有关嵌入式条码数据回传的建议

目的和动机

提高系统健壮性和可扩展性.
本建议主要是针对2个方面的问题:

- 数据回传失败的数据修正机制的问题
- 数据格式表示的问题

数据回传的相关问题

现阶段采用的方式

现阶段,数据回传采用的是回传数据的方式,大致方式如下:

1. 将数据进行分组.
2. 一次一组数据,依次回传到服务器

可能面临的问题

传输机制: 在字符串格式数据传输过程中.由于各种原因(网络,电气,系统,人为干预等)引起的传输中断会打断回传的过程, 而一般采用的保障机制是检测, 比对,重试. 而这些机制本身势必会增加代码复杂性和增加出错的概率.即引入了潜在的风险同时让本身简单的事情复杂化.

数据格式: 另外.现阶段的数据采用字符串格式.一维数据格式.条码之间用字符分割. 识别数据的时候需要对字符串逐一进行再分割.繁琐低效.同时还要约定分隔符种类,注意全角半角和空白问题.对于多级码之间的层级关系,更是缺乏直观的记录和表示方式.

建议的改变

1. 规避重试机制的风险. 数据传输的机制由原来的**数据传输**改为**文件传输**
2. 采用标准化数据传输格式.文件的格式由普通的**字符串格式**改为**JSON格式**

传输的不是字符串数据而是一个文件

使用文件的目的是:

- 降低回传作业的规模.减少出错概率.
- 出错后,易于排查问题.
- 数据以文件保存,便于备份和维护.

各种语言都有很多实现文件上传的的类库.类库本身在设计当中,已经充分考虑了上文描述中可能遇到的各种问题,并已采取了相应的措施.出于尽量不重复发明轮子的原则.只要我们对这些类库使用得当,完全可以利用类库本身的机制来保障传输的可靠性.这样的结果就是:

- 借助成熟的技术保障文件传输的可靠性. 服务端要么收到完整的数据,要么什么也不做(数据不完整服务端不处理).不存在接收一部分数据的情况.这也避免了复杂的比对和重试机制.
- 由于数据以文件形式上传,服务端一旦解析出错.打开文件基本上就能看出来问题的大概.无需再做日志收集,打断点之类的工作.处理问题简单便捷.

使用标准化的JSON数据结构

JSON是国际上跨行业,跨语言的两大数据交换格式之一(另一种格式是XML).经历过全球各种行业数十年的生产考验.通用性和稳定性已获得肯定.在这里,使用标准化格式的带来的主要好处是:

- 保持良好的扩展性
- 简化处理过程
- 直观,易修改

关于**JSON格式**

> **JSON**是一种轻量级的数据交换格式。它基于 ECMAScript (欧洲计算机协会制定的js规范)的一个子集，采用完全独立于编程语言的文本格式来存储和表示数据。简洁和清晰的层次结构使得 JSON 成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成，并有效地提升网络传输效率。
> *一般来说,和xml格式相比,json格式具有:更小的空间,更快的解析速度*

一个JSON数据格式的例子:

```
/*三层嵌套关系*/
[
    {
        "level3":"23132102619788932957244240019615567",
        "children": [
            {
                "level2": "23132102619788932957244240019615566",
                "children": [
                    "23132102619788932957244240019615568",
                    "23132102619788932957244240019615512",
                    "23132102619788932957244424001961545",
                    ...
                    /*一级码, 表示共计xx瓶*/
                ]
            },
            {
                "level2": "23132102619788932957244240196155632",
                "children": [
                    "23132102619788932957244240019615568",
                    "23132102619788932957244240019615512",
                    "23132102619788932957244424001961545",
                    ...
                ]
            },
            ...共计xx小箱
        ]
    },
    ...共计xx大箱
]
/*这个格式用户还可以在规则的基础上进行自定义,理论上可以标识任何层级关系的数据*/
/*二层嵌套关系*/
[
    {
        "23132102619788932957244240019615567":
          [
            "23132102619788932957244240019615568",
            "23132102619788932957244240019615512",
            "23132102619788932957244424001961545",
            ...
             /*一级码, 表示共计xx瓶*/
          ]
    },
    {
        "23132102619788932957244240019615567":
          [
            "23132102619788932957244240019615568",
            "23132102619788932957244240019615512",
            "23132102619788932957244424001961545",
            ...
             /*一级码, 表示共计xx瓶*/
          ]
    },
    ...
]
```

综合以上所述,建议最终的条码回传方案如下:

1. 嵌入式设备按照条码的层级关系将当日的生产数据转换成JSON格式.
2. 将待回传的条码转成文件格式
3. 开始向服务器上传条码文件
4. 服务器接收完文件后,读取文件内容进行解析.
5. 服务器文件成功后,写入数据库.

在这当中:

- 如果网络中断或者其他原因导致上传文件失败,只需重新上传文件即可.
- 如果服务器解析文件错误,可以在服务端直接打开文件,检查文件格式内容即知解析错误的原因

以上就是方案的全部.