**LLSync蓝牙设备接入协议**

未经授权，请勿扩散

**修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 修订日期 | 修订版本 | 修改描述 | 作者 |
| 20200817 | V0.1.0 | 发布第一版draft规范  协议版本号为0 | willssong  esma  markyyao  zekwang |
| 20200910 | V1.0.0 | event数据增加分片功能 | esma |
| 20200917 | V1.1.0 | 协议版本号变更  增加数据分片 | esma |
| 20201102 | V1.2.0 | 协议版本号变更  支持绑定、解绑数据分片 | zekwang |
| 20201119 | V1.3.0 | 协议版本号变更  增加OTA功能 | esma |
| 20210201 | V1.4.0 | 物模型支持结构体  修复一些bug | esma |
| 20210220 | V1.4.1 | 增加安全绑定功能 | esma |
| 20210515 | V1.5.0 | 增加配网功能 | esma |
| 20211129 | V1.6.0 | 增加数组类型 | wisonhu |

**修订流程**

对于本文档中任何内容的增删改以及相关其它文档的创建，都应该知会作者或者相关接口人。

**接口人**

本文档中的任何信息都应该被仔细的阅读。如果有任何疑虑，意见或问题，请直接联系下表中的接口人。

|  |  |  |  |
| --- | --- | --- | --- |
| **姓名** | **邮箱** | **电话** | **组织** |
| *willssong* | *willssong@tencent.com* |  | *腾讯云物联网产品中心* |
| *esma* | *esma@tencent.com* |  | *腾讯云物联网产品中心* |
| *markyyao* | *markyyao@tencent.com* |  | *腾讯云物联网产品中心* |
| *zekwang* | *zekwang@tencent.com* |  | *腾讯云物联网产品中心* |

**缩略语清单**

|  |  |  |
| --- | --- | --- |
| **缩略语** | **英文全名** | **中文解释** |
| *LLSync* |  | *腾讯连连Sync协议* |
| *BLE* | *Bluetooth Low Energy* | *低功耗蓝牙* |
| *LLDevice* |  | *蓝牙Sync设备管理属性* |
| *LLData* |  | *蓝牙Sync数据属性* |
| *LLEvent* |  | *蓝牙Sync事件属性* |
| *LLOTA* |  | *蓝牙Sync OTA属性* |

目录

[1. 引言 5](#_Toc89088940)

[1.1 背景 5](#_Toc89088941)

[1.2 目的 5](#_Toc89088942)

[2. 设备参数要求 5](#_Toc89088943)

[3. LLSync TLV格式 5](#_Toc89088944)

[4. LLSync Profile定义 7](#_Toc89088945)

[4.1 LLDeviceInfo 8](#_Toc89088946)

[4.2 LLData 10](#_Toc89088947)

[4.3 LLEvent 12](#_Toc89088948)

[4.4 LLOTA 13](#_Toc89088949)

[4.5 UUID说明 13](#_Toc89088950)

[5. LLSync Advertisement定义 13](#_Toc89088951)

[6. BLE通信数据流 15](#_Toc89088952)

[6.1子设备绑定 15](#_Toc89088953)

[6.2子设备连接 18](#_Toc89088954)

[6.3子设备解绑 19](#_Toc89088955)

[6.4数据模板协议交互 20](#_Toc89088956)

[6.5 设备信息上报 25](#_Toc89088957)

[6.6 设备OTA 26](#_Toc89088958)

[7. 蓝牙辅助配网 31](#_Toc89088959)

[7.1 广播数据和服务 31](#_Toc89088960)

[7.2 配网数据交互 32](#_Toc89088961)

## 引言

### 背景

腾讯连连是腾讯云面向物联网行业提供的一整套C to B开放平台服务，借助腾讯连连可以降低物联网产品的研发门槛并加快研发速度，同时提供以微信小程序为载体的、面向消费者的应用入口，整合腾讯内部的品牌以及多项优势内容服务，助力万物互联时代真正到来。

BLE设备在IoT设备中占比高，适用范围广，但由于BLE无法直接接入互联网，BLE类设备上云比较困难，开发门槛较高。为解决此问题，腾讯云物联网开发平台制定LLSync协议帮助BLE设备快速简单的完成上云。

### 目的

本文档描述了BLE设备接入腾讯连连平台的流程标准，帮助开发者更好的理解LLSync协议。

## 设备参数要求

|  |  |
| --- | --- |
| **参数项** | **要求** |
| BLE ATT MTU | >= 23 |
| BLE协议 | >= BLE 4.2 |

## LLSync TLV格式

腾讯云物联网为接入平台定义一套[数据模板协议](https://cloud.tencent.com/document/product/1081/34916)，将设备的接入形式通过JSON模板标准化。多数BLE设备受资源限制，较难承载JSON格式的数据交互，针对此定义了TLV格式的二进制数据包来表示数据模板，最大程度的减少资源占用。如无特殊说明，本文所有数据均使用网络序传输。

LLSync TLV二进制数据包中有用户数据、数据长度和数据类型，TLV格式被广泛应用在LLData数据包和LLEvent数据包中。

**LLSync TLV格式:**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | Type | Length | Value |
| 长度 | 1 Byte | N Bytes | N Bytes |
| 说明 | [Type字段定义](#type字段说明) | 可选 | 无 |

*说明：Type 字段决定 Length 字段是否存在。*

**Type字段说明:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 字段 | [数据类型定义](#数据类型定义) | | | [ID定义](#id含义说明) | | | | |

*说明：Type 字段高 3 Bits表示数据类型，低 5 Bits表示 ID。*

**数据类型定义**：

|  |  |  |  |
| --- | --- | --- | --- |
| 数据类型 | 值 | 数据长度 | 数据范围 |
| 布尔 | 0 | 1 Byte | 0/1 |
| 整数 | 1 | 4 Bytes | -2^31 ~ 2^31 - 1 |
| 字符串 | 2 | N(<= 2048)Bytes | 用户自定义数据 |
| 浮点数 | 3 | 4 Bytes | 1.2E-38 ~ 3.4E+38 |
| 枚举 | 4 | 2 Bytes | 0 ~ 2^16 - 1 |
| 时间 | 5 | 4 Bytes | 0 ~ 2^64 - 1 |
| 结构体 | 6 | N(<= 2048)Bytes | TLV格式数据 |
| 数组 | 7 | N(<= 2048)Bytes | TLV格式数据 |

**ID 含义说明**：

1. 在不同的数据包中 ID 含义略有不同：

* 属性(property)数据包中表示属性ID(property id)。
* 事件(event)数据包中表示事件的参数ID(params id)。
* 行为(action)数据包中表示行为的 input id 或 output id。

*说明及限制：*

1. *数据 ID 占据 5 Bits，最大值为****31****。*
2. *浮点数指的是****单精度浮点数****。*
3. *只有****字符串、结构体和数组类型****拥有****Length字段****。其他类型长度固定， TLV中省略Length字段。*

*示例如下：*

* *00 01表示id = 0， value = 1的布尔数据。此处省略了Length字段。*
* *41 00 05 68 65 6C 6C 6F表示 id = 1，length = 5，value = hello的字符串数据。此处Length字段为00 05。*
* *C2 00 0A 00 01 41 00 05 68 65 6C 6C 6F 表示 id = 2，length = 10的结构体数据。其成员1是 id = 0，value = 1的布尔数据，成员2是id = 1，length = 5，value = hello的字符串数据。此处结构体Length字段为00 0A，字符串Length字段为00 05。*

1. *结构体不支持嵌套，结构体成员只能是其他数据类型。*
2. *数组类型说明如下：*

* *不需要带长度类型的数组：eg：E8 00 08 00 00 04 00 00 00 08 00，表示id=8，内部元素为0x400和0x800。*
* *需要带长度类型的数组：eg：E0 00 0C 00 03 79 65 73 00 05 68 65 6C 6C 6F，表示id=0，其内容为"yes"和"hello"。*

*eg：E0 00 19 00 0B 80 00 07 40 00 05 68 65 6C 6C 6F 00 0A 80 00 03 40 00 04 74 65 73 74，表示id=0，第一个元素为0x80枚举，第二个元素为0x40字符串，其数值为：{{.enum=7, .string="hello"}，{.enum=3, .string="test"}}。*

## LLSync Profile定义

Profile总架构如图：

图形用户界面

描述已自动生成

LLSync Profile包含4个characteristics：

**LLDeviceInfo**：设备信息写入特征值，用于设备连接、绑定和身份确认。

**LLData**：数据模版操作特征值，用于通知设备端执行数据模版操作。

**LLEvent**：事件上报特征值，用于设备端向小程序上报数据。

**LLOTA**：升级数据特征值，用于控制设备进行版本更新。

LLSync数据包最大长度为 **2048** 字节，包括数据包头和用户数据。同时支持数据分片，当数据包长度大于ATT MTU时，LLSync 协议会将数据分片发送，接收方收到分片数据后需要将数据组包后处理。

### 4.1 LLDeviceInfo

LLDeviceInfo有两种格式的数据包。

LLSync协议版本等于0：

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 数据 |
| 长度 | 1 Byte | N Bytes |
| 说明 | [类型定义](#类型说明) | [数据格式定义](#数据格式定义) |

LLSync协议版本大于 0:

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 长度 | 数据 |
| 长度 | 1 Byte | 2 Bytes | N Bytes |
| 说明 | [类型定义](#类型说明) | [长度格式定义](#长度格式定义) | [数据格式定义](#数据格式定义) |

**类型定义**：

|  |  |
| --- | --- |
| 数据类型 | 值 |
| 时间同步 | 0 |
| 连接鉴权 | 1 |
| 绑定成功 | 2 |
| 绑定失败 | 3 |
| 解绑请求 | 4 |
| 连接成功 | 5 |
| 连接失败 | 6 |
| 解绑成功 | 7 |
| 解绑失败 | 8 |
| MTU设置结果 | 9 |
| 绑定确认超时 | 0x0A |
| 动态注册回复 | 0x0B |

**数据格式定义：**

数据格式字段由具体的数据类型定义。

|  |  |  |
| --- | --- | --- |
| 数据类型 | 数据格式 | 说明 |
| 0 | 4 Bytes Nonce +  4 Bytes Unix TS | 向设备端发送计算签名所需信息 |
| 1 | 4 Bytes Unix Ts +  20 Bytes Hmac-sha1 | Local Psk对Ts签名得到Hmac-sha1 |
| 2 | 1 Byte bind result +  4 Bytes local psk +  8 Bytes bind string | 小程序生成Local Psk和Bind String。  小程序记录Local psk和Bind string与设备的对应关系 |
| 3 | N/A | 绑定失败 |
| 4 | 20 Bytes Hmac-sha1 | 使用Local Psk对“UnbindRequest”签名得到Hmac-sha1 |
| 5 | N/A | 连接成功 |
| 6 | N/A | 连接失败 |
| 7 | N/A | 解绑成功 |
| 8 | N/A | 解绑失败 |
| 9 | 2 Bytes Result | 小程序设置MTU的结果 |
| 0x0A | N/A | 等待绑定确认超时 |

**长度格式定义**：

由于ATT MTU限制，当数据包长度大于 ATT MTU 时，LLSync 会对数据报文进行分片。

长度字段的15～14 Bits用来表示分片标记，13 Bit用来表示确认绑定，11～0 Bits表示数据长度，数据长度最大值为2^11 – 1字节,已经可以满足绝大多数的使用场景。**长度格式定义适用于本文档中所有涉及到数据分片的length字段**。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | … | 1 | 0 |
| 说明 | [分片标记定义](#分片标记定义) | | [绑定标记](#绑定标记) |  | [数据长度](#数据长度) | | | | |

**分片标记定义:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 分片值 | 00 | 01 | 10 | 11 |
| 说明 | 不分片 | 分片，首包 | 分片，中间包 | 分片，尾包 |

**长度计算方式：分片标记 << 14 ｜ 绑定标记 << 13 | 数据长度。**

**绑定标记**：

|  |  |
| --- | --- |
| Bind Flag | 说明 |
| 0 | 同意绑定 |
| 1 | 拒绝绑定 |

*说明：*

1. *数据分为 2 包时，只有分片首包和尾包。*
2. *数据长度指的是本包内有效载荷的长度，不包含长度字段本身占用的长度。*
3. *连接鉴权期间LLSync认为设备使用的ATT\_MTU固定是23。*
4. *绑定标记只有启用安全绑定功能时才有效，其他场景默认为0。*
5. *数据分片是将较长的用户数据分割为多个短数据包后进行传输，每个短数据包的传输格式都需要符合本文档中相应的报文格式。*

*示例：完整的连接鉴权信息共 24 字节，如下：*

*0xA1,0xA2,0xA3,0xA4,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0xBA,0xBB,0xBC,0xBD,0xBE,0xBF,0xC0,0xC1,0xC2,0xC3。*

*当ATT MTU为23字节时，单包用户数据长度最大为20字节。因此需要将连接鉴权信息分为2包发送，第一包数据如下：*

*0x01,0x40,0x11,0xA1,0xA2,0xA3,0xA4,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5, 0xB6,0xB7,0xB8,0xB9,0xBA,0xBB,0xBC*

*其中，0x01表示连接鉴权信息，0x40，0x11表示分片数据的第一包，数据长度17字节。*

*第二包数据如下：*

*0x01,0xC0,0x07,0xBD,0xBE,0xBF,0xC0,0xC1,0xC2,0xC3*

*其中，0x01表示连接鉴权信息，0xC0，0x07表示分片数据最后一包，数据长度7字节。*

### 4.2 LLData

LLData用于操作数据模版，不同的业务数据包中字段含义略有不同。

**LLData 格式定义**：

|  |  |  |
| --- | --- | --- |
| 字段 | Fixed header | Payload |
| 长度 | 1 Byte | N Bytes |
| 说明 | [Fixed header定义](#_固定头部定义) | [Payload定义](#payload定义) |

#### 4.2.1 Fixed header定义:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 说明 | [数据模版类型定义](#数据模版类型定义) | | [数据作用定义](#数据作用定义) | [ID 定义](#id定义) | | | | |

数据模版操作包括属性、事件、行为三类，通过7 ～ 6 Bits标记。Bit 5用来标记是物联网平台下发的请求报文还是物联网平台对设备的应答报文。Bits 4 ～ 0 表示数据模版的ID。

**数据模版类型定义**：

|  |  |
| --- | --- |
| 字段 | 数值 |
| property | 0 |
| event | 1 |
| action | 2 |

**数据作用定义**：

|  |  |  |
| --- | --- | --- |
| 字段 | 数值 | 说明 |
| Request | 0 | 向设备下发请求 |
| Reply | 1 | 应答设备请求 |

**ID定义**：

|  |  |  |  |
| --- | --- | --- | --- |
| 数据模版类型 | 数据作用 | ID | 说明 |
| 0 | 0 | 0 | N/A |
| 1 | 0 | 表示report\_reply方法 |
| 2 | 表示get\_status\_reply方法 |
| 1 | 1 | event id | 表示事件id |
| 2 | 0 | action id | 表示行为 id |

*说明：event id/action id在不得超过31。*

#### 4.2.2 Payload定义

不同的Fixed header对应的Payload格式不同。

**Payload格式定义**：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据模版类型 | 数据作用类型 | ID | Payload | 说明 |
| 0 | 0 | 0 | [TLV格式](#tlv格式) | [见6.4.2](#_6.4.2_设备远程控制) |
| 1 | 0 | [Reply\_Result定义](#应答结果) | [见6.4.1](#_6.4.1_设备属性上报) |
| 2 | [TLV格式](#tlv格式) | [见6.4.3](#_6.3子设备解绑) |
| 1 | 1 | event id | [Reply\_Result定义](#应答结果) | [见6.4.4](#_6.4.4_设备事件上报) |
| 2 | 0 | action id | [TLV格式](#tlv格式) | [见6.4.5](#_6.4.5_设备行为调用) |

**Reply\_Result定义:**

|  |  |
| --- | --- |
| 数值 | 说明 |
| 0 | 成功 |
| 1 | 失败 |
| 2 | 数据解析错误 |

*说明：表示本次操作操作结果。*

### 4.3 LLEvent

LLEvent用于设备主动上报报文，主要用于对LLDeviceInfo、LLData和LLOTA的回复。

**LLEvent格式定义:**

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | type | length | value |
| 长度 | 1 byte | 2 bytes | N bytes |
| 说明 | [类型定义](#event类型定义) | [长度定义](#_长度格式定义) | 无 |

**Event类型定义**：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型值 | 说明 | Value |
| 属性上报 | 0 | 数据模版中的report | [见6.4.1](#_6.4.1_设备属性上报) |
| 控制回复 | 1 | 数据模版中的control\_reply | [见6.4.2](#_6.4.2_设备远程控制) |
| 获取最新信息 | 2 | 数据模版中的get\_status | [见6.4.3](#_6.4.3_获取设备最新信息) |
| 事件上报 | 3 | 数据模版中的event\_post | [见6.4.4](#_6.4.4_设备事件上报) |
| 行为响应 | 4 | 数据模版中的action\_reply | [见6.4.5](#_6.4.5_设备行为调用) |
| 绑定鉴权信息 | 5 | 绑定后设备返回的信息 | [见6.1](#_6.1子设备绑定) |
| 连接鉴权信息 | 6 | 连接后设备返回的信息 | [见6.2](#_6.2子设备连接) |
| 解绑鉴权信息 | 7 | 解绑后设备返回的信息 | [见6.3](#_6.3子设备解绑) |
| 设备信息 | 8 | 上报MTU长度和协议版本 | [见6.5](#_6.5_设备信息上报) |
| 升级请求回复 | 9 | 回复设备升级请求 | [见6.6.3](#_6.6.3_升级请求应答包) |
| 升级数据包回复 | 10 | 回复升级数据包 | [见6.6.5](#_6.6.5_升级数据应答包) |
| 升级校验结果回复 | 11 | 回复升级文件的校验结果 | [见6.6.7](#_上报固件检查结果) |
| MTU协商结果上报 | 12 | 上报最终协商确定的MTU | [见6.5](#MTU协商) |
| 等待绑定时间上报 | 13 | 等待绑定确认的最大时间 | [见6.1](#_6.1子设备绑定) |
| 动态注册请求 | 14 | 设备请求动态注册 |  |

### 4.4 LLOTA

LLOTA用于对设备进行版本更新。

**LLOTA格式定义**：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | type | length | value |
| 长度 | 1 byte | 1 byte | N bytes |
| 说明 | [类型定义](#ota类型定义) | 无 | [见6.6](#_6.6_设备OTA) |

**OTA类型定义**：

|  |  |
| --- | --- |
| 类型定义 | 值 |
| 升级请求 | 0 |
| 升级数据包 | 1 |
| 升级结束通知 | 2 |

### 4.5 UUID说明

LLSync Bluetooth Base UUID为 00000000-65d0-4e20-b56a-e493541ba4e2。

按照BLE协议，16 Bits UUID和128 Bits UUID转换关系为

128 Bits value = 16 Bits value \* 2^96 + BluetoothBaseUUID

即 0000xxxx-65d0-4e20-b56a-e493541ba4e2 中的 xxxx 替换为16 Bits UUID，例如 Service 16 BitsUUID FFE0 转换为128 Bits 的UUID 为 0000ffe0-65d0-4e20-b56a-e493541ba4e2，Characteristic的UUID的转换类似。

## LLSync Advertisement定义

自定义广播数据按照BlueTooth协议要求，添加到0xFF Manufacturer Specific Data的字段当中，company ID使用0xFEE7（Tencent Holdings Limited），0xFEE7和0xFEBA均为腾讯申请的Company ID。

**广播包格式**：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 说明  状态 | 设备状态 | | 设备标识 | | 附加标识 | |
| 长度 | 取值 | 长度 | 取值 | 长度 | 取值 |
| 未绑定 | 1 | [设备](#设备状态定义)  [状态](#设备状态定义)  [定义](#设备状态定义) | 6 | MAC地址 | 10 | Product ID |
| 绑定中 | 1 | 6 | MAC地址 | 10 | Product ID |
| 已绑定 | 1 | 8 | [设备标识计算](#设备标示计算) | 8 | [绑定标识计算](#绑定标示计算) |

*说明：未绑定和绑定中广播内容相同。当启用 BLE\_QIOT\_BUTTON\_BROADCAST 功能后，可以控制在未绑定状态下不广播，当通过按键或其他操作进入绑定中状态后才开启广播。*

**设备状态定义**：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 说明 | 协议版本 | | | |  | 动态注册 | [绑定状态](#绑定状态说明) | |

*说明：*

* *LLSync SDK版本号见 BLE\_QIOT\_LLSYNC\_PROTOCOL\_VERSION 定义。*
* *当设备需要动态注册时，动态注册标记置为 1。*

**绑定状态说明:**

|  |  |  |
| --- | --- | --- |
| 状态 | 值 | 说明 |
| 未绑定 | 0 | 初始状态，可以选择不发送BLE Advertisement以省电 |
| 绑定中 | 1 | 按下绑定触发按键，在超时之前处于绑定中，超时前按照要求发送广播包 |
| 已绑定 | 2 | 正确完成绑定状态，需要持续发送广播包 |

**设备标识计算**：

Temp = md5sum(蓝牙设备的product\_id | 蓝牙设备的device\_name)

设备标识Result = Temp前8位 ^ Temp后8位

比如：蓝牙设备的ProductID为ABCDEFGHIJ，DeviceName为Dev01

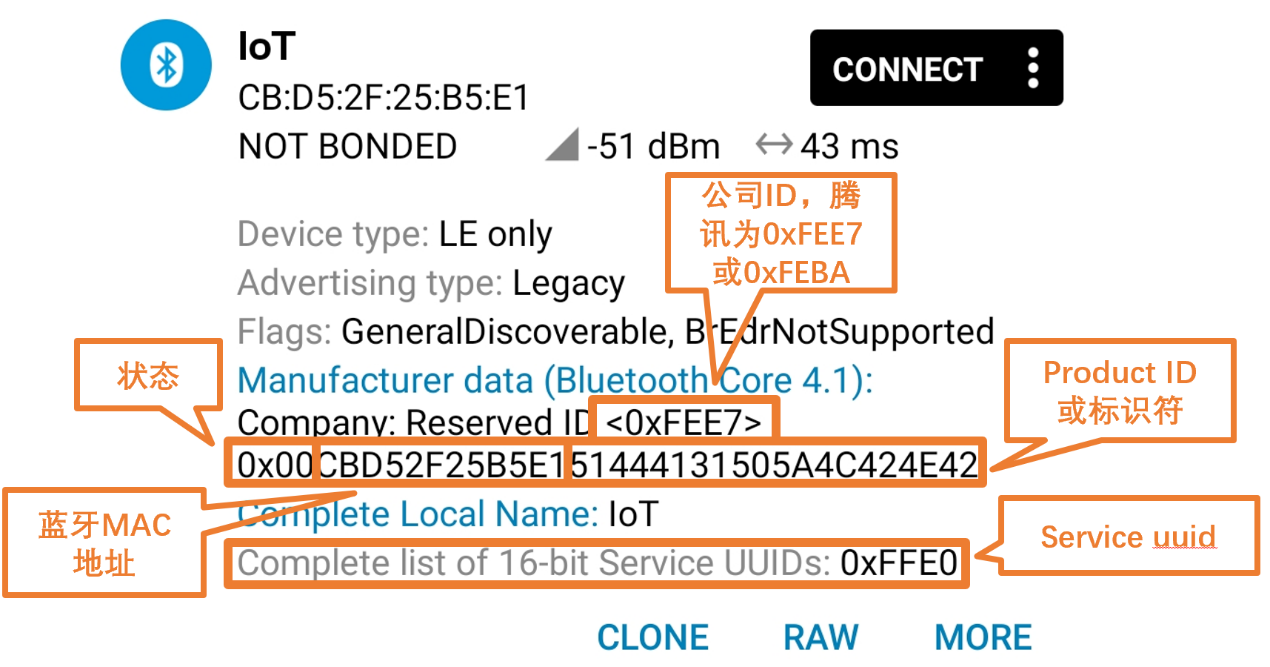
那么Temp = md5sum(“ABCDEFGHIJDev01”) = { 0x61, 0x2a, 0xf7, 0x9d, 0x50, 0x17, 0x93, 0x87, 0x2a, 0x4a, 0x97, 0xe8, 0xcb, 0xe4, 0x5a, 0x10}

Result为 {0x4b, 0x60, 0x60, 0x75, 0x9b, 0xf3, 0xc9, 0x97}

**绑定标识计算**：

网关或小程序在绑定成功时提供，计算方式和绑定标识符一致。

**广播包示例如下**：



## BLE通信数据流

### 6.1子设备绑定

场景：BLE终端尚未绑定，需要先进行绑定才可以连接。

图示

描述已自动生成

1. 往LLDeviceInfo上写入Unix TS。

|  |  |  |
| --- | --- | --- |
| Type | Value | |
| nonce | timestamp |
| 0x00 | 1. Bytes nonce | 4 Bytes timestamp |

1. LLSync 支持设备动态注册，在设备绑定时可以自动在物联网开发平台注册设备。当设备收到 Unix TS时回复报文，报文内包含设备动态注册所需信息。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Length | Value | | |
| Device Name Length | Device Name | Sign |
| 0x0E | 2 Bytes length | 1 Bytes | N Bytes | (Length – 1 - N) Bytes |

小程序将签名信息透传到后台，后台进行动态注册并且回复小程序。小程序将返回信息透传给设备，设备继续绑定流程。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type | Length | Value | | | | |
| Result | Payload Length | Payload | Nonce | Timestamp |
| 0x0B | 2 Bytes length | 1 Byte | 1 Byte | N Bytes | 4 Bytes | 4 Bytes |

* 通过配置SDK的BLE\_QIOT\_DYNREG\_ENABLE 可以启用动态注册功能。

1. 出于设备安全考虑，LLSync SDK支持安全绑定功能，绑定前需要设备端确认。

* 该功能是可选功能，通过配置SDK的BLE\_QIOT\_SECURE\_BIND可以启用安全绑定功能。若不启用该功能则设备进入绑定验证签名流程(见步骤3)；
* 若启用安全绑定功能，设备端需要等待用户确认后才可以继续绑定流程。

启用安全绑定功能时，设备端需要先通过LLEvent告知小程序绑定过程的最大超时时间，用户超时未操作时结束绑定流程。

|  |  |  |
| --- | --- | --- |
| Type | Length | Value |
| 0x0D | 2 Bytes length | 2 Bytes Wait Time |

*说明：绑定确认超时时间单位为秒，占用2字节。默认60秒。*

启用安全绑定功能后，用户选择确认绑定、拒绝绑定时，设备端会通过验证签名报文(见步骤3)上报小程序用户操作结果。

如果用户不做操作导致绑定超时或者在小程序上取消绑定，小程序会通过LLDeviceInfo报文通知设备。

|  |  |
| --- | --- |
| Type | Result |
| 0x0A | 0/1 |

*说明：*

1. *Result 为 0 表示用户在小程序上点击取消，为 1 表示连接超时。*
2. *小程序通知设备超时，是考虑到节约设备资源。设备也可以依靠自身能力检测绑定超时。*
3. 设备验证签名后返回的LLEvent数据包。

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Length | Value | |
| sign info | device name |
| 0x05 | 2 Bytes length | 20 Bytes sign info | N Bytes device name |

*说明：*

1. *sign info是通过设备的psk对设备信息签名得到，签名算法使用hamc-sha1。*
2. *deviceinfo = product id + devicename + ; + nonce + ; + expiration time*
3. *expiration time = timestamp + 60*
4. *计算签名时对于 nonce 和 timestamp，将其转换为字符串类型后再计算签名，避免大小端问题导致的签名错误。示例：timestamp = 0x5f3279fa，转换为对应数值的字符串为“1597143546”*
5. *启用安全绑定功能后，Length字段需要指定[绑定标记](#长度格式定义)。*
6. 往FFE1写入绑定成功结果格式见下表

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Value | | |
| Result | Local Psk | 绑定标识符 |
| 0x02 | 02 | 4 Bytes Local Psk | 8 Bytes |

*说明:*

1. *Local Psk和绑定标识符由小程序生成。*
2. *Result表示绑定状态，绑定成功固定为0x02。*
3. 往FFE1写入绑定失败结果格式见下表

|  |  |
| --- | --- |
| type | value |
| 0x03 | 1 Byte [Reply\_Result](#应答结果) |

*说明：*

1. *BLE终端不会校验网关/小程序的身份，存在BLE终端被恶意绑定的可能，BLE终端可以配置通过按键进入待绑定状态，默认2分钟有效。*
2. *设备连接成功之后，不会再广播beacon，小程序/网关无法再次扫描。*
3. *如果绑定成功，需要在设备上存储 Local Psk 用于后续的网关 + 子设备连接鉴权。*

### 6.2子设备连接

场景：设备广播Beacon标识设备已绑定，需要进行连接

日程表

描述已自动生成

*说明：如果是小程序，写入上线结果认为是写入小程序和设备的连接结果。*

1. 往LLDeviceInfo写入签名信息数据格式见下表

|  |  |  |
| --- | --- | --- |
| Type | Value | |
| Timestamp | Sign info |
| 0x01 | 4 Bytes timestamp | 20 Bytes sign info |

*说明：sign info是使用Local Psk对timestamp进行签名，算法选择hmac-sha1。*

1. 设备验证签名后返回的LLEvent格式数据包

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Length | Value | |
| Sign info | Device name |
| 0x06 | 2 Bytes length | 20 Bytes sign info | N Bytes device name |

*说明：*

1. *sign info是使用Local Psk对设备信息进行签名，算法选择hmac-sha1。设备信息包括 expiration time + product id + device name，其中expiration time = timestamp + 60。*
2. *计算签名时对于timestamp，将其转换为字符串类型后再计算签名，避免大小端问题导致的签名错误。示例：timestamp = 0x5f3279fa，转换为对应数值的字符串为“1597143546”。*

### 6.3子设备解绑

场景：子设备已经绑定且完成连接，小程序端请求解绑

图示, 日程表

描述已自动生成

1. 往LLDeviceInfo写入解绑请求

|  |  |
| --- | --- |
| Type | Value |
| 0x04 | 1. Bytes sign info |

*说明：sign info是使用Local Psk对固定字符串“UnbindRequest”进行签名，算法选择hmac-sha1。*

1. 验签后返回的LLEvent信息数据格式如下表

|  |  |  |
| --- | --- | --- |
| Type | Length | Value |
| 0x07 | 2 Bytes length | 20 Bytes sign info |

*说明：sign info是使用Local Psk对固定字符串“UnbindResponse”进行签名，算法选择hmac-sha1。*

3.往LLDeviceInfo写入解除绑定成功。

|  |
| --- |
| Type |
| 0x07 |

4.往LLDeviceInfo写入解除绑定失败。

|  |
| --- |
| Type |
| 0x08 |

### 6.4数据模板协议交互

#### 设备属性上报

图片包含 日程表

描述已自动生成

1. 设备属性上报LLEvent 数据格式，对应数据模版的Report操作。

|  |  |  |
| --- | --- | --- |
| Type | Length | Property Value |
| 0x00 | 2 Bytes length | [TLV数据](#tlv格式) |

*说明：属性支持增量上报，即可以省略某些属性。*

Property Value中可以包含多个Property的数据。示例

|  |  |
| --- | --- |
| 数值 | 描述 |
| 00 | Type |
| 00, 0F | Length |
| 00，01 | Propert Power Switch = 1 |
| 81，00，01 | Property Color = 1 |
| 22，00，00，00，23 | Property Brightness = 0x23 |
| 43，00，02，31，32 | Property Name = “12” |

1. 属性上报结果通过LLData通知设备，对应数据模版的report\_reply操作

|  |  |
| --- | --- |
| Header | Value |
| 0x20 | 1 Byte [Reply\_Result](#应答结果) |

#### 6.4.2 设备远程控制

图片包含 图示, 日程表

描述已自动生成

1. 通过LLData远程控制设备，对应数据模版的control操作.

|  |  |  |
| --- | --- | --- |
| Type | Length | Property Value |
| 0x00 | 2 Bytes length | [TLV数据](#tlv格式) |

*说明：属性支持增量下发，即可以省略某些属性。*

Property Value中可以包含多个Property的数据。示例

|  |  |
| --- | --- |
| 数值 | 描述 |
| 00 | Type |
| 00，0F | Length |
| 00，01 | Property Power Switch = 1 |
| 81，00，01 | Property Color = 1 |
| 22，00，00，00，23 | Property Brightness = 0x23 |
| 43，00，02，31，32 | Property Name = “12” |

1. 设备通过LLEvent上报操作结果，对应数据模版的control\_reply操作

|  |  |  |
| --- | --- | --- |
| Type | Length | Value |
| 0x01 | 2 Bytes length | 1 Byte [Reply\_Result](#应答结果) |

#### 

#### 6.4.3 获取设备最新信息

图示

描述已自动生成

1. 设备通过LLEvent获取最新信息，对应数据模版的get\_status操作

|  |
| --- |
| Type |
| 0x02 |

1. 小程序通过LLData下发最新信息，对应数据模版的get\_status\_reply操作

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Result | Length | Property Value |
| 0x22 | 1 Byte Reply\_Result | 2 Bytes value length | [TLV数据](#tlv格式) |

*说明：设备最新信息支持增量下发，即可以省略某些属性。*

Property Value中可以包含多个property的数据。

|  |  |
| --- | --- |
| 数值 | 描述 |
| 22 | Type |
| 00 | Reply\_Result = 成功 |
| 00，0F | Length |
| 00，01 | Property Power Switch = 1 |
| 81，00，01 | Property Color = 1 |
| 22，00，00，00，23 | Property Brightness = 0x23 |
| 43，00，02，31，32 | Property Name = “12” |

#### 

#### 6.4.4 设备事件上报

图片包含 图示

描述已自动生成

1. 设备通过LLEvent上报事件，对应数据模版中的event\_post操作

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Length | Event id | Event value |
| 0x03 | 2 Bytes length | 1 Byte event id | [TLV数据](#tlv格式) |

*说明：事件支持增量上报。*

Event value中可以包含多个event 参数。示例

|  |  |
| --- | --- |
| 数值 | 描述 |
| 03 | Type |
| 00，11 | Length |
| 02 | Event id |
| 40， 00，08， 31，32，33，34，35，36，37，38 | Event Param Name = “12345678” |
| 21，00，00，04，00 | Event Param Error Code = 0x400 |

1. 通过LLData返回操作结果，对应数据模版中的event\_reply操作

|  |  |
| --- | --- |
| Type | Value |
| [见章节4.3](#_4.3_LLEvent) | 1 Byte [Reply\_Result](#应答结果) |

*说明：假设event id = 0，那么Type字段应该是0x60。*

#### 

#### 6.4.5 设备行为调用

图示

描述已自动生成

1. 通过LLData向设备发起行为调用请求，对应数据模版中的action操作

|  |  |  |
| --- | --- | --- |
| Type | Length | Action Value |
| [见章节4.2](#_4.2_LLData) | 2 Bytes length | [TLV数据](#tlv格式) |

*说明：假设action id = 0，那么Type字段应该是0x80*。

Action value中可以包含多个 input 参数。示例

|  |  |
| --- | --- |
| 数值 | 描述 |
| 80 | Type |
| 00, 0B | Length |
| 20，00，00，00，04 | input id interval = 0x04 |
| 41，00，04，31，32，33，34 | input id message = “1234” |

1. 设备通过LLEvent上报行为调用结果，对应数据模版中的action\_reply操作

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Length | Value | | |
| Result | Action id | Response params |
| 0x04 | 2 Bytes length | 1 Byte [Reply\_Result](#应答结果) | 1 Byte action id | [TLV数据](#tlv格式) |

Response param中可以包含多个 response 参数。示例

|  |  |
| --- | --- |
| **数值** | **描述** |
| 04 | Type |
| 00，0F | Length |
| 00 | Reply Result = 成功 |
| 00 | action id = 0 |
| 00，01 | Response Result = 1 |
| 41， 00，08， 31，32，33，34，35，36，37，38 | Response message = “12345678” |

### 6.5 设备信息上报

1. 连接成功后，设备通过LLEvent主动向小程序/网关上报设备信息，包括协议版本号，设备需要设置的MTU大小和设备固件版本号。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type | Length | Value | | | |
| LLSync version | [MTU Filed](#mtufiled定义) | Firmware version | |
| Length | Payload |
| 0x08 | 2 Bytes | 1 Byte | 2 Bytes | 1 Byte | N (<=32) Bytes |

*说明：*

1. *版本号与[LLSync Advertisement](#_LLSync_Advertisement定义)中必须一致。*
2. *示例，08 00 09 02 00 14 05 30 2e 30 2e 31，LLSync版本号为2，mtu大小设置为0x14，固件版本号长度5字节，固件版本号为”0.0.1”。*

MTU Filed 定义：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bit | 15 | 14 | … | 11 | 10 | … | 0 |
| 说明 | mtu flag | Reserved | | | MTU大小 | | |

*说明：*

1. *Bits 0 – Bits 10用来表示设备端通信使用的 MTU 大小mtu\_size；*
2. *Bits 15 用来向小程序表示是否设置 MTU。当mtu flag为 1 时，小程序需要按照设备上传的mtu\_size 进行MTU设置；当mtu flag为0时，小程序不设置 MTU ，使用mtu\_size进行分片。*

*需要小程序去设置 MTU 的原因是：在安卓手机上如果小程序不显式设置 MTU，双方会使用默认MTU为23进行通信；在IOS上不存在该问题。*

1. *Bits 11 – Bits 14预留。*

2. 小程序收到设备信息上报后，需要检查[mtu flag](#mtuflag)。当mtu flag设置为1时：

安卓系统上，小程序需要调用 MTU 设置接口修改 MTU，并通过LLDeviceInfo通知设备端设置结果。

|  |  |
| --- | --- |
| Type | Value |
| 0x09 | 2 Byte Result |

*说明：*

1. *0表示设置成功，0xFFFF表示设置失败，其他表示设置成功的MTU值。*
2. *当前小程序只能获取到设置成功失败，无法获取到设置成功的具体MTU值。*

IOS系统上，小程序无法设置MTU，在蓝牙连接时IOS系统会设置MTU，LLSync SDK可以直接上报IOS系统设置的MTU给小程序用来通信。

3. 在手机上设置 MTU 后，由于小程序无法得知设置成功的MTU数值，因此还需要设备通过LLEvent将最终的MTU 数值上报给小程序，最终完成MTU的协商。

|  |  |  |
| --- | --- | --- |
| Type | Length | Value |
| MTU size |
| 0x0C | 2 Bytes | 2 Bytes |

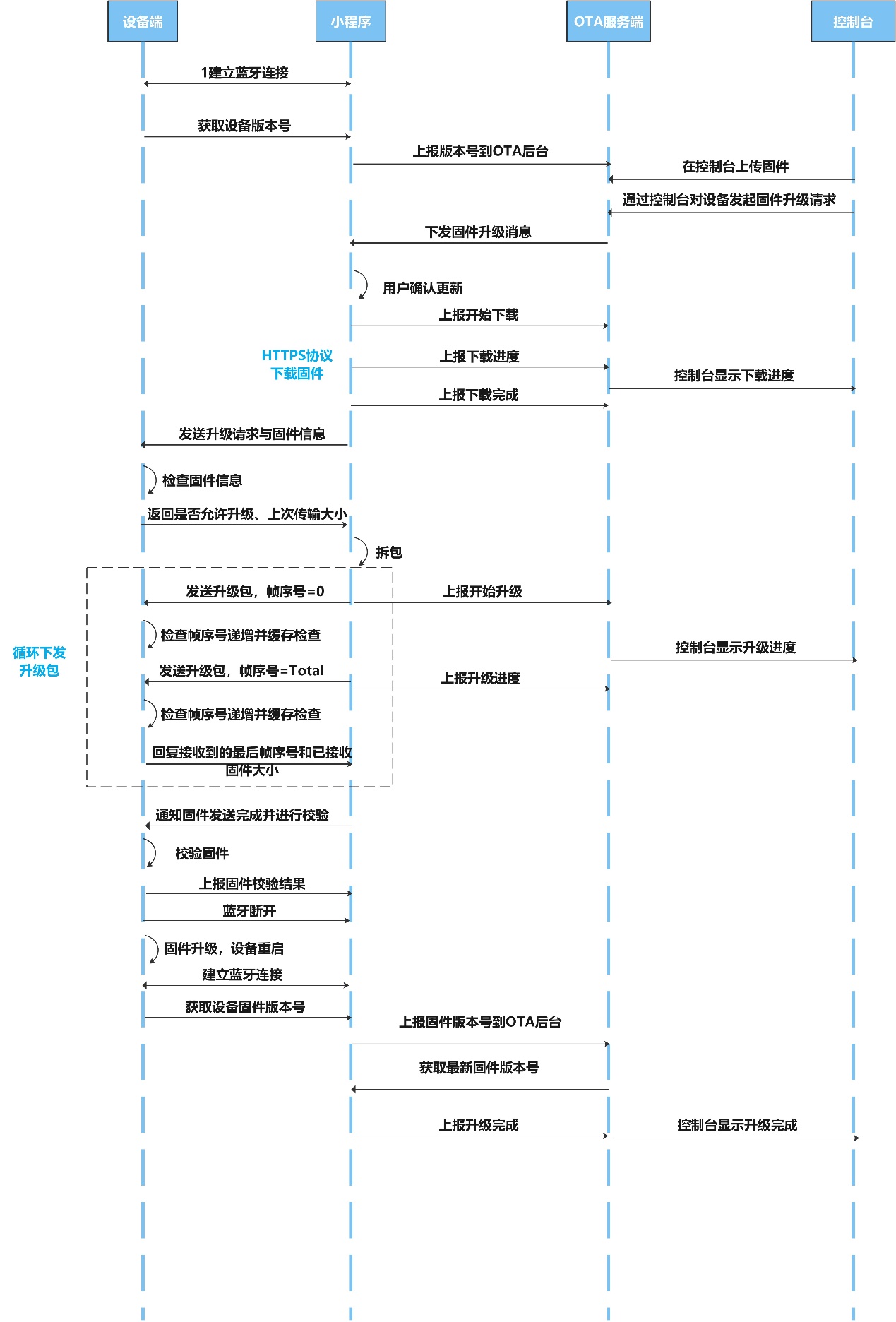
*说明：*

1. *在安卓上小程序设置 MTU 失败时，设备端上报MTU = 20，即默认ATT\_MTU – 3。*
2. *在安卓上小程序设置 MTU 成功，设备端会将蓝牙SDK获取到最新MTU值上报给小程序。*
3. *在IOS上，当IOS系统设置MTU后，设备端会将蓝牙SDK获取到最新MTU值上报给小程序。*

### 6.6 设备OTA

设备OTA流程图如下，设备端只关心和小程序的数据交互。包括：

* 设备端主动上报版本号
* 小程序下发升级请求
* 设备端应答升级请求
* 小程序下发升级数据包
* 设备端应答升级数据包
* 小程序通知下发结束
* 设备端上报文件校验结果



#### 6.6.1 固件版本上报

设备通过 LLEvent 进行固件版本号上报，见 [6.5](#_6.5_设备信息上报) 中一并上报。

#### 6.6.2 升级请求包

小程序通过 LLOTA 下发升级请求包到设备。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type | Length | Value | | | |
| File size | File crc | File version len | File version |
| 0x00 | 2 Bytes | 4 Bytes | 4 Bytes | 1 Byte | 1 ~ 32 Bytes |

*说明：*

1. *约定使用CRC32 进行文件校验。*
2. *升级请求包分片规则请参见* [*LLEvent 分片规则*](#长度格式定义)*。*

*示例：*

*00 00 0e 00 00 00 ff 18 70 16 3c 05 30 2e 30 2e 31，文件大小为0xFF，文件CRC为0x1870163C，文件版本为0.0.1*

*对上述数据应用分片规则，可以分为三包：*

*00 40 04 00 00 00 ff*

*00 80 04 18 70 16 3c*

*00 c0 06 05 30 2e 30 2e 31*

*也可以分为两包:*

*00 40 08 00 00 00 ff 18 70 16 3c*

*00 c0 06 05 30 2e 30 2e 31*

*分包数量也可以大于三包，大于三包时会有多个0x00,0x80开头的数据包。*

*分包数量取决于数据长度和ATT MTU大小，LLSync会自动处理分包和组包，用户无需关心。*

#### 6.6.3 升级请求应答包

设备通过 LLEvent 对升级请求作出应答。

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Length | Value | |
| Indicate | Payload |
| 0x09 | 2 Bytes | 1 byte | N Bytes |

升级请求应答包中value由1字节indicate和N字节的payload构成。

* indicate表示升级请求的请求结果。
* payload是请求结果的延伸字段。

**indicate定义**：

|  |  |
| --- | --- |
| Bit | 说明 |
| 0 | 0: 禁止升级  1: 允许升级 |
| 1 | 0: 不支持断点续传  1: 支持断点续传 |
| 2 ～ 7 | Reserved |

不同的indicate字段会有不同的payload。

**当允许升级时payload定义如下**：

|  |  |
| --- | --- |
| 字段 | 说明 |
| 1 byte total package numbers | 单次循环中可以连续传输的数据包个数，取值范围0x00 ~ 0xFF。 |
| 1 byte package length | 单个数据包大小，取值范围 0x00 ~ 0xF0。 |
| 1 byte data retry time | 数据包的超时重传周期，单位：秒 |
| 1 byte device reboot time | 设备重启最大时间，单位：秒 |
| 4 bytes last received file size | 断点续传前已接收文件大小 |
| 1 byte package send interval | 小程序连续两个数据包的发包间隔 |

*说明：*

1. *不支持断点续传时，已接收文件大小恒为0。*
2. *小程序连续 5 个超时重传周期内没有收到设备端回应，认为升级失败。*
3. *设备重启最大时间是设备下载成功后重启设备，小程序等待设备上报新版本号的最大时间，超出此时间小程序认为升级失败。*
4. *升级请求应答包分片规则请参见* [*LLEvent 分片规则*](#长度格式定义)*。*

*示例：*

*0a 00 09 03 10 0f 05 14 00 00 00 00，表示设备端允许升级且支持断点续传，单次循环传输0x10个数据包，每个数据包数据长度为0x0F，数据包超时设置为5秒，设备重启时间最大为20秒，断点续传前文件大小为0*

**当禁止升级时payload表示禁止升级的原因**：

|  |  |
| --- | --- |
| 错误码 | 说明 |
| 2 | 设备电量不足 |
| 3 | 版本号错误 |

*示例：*

*0a 00 02 00 02,表示设备端禁止升级，因为设备电量过低*

#### 6.6.4 升级数据包

小程序通过 LLOTA 下发升级数据包到设备。

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Length | Value | |
| Seq | Payload |
| 0x01 | 1 Byte | 1 Byte | N Bytes |

*说明：*

1. *length字段表示seq和payload的长度之和。*
2. *seq表示数据包在单次循环中的序列号，从0开始，每一包数据增加1，直到total package numbers – 1结束，单次循环结束后重新从0开始。*

*示例：*

*01 10 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01，表示seq为0x01的数据包，数据总长度为0x10，有效数据长度为0x0F，即0x0F个0x01。*

#### 6.6.5 升级数据应答包

设备通过 LLEvent 对升级数据包作出应答。

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Length | Value | |
| Next Seq | File size |
| 0x0A | 2 Bytes | 1 byte | 4 Bytes |

*说明：*

1. *next seq是设备收到的数据包的seq的下一个seq，file size是设备已接收的正确文件的大小。*
2. *设备收到单个循环的所有数据包后，使用next seq和file size对此次循环作出应答，小程序收到应答后再发送下一循环的数据包数据包。*
3. *设备收到错误的seq时，发送应答包给小程序请求重传，小程序根据设备上报的next seq和file size重新传输数据，小程序应该从file size处开始传输，seq等于next seq。*
4. *当传输出错时，在一个数据重传周期内，设备端只会上报一次数据应答包。*
5. *连续5个数据重传周期内没有收到正确的数据包，设备端认为升级失败，用户可以控制断开连接。*
6. *升级数据包最后一个循环中数据包可能不足total package numbers，设备会根据文件大小计算，以便在收到最后一个数据包时仍然可以发送数据应答包。*

*示例：*

*0b 00 05 0f 00 00 00 f0，表示设备端收到的最后一个数据包的seq为0x0F，设备当前接收的正确文件的大小为0xF0*

#### 6.6.6 升级数据结束通知包

小程序通过 LLOTA 通知设备升级数据包下发结束。

|  |
| --- |
| Type |
|
| 0x02 |

*说明：*

1. *小程序文件下发结束后通知设备端进行固件检查并上报结果。*

#### 6.6.7 上报固件检查结果

设备通过 LLEvent 上报升级文件的校验结果。

|  |  |  |
| --- | --- | --- |
| type | length | value |
| 0x0B | 2 Bytes | [校验结果定义](#校验结果定义) |

**校验结果定义**：

|  |  |
| --- | --- |
| Bit | 说明 |
| 7 | 1 ： 校验通过  0 ： 校验失败 |
| 6 ～ 0 | 0 ： 文件CRC错误  1 ： flash操作失败  2 ： 文件内容错误 |

*说明：*

1. *使用 1 字节表示校验结果，Bit 7 表示校验是否通过，如果文件校验错误，Bit 6 ～ 0表示具体的错误原因。*

*示例：0c 00 01 80，表示文件校验通过*

## 7. 蓝牙辅助配网

本章节介绍基于LLSync协议的蓝牙辅助配网规范，LLSync配网功能沿用了LLSync标准蓝牙功能的部分协议，例如数据分片协议、MTU设置协议等。LLSync配网功能只做网络配置，不涉及连接鉴权、设备绑定等功能。

### 7.1 广播数据和服务

LLSync辅助配网功能广播数据包和LLSync标准蓝牙功能广播数据包不同。

1. Service UUID为**0xFFF0**。
2. 厂商数据格式固定。

|  |  |  |
| --- | --- | --- |
| LLSync协议版本号 | MAC | Product ID |
| 0x02 | 6 字节设备地址 | 10 字节产品ID |

广播数据包示例：

文本

描述已自动生成

LLSync 辅助配网功能使用了LLDeviceInfo（0xFFE1）和LLEvnet（0xFFE3）两个特征值。小程序通过LLDeviceInfo向设备写入数据，设备通过LLEvent向小程序上报数据。

蓝牙服务示例：

图形用户界面, 应用程序

描述已自动生成

### 7.2 配网数据交互

#### 7.2.1 设备信息获取

1. 蓝牙连接成功后，小程序通过LLDeviceInfo向设备下发信息获取指令。

|  |
| --- |
| Type |
| 0xE0 |

*该报文不携带任何内容。*

2. 设备通过LLevnet向小程序回复设备信息。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type | Length | Value | | | |
| LLSync version | [MTU Filed](#mtufiled定义) | Device name | |
| Length | name |
| 0x08 | 2 Bytes | 1 Byte | 2 Bytes | 1 Byte | N (<=48) Bytes |

1. *配网功能和标准蓝牙功能的该报文Value部分不一致，对比章节[6.5](#_6.5_设备信息上报)。*
2. *MTU设置功能参考章节6.5。*

#### 7.2.2 WIFI模式设置

1. 小程序通过LLDeviceInfo向设备下发指令设置WIFI模式，当前仅支持STA模式。

|  |  |
| --- | --- |
| Type | Value |
| 0xE1 | 1 Byte WIFI-Mode |

WIFI-Mode：

|  |  |
| --- | --- |
| Mode | 值 |
| NULL | 0x00 |
| STA | 0x01 |

2. 设备通过LLEvent向小程序回复设置结果。

|  |  |  |
| --- | --- | --- |
| Type | Length | Value |
| 0xE0 | 2 Bytes length | 1 Byte Result |

*说明：0x00表示切换成功，0x01表示切换失败。*

#### 7.2.3 WIFI信息下发

1. 小程序通过LLDeviceInfo向设备传输WIFI信息，包括SSID和PASSWORD。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type | Length | value | | | |
| SSID | | PASSWORD | |
| 0xE2 | 2 Bytes | 1 Byte Length | SSID | 1 Byte Length | PASSWORD |

2. 设备通过LLEvent向小程序回复WIFI信息设置结果。

|  |  |  |
| --- | --- | --- |
| Type | Length | Result |
| 0xE1 | 2 Bytes | 1 Byte |

*说明：0x00表示设置成功，0x01表示切换失败*

#### 7.2.4 请求WIFI连接

1. 小程序通过LLDeviceInfo向设备下发连接WIFI指令。

|  |
| --- |
| Type |
| 0xE3 |

2. 设备开始连接WIFI，连接结束后将结果上报给小程序。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Type | Length | Value | | | | |
| WIFI-Mode | Station状态 | SoftAp状态 | SSID Length | SSID |
| 0xE2 | 2 Bytes | 1 Byte | 1 Byte | 1 Byte | 1 Byte | N Bytes |

*说明：*

*1.WIFI-Mode当前仅支持STA。*

*2.Station状态表示WIFI是否连接，0x00表示已连接，0x01表示未连接。*

*3.SoftAp状态当前不使用。*

*4.已连接时需要将当前的SSID上报，未连接时SSID Length设置为0。*

#### 7.2.5 设备Token下发

1. 小程序从后台获取设备Token后通过LLDeviceInfo下发给设备。设备通过C-SDK将Token上报给IoT后台进行验证。

|  |  |  |
| --- | --- | --- |
| Type | Value | |
| Token Length | Token |
| 0xE4 | 2 Bytes Length | N Bytes Data |

2. IoT后台将Token的验证结果返回给设备，设备通过LLEvent上报给小程序。

|  |  |  |
| --- | --- | --- |
| Type | Length | Value |
| 0xE3 | 2 Bytes | 1 Byte Result |

*说明：0x00表示设置成功，0x01表示切换失败*

#### 7.2.6 配网日志获取

1. 配网失败后，可以在小程序端获取设备配网日志进行问题分析。小程序通过LLDeviceInfo向设备下发日志上报指令。

|  |
| --- |
| Type |
| 0xE5 |

2. 设备通过LLEvent向小程序上报配网过程中产生的日志。

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Length | Value | |
| 0xE4 | 2 Bytes | Log Type | Msg Content |
| 1 Byte | N Bytes |

*说明：Log Type为0x00表示设备可能存储的未上报的错误日志，0x01表示设备本次产生的错误日志，0x02表示普通的配网过程日志。*