**ESL API接口**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Change Description** | **Author** |
| V1.0 | 2018/8/30 | 初始版本 | Ning |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**C O N F I D E N T I A L**

This document is the property of KKM Co.Ltd. KKM retains all rights pertaining to industrial property including patent applications. This document is only for the recipient(s) which authorized by KKM. It contains confidential information and any use, dissemination, distribution, or reproduction of this message by unintended recipients is not authorized and may be unlawful.

. **Catalogue**

[1. 目的 3](#_Toc523412327)

[2. 产品架构 3](#_Toc523412328)

[3. 功能介绍 3](#_Toc523412329)

[3.1 标签状态上报 3](#_Toc523412330)

[3.2 更新标签图片 4](#_Toc523412331)

[3.2.1 连接鉴权 5](#_Toc523412332)

[3.2.1 更新标签图片 8](#_Toc523412333)

[3.2.2 更新标签图片（基于ASCII压缩图片） 11](#_Toc523412334)

[3.3 更新标签参数 13](#_Toc523412335)

# 目的

本文描述了电子标签的API接口，用于指导第三方App接入电子标签。

# 产品架构

电子标签

电子标签

Bluetooth4.0/5.0

App(Android/IOS)

接口：

* 电子标签和基站之间采用蓝牙4.1/5.0进行通讯，为了确保电子标签和App不被非法接入，电子标签和App通讯前需要采用双向MD5鉴权协议以确保安全。

**工作机制：**

1、电子标签：标签开机后，会周期性向广播标签状态信息，该消息包括：电子标签ID，正在显示的图片ID，电量，环境温度，故障状态，版本等信息。

2、App：App可以通过蓝牙链接到电子标签进行更新图片或者修改参数。

约束：目前标签要求App支持的蓝牙协议版本最低为4.1版本，标签建立连接后会发起MTU协商，把MTU协商为50字节以上，以加快图片传输速度。如果低于4.1版本（不支持MTU协商更新），更新图片会失败。

# 功能介绍

## 标签状态上报

标签开机后，会周期性状态在广播消息中上报如下数据：

* 标签的ID
* 标签的信号强弱（到基站的距离）；
* 标签当前显示的图片ID；
* 标签运行的故障状态；
* 标签电池电量；
* 标签环境的温度；

说明：

其中电池的电量是采用电压的形式提供，单位是mV，比如3.3V对应3300mV，实际电量可以根据电压进行换算；

标签传感器监测的环境温度（误差范围为±3℃），可以提供一个粗略的温度监测；

如广播消息：

0201060302A0FE10FF4B4D100000000B371E0001020304CA

则表示如下含义：

* 0201060302：蓝牙协议定义，保留
* A0FE：表示Beacon类型服务数据，所有的标签本字段固定为A0FE，如果不是A0FE，则可以丢弃本消息；
* 10：后续消息字段长度，即16字节
* FF4B4D：保留
* 10：传感器类型，10表示电子标签，如果不是10，则可以丢弃本设备上报消息；
* 00：标签的固件版本号，不同版本号的标签支持的能里会有差异。
* 00：能力字段1，高4bit表示工作状态，0x1标识电子标签显示故障，低4bit预留；
* 00：能力字段2，预留；
* 0C83：电池电压信息，采用大端编码，（0C83电压为3203mV）
* 2100：环境温度，有符号数表示，大端编码，（2100表示33℃）
* 01020304：当前显示的图片ID，更新图片时由客户端管理软件下发指定，建议每次更新图片都分配一个唯一的图片，这样可以监控各个标签当前显示的图片。
* CA：标签在1米处的信号功率，采用符号数，（0xCA=-54dBm）用于计算标签和基站之间的距离；

## 更新标签图片

App通过蓝牙连接到标签后，需要先采用MD5算法进行连接鉴权（20秒内完成），才可以可以发起图片更新。如果鉴权失败，则不允许更新图片。

### 连接鉴权

KBeacon

KGateway

3.MD5 request(random1)

2.BLE connection setup

1. Scanning device

4. MD5 response(auth1\_value, random2)

6. MD5 response(auth2\_value)

8. MD5 success

下发更新的图片内容

7. Verify Gateway Auth success

5. Verify Device auth success

第3步消息内容如下：

|  |  |  |
| --- | --- | --- |
| 字段 | 字段长度 | 含义 |
| DataType | 4bit | 数据帧类型：0x1  0x1: 鉴权数据帧  0x0: 二进制类型  0x2：JSON类型数据帧  0x4：ASCII类型数据帧 |
| PduTag | 4bit | 数据帧分段标识，对于鉴权，固定为0x3   * FrameStart = 0, * FrameMiddle = 1, * FrameEnd = 2, * FrameSingle = 3 |
| AuthAlg | 1 Byte | 鉴权类型：0x1  0x1: App鉴权  0x2: 设备鉴权 |
| AppRandom | 4 Byte | 鉴权随机数，由app随机生成，供标签输入计算MD5值 |

第4步消息内容如下：

|  |  |  |
| --- | --- | --- |
| 字段 | 字段长度 | 含义 |
| DataType | 4bit | 数据帧类型：0x1  0x1: 鉴权数据帧  0x0: 二进制类型  0x2：JSON类型数据帧  0x4：ASCII类型数据帧 |
| PduTag | 4bit | 数据帧分段标识，对于鉴权，固定为0x3   * FrameStart = 0, * FrameMiddle = 1, * FrameEnd = 2, * FrameSingle = 3 |
| AuthAlg | 1 Byte | 鉴权类型：0x1  0x1: App鉴权  0x2: 设备鉴权 |
| deviceRandom | 4 Byte | 鉴权随机数，由标签随机生成，供app输入计算MD5值 |
| auth1\_value | 16 Byte | 根据app输入的随机数，以及标签的mac地址，设备的密码，计算出一个MD5值 |

MD5按照顺序输入格式如下，标签按照如下顺序计算出MD5值。

* uint8\_t macAddr[6]; //设备的mac地址，请注意顺序
* uint8\_t sourceKey[2];
* uint8\_t inputRandom[4]; //app的输入随机值，跟下发的顺序保持一致
* uint8\_t devicePassword[8]; //设备密码，ASCII字符，默认为00000000
* 其中sourceKey固定为：0xA9，0xB1

第5~6步：app按照相同的方式计算MD5值，如果非法直接关闭连接。如果设备验证合法，则根据设备输入的随机数，以及上述规则计算出MD5值（用于应答设备的）。

|  |  |  |
| --- | --- | --- |
| 字段 | 字段长度 | 含义 |
| DataType | 4bit | 数据帧类型：0x1  0x1: 鉴权数据帧  0x0: 二进制类型  0x2：JSON类型数据帧  0x4：ASCII类型数据帧 |
| PduTag | 4bit | 数据帧分段标识，对于鉴权，固定为0x3   * FrameStart = 0, * FrameMiddle = 1, * FrameEnd = 2, * FrameSingle = 3 |
| AuthAlg | 1 Byte | 鉴权类型：0x2  0x1: App鉴权  0x2: 设备鉴权 |
| Auth2\_value | 16 Byte | 根据设备在上一步输入的随机数，以及标签的mac地址，设备的密码，计算出一个MD5值 |

MD5按照顺序输入格式如下，APP按照如下顺序计算出MD5值。

* uint8\_t macAddr[6]; //设备的mac地址，请注意顺序
* uint8\_t sourceKey[2];
* uint8\_t inputRandom[4]; //标签的输入随机值
* uint8\_t devicePassword[8]; //设备密码，ASCII字符，默认为00000000
* 其中sourceKey固定为：0xA9，0xB1

第7~8步：设备计算App返回的MD5值是否合法，如果合法，则向App返回成功应答消息。

|  |  |  |
| --- | --- | --- |
| 字段 | 字段长度 | 含义 |
| DataType | 4bit | 数据帧类型：0x1  0x1: 鉴权数据帧  0x0: 二进制类型  0x2：JSON类型数据帧  0x4：ASCII类型数据帧 |
| PduTag | 4bit | 数据帧分段标识，对于鉴权，固定为0x3   * FrameStart = 0, * FrameMiddle = 1, * FrameEnd = 2, * FrameSingle = 3 |
| AuthRslt | 1 Byte | 鉴权应答：0x2表示鉴权成功  其它：鉴权失败 |

### 更新标签图片

ESL

Gateway

3. 下发图片字节

2.鉴权成功

1. 连接建立

4. 下发图片字节应答

5. 下发图片字节

6. 下发图片字节应答

步骤3：Gateway下发图片字节，对于第一帧，需要包含图片的头信息。

|  |  |  |
| --- | --- | --- |
| 字段 | 字段长度 | 含义 |
| DataType | 4bit | 数据帧类型：0x0  0x1: 鉴权数据帧  0x0: 二进制类型  0x2：JSON类型数据帧  0x4：ASCII类型数据帧 |
| PduTag | 4bit | 图片的数据帧，第一帧是FrameStart，中间帧为FrameMiddle，最后一帧是FrameEnd。   * FrameStart = 0, * FrameMiddle = 1, * FrameEnd = 2, * FrameSingle = 3 |
| DataIndex | 2 Byte | 内容字段在图片数据帧中的序号，从1开始计算，数据内容包含图片头。 |
| Data Content | 变长 | 图片内容，采用二进制编码 |

第一帧的图片头信息如下：

|  |  |  |
| --- | --- | --- |
| 字段 | 字段长度 | 含义 |
| PictureType | 1Byte | 图片数据类型：  0x1： 二进制非压缩格式图片 |
| PictureID | 4Byte | 图片的ID，图片更新成功后，本图片ID会在广播消息中携带 |
| PictureLen | 2Byte | 采用小端编码（高位在第1字节，低位在第2字节），对于2.9寸，图片长度为4736字节 |
| Picture Content | 变长 | 图片内容，BIN格式，通过Image2Lcd工具可以生成，参见第4章节。 |

步骤4：ESL收到数据后，将在5秒内向App返回应答，应答消息格式如下：

|  |  |  |
| --- | --- | --- |
| 字段 | 字段长度 | 含义 |
| DataType | 4bit | 数据帧类型：0x0  0x1: 鉴权数据帧  0x0: 二进制类型  0x2：JSON类型数据帧  0x4：ASCII类型数据帧 |
| PduTag | 4bit | 应答消息，固定为FrameSingle。   * FrameStart = 0, * FrameMiddle = 1, * FrameEnd = 2, * FrameSingle = 3 |
| AckDataSeq | 2Byte | 表示ESL已经收到的数据长度。 |
| RxWindow | 2Byte | 接收窗，ESL不支持 |
| AckCause | 2Byte | 成功应答码：  4: 表示接收数据成功，等待下一帧；  0：表示图片数据接收完成，消息执行成功  错误应答如下：  259 输入的内容字段非法  289 请求刷新图片尺寸和实际墨水屏尺寸定义不符合  290 下发的图片数据长度错误  291 墨水屏故障  298 未知的消息内容 |

步骤5：app收到成功应答4后，下发下一帧的内容，下一帧的起始根据应答消息中的AckDataSeq计算。如果收到错误应答，则直接结束下发图片更新流程。 App如果成功收到应答0，则表示图片更新完成。

|  |  |  |
| --- | --- | --- |
| 字段 | 字段长度 | 含义 |
| DataType | 4bit | 数据帧类型：0x0  0x1: 鉴权数据帧  0x0: 二进制类型  0x2：JSON类型数据帧  0x4：ASCII类型数据帧 |
| PduTag | 4bit | 图片的数据帧，中间帧为FrameMiddle   * FrameStart = 0, * FrameMiddle = 1, * FrameEnd = 2, * FrameSingle = 3 |
| DataIndex | 2 Byte | 内容字段在图片数据帧中的序号，从1开始计算，数据内容包含图片头。 |
| Data Content | 变长 | 图片内容 |

### 更新标签图片（基于ASCII压缩图片）

ESL

Gateway

3. 下发图片字节

2.鉴权成功

1. 连接建立

4. 下发图片字节应答

5. 下发图片字节

6. 下发图片字节应答

步骤3：Gateway下发图片字节，对于第一帧，需要包含图片的头信息。

|  |  |  |
| --- | --- | --- |
| 字段 | 字段长度 | 含义 |
| DataType | 4bit | 数据帧类型：0x4  0x1: 鉴权数据帧  0x0: 二进制类型  0x2：JSON类型数据帧  0x4：ASCII类型数据帧 |
| PduTag | 4bit | 图片的数据帧，第一帧是FrameStart，中间帧为FrameMiddle，最后一帧是FrameEnd。   * FrameStart = 0, * FrameMiddle = 1, * FrameEnd = 2, * FrameSingle = 3 |
| DataIndex | 2 Byte | 内容字段在图片数据帧中的序号，从1开始计算，数据内容包含图片头。 |
| Data Content | 变长 | 图片内容，基于ASCII编码 |

第一帧的图片头信息如下：

|  |  |  |
| --- | --- | --- |
| 字段 | 字段长度 | 含义 |
| PictureType | 2Byte | 图片数据类型，采用十六进制ASCII编码：   * 第一字节为ASCII的0 * 第2字节为ASCII的2   组合起来为0x02，即ASCII压缩格式图片 |
| PictureID | 8Byte | 图片ID，采用十六进制ASCII编码：  举例：A0000101，图片的ID，图片更新成功后，本图片ID会在广播消息中携带 |
| PictureLen | 4Byte | 图片ID，采用十六进制ASCII编码：  采用小端编码（高位在第1字节，低位在第2字节），对于2.9寸，图片长度为4736字节 |
| DictoryLen | 2Byte | 压缩字典长度，采用十六进制ASCII编码： |
| Dictory | DictoryLen\*2 | 压缩字典，参见第4章节。 |
| Picture Content | 变长 | 图片内容，采用ASCII编码。 |

步骤4：ESL收到数据后，将在5秒内向App返回应答，应答消息格式如下：

|  |  |  |
| --- | --- | --- |
| 字段 | 字段长度 | 含义 |
| DataType | 4bit | 数据帧类型：0x4  0x1: 鉴权数据帧  0x0: 二进制类型  0x2：JSON类型数据帧  0x4：ASCII类型数据帧 |
| PduTag | 4bit | 应答消息，固定为FrameSingle。   * FrameStart = 0, * FrameMiddle = 1, * FrameEnd = 2, * FrameSingle = 3 |
| AckDataSeq | 2Byte | 表示ESL已经收到的数据长度。 |
| RxWindow | 2Byte | 接收窗，ESL不支持 |
| AckCause | 2Byte | 成功应答码：  4: 表示接收数据成功，等待下一帧；  0：表示图片数据接收完成，消息执行成功  错误应答如下：  259 输入的内容字段非法  289 请求刷新图片尺寸和实际墨水屏尺寸定义不符合  290 下发的图片数据长度错误  291 墨水屏故障  298 未知的消息内容 |

步骤5：app收到成功应答4后，下发下一帧的内容，下一帧的起始根据应答消息中的AckDataSeq计算。如果收到错误应答，则直接结束下发图片更新流程。 App如果成功收到应答0，则表示图片更新完成。

|  |  |  |
| --- | --- | --- |
| 字段 | 字段长度 | 含义 |
| DataType | 4bit | 数据帧类型：0x0  0x1: 鉴权数据帧  0x0: 二进制类型  0x2：JSON类型数据帧  0x4：ASCII类型数据帧 |
| PduTag | 4bit | 图片的数据帧，中间帧为FrameMiddle   * FrameStart = 0, * FrameMiddle = 1, * FrameEnd = 2, * FrameSingle = 3 |
| DataIndex | 2 Byte | 内容字段在图片数据帧中的序号，从1开始计算，数据内容包含图片头。 |
| Data Content | 变长 | 图片内容，基于ASCII编码 |

## 更新标签参数

考虑到ESL标签未来支持iBeacon/Eddystone等功能，为了方便后续参数扩充，因此内置JSON解析器。所有的参数采用JSON格式。

ESL

Gateway

3. 下发JSON参数

2.鉴权成功

1. 连接建立

4. JSON参数应答

步骤3：Gateway下发图片字节，对于第一帧，需要包含图片的头信息。

|  |  |  |
| --- | --- | --- |
| 字段 | 字段长度 | 含义 |
| DataType | 4bit | 数据帧类型：0x2  0x1: 鉴权数据帧  0x0: 二进制类型  0x2：JSON类型数据帧  0x4：ASCII类型数据帧 |
| PduTag | 4bit | 图片的数据帧，JSON参数固定FrameSingle。   * FrameStart = 0, * FrameMiddle = 1, * FrameEnd = 2, * FrameSingle = 3 |
| DataIndex | 2 Byte | 内容字段在图片数据帧中的序号，从1开始计算，数据内容包含图片头。 |
| Data Content | 变长 | JSON格式的参数，采用ASCII编码 |

Data Content内容：

修改密码：

{"tag": "mdf","pwd":"12345678"}

修改发射功率为0dBm（范围-20~5dBm，默认为5dBm）：

{"tag": "mdf","txPwr":0}

修改广播周期为1秒（范围为100~10000ms）：

{"tag": "mdf","advPeriod":1000}

步骤4：ESL收到数据后，将在3秒内向App返回应答，应答消息格式如下：

|  |  |  |
| --- | --- | --- |
| 字段 | 字段长度 | 含义 |
| DataType | 4bit | 数据帧类型：0x2  0x1: 鉴权数据帧  0x0: 二进制类型  0x2：JSON类型数据帧  0x4：ASCII类型数据帧 |
| PduTag | 4bit | 应答消息，固定为FrameSingle。   * FrameStart = 0, * FrameMiddle = 1, * FrameEnd = 2, * FrameSingle = 3 |
| AckDataSeq | 2Byte | 表示ESL已经收到的数据长度。 |
| RxWindow | 2Byte | 接收窗，ESL不支持 |
| AckCause | 2Byte | 成功应答码：  0：消息执行成功  错误应答如下：  259 输入的内容字段非法(JSON解析错误或者字段不支持) |

# 其它资源

图片压缩算法：

<https://github.com/kkmhogen/ESLBin2Json.git>

BIN图片生成工具：

https://github.com/kkmhogen/ESLIntroduction.git