# 物联网（二进制）通讯协议详解

1.协议详解

|  |  |  |  |
| --- | --- | --- | --- |
| 头指令 | N段的长度 | N段 | 内容 |
| 1bit（固定长度） | 1bit（固定长度） | 不固定长度 | 不固定长度 |

头指令：

从0x00-0xff，其中0xff和0x99为保留字段不得使用。

N段的长度：

为N段的数据位的长度，比如N段的数据为FF，此数据占用1位bit，则N段的长度为1,如果N段数据为，0x100~0xffff,这个数据占位为2BIT，则N段的长度为2，

如果N段的数据为0x10000~0xffffff,则占位3bit,N段的的长度为3，依此类的。

N段：

1此段数据为内容数据的长度，内容如果是1BIT那，N段的数值就是1，如果内容是N bit位,N段的数据就是N。

内容：

此段可自己定义想要的内容可以是，十六进制内容，也可以是其他格式内容,形式不固定，内容不固定。

(在制作物联网项目中，至少要把设备ID，在内容中体现)

## 2 特殊协议

心跳包：心跳包是从服务器发出的，所以客户端不需要发送心跳包，

客户端需要接受心跳包，并将心跳包过滤。

心跳包为，单字节0x99,客户端如遇到单字节0X99，需要把接收时间调整到最新。接收到非心跳包也要更新接收时间。

客户端如果超过60-90秒，未更新最新接收时间，则认为连接超时，需要重新连接。

内置指令：

0xff，为服务内置指令，此指令头包含作用为

1.服务已满员，需要连接其他服务、

2.获取中间层的tokenID,如果用到中间转发层的话，tokenid是唯一标识，否则不予以转发也不能收到信息。

3.服务器其他通知或内容。（可自定义）

### 示例

简单描述就是：

头指令，是代表了一个CLASS类，每定义一个头指令，后台处理就要对应一个CLASS

，我这里为了方便，全部定义为0X01，也就是后台只需要用一个CLASS就能处理。

|  |  |
| --- | --- |
| N段的长度 | N段 |

都是通过计算获得的，不固定。

内容是可以自己随便定义的。

我这里定义的前两个字节，定义为具体的方法。你可以认为是，你后端处理CLASS类的具体调用的某一个方法。

登录（客户端发送）：

|  |  |  |  |
| --- | --- | --- | --- |
| 头指令 | N段的长度 | N段 | 内容 |
| 0x01 | 通过计算N段的长度 | 通过计算内容的长度 | （固定2BIT-代表后端的登录方法）  0X000A  （N bit -设备ID编号）  例如设备ID编号是固定的8位4BIT  0x00000001 |

登录组合示例：

0x01(头指令) 0X01（0x0C的长度） 0X6（后面内容的长度，十进制就是12） 0X000A00000001

返回验证及令牌（客户端接收）：

|  |  |  |  |
| --- | --- | --- | --- |
| 头指令 | N段的长度 | N段 | 内容 |
| 0x01 | 通过计算N段的长度 | 通过计算内容的长度 | （固定2BIT-代表后端的登录方法）  0X000A  （N bit -返回的TOKEN）  例如 固定8位4BIT  0x10000011 |

示例：

0x01(头指令) 0X01（0x0C的长度） 0X6（后面内容的长度，十进制就是12） 0X000A10000011

查询（客户端/服务端-发送）：

|  |  |  |  |
| --- | --- | --- | --- |
| 头指令 | N段的长度 | N段 | 内容 |
| 0x01 | 通过计算N段的长度 | 通过计算内容的长度 | （固定2BIT-代表查询的某个方法）  0X00A1  （N bit -登录的令牌TOKEN）  例如 固定8位  0x10000011 |

查询结果（客户端/服务端-接收）：

|  |  |  |  |
| --- | --- | --- | --- |
| 头指令 | N段的长度 | N段 | 内容 |
| 0x01 | 通过计算N段的长度 | 通过计算内容的长度 | （固定2BIT-代表查询的某个方法）  0X00A1  （N bit -返回的结果）  可以是UTF-8字符的的二进制形式。可以是自己定义的值 |

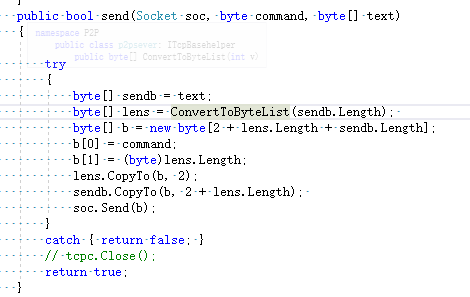
心跳：服务端架构自带，无需定义：

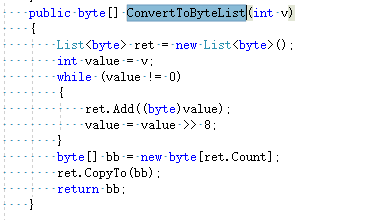
客户端只需要按照 特殊协议，心跳包的解析方法就可以了。

## 3发包例子

其中ConvertToByteList 是把INT类型，转换长BTYE[]数组，然后得到数组的长度（占位）。

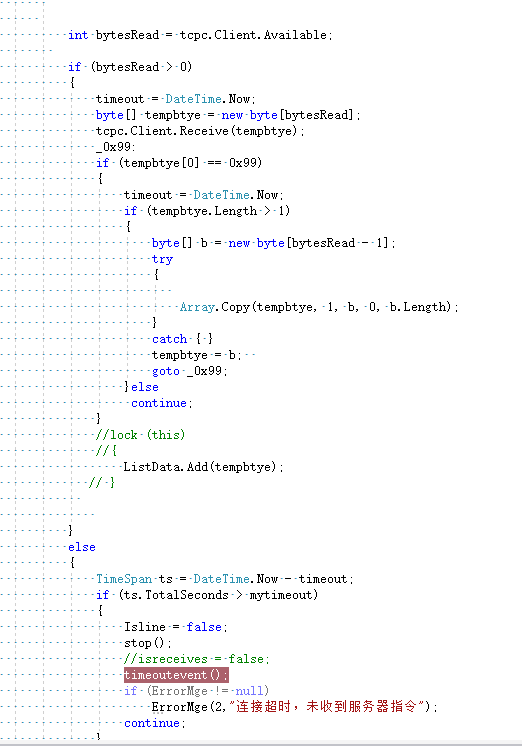
非常简单的组装。2是头部的固定2BIT，一个是头指令，一个是N段的长度。





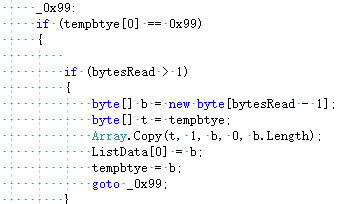
## 解包例子

接收到数据如果第一位是0X99则消除它，然后将其他的内容，增加到等待解包队列。更新接收时间，防止出错。

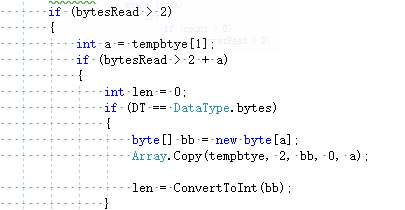


## 解包队列方法

发现第一位是0x99，则消除它

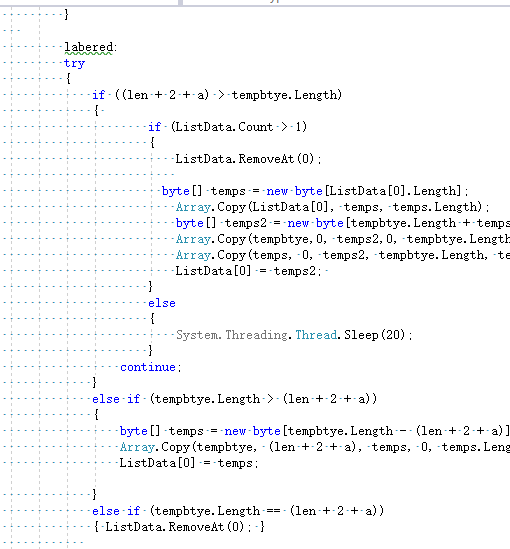


如果过滤后，字符大于2则，看看整个协议的内容到底有多长



如果当前的的值太少了，则吧队列的下一组数据合并到这组数的后面，如果

当前数值太多，则把多余的数值，合并到下组数据的最前面。然后把当前的数值存入临时变量中。从列队中删除这组数据。



最后把正确的数据，拆开分成，头指令和具体内容。

