**连连Sync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 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**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* |

[1. 引言 5](#_Toc55210707)

[1.1 背景 5](#_Toc55210708)

[1.2 目的 5](#_Toc55210709)

[2. 设备参数要求 5](#_Toc55210710)

[3. LLSync TLV格式定义 5](#_Toc55210711)

[3.1 LLSync数据包 6](#_Toc55210712)

[3.2 LLSync固定报头 6](#_Toc55210713)

[3.3 LLSync报文参数 7](#_Toc55210714)

[4.LLSync Profile定义 9](#_Toc55210715)

[4.1 LLSync Profile 11](#_Toc55210716)

[4.2 UUID说明 14](#_Toc55210717)

[5.LLSync Advertisement定义 15](#_Toc55210718)

[6． BLE通信数据流 17](#_Toc55210719)

[6.1子设备绑定 17](#_Toc55210720)

[6.2子设备连接 18](#_Toc55210721)

[6.3子设备解绑 20](#_Toc55210722)

[6.4数据模板协议交互 21](#_Toc55210723)

[6.5 设备信息上报 28](#_Toc55210724)

[7.蓝牙辅助配网 28](#_Toc55210725)

[7.1概述 28](#_Toc55210726)

[7.2蓝牙辅助配网流程 28](#_Toc55210727)

[7.3传输格式 30](#_Toc55210728)

## 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以及加快研发速度，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。

### 目的

本文档旨在将BLE设备接入腾讯连连平台的流程标准，降低BLE类设备上云门槛。同时可以支持BLE和Wi-Fi的混合配网。

## 设备参数要求

|  |  |
| --- | --- |
| **参数项** | **要求** |
| BLE ATT MTU | 28及以上，最大值需要结合数据模板格式而定 |
| BLE协议 | BLE4.2及以上 |

## LLSync TLV格式定义

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)，将设备的接入按照JSON模板的形式进行了标准化。但由于BLE协议中MTU较小，无法承载JSON格式的协议交互，因为定义了一套标准TLV格式用来表示数据模板。如无特殊说明，本文所有数据均使用网络序传输。

LLSync数据包最大长度为2048字节，包括LLSync包头和用户数据。当LLSync 数据包大于Gatt MTU时，LLSync 协议会将数据分片发送，接收方收到分片数据后应该先将数据拼接再进行处理。

### 3.1 LLSync数据包

LLSync 数据包用于 LLData 数据的下发，其报文结构如下：

**LLSync 数据包结构**

|  |  |
| --- | --- |
| **Fixed header** | 1 Byte固定报头，见 3.2 |
| **Payload** | N Bytes报文参数，见 3.3 |

### 3.2 LLSync固定报头

1 Byte固定报头说明如下：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **含义** | 数据模版报文  类型，见3.2.1 | | 数据作用，见3.2.2 | 数据ID字段，见3.3.3 | | | | |

#### 3.2.1 数据模版报文类型

位置：二进制位 7 - 6

表示为 2 位无符号值，这些值的定义见表格 数据模版报文类型

**数据模版报文类型**

|  |  |  |
| --- | --- | --- |
| **名字** | **值** | **描述** |
| **Property** | 0 | 对应数据模版中的属性 |
| **Event** | 1 | 对应数据模版中的事件 |
| **Action** | 2 | 对应数据模版中的行为 |

#### 3.2.2 数据作用类型

位置：二进制位 5

表示为 1 位无符号值，这些值的定义见表格 数据作用类型

**数据作用类型**

|  |  |  |
| --- | --- | --- |
| **名字** | **值** | **描述** |
| **Request** | 0 | 该数据用作请求 |
| **Reply** | 1 | 该数据用作应答 |

用于区分小程序/网关下发给设备的报文类型，是主动请求还是对设备请求的应答。

#### 3.2.3 数据ID

位置：二进制位 4 - 0

表示为 5 位无符号值，与数据模版报文类型和数据作用类型相关，见表格 数据ID定义

**数据ID定义**

|  |  |  |  |
| --- | --- | --- | --- |
| **数据模版报文类型** | **数据作用类型** | **数据ID** | **描述** |
| Property | Request | 0 | 无意义 |
| Reply | 0 | 表示report\_reply |
| 2 | 表示get\_status\_reply |
| Event | Reply | event\_id | 表示event对应的枚举数值 |
| Action | Request | action\_id | 表示action对应的枚举数值 |

*注意：event\_id/action\_id在数据模版json文件中为字符串形式，需要通过转换脚本转换成枚举类型，不得超过63。*

### 3.3 LLSync报文参数

报文参数和固定报头存在关联，见表格 报文参数定义

**报文参数定义**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **报文类型** | **数据作用类型** | **数据ID** | **参数类型** | **参数描述** |
| **Property** | Request | 0 | TLV类型 | 见 3.3.2 |
| Reply | 0 | Reply\_Result  类型 | 见 3.3.1 |
| 2 | TLV类型 | 见 3.3.2 |
| **Event** | Reply | event\_id | Reply\_Result  类型 | 见 3.3.1 |
| **Action** | Request | action\_id | TLV类型 | 见 3.3.2 |

#### 3.3.1 Reply\_Result类型

Reply报文返回值的定义，见表格 Reply返回值定义

**Reply返回值定义**

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

#### 3.3.2 TLV类型

TLV用来表示用户数据，其内容包括数据类型，数据长度，用户数据。TLV可以用在LLData和LLEvent数据中。数据模版中定义了 6 种基本数据类型，对于不同的基本数据类型，TLV 的格式略有不同，具体定义见表格 TLV类型定义

**TLV类型定义**

|  |  |
| --- | --- |
| **Type** | 1 Byte数据类型 |
| **Length** | 数据长度 |
| **Value** | 用户数据 |

##### 3.3.2.1数据类型定义

数据模板中定义了6种基本数据类型，占据二进制位 7 - 5，二进制位 4 - 0 表示数据ID。具体定义见表格 数据头定义

**数据头定义**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Property** | 见表格 tlv数据类型定义 | | | property\_id | | | | |
| **Event** | params\_id | | | | |
| **Action** | input\_id | | | | |

*备注：数据ID占据5位，最大值为63。*

##### 3.3.2.2 tlv数据类型定义

|  |  |  |  |
| --- | --- | --- | --- |
| **类型值** | **类型** | **长度** | **取值范围** |
| 0 | 布尔型 | 1 Byte | 0/1 |
| 1 | 整数型 | 4 Bytes | -2^31 ~ 2^31-1 |
| 2 | 字符串 | N (< 2048) Bytes | 用户自定义数据 |
| 3 | 浮点型 | 4 Bytes | 1.2E-38 ~ 3.4E+38 |
| 4 | 枚举型 | 2 Bytes | 0 ~ 65535 |
| 5 | 时间型 | 4 Bytes | 0 ~ 4,294,967,295 |

1. 类型值表示该数据类型在数据头中 7 - 5 bit的值。
2. 长度和取值范围都是用来形容该TLV中数值部分，字符串头部2字节表示字符串长度。因为其他类型长度是固定的，所以在数据包中省略其他类型数据长度。例如00 01表示id = 0， value = 1的布尔数据，41 00 05 68 65 6C 6C 6F表示 id = 1，length = 5，value = hello的字符串。
3. 浮点型指的是单精度浮点数。
4. 忽略掉一些和数据流无关的非重点信息，如：读写类型、描述等

## 4.LLSync Profile定义

Profile总架构如图：

图形用户界面

描述已自动生成

LLSync Primary Service作为整个Explorer功能服务的总入口

### 4.1 LLSync Profile

#### 4.1.1 LLDeviceInfo

LLDeviceInfo进行设备信息的读取/写入，用于进行设备鉴权、连接及操作等信息交互，数据格式见表格 LLDeviceInfo数据格式，注意版本号差别，请以最新版本为基础进行开发。

**LLDeviceInfo数据格式（协议版本号0）**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **数据格式** | | **描述** |
| **数据类型**  **1 Bytes** | **数据内容**  **n Bytes** |
| **时间同步信息** | 0 | 4 Bytes nonce +  4 Bytes Unix TS | 绑定前向BLE设备发送验签信息 |
| **连接鉴权信息** | 1 | 4 Bytes Unix Ts +  20 Bytes Hmac-sha1 | 使用local psk对Ts签名得到Hmac-sha1 |
| **绑定结果通知** | 2 | 1 Byte绑定结果 +  4 Bytes local psk +  8 Bytes bind string | 绑定成功，小程序或网关生成local psk和bind string，小程序或网关需要记录local psk和bind string与设备的对应关系 |
| 3 | 无 | 绑定失败 |
| **解绑请求** | 4 | 20 Bytes Hmac-sha1 | 使用local psk对“UnbindRequest”签名得到Hmac-sha1 |
| **连接结果通知** | 5 | 无 | 连接成功 |
| 6 | 无 | 连接失败 |
| **解绑结果通知** | 7 | 无 | 解绑成功 |
| 8 | 无 | 解绑失败 |

上表所述数据内容为数据不分片时的格式与长度，分片时请按分片格式拆分发送，以“连接鉴权信息”为例：

完整数据包为“0x01,4 Bytes Unix Ts,20 Bytes Hmac-sha1”，0x01为连接鉴权信息标记，之后为4字节时间戳和20字节校验签名。

**LLDeviceInfo数据格式（协议版本号1）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **数据格式** | | | **描述** |
| **数据类型**  **1 Bytes** | **分片及长度**  **2 Bytes** | **数据内容**  **n Bytes** |
| **时间同步信息** | 0 | 见表格“数据分片及数据长度定义” | 4 Bytes nonce +  4 Bytes Unix TS | 绑定前向BLE设备发送验签信息 |
| **连接鉴权信息** | 1 | 见表格“数据分片及数据长度定义” | 4 Bytes Unix Ts +  20 Bytes Hmac-sha1 | 使用local psk对Ts签名得到Hmac-sha1 |
| **绑定结果通知** | 2 | 见表格“数据分片及数据长度定义” | 1 Byte绑定结果 +  4 Bytes local psk +  8 Bytes bind string | 绑定成功，小程序或网关生成local psk和bind string，小程序或网关需要记录local psk和bind string与设备的对应关系 |
| 3 | 无 | 无 | 绑定失败 |
| **解绑请求** | 4 | 见表格“数据分片及数据长度定义” | 20 Bytes Hmac-sha1 | 使用local psk对“UnbindRequest”签名得到Hmac-sha1 |
| **连接结果通知** | 5 | 无 | 无 | 连接成功 |
| 6 | 无 | 无 | 连接失败 |
| **解绑结果通知** | 7 | 无 | 无 | 解绑成功 |
| 8 | 无 | 无 | 解绑失败 |

上表所述数据内容为数据不分片时的格式与长度，分片时请按分片格式拆分发送，以“连接鉴权信息”为例：

假设mtu长度足够，不分片时完整数据包为“0x01,0x00,0x18,0xA1,0xA2,0xA3,0xA4, 0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0xBA,0xBB,0xBC,0xBD,0xBE,0xBF,0xC0,0xC1,0xC2,0xC3”，0x01为连接鉴权信息标记，0x00,0x18为分片标记及本包数据长度，0xA1,0xA2,0xA3,0xA4为时间戳，其余为20字节校验签名，（仅举例，实际产品不使用这种方式）。

考虑到不同芯片的兼容性，实际产品中连接时统一按照mtu最小20字节进行分片，则第一包数据为“0x01,0x40,0x11,0xA1,0xA2,0xA3,0xA4,0xB0,0xB1,0xB2,0xB3,0xB4,0xB5, 0xB6,0xB7,0xB8,0xB9,0xBA,0xBB,0xBC”，0x01为连接鉴权信息标记，0x40,0x11为分片标记及本包数据长度，之为4字节时间戳以及校验签名的前13字节；第二包数据为“0x01,0xC0,0x07,0xBD,0xBE,0xBF,0xC0,0xC1,0xC2,0xC3”，0x01为连接鉴权信息标记，0xC0,0x07为分片标记及本包数据长度，之后为剩余7字节的校验签名。

#### 4.1.2 LLData

LLData进行连连业务数据的读写，对应着数据模板中的control、report\_reply、get\_status\_reply、event\_reply、action\_reply操作。具体格式与操作有关，详细内容请查阅相关操作描述章节。

#### 4.1.3 LLEvent

LLEvent用于设备主动上报通知，包括对LLDeviceInfo和LLData的回复，见表格 LLEvent消息格式定义

**LLEvent消息格式定义**

|  |  |  |
| --- | --- | --- |
| **名称** | **长度** | **描述** |
| **Type** | 1 Byte | 见表格 LLEvent Type定义 |
| **Length** | 2 Bytes | 见表格“数据分片及数据长度定义” |
| **Value** | N Bytes | Type决定具体内容 |

**LLEvent Type定义**

|  |  |  |
| --- | --- | --- |
| **名称** | **数值** | **说明** |
| **属性上报** | 0 | 对应数据模版中的report |
| **控制回复** | 1 | 对应数据模版中的control\_reply |
| **获取最新信息** | 2 | 对应数据模版中的get\_status |
| **事件上报** | 3 | 对应数据模版中的event\_post |
| **行为响应** | 4 | 对应数据模版中的action\_reply |
| **绑定鉴权信息** | 5 | 绑定后设备返回的信息 |
| **连接鉴权信息** | 6 | 连接后设备返回的信息 |
| **解绑鉴权信息** | 7 | 解绑后设备返回的信息 |
| **设备信息** | 8 | 上报MTU长度和协议版本 |

不同的LLEvent Type对应的数据内容不同，见表格 LLEvent Value定义

**LLEvent Value定义**

|  |  |
| --- | --- |
| **LLEvent Type** | **Value** |
| 0 | 见6.4.1 |
| 1 | 见6.4.2 |
| 2 | 见6.4.3 |
| 3 | 见6.4.4 |
| 4 | 见6.4.5 |
| 5 | 见6.1 |
| 6 | 见6.2 |
| 7 | 见6.3 |
| 8 | 见6.5 |

**数据分片及数据长度定义**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | 15 | 14 | 13 | 12 | 11 | 10 | … | 1 | 0 |
| **值** | 见分片标记定义 | | 数据长度 | | | | | | |

**分片标记定义**

|  |  |
| --- | --- |
| **值** | **含义** |
| **00** | 不分片 |
| **01** | 数据分片，表示第一包数据 |
| **10** | 数据分片，表示中间包数据 |
| **11** | 数据分片，表示最后一包数据 |

1. 由于gatt mtu限制，较大数据需要分为多个数据包传输。
2. 如果数据包分为2包，那么只有分片标记为01和11的数据包。
3. 数据长度指的是本包内有效载荷数据的长度，例如一包数据总长度为20字节，除去type 1字节，length和分片共2字节，其余为有效数据17字节，所以长度应填17。
4. 数据分片后每一包的数据格式都和未分包的数据格式相同，数据分片涉及到LLEvent数据包、部分LLData数据包及部分LLDeviceInfo数据包。
5. 如果是分片的event、property、action上行数据，设备端会持续发起notification直至分片结束，APP/小程序/网关端 需要等待分片结束后再进行后续的业务流程。

### 4.2 UUID说明

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

按照BLE协议，16bit UUID和128bit UUID转换关系为

128-bit value = 16-bit-value \* 2^96 + BluetoothBaseUUID

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

## 5.LLSync Advertisement定义

自定义广播数据按照BlueTooth协议要求，放在0xFF Manufacturer Specific Data的字段当中，company ID使用0xFEE7（Tencent Holdings Limited），0xFEE7和0xFEBA均为我司申请的Company ID。格式为：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 | 见后面说明1 | 6 | MAC地址 | 10 | Product ID |
| **绑定中** | 1 | 6 | MAC地址 | 10 | Product ID |
| **已绑定** | 1 | 8 | 计算方式见说明2 | 8 | 绑定标识，计算方式见说明3 |

说明：

1. 设备状态定义

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **含义** | 协议版本 | | | | Reserved | | 绑定状态 | |

1. 当前协议版本号为 1，不同协议版本号之间数据交互格式不同
2. 绑定状态为 0 表示未绑定，为1表示绑定中，为2表示已绑定
3. 已绑定状态下设备标识计算方式

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

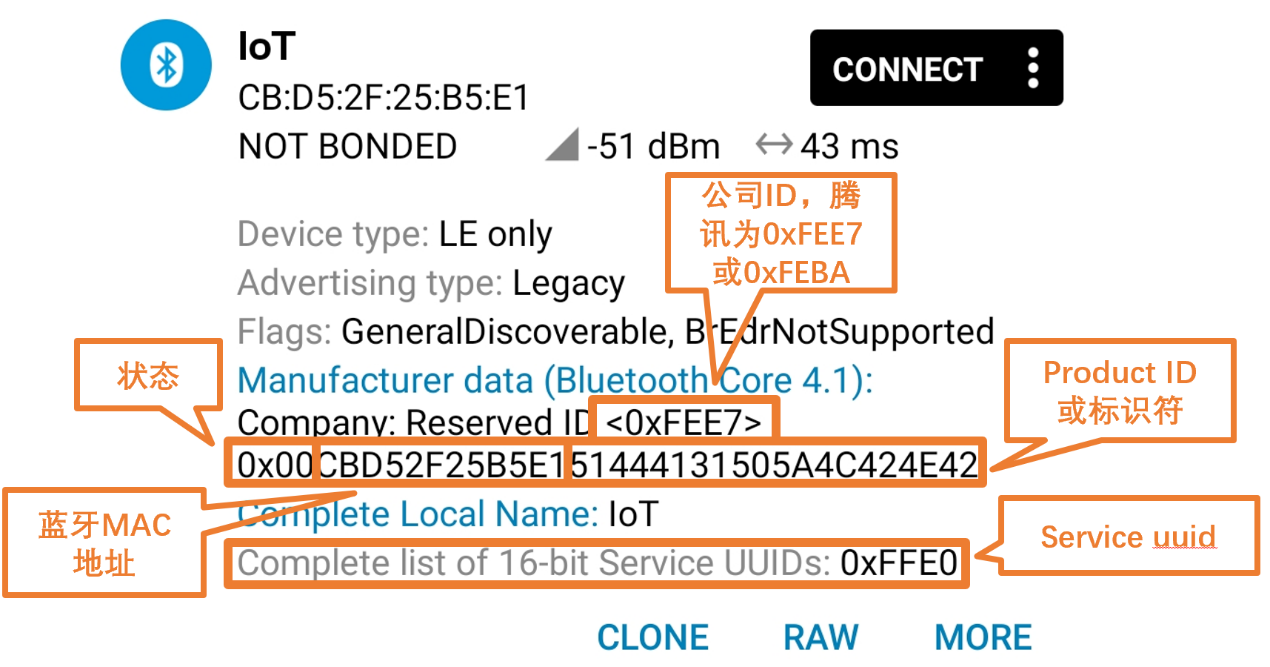
那么Temp = md5sum(“ABCDEFGHIJDev01”) = { 0x61, 0x2a, 0xf7, 0x9d, 0x50, 0x17, 0x93, 0x87, 0x2a, 0x4a, 0x97, 0xe8, 0xcb, 0xe4, 0x5a, 0x10}

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

1. 绑定标识计算方式

由网关或小程序在绑定成功时提供，网关的计算方式和说明2一致，使用网关的product\_id和device\_name。

扫描看到的广播包如下图所示



**绑定状态说明**

|  |  |
| --- | --- |
| **状态** | **说明** |
| **未绑定** | 初始状态，可以选择不发送BLE Advertisement以省电 |
| **绑定中** | 按下绑定触发按键，在超时之前处于绑定中，这期间必须要按照格式要求发送advertisement |
| **已绑定** | 正确完成绑定状态，需要发送advertisement，interval可以在不影响业务的情况根据需要改大 |

## BLE通信数据流

### 6.1子设备绑定

场景：BLE终端尚未绑定需要进行绑定

图示

描述已自动生成

1. 往UUID FFE1上写入Unix TS数据格式见下表

|  |  |  |
| --- | --- | --- |
| **type** | **value** | |
| **nonce** | **timestamp** |
| 00 | 1. Bytes nonce | 4 Bytes timestamp |

1. 设备验证签名后返回的LLEvent数据格式见下表

|  |  |  |  |
| --- | --- | --- | --- |
| **type** | **length** | **value** | |
| **sign info** | **device name** |
| 05 | 2 Bytes value length | 20 Bytes sign info | N Bytes device name |

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。deviceinfo=

product id + devicename + nonce + expiration time，其中expiration time = timestamp + 60。

**约定：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”。**

设备返回的数据可能是分片的，小程序/网关需要检查分片标记。

1. 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** |
| 02 | 02 | 4 Bytes Array local psk | 8 Bytes Array |

1. 往FFE1写入绑定失败结果格式见下表

|  |  |
| --- | --- |
| **type** | **value** |
| 03 | 1 Bytes Reply\_Result |

注意：

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终端需要通过按键进入待绑定状态，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. 设备绑定成功之后，不会再响应UUID FFE1的读取请求
4. 计算签名时，如果BLE设备没有获取到实时时间，以最大时间为ts来计算签名
5. 如果绑定成功，需要在设备上存储LocalPSK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，需要重新连接

日程表

描述已自动生成

注：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FFE1写入签名信息数据格式见下表

|  |  |  |
| --- | --- | --- |
| **type** | **value** | |
| **timestamp** | **sign info** |
| 01 | 4 Bytes timestamp | 20 Bytes sign info |

sign info是使用local psk对timestamp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** | |
| **sign info** | **device name** |
| 06 | 2 Bytes value length | 20 Bytes sign info | N Bytes device name |

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。

**约定：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”。**

设备返回的数据可能是分片的，小程序/网关需要检查分片标记。

### 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FFE1写入解绑请求

|  |  |
| --- | --- |
| **type** | **value** |
| 04 | 1. Bytes sign info |

1. 验签后返回的LLEvent信息数据格式如下表

|  |  |  |
| --- | --- | --- |
| **type** | **length** | **value** |
| 07 | 2 Bytes value length | 20 Bytes sign info |

注意：

1. 解绑请求固定字符串UnbindRequest，解绑回复固定字符串UnbindResponse
2. 使用local psk对固定字符串签名，算法选择hmac-sha1

3.往FFE1写入解绑成功结果格式见下表

|  |  |
| --- | --- |
| **type** | **value** |
| 07 | 1 Byte Reply\_Result |

4.往FFE1写入绑定失败结果格式见下表

|  |  |
| --- | --- |
| **type** | **value** |
| 08 | 1 Byte Reply\_Result |

### 6.4数据模板协议交互

数据模板协议交互均以网关MQTT数据流为例，如果是小程序，和IoT后台的交互要更换成对应的API调用。

往UUID FFE2上写入/读取的TLV值 和 数据模板协议中的上下行JSON中的param部分对应，为了简化，省掉了clientToken和timestamp。

#### 6.4.1 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** |
| 00 | 2 Bytes value length | tlv数据 |

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 tlv data |
| 81，00，01 | property color tlv data |
| 22，00，00，00，23 | property brightness tlv data |
| 43，00，02，31，32 | property name tlv data |

备注：如果数据分片，每包的数据格式都是type，lenth，property value，需要服务端将property value拼接后进行解析。

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 |

#### 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.

如果协议版本号为1，其数据格式如下：

|  |  |
| --- | --- |
| **header** | **proterpy value** |
| 00 | tlv数据 |

property value中可以包含多个property的数据。示例

|  |  |
| --- | --- |
| **数值** | **描述** |
| 00 | header |
| 00，01 | property power switch tlv data |
| 81，00，01 | property color tlv data |
| 22，00，00，00，23 | property brightness tlv data |
| 43，00，02，31，32 | property name tlv data |

如果协议版本号为2，其数据格式如下：

|  |  |  |
| --- | --- | --- |
| **header** | **length** | **proterpy value** |
| 00 | 2 Bytes value length | tlv数据 |

*备注：此处length格式与event中length格式相同，其定义见表格“数据分片及数据长度定义”*

property value中可以包含多个property的数据。示例

|  |  |
| --- | --- |
| **数值** | **描述** |
| 00 | header |
| 00，0F | value length |
| 00，01 | property power switch tlv data |
| 81，00，01 | property color tlv data |
| 22，00，00，00，23 | property brightness tlv data |
| 43，00，02，31，32 | property name tlv data |

备注：如果数据分片，每包的数据格式都是header，length，property value，设备端会将数据包中property value进行拼接，同时将数据中的value length求和来还原完整数据包。

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** | **result** |
| 01 | 2 Bytes value length | 1 Byte Reply\_Result |

#### 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上报最新信息，对应数据模版的get\_status操作

|  |
| --- |
| **type** |
| 02 |

get\_status操作不需要携带数据。

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

|  |  |  |  |
| --- | --- | --- | --- |
| **header** | **result** | **length** | **property value** |
| 0x22 | 1 Byte Reply\_Result | 2 Bytes value length | tlv数据 |

*备注：此处length格式与event中length格式相同，其定义见表格“数据分片及数据长度定义”*

如果数据分片，每包的数据格式都是header，result, length，property value，设备端会将数据包中property value进行拼接，同时将数据中的value length求和来还原完整数据包。

如果result结果失败时，则没有length等后续字段。

property value中可以包含多个property的数据。示例

|  |  |
| --- | --- |
| 数值 | 描述 |
| 22 | header |
| 00 | Reply\_Result |
| 00，0F | property value length |
| 00，01 | property power switch tlv data |
| 81，00，01 | property color tlv data |
| 22，00，00，00，23 | property brightness tlv data |
| 43，00，02，31，32 | property name tlv data |

#### 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** |
| 03 | 2 Bytes value length | 1 Byte event id | tlv数据 |

event value中可以包含多个event 参数。示例

|  |  |
| --- | --- |
| 数值 | 描述 |
| 03 | type |
| 00，11 | event value length |
| 02 | event id，对应hardware fault |
| 40， 00，08， 31，32，33，34，35，36，37，38 | param name tlv data |
| 21，00，00，04，00 | param error code tlv data |

备注：如果数据分片，每包的数据格式都是type，length，event id，event value，需要服务端将event value拼接后解析。

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

|  |  |
| --- | --- |
| **header** | **value** |
| *参考章节3.2* | 1 Byte Reply\_Result |

*假设event id = 0，那么header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

如果协议版本号为1，其数据格式如下：

|  |  |
| --- | --- |
| **header** | **action value** |
| *参考章节3.2* | tlv数据 |

*假设action id = 0，那么header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 | header |
| 20，00，00，00，04 | input interval tlv data |
| 41，00，04，31，32，33，34 | input message tlv data |

如果协议版本号为2，其数据格式如下：

|  |  |  |
| --- | --- | --- |
| **header** | **length** | **action value** |
| *参考章节3.2* | 2 Bytes value length | tlv数据 |

*假设action id = 0，那么header字段应该是0x80*。

*备注：此处length格式与event中length格式相同，其定义见表格“数据分片及数据长度定义”*

如果数据分片，每包的数据格式都是header，length，action value，设备端会将数据包中action value进行拼接，同时将数据中的length求和来还原完整数据包。

action value中可以包含多个 input 参数。示例

|  |  |
| --- | --- |
| **数值** | **描述** |
| 80 | header |
| 00, 0B | action value length |
| 20，00，00，00，04 | input interval tlv data |
| 41，00，04，31，32，33，34 | input message tlv data |

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 |
| 4 | 2 Bytes value length | 1 Byte Reply\_Result | 1 Byte action\_id | tlv数据 |

1. result结果失败时，没有length等后续字段
2. 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 |
| 00 | action id，对应loop |
| 00，01 | response result tlv data |
| 41， 00，08， 31，32，33，34，35，36，37，38 | response message tlv data |

备注：如果数据分片，每包的数据格式都是type，length，result，action id，response params，需要服务端将response params拼接后解析。

### 6.5 设备信息上报

设备连接成功后主动向小程序/网关上报设备信息，包括协议版本和设备MTU大小，其中版本号与LLSync Advertisement中保持一致（见5.LLSync Advertisement定义）。

|  |  |  |  |
| --- | --- | --- | --- |
| type | length | value | |
| version | mtu size |
| 8 | 2 Bytes value length | 1 Byte version | 2 Bytes mtu length |

## 7.蓝牙辅助配网

### 7.1概述

蓝牙辅助方式配网，每个厂商编码方式和报文选择上有自己的协议，本文介绍腾讯蓝牙辅助方式配网基础规范，包括蓝牙接入规范和蓝牙交互服务规范等。

* 蓝牙辅助方式配网是一款基于蓝牙通道的 Wi-Fi 网络配置功能。它通过蓝牙辅助方式配网协议将 Wi-Fi 配置传输到BLE设备，然后BLE设备可基于这些信息连接到WIFI 热点。
* 此时腾讯连连小程序可以通过GATT连接，例如，GATT通讯将后台提供的配网 Token 发送给设备，并由设备转发至物联网后台，依据 Token 可以进行设备绑定。

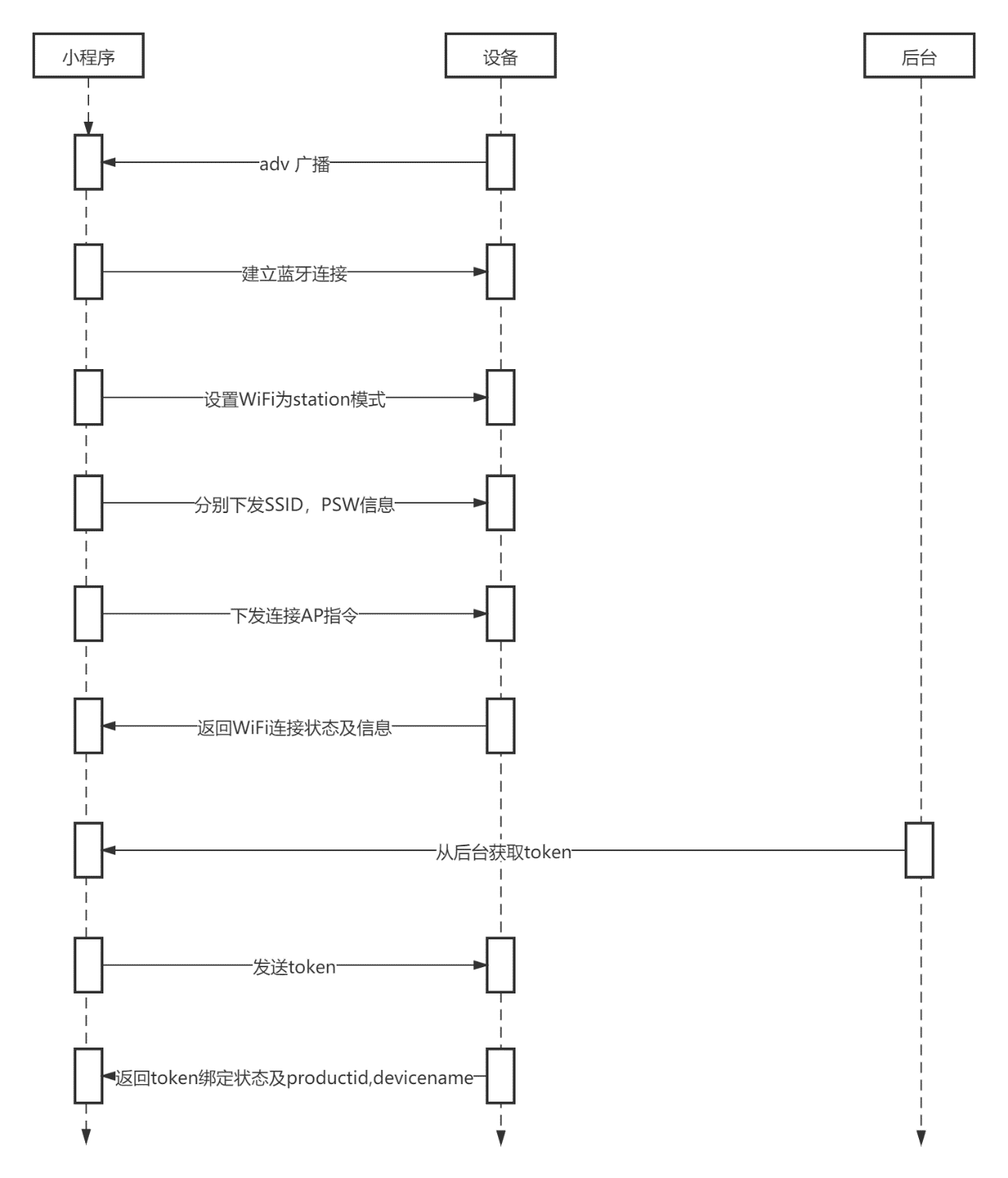
目前腾讯连连小程序已支持采用蓝牙辅助方式配网基础规范议进行蓝牙辅助配网

### 7.2蓝牙辅助配网流程

配网流程

1. BLE设备开启 GATT Server 功能，发送带有特定adv data的广播。
2. 使用腾讯连连小程序搜索到该特定广播，手机作为 GATT Client连接BLE设备。
3. GATT 连接建立成功后，腾讯连连小程序向BLE设备发送“Wi-Fi mode设置为station模组”控制帧
4. 腾讯连连小程序向BLE设备发送定义的 SSID、Password用于Wi-Fi 连接的必要信息。
5. 腾讯连连小程序向BLE设备发送“Wi-Fi 连接请求”控制帧，BLE设备收到之后，识别为腾讯连连小程序已将必要的信息传输完毕，准备连接 Wi-Fi。
6. BLE设备连接到 Wi-Fi 后，发送“Wi-Fi 连接状态报告”控制帧到腾讯连连小程序，以报告连接状态。
7. 腾讯连连小程序收到BLE设备报告的链接状态后，发送“服务端获取的token”用于设备与后台绑定。
8. BLE设备与后台绑定完成后，发送“设备端绑定状态报告”控制帧到腾讯连连小程序，以报告绑定状态。至此配网结束。

配网流程图如下



### 7.3传输格式

腾讯连连小程序与BLE设备之间的通信格式定义如下

**帧不分片时的标准格式** (8 bit)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **描述** | type | Frame control | Sequence number | Data length | data |
| **数值** | 1 | 1 | 1 | 1 | ${ Data length } |

**帧分片格式**（8 bit）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **描述** | type | Frame control | Sequence number | Data length | data | |
| **数值** | 1 | 1 | 1 | 1 | Total Content Length | Content |
| 2 | ${Data Length} - 2 |

**Ack 帧格式**（8 bit）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **描述** | Type(Ack) | Frame control | Sequence number | Data length | data |
| **数值** | 1 | 1 | 1 | 1 | Acked Sequence Number |
| 2 |

#### 1.Type

类型域，占 1 byte。分为 Type 和 Subtype（子类型域）两部分, Type 占低 2 bit，Subtype占高 6 bit。

分为控制帧，数据帧，控制帧（0x0b’00）定义见下表

|  |  |  |  |
| --- | --- | --- | --- |
| **控制帧（二进制）** | **含义** | **释义** | **备注** |
| 0x0(b’000000) | Ack | 用来回复对方发的帧， Ack 帧的 Data 域使用回复对象帧的 Sequence 值。 | Data 域使用1 byte Sequence 值， 与恢复对象帧的Sequence 值相同。 |
| 0x2(b’000010) | 设置WIFI工作模式 | 设置BLE设备的 Wi-Fi 模式，帧包含 opmode 信息。 | Data[0]用于表示wifi mode类型，包括：0x00:NULL;0x01:STA |
| 0x3(b’000011) | 连接BLE设备到AP | 通知BLE设备，必要的信息已经发送完毕，可以连接 AP。 | 不包含data域 |
| 0x5(b’000101) | 获取WiFi状态 | 获取BLE设备的 Wi-Fi 模式和状态等信息。 | 会通过 Wi-Fi 连接状态 报告 (Wi-Fi Connection State Report) 数据帧来回复小程序当前所处的 opmode、连接状态、SSID。 |

数据帧（0x1 b’01）定义见下表

|  |  |  |  |
| --- | --- | --- | --- |
| **数据帧（二进制）** | **含义** | **释义** | **备注** |
| 0x2(b’000010) | Wifi station 的ssid信息 | STA 将要连接的 AP 的 SSID。 | NULL |
| 0x3(b’000011) | Wifi station 的password信息 | STA 将要连接的 AP 的密码 | NULL |
| 0xf (b’001111) | Wi-Fi connection state report. | 通知手机BLE设备的 Wi-Fi 状态， 包括 STA状态， 用于小程序配置 STA 连接时的通知， 或有 STA 连接上 SoftAP 时的通知。 | 回复的ACK中data[0]表示：opmode，包括0x00:NULL;0x01:STA;data[1]表示：STA的连接状态，0x0表示处于连接状态，其他表示处于非连接状态；data[2]表示：softap的连接状态，即表示有多少STA已经连接；data[3]及后面表示：为按照协议格式的SSID信息； |
| 0x13 (b’010011) | Token data | 用户发送token或者接收token绑定状态信息 | 数据较长时可分片发送。 |

#### 2.FrameControl

帧控制域，占 1 byte

#### 3.SequenceControl

序列控制域。帧发送时，无论帧的类型是什么，序列 (Sequence)都会自动加 1，用来防止重放攻击 (ReplayAttack)。每次重现连接后，序列清零。

#### 4.Length

Data 域的长度，不包含 CheckSum。

#### 5.Data

Data表示用户传输的数据，以下为示例数据

**实例1、下发设置切换WIFI mode到STA**

命令：08 08 00 01 01

|  |  |
| --- | --- |
| **数值** | **描述** |
| (0x02<<2) | 0x00=08 | 加载控制帧，0x00为控制命令，0x02设置切换WIFI mode到STA |
| 08 | 加载帧控制域 |
| 00 | sequence序列控制域，每发送一次加1 |
| 01 | 数据长度 |
| 01 | data0 |

**实例2、下发WiFi的SSID信息**

命令：09 00 01 07 74 65 6E 63 65 6E 74

|  |  |
| --- | --- |
| **数值** | **描述** |
| (0x02<<2) | 0x01=0x09 | 加载数据帧，0x01为数据命令，0x02表示发送AP的SSID |
| 00 | 加载帧控制域，无检验，无加密 |
| 01 | sequence序列控制域，每发送一次加1 |
| 07 | 数据长度 |
| 74 | data0 |
| 65 | data1 |
| 6E | data2 |
| 63 | data3 |
| 65 | data4 |
| 6E | data5 |
| 74 | data6 |