节的字符串用hex的方式转化成8个字节。比如 “0001180100000001”表示成 0x0001180100000001，8个字节。
2. 结束符

固定值 0xEEEE

9. 如果长度和校验失败，直接丢弃当前数据包。

## 响应数据格式

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字段 | 起始符 | 校验码 | 包长度 | 协议类型 | 序列号 | 日期时间 | 终端id | 响应内容 | 结束符 |
| 长度 | 2 | 2 | 2 | 2 | 4 | 6 | 8 | N | 2 |

1. 协议类型

原样返回请求类型+1.

1. 序列号

原样返回请求中的序列号。

1. 日期时间

服务器响应时的，服务器时间。

1. 终端ID

请求设备的ID。

1. 响应内容：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 响应码 | 响应内容长度 | 响应内容 |
| 长度 | 1 | 2 | N |

1. 响应码：

0-99 表示成功的返回码，100-255表示失败的返回码。例如：

0：成功。根据接口不同，对应的响应内容可能是“OK”或者对应的数据结构

1：有配置更新。说明请求成功了，同时服务器有配置数据，设备可以根据此返回值发起配置请求。对应的内容为空

100：错误1，响应的内容是错误1的描述文字。

101：错误2，响应的内容是错误2的描述文字

# 协议内容

## 登录包(0x0201)

「使用情境」

设备每一次上电开机时,会上报登录包,服务器确认合法性后,向设备回复响应。

1. 请求包内

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 验证码 | 协议版本 | 自定义数据 |
| 长度 | 4 | 1 | 32 |

1. 设备验证码

32位整数 uint32 类型。

由设备根据ID根据协商的加密算法自动生成,用于校验设备的合法性,服务器校验失败将拒绝登陆,并断开连接。

目前调试阶段，暂时为0x00000000

1. 协议版本。

为了方便平台兼容识别硬件报文格式的变化。默认从0开始，一旦同一个接口协议不兼容了，就需要升级版本号。

1. 自定义数据

可以根据需要，让设备上报个性数据。在头羊和大牲畜的设备中，我们填充20byte的SIM卡的ICCID，剩余的补0。例子:”86891234567890123456”

1. 响应

0：成功：OK

1：有配置，内容为空

2：失败，内容为具体错误描述

* 时间结构体

|  |  |  |
| --- | --- | --- |
| 字段 | 时间结构体 | |
| 长度 | 1 | 6 |
| 内容 | 时区 | UTC时间 |
| 8 | 0x120109110A0C |

1. 协议样例：

Req：ff ff 0a 00 20 00 01 01 01 00 00 00 12 01 17 03 09 36 00 01 12 01 00 00 00 01 00 00 00 00 ee ee

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 头 | 校验 | 长度 | 协议号 | 流水号 | 时间 | Id | 验证码 | 尾 |
| 报文 | ffff | 0a00 | 2000 | 0101 | 01000000 | 120117030936 | 0001120100000001 | 00000000 | eeee |
| 值 |  | 0x000a:异或校验 | 0x0020=32 | 0x0101 | 0x00000001=1 | 18-01-23 3：9：54 | 0001120100000001 | Int类型的0，4个字节 |  |

Rsp：

## 时间请求(0x0501)

「使用情境」

设备需要时间时，通过此接口向设备请求时间。

1. 请求包内容：

空

1. 响应

0：成功：内容为时间结构体

1：有配置，内容为空

2：失败，内容为具体错误描述

1. 协议样例：

Req：

Rsp：

## 头羊羊群位置包(0x0403)

「使用情境」

头羊的小羊数据上报包。

1. 请求包内容：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字段 |  |  | | | 羊群数据 | | | | | | | |
| 电压 | 卫星数 | 纬度 | 经度 | 小羊数 | ID前缀 | 小羊数组 | | | | | |
| 小羊ID | 时间差 | 小羊ID | 时间差 | 小羊ID | 时间差 |
| 长度 | 2 | 1 | 11 | 11 | 2 | 4 | 4 | 2 | 4 | 2 | 4 | 2 |

1. 电压：设备的电压，mv表示，比如0x1234 =4660，表示4.66V
2. 卫星数：定位的卫星数，可选数值，根据需要填充
3. 经纬度：带正负号的字符串表示，比如-22.4862 和114.4981，规格长度为11，支持小数点后面5位。
4. 小羊数：当前包里面包含的小羊数据的个数。
5. ID前缀，二维码ID的前8位字符的hex表示：0x00011801 表示字符串的00011801
6. 小羊ID：4字节的无符号整型。二维码ID的后8位的hex表示，比如二维码是0001180100001122。后8位的是十进制的1122，一千一百二十二，十六进制是0x0462. 所以这里的小羊ID就是0x00000462 。填充的时候低地址值在前 ：0x62040000
7. 时间差：发送时计算出来的小羊的接收时间跟头羊的GPS时间的差值，单位是分钟。是一个两个字节的short值. 填充的时候低地址值在前。
8. 响应：

0：成功：OK

1：有配置，内容为空

2：失败，内容为具体错误描述

1. 参考数据结构
2. 协议样例：

Req：

样例1：

ff ff 24 00 eb 00 03 04 22 00 00 00 12 01 16 0e 05 3a 00 01 12 01 00 00 00 01 08 32 32 2e 32 33 38 34 37 00 00 00 31 31 34 2e 32 33 38 34 37 00 00 0e 10 0f 00 00 00 00 00 00 00 00 e0 e0 08 09 0a 00 00 00 00 00 00 00 e1 e1 08 09 0a 00 00 00 00 00 00 00 e2 e2 08 09 0a 00 00 00 00 00 00 00 e3 e3 08 09 0a 00 00 00 00 00 00 00 e4 e4 08 09 0a 00 00 00 00 00 00 00 e5 e5 08 09 0a 00 00 00 00 00 00 00 e6 e6 08 09 0a 00 00 00 00 00 00 00 e7 e7 08 09 0a 00 00 00 00 00 00 00 e8 e8 08 09 0a 00 00 00 00 00 00 00 e9 e9 08 09 0a 00 00 00 00 00 00 00 ea ea 08 09 0a 00 00 00 00 00 00 00 eb eb 08 09 0a 00 00 00 00 00 00 00 ec ec 08 09 0a 00 00 00 00 00 00 00 ed ed 08 09 0a 00 00 00 00 00 00 00 ee ee 08 09 0a ee ee

样例2：

ff ff 14 00 73 00 03 04 23 00 00 00 12 01 16 0e 06 02 00 01 12 01 00 00 00 01 08 32 32 32 33 2e 38 34 37 00 00 00 31 31 34 32 33 2e 38 34 37 00 00 0e 10 05 00 00 00 00 00 00 00 00 ef ef 08 09 0a 00 00 00 00 00 00 00 f0 f0 08 09 0a 00 00 00 00 00 00 00 f1 f1 08 09 0a 00 00 00 00 00 00 00 f2 f2 08 09 0a 00 00 00 00 00 00 00 f3 f3 08 09 0a ee ee

红色部分解析如下：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 卫星数 | 纬度 | 经度 | 电压 | 小羊数 | ID |  |  |
| 08 | 323232332e383437000000 | 31313432332e3834370000 | 0e10 | 0500 | 00000000000000ef | ef | 08090a |
| 8 | 2223.847 | 11423。847 | 0x100e=4110mV | 0x0005=5 |  | 0xef | 8:9:10 |

Rsp：

## 位置包（大牲畜）(0x0401)

「使用情境」

普通的GPS位置上报包。

1. 请求包内容：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 字段 | 设备信息 | 位置信息 | | | | | | | | |
| 电压 |  |  | | | |  | | | |
| 位置数 | 卫星数1 | 纬度1 | 经度1 | 时间1 | 卫星数n | 纬度n | 经度n | 时间n |
| 长度 | 2 | 1 | 1 | 11 | 11 | 6 | 1 | 11 | 11 | 6 |

1. 卫星数：定位的卫星数。
2. 位置数：当前包中的位置个数。
3. 经纬度：带正负号的字符串表示，比如-22.4862 和114.4981，规格长度为11，支持小数点后面5位。
4. 电压：设备的电压，mv表示，比如0x1234 =4660，表示4.66V
5. 时间: 每一个经纬度定位时的时间，年月日时分秒
6. 响应：

0：成功：OK

1：有配置，内容为空

2：失败，内容为具体错误描述

1. 协议样例：

Req：

Rsp：

## 配置查询包(0x0601)

「使用情境」

需要更新配置时，主动向服务器发送，如果服务器有配置数据，则在回复中下发配置数据。每请求一次，服务器只能回复一个配置数据，不能在一次请求中，回复多个包，因为在使用模块的方案中，模块的接收buffer是有限的，多包会有风险。

1. 请求包内容：

空

1. 响应：

响应包格式：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 响应码 | 响应内容长度 | 响应内容 |
| 长度 | 1 | 2 | N |
| 内容 | 0 | N | 具体内容见下表 |

响应码：

0：成功：配置数据结构

1：有配置，内容为空

2：失败，内容为具体错误描述

响应内容（配置数据结构）：

|  |  |  |
| --- | --- | --- |
| 字段 | 配置类型 | 配置内容 |
| 长度 | 1 | N |

不同的配置类型，表示不同的服务器下发配置数据，配置数据编号表示如下：

|  |  |
| --- | --- |
| 配置类型 | 配置数据类型 |
| 1 | 服务器地址 |
| 2 | 大牲畜定位配置 |
| 3 | 头羊定位配置 |

### 服务器地址

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 字段 | 地址个数 | 服务器信息 | | | | | |
| 类型 | 地址 | 端口 | 类型n | 地址n | 端口n |
| 长度 | 1 | 1 | 32 | 2 | 1 | 32 | 2 |

1. 服务器个数：配置的可用服务器的地址个数
2. 类型：0 表示IP，1表示域名。这里的域名直接存储的就是字符的ascii码，比如www.abc.com，加上0结束符，就是12个字节，前几个字节为0x77 0x77 0x77 0x2E。最大支持的域名长度32。
3. 端口：short类型的数字，0-65535
4. 协议样例：

Req：

Rsp：

### 大牲畜定位参数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段 | 定位时段 | |  | 上报时间点 | | |
| 开始时间 | 结束时间 | 定位间隔 | 上报点个数 | 时间1 | 时间n |
| 长度 | 4 | 4 | 4 | 1 | 4 | 4 |

1. 开始时间，结束时间：时分秒的秒数表示，比如1：01：01 就是3661。默认早8点到晚8点。
2. 定位间隔：间隔的秒表示，比如一分钟，就是60。默认2小时，一天就是6次
3. 上报点数：规格最多10个，服务器默认下发3个。
4. 时间1-n：时分秒的秒数表示 。默认第一次就是开机时间加一个定位间隔内的随机时间。
5. 协议样例

Req：

Rsp：

### 头羊网关定位参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段 | 433工作时段 | |  | |
| 开始时间 | 结束时间 | 上报周期 | 上报计时起始时间点 |
| 长度 | 4 | 4 | 4 | 4 |

1. 开始时间，结束时间：时分秒的秒数表示，比如1：01：01 就是3661。默认12点到24点。
2. 上报起始点：时分秒的秒数表示，硬件默认启动后，随机延时一个时间（小于上报周期），作为起始时间。默认开始时间加个上报周期内的随机值
3. 上报周期：上报的时间周期，表示间隔多久上报一次，单位为分钟。240 就表示4个小时上报一次。默认4小时。
4. 协议样例

Req：

Rsp：