设备集成开发指导

文档版本 01

发布日期 2022-04-19

华为终端有限公司

注意

华为会不定期对本文档的内容进行更新。

本文档仅作为使用指导，文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为终端有限公司

地址： 广东省东莞市松山湖园区新城路 2 号

网址： https://consumer.huawei.com

目 录

1 概述

简介

本文档为 HarmonyOS Connect 生态产品合作伙伴提供基于华为蓝牙应用层开放协议规范开发的产品的设备集成指导，旨在帮助伙伴快速熟悉开发流程，完成产品信息配置、设备注册、设备控制、OTA 升级等功能开发，并基于智慧生活 App 和 FA 进行代理注册测试、设备控制测试和 OTA 升级功能测试。

开发方案

表1-1 蓝牙 BLE 设备开发方案

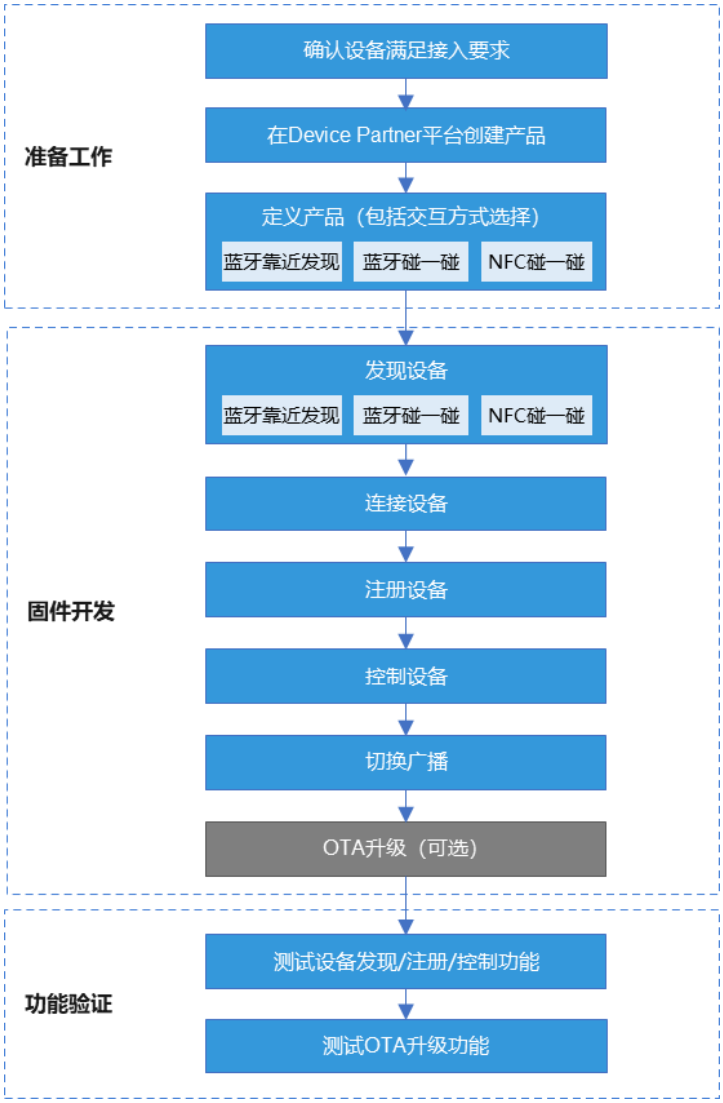
方案名称	适用场景	说明
HarmonyOS Connect 极小硬件方案（BLE）	基于华为蓝牙应用层开放协议规范开发的产品	HarmonyOS Connect 标准开发方案
HarmonyOS Connect 非华为开放协议方案（BLE）	基于非华为蓝牙应用层开放协议规范开发的产品	非标准方案。非运动健康类产品需与品类产品经理确认，否则将无法提交认证申请或被驳回。请优先选择标准方案。

需要根据实际方案进行开发：

- HarmonyOS Connect 极小硬件方案（BLE）：参见本文档。
- HarmonyOS Connect 非华为开放协议方案（BLE）：若产品为运动健康类产品，参见蓝牙功能开发。

开发流程

图1-1 设备集成开发流程



2 准备工作

步骤 1 确认设备接入要求。

蓝牙 BLE 设备接入 HarmonyOS Connect 生态目前有三种极简交互方式：蓝牙靠近发现、蓝牙碰一碰和 NFC 碰一碰，三选一即可。其中，蓝牙靠近发现包含首次配对靠近发现、靠近发现回连、设备重置后靠近发现三个场景；蓝牙碰一碰能力包含首次一碰发现弹框、二碰回连弹框、设备重置后一碰发现弹框三个场景。

须知

蓝牙碰一碰功耗较高，功耗敏感类设备不建议选择该方式。

表2-1 设备接入要求

规则项	说明	NFC 碰一碰	蓝牙碰一碰	蓝牙靠近发现
硬件最低资源要求	RAM ：64KB（支持加解密、适配华为蓝牙开发协议）、128KB（支持加解密、适配华为蓝牙开发协议、升级保留双分区）	必选	必选	必选
	ROM：128KB	必选	必选	必选
广播能力	支持 SIG BT 协议 4.0 及以上版本。	必选	必选	必选
	支持 2.4G BLE 广播功能。	必选	必选	必选
	设备具备明确 UX（如：开机/双击/长按等）可触发设备发起首次靠近发现/一碰广播能力。	-	必选	必选
	设备具备发送不同广播（如：设备首次配对靠近发现/一碰广播、靠近/二碰回连广播）的能力。	必选	可选	必选
	设备具备重置广播的能力，且设备重置后具备再次发送首次靠近发现/一碰广播的能力。	必选	必选	必选
广播类型	支持 BLE Advertising。	可选	必选	必选

规则项	说明	NFC 碰一碰	蓝牙碰一碰	蓝牙靠近发现
	使用 ADV_IND。	可选	必选	必选

步骤 2 在伙伴网站创建产品，参见产品定义指导。

开发方案：选择“HarmonyOS Connect 极小硬件方案（BLE）”。

步骤 3 产品定义。

极简交互方式：选择“蓝牙靠近发现”、“蓝牙碰一碰”或“NFC 碰一碰”。

证书类型：选择“无”。

物模型定义：根据产品实际需要定义产品功能，并在产品详情页下载产品 profile 文件用于固件开发。

步骤 4 配置生成广播名称和靠近发现/碰一碰广播报文。

在集成开发页面可配置生成广播名称和靠近发现/碰一碰广播报文，广播详细内容可参考 3.1 简介。

图2-1 配置生成靠近发现报文

步骤 5 登录伙伴网站后，在产品开发环节，下载产品信息。

图2-2 导出产品信息

----结束

3 固件开发

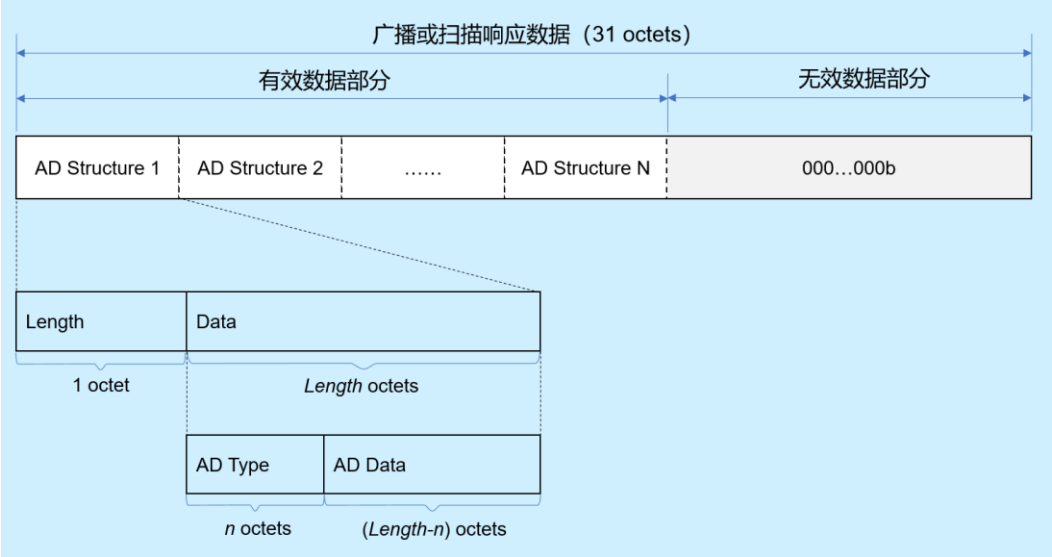
3.1 简介

BLE 广播介绍

BLE 广播分成广播包（Advertising Data）和响应包（Scan Response）两种类型。

每个包都是 31 字节，数据包中分为有效数据（significant）和无效数据（non-significant）两部分。

图3-1 广播包的数据结构



有效数据部分：包含若干个广播数据单元，称为 AD Structure，每个 AD Structure 的格式都是：

- 长度 Length ，表示这个 AD Structure 的长度（除去 len 本身 1）；
- 类型 AD Type，标记这段广播数据代表什么，如设备名、uuid 等；
- 数据 AD data，标记具体的数据内容。

无效数据部分：广播包的长度必须是 31 字节，如果有效数据部分不到 31 字节，剩下的使用 0 补全。这部分的数据是无效的。

BLE 设备广播规范

BLE 设备广播规范，具体包括蓝牙靠近发现和蓝牙碰一碰等几种广播形式。

蓝牙靠近发现广播规范

在蓝牙靠近发现中，BLE 广播分为靠近发现弹框 FA 的广播（简称靠近发现广播）和常态广播。为避免骚扰用户，靠近发现广播需要主动触发（开机/上电/双击等）才能发送，广播的响应距离为小于 30cm，广播间隔为 20ms，广播周期为 1 分钟，其他时间段默认发送常态广播，常态广播间隔根据实际设备自定义。靠近发现广播规范定义了广播包（Advertising Data）和响应包（Scan Response），根据 Advertising Data 可分为一靠广播和二靠广播；常态广播规范只定义了响应包（Scan Response），也就是蓝牙广播名称，根据设备注册状态分为未注册常态广播和已注册常态广播。

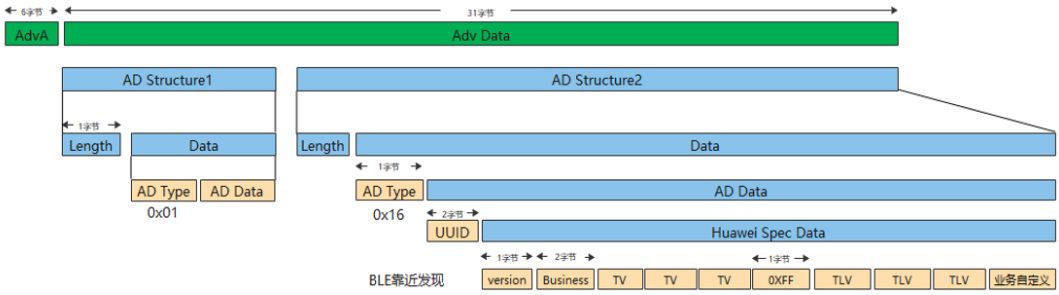
- 一靠广播为设备未注册时发送的靠近发现广播，此时设备与手机靠近后拉起 FA，请求 FA 进行设备代理注册，ProtocolD 值为 0x15。
- 二靠广播为设备注册成功后发送的靠近发现广播，此时设备与手机靠近后，手机 FA 直接进入设备控制页，ProtocolD 值为 0x00/0x01。
- 设备发送一靠广播或者未注册常态广播时，使用智慧生活 App 可以扫描添加设备。

- 用户与设备没有注册关系时，设备发送二靠广播靠近手机，手机侧没有感知，不会拉起 FA。
- 已注册常态广播用于打开设备卡片时连接设备，或者设备连接过程中蓝牙连接断开时重新连接。

说明

当设备支持分享时，设备的分享者称为 owner，被分享者称为 member。ProtocolD 值为 0x00 时表示仅 owner 会弹框，member 不会弹框；ProtocolD 值为 0x01 时表示 owner 和 member 都会弹框。

靠近发现广播包（Advertising Data）结构如下图所示：



说明

AdvA 为 6 字节的 MAC 地址，不占用广播包的长度,此处需要使用 Public MAC，不能使用随机 MAC。

基于上述的广播包结构，我们需要首先了解蓝牙靠近发现广播包规范如表 3-1。

表3-1 蓝牙靠近发现广播包规范

内容		长度（Byte）	值	必选/可选	说明
AD Structure1	Length	1	0x 02	必选	类型和值的总长度
	AD Type	1	0x 01	必选	0x01 表示 flag
	Value	1	0x 06	必选	支持的 LE/BE/EDR 的 Flag 信息
AD Structure2	Length	1	0x XX	必选	类型和值的总长度
	AD Type	1	0x 16	必选	广播类型，0x16 表示蓝牙服务数据
	UUID	2	0xEEFD	必选	华为购买的 UUID，0xFDEE（已购买）小端存

内容			长度（Byte）	值	必选/可选	说明
						储
	Huawei Spec Data	协议版本	1	0x 01	必选	华为蓝牙广播协议版本
		business	1	0x 01	必选	01 表示靠近发现业务
		businessEx	1	0x 0D	必选	0x0D 表示拉起 FA
		subProdIdT	1	0x 04	必选	子产品 ID 类型
		subProdIdV	1	0x XX	必选	子产品 ID 值，默认 00
		AdvPowerT	1	0x 11	必选	测距字段类型
		AdvPowerV	1	0x XX	必选	广播的实际发射功率 AdvPower TRP = TxPower（设备芯片广播发射功率） – OTA（设备天线损耗），取值【-128，127】dBm，建议用芯片可配的最小发射功率以节省功耗。例如芯片的广播 TxPower 为-6dBm，天线 OTA 是 10dBm，则 Adv TRP 配置为-16dBm，取值为 0xF0
		prodId T	1	0x 12	必选	产品 ID 类型
		prodId V	4	0x XXXXXXXX	必选	产品 ID 值，产品 ID 的 ASCII 码值
		0xFF	1	0x FF	必选	分隔符
		ProtocolDT	1	0x 17	必选	蓝牙协议 ID 类型
		ProtocolDL	1	0x 01	必选	蓝牙协议 ID 长度

内容			长度（Byte）	值	必选/可选	说明
		ProtocolDV	1	0x XX	必选	蓝牙 ID 值 bit0：0 表示 member 二靠不弹框，1 表示 member 二靠弹框 bit1：预留字段，默认填 0 bit2~bit7 组成的 int 值标识事件类型，字段说明如下： 0-弹设备控制框 4-设备有异常告警事件，发送告警广播 5-纯蓝牙 BLE 设备请求 App 进行代理注册 通过计算得出： 0x15：请求 App 进行代理注册（一靠广播） 0x01/0x00：弹设备控制框（二靠广播） 0x11/0x10：设备有告警异常事件
		SN T	1	0x 14	必选	设备 SN
		SN L	1	0x 02	必选	设备 SN 长度
		SN V	2	0x XXXX	必选	设备 SN 。设备 SN 最后 2 字节的 ASCII 码值，必须和 deviceinfo 上报的 SN 最后两位一致
		业务自定义	1	0x 91	可选	以下为业务自定义数据
		L	1	自定义 L	可选	业务自定义字段 L
		V	变长	自定义 V	可选	业务自定义字段 V

表3-2 BLE 设备靠近发现广播实例

适用场景	长度	类型	值	说明
以下每个场景都携带	0x02	0x01	0x06	BLE 可被发现
纯 BLE 类设备请求 App 进行代理注册（一靠广播）	0x17	0x16	0xEEFD01010D04SS11PP12XXXXXXXXFF1701151402NNNN	未注册，SS 表示产品子型号；PP 表示发射功率值；XXXXXXXX 表示产品 ID 的 ASCII 码值；NNNN 是 SN 后两位的 ASCII 码值。
二靠广播，owner 弹框，member 不弹框	0x17	0x16	0xEEFD01010D04SS11PP12XXXXXXXXFF1701001402NNNN	已注册，SS 表示产品子型号；PP 表示发射功率值；XXXXXXXX 表示产品 ID 的 ASCII 码值；NNNN 是 SN 后两位的 ASCII 码值。
二靠广播，owner、member 都弹框	0x17	0x16	0xEEFD01010D04SS11PP12XXXXXXXXFF1701011402NNNN	已注册，SS 表示产品子型号；PP 表示发射功率值；XXXXXXXX 表示产品 ID 的 ASCII 码值；NNNN 是 SN 后两位的 ASCII 码值。

响应包（Scan Response）结构如下图所示：

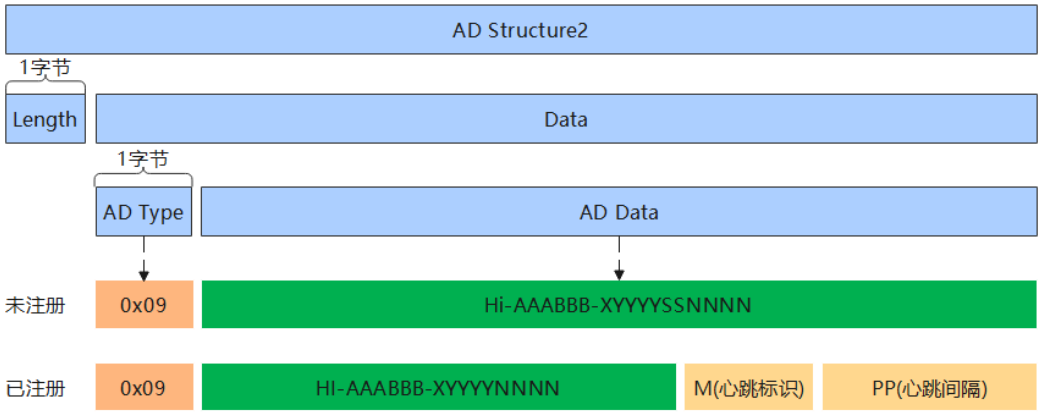


表3-3 响应包（Scan Response）广播实例

适用场景	长度	类型	值	说明
靠近发现广播、	可变	0x09	未注册：“Hi-**AAABB**	未注册： **Hi-**：未注册广播名称固定前缀，3 字节，内容为字符串‘Hi-’对应的 ASCII 码十六进制，必传； **AAABBB** ：设备名称+厂商名称，最长 10 字节，由

适用场景	长度	类型	值	说明
常态广播和碰一碰广播			**B-X** **YYYYSSNNNN**” 已注册： “HI-**AAABBB-X** **YYYYNNNNMPP**”	厂商自定义，内容为对应字符串的 ASCII 码十六进制。可以包含字母、数字、下划线，不支持其他字符，必传； **-**：固定分割符，1 个字节，内容为‘-’对应的 ASCII 码 0x2d，必传； **X**：1 字节版本号，非 0，标识协议的版本号，当前传’1’对应的 ASCII 码十六进制 0x31,必传； **YYYY**：设备类型（ProductId），4 字节，内容为对应字符串 ASCII 码的十六进制，必传； **SS** ：设备子型号(sub ProductID)，默认值为“00”，产品配置多外观时，内容为多外观对应的编号，2 字节，内容为对应字符串 ASCII 码的十六进制，必传； **NNNN**：设备 sn 序列号后四位，4 字节，内容为对应字符串 ASCII 码的十六进制，必选； 已注册： **HI-**：固定前缀，3 字节，内容为字符串‘HI-’对应的 ASCII 码十六进制，必传； **AAABBB** ：设备名称+厂商名称，最长 10 字节，由厂商自定义，内容为对应字符串的 ASCII 码十六进制。可以包含字母、数字、下划线，不支持其他字符，必传； **-**：固定分割符，1 个字节，内容为‘-’对应的 ASCII 码 0x2d，必传； **X**：1 字节版本号，非 0，标识协议的版本号，当前传‘1’对应的 ASCII 码十六进制 0x31，必传； **YYYY**：设备类型（ProductId），4 字节，内容为对应字符串 ASCII 码的十六进制，必传； **NNNN**：设备 sn 序列号后四位，4 字节，内容为对应字符串 ASCII 码的十六进制，必选； **M**: 1 个字节：0x00—0xFF，可选，如果发送该参数，则 PP 为必传；**如果设备侧不携带 MPP 字段，则网关默认不回连该 BLE 设备。** bit0： 0 为心跳广播报文 1 为设备主动回连请求广播报文（比如设备有数据上报需要连接网关） bit1：心跳时长间隔单位。0 为单位毫秒，1 单位为秒。 **PP**：采用小端格式传输， 2 个字节，unsigned int 类型参数，16 进制。标识设备心跳间隔时长， 例如 5000ms，可选。

例如响应包的报文为：48492d48554157454941492d013130314332383031023c00

对应的格式如下：**48492d**（HI-） **4855415745494149**（HUAWEIAI）**2d**（-）**31**（X）**31303143**（YYYY 为 101C）**32383031**（NNNN 为 2801）**02**（M 对应为 0b00000010 为广播心跳，单位为秒）**3c00**（PP 对应 10 进制为 60 秒）

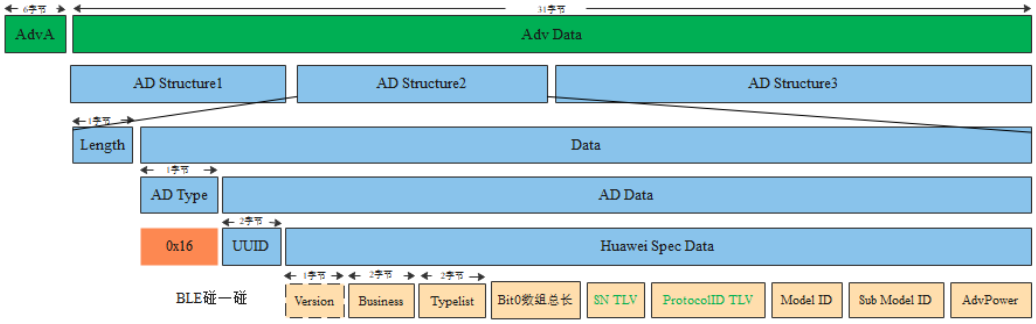
蓝牙碰一碰广播规范

在蓝牙碰一碰中，BLE 广播分为一碰广播、二碰广播。BLE 碰一碰不涉及常态广播。碰一碰广播默认发送，广播间隔根据实际设备自定义。碰一碰广播规范定义了广播包（Advertising Data）和响应包（Scan Response），响应包（Scan Response）装载了蓝牙广播名称，与蓝牙靠近发现相同。

- 一碰广播为设备未注册时发送的碰一碰广播，此时设备与手机碰一碰拉起 FA，请求 FA 进行设备代理注册，ProtocolD 值为 0x15。
- 二碰广播为设备注册成功后发送的碰一碰广播，此时设备与手机碰一碰后，手机 FA 直接进入设备控制页，ProtocolD 值为 0x00/0x01。
- 设备发送一碰广播时，使用智慧生活 App 可以扫描添加设备。

BLE 碰一碰广播结构和 BLE 靠近发现广播整体结构类似，在 Huawei Spec Data 之前的内容二者一致，区别在于 Huawei Spec Data 中 Version 和 Business 字段后，靠近发现协议定义多个 TV 字段，而 BLE 碰一碰则定义了一个 TypeList 后面跟一组 Value 信息。

碰一碰广播包（Advertising Data）结构如下图所示：



说明

AdvA 为 6 字节的 MAC 地址，不占用广播包的长度，此处需要使用 Public MAC，不能使用随机 MAC。

基于上述的广播包结构，我们需要首先了解蓝牙碰一碰广播包规范如表 3-4。

表3-4 蓝牙碰一碰广播包规范

内容			长度（Byte）	值	必选/可选	说明
AD	Length		1	0x 02	必	类型和值的总长度

内容				长度（Byte）	值	必选/可选	说明
Structure1						选	
	AD Type			1	0x 01	必选	0x01 表示 flag
	Value			1	0x 06	必选	支持的 LE/BE/EDR 的 Flag 信息
AD Structure2	Length			1	0x XX	必选	类型和值的总长度
	AD Type			1	0x 16	必选	广播类型，0x16 表示蓝牙服务数据
	UUID			2	0xEEFD	必选	华为购买的 UUID，0xFDEE（已购买）小端存储
	Huawei Spec Data	协议版本		1	0x 01	必选	华为蓝牙广播协议版本
		business		1	0x 06	必选	06 表示碰一碰业务
		businessEx		1	0x 00	必选	默认为 0x00
		typelist		2	0x00xx	必选	根据所需业务使能对应 bit 位，碰一碰传 0x1700（使能的 bit0、bit1、bit2、bit4） bit0：自定义数据，不定长，Length+Value 格式 bit1：prodId bit2：Sub prod ID bit4：AdvPower
		bit0 Length		1	0x0A		bit0 的长度
		bit0 Value	Protocol D T	1	0x 17	必选	蓝牙协议 ID 类型
			Protocol D L	1	0x 01	必选	蓝牙协议 ID 长度
			Protocol D V	1	0x XX	必	蓝牙 ID 值

内容				长度（Byte）	值	必选/可选	说明
						选	bit0：0 表示 member 二碰不弹框，1 表示 member 二碰弹框 bit1：预留字段，默认填 0 bit2~bit7 组成的 int 值标识事件类型，字段说明如下： 0-弹设备控制框 4-设备有异常告警事件，发送告警广播 5-纯蓝牙 BLE 设备请求 App 进行代理注册
			SN T	1	0x 14	必选	设备 SN
			SN L	1	0x 02	必选	设备 SN 长度
			SN V	2	0x XXXX	必选	设备 SN 。设备 SN 最后 2 字节的 ASCII 码值，必须和 deviceinfo 上报的 SN 最后两位一致
			业务自定义	1	0x 91	可选	以下为业务自定义数据
			L	1	自定义 L	可选	业务自定义字段 L
			V	变长	自定义 V	可选	业务自定义字段 V
		prodId V		4	0x XXXXXXXX	必选	产品 ID 值，产品 ID 的 ASCII 码值
		subProdId V		1	0x XX	必选	子产品 ID 值，默认 00
		AdvPower V		1	0x XX	必选	广播的实际发射功率 AdvPower TRP = TxPower （设备芯片广

内容				长度（Byte）	值	必选/可选	说明
							播发射功率）- OTA（设备天线损耗），取值【-128，127】dBm，建议用芯片可配的最小发射功率以节省功耗。例如芯片的广播 TxPower 为-6dBm，天线 OTA 是 10dBm，则 Adv TRP 配置为-16dBm，取值为 0xF0

表3-5 BLE 设备碰一碰广播实例

适用场景	长度	类型	值	说明
以下每个场景都携带	0x02	0x01	0x06	BLE 可被发现
纯 BLE 类设备请求 App 进行代理注册（一碰广播）	0x16	0x16	0xEEFD0106001700071402NNNN170115XXXXXXXXSSPP	未注册，SS 表示产品子型号；PP 表示发射功率值；XXXXXXXX 表示产品 ID 的 ASCII 码值；NNNN 是 SN 后两位的 ASCII 码值。
二碰广播，owner 弹框，member 不弹框	0x16	0x16	0xEEFD0106001700071402NNNN170100XXXXXXXXSSPP	已注册，SS 表示产品子型号；PP 表示发射功率值；XXXXXXXX 表示产品 ID 的 ASCII 码值；NNNN 是 SN 后两位的 ASCII 码值。
二碰广播，owner、member 都弹框	0x16	0x16	0xEEFD0106001700071402NNNN170101XXXXXXXXSSPP	已注册，SS 表示产品子型号；PP 表示发射功率值；XXXXXXXX 表示产品 ID 的 ASCII 码值；NNNN 是 SN 后两位的 ASCII 码值。

交互流程介绍

蓝牙设备与主控端的交互流程包括设备发现、建立连接、代理注册、设备控制四个部分。具体开发内容参考对应的章节，先了解这四个部分的功能简介。

步骤 1 发现设备。

根据产品定义时选择的方式：

- 若选择 NFC 碰一碰，产品伙伴需要完成 NFC 标签相关流程操作，遵循标签规范。
- 若选择蓝牙靠近发现或者蓝牙碰一碰，产品伙伴需要遵循 BLE 广播规范，对外广播 BLE 报文。

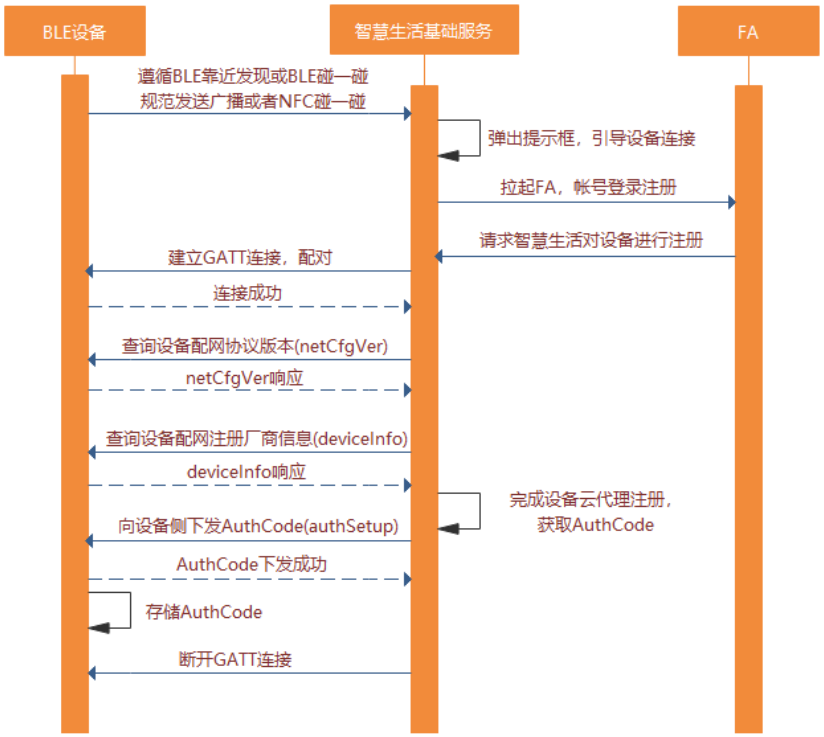
步骤 2 连接设备。

设备采用华为通用 UUID 与智慧生活基础服务建立 GATT 连接。

步骤 3 注册设备。

智慧生活基础服务获取到设备信息后在 HiLink 云侧进行代理注册。

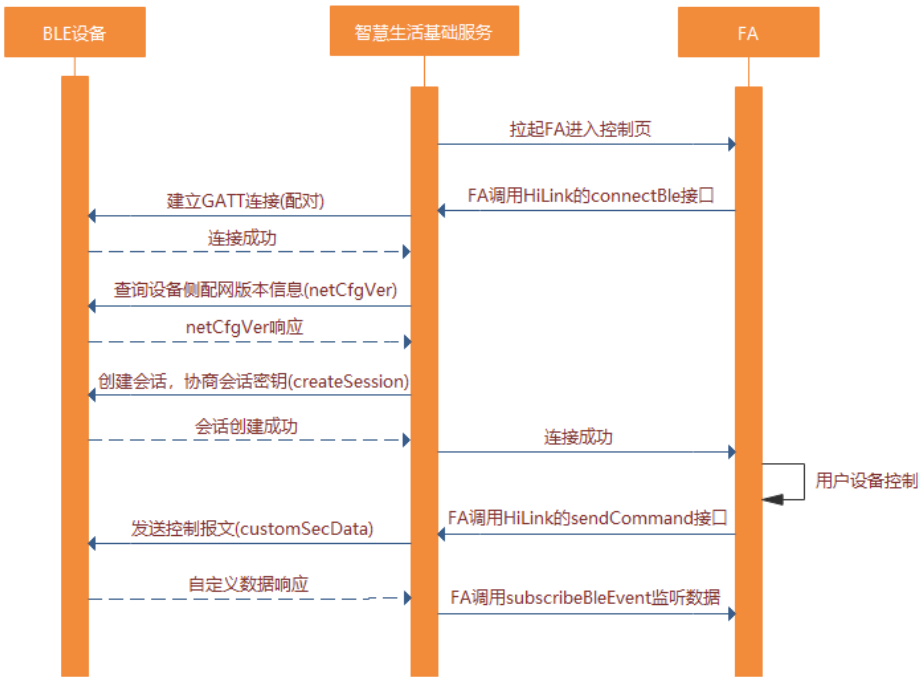
图3-2 设备注册流程



步骤 4 控制设备。

智慧生活基础服务与设备协商 session 会话，并下发控制指令，设备侧响应指令，并主动上报数据。

图3-3 设备控制流程



----结束

3.2 发现设备

3.2.1 蓝牙靠近发现设备

若产品定义时选择的极简交互方式为蓝牙靠近发现，产品伙伴需要遵循蓝牙靠近发现广播包规范，对外广播蓝牙报文。

参考代码如下：

```
int myAdvId = 0; // 广播 id
StartAdvRawData mStartAdvRawData; // 广播的数据参数，内容格式请参考蓝牙广播结构规范

/* 靠近发现，一靠广播 */
unsignedcharmyadvData_0001[30] = {
0x2, 0x1, 0x6, 0x1A, 0x16, 0xEE, 0xFD, 0x01, 0x01, 0x0D, 0x04, 0x00, 0x11, 0xF8,
0x12,
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
0xFF, 0x17, 0x01, 0x15, 0x14, 0x02, demoDevInfo->sn[10], demoDevInfo->sn[11], 0x91,
0x01, 0x01
    };
unsignedcharmyrspData_0001[] = {
0x13, 0x9,  'H', 'i', '-', 'A', 'A', 'A', '-', 0x31,
```



```
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
demoDevInfo->subProductId[0], demoDevInfo->subProductId[1],
demoDevInfo->sn[8], demoDevInfo->sn[9], demoDevInfo->sn[10],demoDevInfo->sn[11]
    };
/* 靠近发现，二靠广播 */
unsignedcharmyadvData_0011[30] = {
0x2, 0x1, 0x6, 0x1A, 0x16, 0xEE, 0xFD, 0x01, 0x01, 0x0D, 0x04, 0x00, 0x11, 0xF8,
0x12,
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
0xFF, 0x17, 0x01, 0x01, 0x14, 0x02, demoDevInfo->sn[10], demoDevInfo->sn[11], 0x91,
0x01, 0x01
    };
unsignedcharmyrspData_0011[] = {
0x14, 0x9,  'H', 'I', '-', 'A', 'A', 'A', '-', 0x31,
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
demoDevInfo->sn[8], demoDevInfo->sn[9], demoDevInfo->sn[10],demoDevInfo->sn[11],
0x02,0x3C,0x00
    };
mStartAdvRawData.advData = myAdvData; // 广播数据
mStartAdvRawData.advDataLen = sizeof(myAdvData); // 广播数据长度
mStartAdvRawData.rspData = myRspData; // 广播响应数据
mStartAdvRawData.rspDataLen = sizeof(myRspData); // 响应数据长度
BleAdvParams mBleAdvParams; // BLE广播参数
mBleAdvParams.minInterval = 32; // 最小广播间隔
mBleAdvParams.maxInterval = 64; // 最大广播间隔
mBleAdvParams.advType = 0x00; // 可连接和可扫描的无定向广播
mBleAdvParams.ownAddrType = 0; // 本地地址类型
mBleAdvParams.peerAddrType = 0; // 对端地址类型
mBleAdvParams.txPower = 0; // 传输功率
mBleAdvParams.duration = 0; // 广播时长
mBleAdvParams.channelMap = 7; // 广播通道
mBleAdvParams.advFilterPolicy = 0; // 基于白名单的广播过滤策略
BleStartAdvEx(&myAdvId, mStartAdvRawData, mBleAdvParams); //发送广播
```

3.2.2 蓝牙碰一碰发现设备

若产品定义时选择的极简交互方式为蓝牙碰一碰，产品伙伴需要遵循蓝牙碰一碰广播包规范，对外广播蓝牙报文。

参考代码如下：

```
int myAdvId = 0; // 广播 id
StartAdvRawData mStartAdvRawData; // 广播的数据参数，内容格式请参考蓝牙广播结构规范
/* 碰一碰，一碰广播 */
unsignedcharmyadvData_1000[29] = {
0x2, 0x1, 0x6, 0x19, 0x16, 0xEE, 0xFD, 0x01, 0x06, 0x0, 0x17, 0x0, 0xA, 0x17, 0x01,
0x15, 0x14, 0x02,
demoDevInfo->sn[10], demoDevInfo->sn[11], 0x91, 0x01, 0x56, demoDevInfo-
>productId[0],
demoDevInfo->productId[1], demoDevInfo->productId[2], demoDevInfo->productId[3],
0x0, 0xF8};
unsignedcharmyrspData_1000[] = {
0x13, 0x9, 'H', 'i', '-', 'A', 'A', 'A', '-', 0x31,
```



```
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
demoDevInfo->subProductId[0], demoDevInfo->subProductId[1],
demoDevInfo->sn[8], demoDevInfo->sn[9], demoDevInfo->sn[10], demoDevInfo->sn[11]};
/* 碰一碰，二碰广播 */
unsignedcharmyadvData_1010[29] = {
0x2, 0x1, 0x6, 0x19, 0x16, 0xEE, 0xFD, 0x01, 0x06, 0x0, 0x17, 0x0, 0xA, 0x17, 0x01,
0x01, 0x14, 0x02,
demoDevInfo->sn[10], demoDevInfo->sn[11], 0x91, 0x01, 0x56, demoDevInfo-
>productId[0],
demoDevInfo->productId[1], demoDevInfo->productId[2], demoDevInfo->productId[3],
0x0, 0xF8};
unsignedcharmyrspData_1010[] = {
0x11, 0x9, 'H', 'I', '-', 'A', 'A', 'A', '-', 0x31,
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
demoDevInfo->sn[8], demoDevInfo->sn[9], demoDevInfo->sn[10], demoDevInfo->sn[11]};

mStartAdvRawData.advData = myAdvData; // 广播数据
mStartAdvRawData.advDataLen = sizeof(myAdvData); // 广播数据长度
mStartAdvRawData.rspData = myRspData; // 广播响应数据
mStartAdvRawData.rspDataLen = sizeof(myRspData); // 响应数据长度
BleAdvParams mBleAdvParams; // BLE广播参数
mBleAdvParams.minInterval = 32; // 最小广播间隔
mBleAdvParams.maxInterval = 64; // 最大广播间隔
mBleAdvParams.advType = 0x00; // 可连接和可扫描的无定向广播
mBleAdvParams.ownAddrType = 0; // 本地地址类型
mBleAdvParams.peerAddrType = 0; // 对端地址类型
mBleAdvParams.txPower = -650; // 传输功率
mBleAdvParams.duration = 0; // 广播时长
mBleAdvParams.channelMap = 7; // 广播通道
mBleAdvParams.advFilterPolicy = 0; // 基于白名单的广播过滤策略
BleStartAdvEx(&myAdvId, mStartAdvRawData, mBleAdvParams); //发送广播
```

3.2.3 NFC 碰一碰发现设备

若产品定义时选择的极简交互方式为 NFC 碰一碰，则完成 NFC 标签相关流程操作，参见 NFC 标签认证。

同时需要按照华为的规范发送广播响应包，详情请参考表 3-3。

3.3 连接设备

BLE 设备进行蓝牙配对需要采用 Justworks 以上安全配对模式，若设备没有显示屏显示 PIN 码，建议采用 Justworks 安全配对。与 IOS 设备对接需要支持配对协商，由于 IOS 不支持配对，可以协商成免配对模式，设备侧需要支持 IOS 协商成免配对接入，如有疑问请咨询芯片厂商。配对完成后，设备采用华为通用 UUID 与智慧生活基础服务建立 GATT 连接，UUID 为固定值，不能修改。建立连接时手机侧会主动进行 MTU 协商，设备侧需要关闭 MTU 主动协商，被动响应 MTU 协商。

表3-6 BLE 通用 UUID 字段描述

服务实例 sid	serviceUuid	服务名称(中文)	服务类型 ServiceType	属性characteristics	characteristic Uuid	操作权限 Permissions	数据类型 Format
hilinkData	15f1e600-a277-43fc-a484-dd39ef8a9100	批量数据读取	hilinkData	readData	15f1e601-a277-43fc-a484-dd39ef8a9100	indicate/ read	String
				writeData	15f1e602-a277-43fc-a484-dd39ef8a9100	write	String

参考代码（以 OpenHarmony 蓝牙协议 api 为例）如下：

```
BleGattService mBleGattService; // GATT 服务
BleGattAttr attrList[3]; // 属性列表

// 使能蓝牙协议栈
EnableBtStack();
// 设置服务的句柄 id
int hilinkServiceId = 0;
// 设置 service 包含的 attr 数量，和 attrList 数组长度一致
mBleGattService.attrNum = 3;
// attrList 代表 attr 属性列表。attrType 代表 attr 类型，0x0 为 service 服务，0x1 为
characteristic 特征
attrList[0].attrType = 0;
// attr 权限，0x01 代表读权限，0x10 代表写权限
attrList[0].permission = 0;
// uuid 类型，0x1 代表 16bit uuid，0x2 代表 32bit uuid，0x3 代表 128bit uuid
attrList[0].uuidType = 3;
// uuid 值赋予 attrList[0].uuid，倒序赋值
unsigned char tempServiceUuid[] = {0x0, 0x91, 0x8a, 0xef, 0x39, 0xdd, 0x84, 0xa4,
0xfc, 0x43, 0x77, 0xa2, 0x0, 0xe6, 0xf1, 0x15};
for(int i = 0; i < sizeof(tempServiceUuid); i++) {
    attrList[0].uuid[i] = tempServiceUuid[i];
}
// attr 的数据
attrList[0].value = "";
// 数据长度
attrList[0].valLen = 0;
// characteristic 特征属性。默认值
attrList[0].properties = 0;
// GATT 客户端请求从服务器端读数据时，被调用
attrList[0].func.read = &myBleGattServiceRead;
```



```
// GATT 客户端请求向服务器端写数据时，被调用
attrList[0].func.write = &myBleGattServiceWrite;
// 当 indication 或者 notification 发送给服务时调用
attrList[0].func.indicate = &myBleGattServiceIndicate;

// attrType ， 0x1 为 characteristic 特征
attrList[1].attrType = 1;
// 可读写权限
attrList[1].permission = 0x11;
// 0x3 代表 128bit uuid
attrList[1].uuidType = 3;
// GATT characteristic UUID 为"15F1E602A27743FCA484DD39EF8A9100"，倒序赋值
unsigned char tempWriteUuid[] = {0x0, 0x91, 0x8a, 0xef, 0x39, 0xdd, 0x84, 0xa4,
0xfc, 0x43, 0x77, 0xa2, 0x2, 0xe6, 0xf1, 0x15};
for(int i = 0; i < sizeof(tempWriteUuid); i++) {
    attrList[1].uuid[i] = tempWriteUuid[i];
}
attrList[1].value = "";
attrList[1].valLen = 0;
// characteristic 可以被写，响应需要发送给客户端
attrList[1].properties = 8;
attrList[1].func.read = &myBleGattServiceRead;
attrList[1].func.write = &myBleGattServiceWrite;
attrList[1].func.indicate = &myBleGattServiceIndicate;
// characteristic 特征
attrList[2].attrType = 1;
// 可读写权限
attrList[2].permission = 0x11;
// 0x3 代表 128bit uuid
attrList[2].uuidType = 3;
// 创建 GATT characteristic UUID 为"15F1E601A27743FCA484DD39EF8A9100"，倒序赋值
unsigned char tempReadUuid[] = {0x0, 0x91, 0x8a, 0xef, 0x39, 0xdd, 0x84, 0xa4, 0xfc,
0x43, 0x77, 0xa2,0x1, 0xe6, 0xf1, 0x15};
for(int i = 0; i < sizeof(tempReadUuid); i++) {
    attrList[2].uuid[i] = tempReadUuid[i];
}
attrList[2].value = "";
attrList[2].valLen = 0;
// characteristic 可以读和发送 indication
attrList[2].properties = 0x22;
attrList[2].func.read = &myBleGattServiceRead;
attrList[2].func.write = &myBleGattServiceWrite;
attrList[2].func.indicate = &myBleGattServiceIndicate;
mBleGattService.attrList = attrList;
// 创建 GATT 服务
BleGattsStartServiceEx(&hilinkServiceId, &mBleGattService);
```

3.4 注册设备

智慧生活基础服务获取到设备信息后在 HiLink 云侧进行代理注册，主要包括：

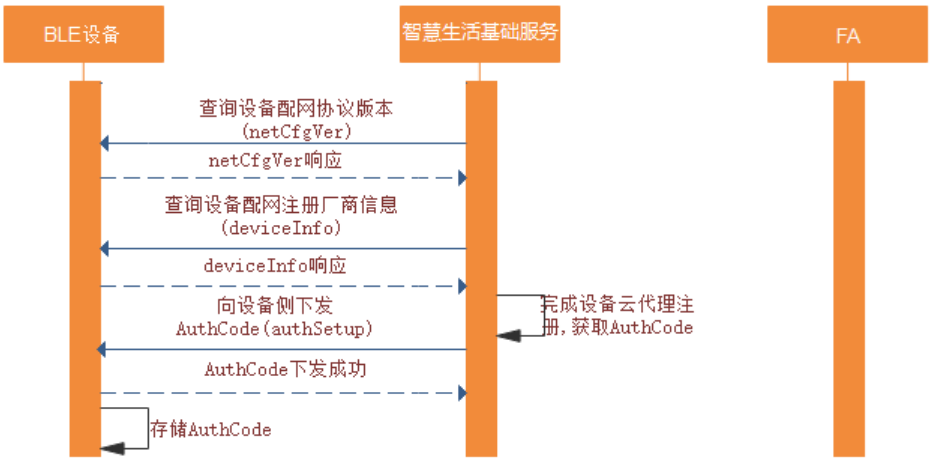
步骤 1 查询设备配网协议版本（netCfgVer）；

步骤 2 查询设备配网注册厂商信息（deviceInfo）；

步骤 3 向设备下发 AuthCode（authSetup）；

步骤 4 设备存储 AuthCode。

图3-4 设备代理注册流程



----结束

代理注册过程涉及到设备和智慧生活基础服务之间的消息交互，首先要重点了解这部分蓝牙数据段数据结构定义。我们以一个具体的消息内容为例，其定义和说明详见表3-7。

如：01 34 01 00 00 00 00 11 09 6e 65 74 43 66 67 56 65 72 09 00 7b 22 76 65 72 22 3a 32 7d

表3-7 蓝牙数据段数据结构定义

内容		长度	值	必选/可选	说明
Header	Version	4 bit	0b0000	必选	version，默认值：0b0000
	CMD Type	4 bit	0bxxxx	必选	0b0000:REQ 请求消息； 0b0001:RSP 响应消息； 0b0010:RPT 主动上报消息
	Message Id	1 byte	0xXX	必选	消息 ID，范围 0~255 • 每发送一条消息 ID 加 1 • 如果消息有应答时，应答的消息 ID 需要和请求的消

内容		长度	值	必选/可选	说明
					息 ID 匹配 • ID 超过 255，自动循环到 0
	Total Frame	1 byte	0xXX	必选	总包数，范围 0~255 • 不分包则该字段设置为 1 • 分包则字段设置为大于等于 2
	Frame Seq	1 byte	0xXX	必选	分包报文序列号 • 如果不需要分片则置为 0 • 如果需要分片则该字段为帧序号，从 0 开始依次添增；比如，首包填 0，第二个包填 1，依次递增。
	Rev	1 byte	XX	必选	00--Rev 保留字段
	Encry	1 byte	XX	必选	数据加密类型 0x00：不加密（BLE 需要 SMP 安全配对， 配对建议采用 Justwork） 0x03：sessionKey 加密
	Return	1 byte	XX	必选	请求、上报消息填写 0x00，响应消息填写具体结果 0x00：执行成功 0x01：操作失败
Payload	Rev	1 byte	0x11	必选	保留字段，便于后续扩展，默认填 0x11
	Service Length	1 byte	XX	必选	服务名长度，比如 netCfgVer 服务名字的长度为 9
	Service Name	n bytes	根据 Service Length 值	必选	服务名的 ASCII 码十六进制，比如 netCfgVer 对应 6e 65 74 43 66 67 56 65 72
	Body Length	2 bytes	0xXX XX	必选	数据长度，为两个字节，采用小端格式传输，例如： 例如 0x64 0x00 表示长度为 0x0064

内容		长度	值	必选/可选	说明
	Body	n bytes	根据 Body Length 值	必选	json 数据内容
	Hmac	32bytes	根据计算得到	可选	数据包校验值，使用 customSecData 时需要

结合蓝牙数据段数据结构定义，我们以蓝牙代理注册过程中设备和智慧生活基础服务之间交互的消息内容为例，说明如表 3-8。

表3-8 代理注册过程通用服务描述

服务名	说明	请求格式	响应格式
netCfgVer	蓝牙代理注册配网协议版本信息。 加密方式：不加密。 响应值为 100-199，默认值为 100	**netCfgVer** 例如：0 34 1 0 0 0 0 11 **9 6e 65 74 43 66 67 56 65 72** 0 0 说明：**9** 即为 Service Length(9 字节)，**6e 65 74 43 66 67 56 65 72** 即为 Service Name（netCfgVer） 另外，Version 是连接配网协议的版本信息，是固定的，不是设备版本信息。	**{"ver":100}** 例如：1 34 1 0 0 0 0 11 9 6e 65 74 43 66 67 56 65 72 **a 0 7b 22 76 65 72 22 3a 331 30 30 7d** 说明：响应数据，从 Message Id 到 Service Name 这部分内容和请求内容是一致的。 **a 0** 即为 Body Length（11 字节），**7b 22 76 65 72 22 3a 31 30 30 7d** 即为 Body（{"ver":100}）。
deviceInfo	设备配网注册产品伙伴信息。 加密方式：不加密。 说明： sn: 设备 SN 号 model: 设备型号 manu: 厂商名称 prodId: 华为给厂商分配的产品唯一标识 **mac**：BLE MAC 地址 blemac: BLE MAC 地址	**deviceInfo** 例如：0 35 1 0 0 0 0 11 **a 64 65 76 69 63 65 49 6e 66 6f** 0 0 说明：**a** 即为 Service Length(10 字节)，**64 65 76 69 63 65 49 6e 66 6f** 即为 Service Name（deviceInfo）	**{** **"productId":"2EKT",** **"sn":"701d080c1fe3",** **"vendor":{** **"devId":"",** **"deviceInfo":{** **"model":"12345",** **"manu":"huawei",** **"prodId":"2EKT",** **"mac":"70:1D:08:0C:1F:E3",** **"blemac":"70:1D:08:0**

服务名	说明	请求格式	响应格式
	fwv: 固件版本号 hwv:硬件版本号 swv: SDK 版本号 prot_t: 协议类型，默认填写 4:为 BLE 连接协议		**C:1F:E3",** **"fwv":"123456",** **"hwv":"123456",** **"swv":"",** **"prot_t":"4"** **}** **}** **}** 原始数据：数据量比较大，需要分包发送，具体参考表 3-9
authSetup	设备注册后将设备 authCode 发送至设备，基于 authCode 派生秘钥。 加密方式：不加密 说明： • **authCode** 和 **authCodeId** 的长度为 **32** 字节，由 **16 bytes** 的二进制数据转成 **16** 进制字符串而成，用于生成密钥。本地存储。需要转换成 hex 格式使用	**{** **"devId":"49e0c206-9a3f-427c-98fa-2a95dea4e0e2",** **"authCode":"8dcdfb4c9ddc2f57c72d4d211d4c224b",** **"uidHash":"5c8ca2dfc61e07fd58961a879103edf095ad68722b7476ee822dd5cb7847e227",** **"authCodeId":"d8fb814bac98c1b55f27f7b417cb9a75"** **}**	**{"errcode":"0"}**

在收发消息时，如果消息长度较大，超过了 MTU 的协商值，需要拆包和合包。分包是对 Payload 进行分包发送，每个分包是由 Header+Payload 分包组成，每个包的长度不能超过 MTU 值；Header 中的 Total Frame 表示总包数，Frame Seq 表示当前是第几个包。

下面我们以 deviceInfo 数据包为例进行分包：

表3-9 deviceInfo 数据包响应内容实例

分包前	分包后	
Payload 内容	数据包序号	内容
11 0a 64 65 76 69 63 65 49 6e 66 6f	数据	01 35 **03 00** 00 00 00 **11 0a 64 65 76 69**

<table>
<tr><th>分包前</th><th colspan="2">分包后</th></tr>
<tr><td rowspan="3">25 01 7b 22 70 72 6f 64 75 63 74 49 64 22 3a 22 32 45 4b 54 22 2c 22 73 6e 22 3a 22 37 30 31 64 30 38 30 63 31 66 65 33 22 2c 22 76 65 6e 64 6f 72 22 3a 7b 22 64 65 76 49 64 22 3a 22 22 2c 22 64 65 76 69 63 65 49 6e 66 6f 22 3a 7b 22 73 6e 22 3a 22 37 30 31 64 30 38 30 63 31 66 65 33 22 2c 22 6d 6f 64 65 6c 22 3a 22 31 32 33 34 35 22 2c 22 64 65 76 5f 74 22 3a 22 38 37 32 30 22 2c 22 6d 61 6e 75 22 3a 22 68 75 61 77 65 69 22 2c 22 70 72 6f 64 49 64 22 3a 22 32 45 4b 54 22 2c 22 6d 61 63 22 3a 22 37 30 3a 31 44 3a 30 38 3a 30 43 3a 31 46 3a 45 33 22 2c 22 62 6c 65 6d 61 63 22 3a 22 37 30 3a 31 44 3a 30 38 3a 30 43 3a 31 46 3a 45 33 22 2c 22 68 69 76 22 3a 22 31 32 33 34 35 36 37 38 22 2c 22 66 77 76 22 3a 22 31 32 33 34 35 36 22 2c 22 68 77 76 22 3a 22 31 32 33 34 35 36 22 2c 22 73 77 76 22 3a 22 31 32 33 34 35 36 22 2c 22 70 72 6f 74 5f 74 22 3a 22 31 32 33 34 22 7d 7d7d</td><td>包 1</td><td>63 65 49 6e 66 6f 25 01 7b 22 70 72 6f 64 75 63 74 49 64 22 3a 22 32 45 4b 54 22 2c 22 73 6e 22 3a 22 37 30 31 64 30 38 30 63 31 66 65 33 22 2c 22 76 65 6e 64 6f 72 22 3a 7b 22 64 65 76 49 64 22 3a 22 22 2c 22 64 65 76 69 63 65 49 6e 66 6f 22 3a 7b 22 73 6e 22 3a 22 37 30 31 64 30 38 30 63 31 66 65 33 22 2c 22 6d 6f 64 65 6c 22 3a 22 31 32 33 34 35 22 2c 22 64 65 76 5f 74 22 3a 22 38 37 32 30 22 2c 22 6d 61 6e 75 22 3a 22 68 75 61 77 65</td></tr>
<tr><td>数据包 2</td><td>01 35 03 01 00 00 00 69 22 2c 22 70 72 6f 64 49 64 22 3a 22 32 45 4b 54 22 2c 22 6d 61 63 22 3a 22 37 30 3a 31 44 3a 30 38 3a 30 43 3a 31 46 3a 45 33 22 2c 22 62 6c 65 6d 61 63 22 3a 22 37 30 3a 31 44 3a 30 38 3a 30 43 3a 31 46 3a 45 33 22 2c 22 68 69 76 22 3a 22 31 32 33 34 35 36 37 38 22 2c 22 66 77 76 22 3a 22 31 32 33 34 35 36 22 2c 22 68 77 76 22 3a 22 31 32 33 34 35 36 22 2c 22 73 77 76 22 3a 22 31 32 33 34 35 36 22 2c 22 70 72 6f 74 5f 74 22 3a 22 31 32 33 34 22 7d 7d</td></tr>
<tr><td>数据包 3</td><td>01 35 03 02 00 00 00 7d</td></tr>
</table>

3.5 控制设备

智慧生活基础服务与设备协商 session 会话，并下发控制指令，主要包括：

步骤 1 查询设备配网协议版本（netCfgVer），实现方式参考代理注册流程；

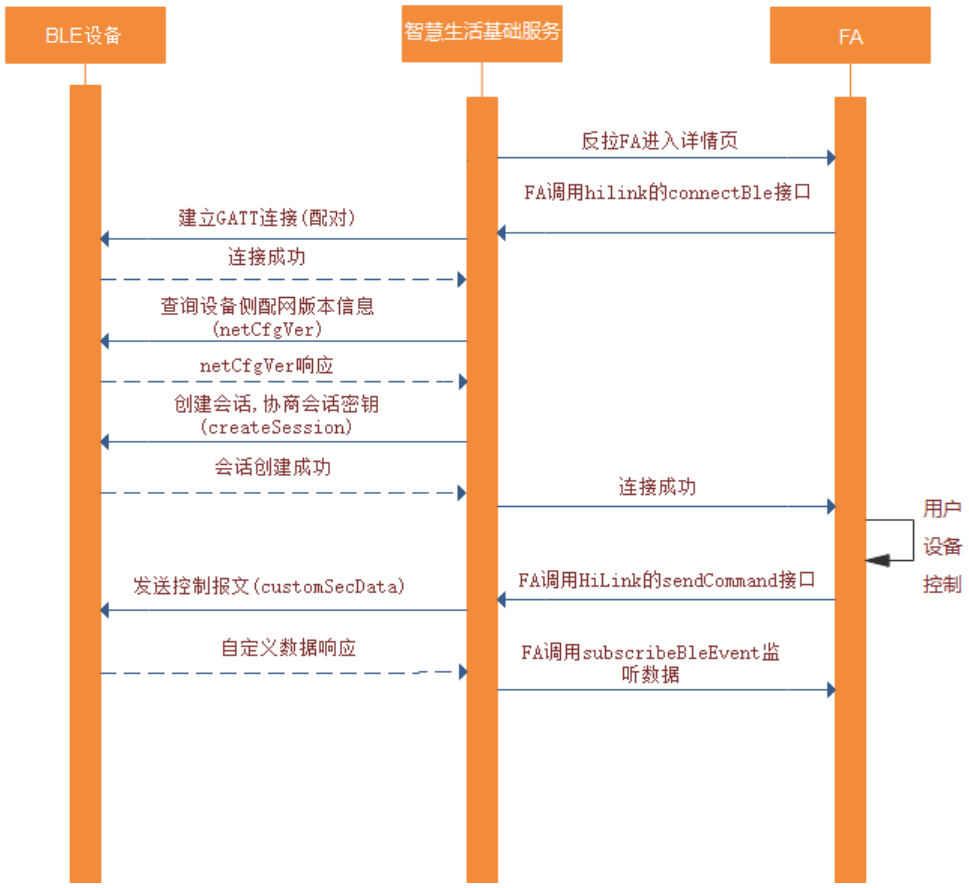
步骤 2 创建会话（createSession）；

步骤 3 生成秘钥；

步骤 4 设备控制（customSecData）。

----结束

图3-5 设备控制流程



3.5.1 创建会话

表3-10 代理注册过程通用服务描述

服务项	内容	说明
服务名	createSession	设备控制前创建会话的服务。 蓝牙代理注册协议版本信息加密方式：不加密。

请求格式	{
	"seq":31330,
	"sn1":"8579eaa63ce56975",
	"uidHash":"5c8ca2dfc61e07fd58961a879103edf095ad68722b7476ee822dd5cb7847e227",
	"uuid":"27c3ff9a-4f1d-45e7-9280-e9deeb0e26db"
	}
		入参 sn1 为字符串格式，需要转换成 16 进制数，才能使用。
		例如：
		"sn1": "8579eaa63ce56975"
		转换为：
		sn1[]={0x85,0x79,0xea,0xa6,0x3c,0xe5,0x69,0x75}

响应格式	{
	"seq": 31330,
	"uuid": "27c3ff9a-4f1d-45e7-9280-e9deeb0e26db",
	"sessId": "3030303132333435363738393031323334356373839303132333435363738930313233334356373839ef",
	"sn2": "3132333435363738",
	"authCodeId": "d8fb814bac98c1b55f27f7b417cb9a75"
	}
		sn2 和 sessId 是设备侧自定义生成的 16 进制数，需要转换为字符串格式使用。
		例如：
		sn2[]={0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38}
		转换为："sn2": "3132333435363738"

表3-11 设备控制涉及参数说明

参数	长度	类型	说明
sn1	8 Byte	unsigned char	应用侧生成的动态随机数，用于生成盐值
sn2	8 Byte	unsigned char	设备侧生成的动态随机数，用于生成盐值
salt	16 Byte	unsigned char	SN1 | SN2 生成的盐值，用于生成 sessionKey 和 hmacKey
authCode	16 Byte	unsigned char	设备注册时应用侧下发，用于生成 sessionKey
sessIdHex	32 Byte	unsigned char	会话 ID，设备侧生成动态随机数
sessionKey	16 Byte	unsigned char	加解密秘钥，PBKDF2（authCode，SN1 | SN2）用于后续 AES128 加解密运算。
hmacKey	32 Byte	unsigned char	校验码秘钥，PBKDF2（Sessionkey，SN1 | SN2）用于后续 HMacSHA256 运算。
authCodeId	16 Byte	unsigned char	设备注册时应用侧下发,在创建会话的响应消息中需要携带

参数	长度	类型	说明
Encrydata	变长	unsigned char	加密数据
tag	16 Byte	unsigned char	数据加密时生成

3.5.2 协商生成秘钥

步骤 1 使用 SN1 | SN2 生成盐值，盐值用于后面生成密钥。参考代码如下：

```
static int SALT_LEN = 16;
int SN_LEN = 8;
memcpy_s(salt, sizeof(salt), sn1, SN_LEN);                    //把 sn1 放在 salt 的前
8Byte
memcpy_s(salt + SN_LEN, sizeof(salt) - SN_LEN , sn2, SN_LEN);    //把 sn2 放在 salt 的后
8Byte
```

样例数据（以 16 进制为例）：

```
输入：sn1：0x85,0x79,0xea,0xa6,0x3c,0xe5,0x69,0x75
    sn2：0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38
输出 salt：
0x85,0x79,0xea,0xa6,0x3c,0xe5,0x69,0x75,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38
```

步骤 2 以 mbedtls 开源库的 2.16.10 版本为例，计算 sessionKey(真正加解密的密钥)，参考代码如下：

```
mbedtls_md_context_t ctx;
const mbedtls_md_info_t *mdInfo = NULL;
mdInfo = mbedtls_md_info_from_type(MBEDTLS_MD_SHA256);
mbedtls_md_init(&ctx);
/* 数字 1 表示使用 HMAC */
mbedtls_md_setup(&ctx, mdInfo, 1);
mbedtls_pkcs5_pbkdf2_hmac(&ctx,
    authCode,         //设备代理注册生成的 authCode
    authCodeLen,      //authCode 的长度，16Byte
    salt,            //SN1|SN2 生成的盐值
    saltLen,          //salt 长度，16Byte
    iterCount,        //设置值为 1
    keyLen,          //sessionKey 的长度，16Byte
    output);          //sessionKey
```

样例数据（以 16 进制为例）：

```
输入：authCode：
0xb5,0x50,0x08,0x67,0x50,0x03,0x05,0xed,0x92,0xbb,0xb4,0x68,0xcd,0xc6,0x23,0x6d
    salt：
0x85,0x79,0xea,0xa6,0x3c,0xe5,0x69,0x75,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38
输出 sessionKey：
0x4b,0x2f,0x68,0x3b,0xa3,0x6e,0x8e,0xfa,0x4d,0xb2,0x0e,0xf2,0x5b,0xc2,0x9e,0x4b
```

步骤 3 计算 hmacKey(用于生成 hmac 校验码)，参考代码如下：

```
mbedtls_md_context_t ctx;
const mbedtls_md_info_t *mdInfo = NULL;
```



```
mdInfo = mbedtls_md_info_from_type(MBEDTLS_MD_SHA256);
mbedtls_md_init(&ctx);
/* 数字 1 表示使用 HMAC */
mbedtls_md_setup(&ctx, mdInfo, 1);
mbedtls_pkcs5_pbkdf2_hmac(&ctx,
    sessionKey,         //上一步生成的 sessionKey
    sessionKeyLen,      //sessionKey 的长度，16Byte
    salt,              //SN1|SN2 生成的盐值
    saltLen,           //salt 长度，16Byte
    iterCount,         //设置值为 1
    keyLen,            //hmacKey 长度,32 Byte
    output);           //hmacKey
```

样例数据：

```
输入 sessionKey:
0x4b,0x2f,0x68,0x3b,0xa3,0x6e,0x8e,0xfa,0x4d,0xb2,0x0e,0xf2,0x5b,0xc2,0x9e,0x4b
    salt:
0x85,0x79,0xea,0xa6,0x3c,0xe5,0x69,0x75,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38
输出 hmacKey:
0x13,0xf4,0xdb,0x60,0xff,0xba,0x91,0xbd,0x3e,0x36,0xe2,0x1,0x1c,0x94,0xaa,0xf4,0x54
,0xbf,0xb5,0x65,0xf6,0x7,0xd8,0x47,0xe,0x7c,0x50,0x81,0x52,0x2a,0xf0,0x82
```

----结束

3.5.3 控制设备

设备控制的数据，是经过加密处理的，加解密需要占用的 RAM 为 32KB 左右。

服务描述

app 侧与设备侧的控制消息交互及设备侧主动上报数据均使用 customSecData 服务名，为了确保设备能够正常接入蓝牙网关，消息内容需要遵循 Device Partner 平台配置的 profile，并按照 json 格式发送。

表3-12 设备控制服务描述

服务项	内容	说明
服务名	customSecData	设备控制的数据，是经过加密处理的，控制加密方式：sessionKey，Header 中的 Encry 字段需要设置为 0x03。
请求格式	• Rev:应用侧下发 • Service Length:0x0D • Service Name:customSecData • Body Length:加密数据包长度 • Body:加密数据包	完整的 payload 格式。 Body 的前 12Byte 是 IV（明文），中间是 EncryData（加密数据，最后 16Byte 是加密时 mbedtls 生成的 tag），最后 32Byte 是 sessIdHex

	• Hmac：32Byte 的 Hmac 校验值	（明文）。 加密前控制指令或主动上报数据格式 {"seq":xxxx,// xxxx 为递增数字 "vendor":"%s"}
响应格式	• Rev:保留字段，填 0x11。 • Service Length:0x0D • Service Name:customSecData • Body Length:加密数据包长度 • Body:加密数据包 • Hmac：Hmac 校验值	

vendor 的内容**必须遵循华为 Profile 协议规范**，内容为 JSON 格式字符串，示例如下：

服务 sid	服务(中文)	服务类型 ServiceType	属性	属性类型 Character Type	操作权限	数据类型	数据约束 (IT系统录入时使用)	取值范围	描述
onoff	开关								
			onoff		GET/REPORT	enum		0-已连接 1-未连接 2-工作中	设备和应用连接时的状态，以及信息

设备侧收到的控制指令格式

```
{
    "seq":xxxx,        // xxxx 为递增数字
    "vendor":{
        "sid":"onoff",
        "data":{
            "onoff":0
        }
    }
}
```

收到指令后返回格式（可选），也可以直接返回执行结果

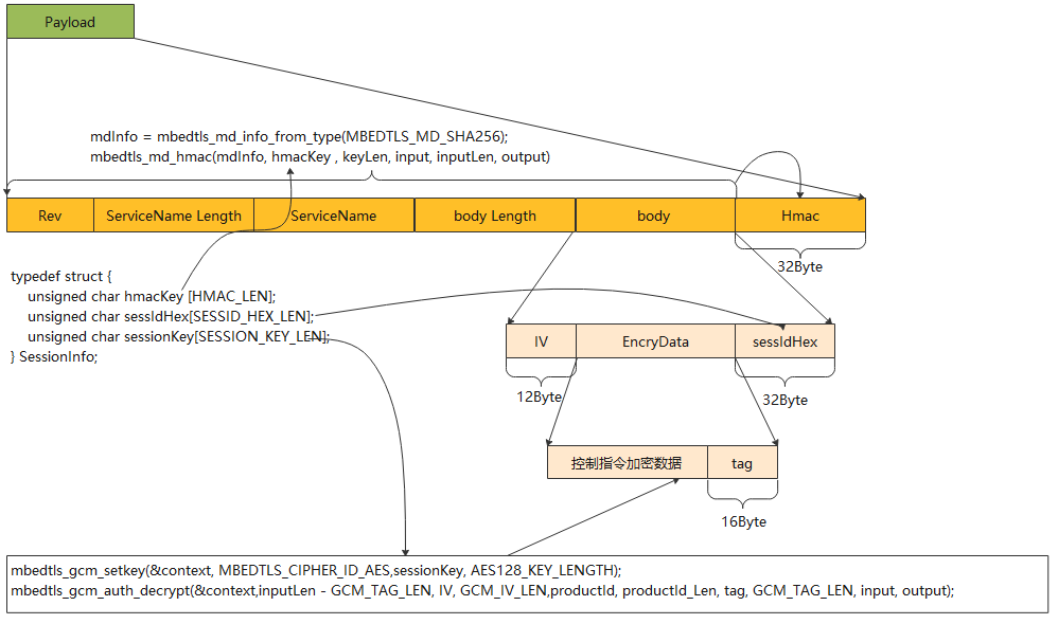
```
{
   "errcode": 0  //返回成功或者 "errcode": xxxx 返回出错
}
```

设备侧执行指令后上报执行结果

```
{
    "seq":xxxx,        // xxxx 为递增数字
    "vendor":{
        "sid":"onoff",
        "data":{
            "onoff":0
        }
    }
}
```


Payload 数据包格式说明

图3-6 Payload 数据包格式



设备控制

步骤 1 解密控制指令

以 mbedtls 开源库(推荐 2.16.10 版本)为例，使用 createSession 过程生成的密钥 sessionKey 对控制命令数据包(EncryData)进行解密；使用 AES128 加密算法对数据进行解密。参考代码如下：

```
#define AES128_KEY_LENGTH 128
#define GCM_TAG_LEN 16

unsigned char tag[GCM_TAG_LEN] = {0};
mbedtls_gcm_context context;
mbedtls_gcm_init(&context);
// 取 Encrydata 后 16Byte 生成 tag
memcpy_s(tag, sizeof(tag), input + inputLen - GCM_TAG_LEN, GCM_TAG_LEN);
// 设置解密使用的 sessionKey
mbedtls_gcm_setkey(&context, MBEDTLS_CIPHER_ID_AES, sessionKey, AES128_KEY_LENGTH);
mbedtls_gcm_auth_decrypt(&context,
    inputLen - GCM_TAG_LEN,    // 需要解密数据的长度
    IV,                       //IV 值取 body 前 12Byte
    GCM_IV_LEN,                 //IV 长度 12Byte
    productId,                  //productId，字符串
    productIdLen,               //productId 长度，值为 4
    tag,                      //取自 EncryData 末尾 16Byte
    GCM_TAG_LEN,                //tag 长度 16Byte
    input,                    //Encrydata
    output)                   //解密输出的数据
```


样例数据（因为 tag 有时效性，所以不能直接照搬样例数据，样例数据仅供参考）：

```
Encrydata:
0x13,0x08,0x61,0x6a,0xf1,0x4e,0x84,0x8a,0x3b,0xf1,0x74,0x97,0x20,0xbc,0x26,0x34,0x4
a,0x6e,0x5f,0x90,0x87,0xea,0xc0,0x75,0x58,0x19,0x03,0xde,0x8b,0xa2,0x83,0x6d,0x79,0
xd9,0x38,0x12,0xc9,0x44,0x04,0x6c,0x84,0x7c,0x33,0x82,0x6f,0x68,0xb9,0x5e,0x42,0x15
,0xcf,0x88,0x2a,0xf3,0x5a,0x13,0xe5,0x74,0x1a,0x15,0x65,0x60,0xf5,0x6b,0x16,0x7d,0x
0b,0xef,0xc0,0xb8,0xf8
需要解密的数据:
0x13,0x08,0x61,0x6a,0xf1,0x4e,0x84,0x8a,0x3b,0xf1,0x74,0x97,0x20,0xbc,0x26,0x34,0x4
a,0x6e,0x5f,0x90,0x87,0xea,0xc0,0x75,0x58,0x19,0x03,0xde,0x8b,0xa2,0x83,0x6d,0x79,0
xd9,0x38,0x12,0xc9,0x44,0x04,0x6c,0x84,0x7c,0x33,0x82,0x6f,0x68,0xb9,0x5e,0x42,0x15
,0xcf,0x88,0x2a,0xf3,0x5a
tag:
0x13,0xe5,0x74,0x1a,0x15,0x65,0x60,0xf5,0x6b,0x16,0x7d,0x0b,0xef,0xc0,0xb8,0xf8
productId: 26W5
IV Data: 0x62,0x30,0xfc,0x88,0x79,0x87,0x4d,0xde,0x59,0x53,0xb9,0xdc
sessKey:
0x4b,0x2f,0x68,0x3b,0xa3,0x6e,0x8e,0xfa,0x4d,0xb2,0x0e,0xf2,0x5b,0xc2,0x9e,0x4b
解密输出的数据：{"vendor":{"data":{"on":1},"sid":"switch"},"seq":15572}
```

步骤 2 数据处理

参考同以上明文数据处理。

步骤 3 数据加密

以 mbedtls 开源库(推荐 2.16.10 版本)为例，使用 createSession 过程生成的密钥 sessionKey 对响应数据或主动上报数据进行加密；使用 AES128 加密算法对数据进行加密。代码参考如下：

```
#define AES128_KEY_LENGTH 128
unsigned char tag[GCM_TAG_LEN] = {0};
mbedtls_gcm_context context;
mbedtls_gcm_init(&context);
//设置加密使用的 sessionKey。
mbedtls_gcm_setkey(&context, MBEDTLS_CIPHER_ID_AES, sessionKey, AES128_KEY_LENGTH);
mbedtls_gcm_crypt_and_tag(&context,
    MBEDTLS_GCM_ENCRYPT,
    inputLen,                    //需要加密数据的长度
    IV,                          //设备侧生成 12 位随机数,拼包时放在 body 的前 12Byte
    GCM_IV_LEN,                  //IV 长度 12Byte
    productId,                   //productId，字符串
    productIdLen,                //productId 长度，值为 4
    input,                       //需要加密的数据
    output,                      //加密后的数据
    GCM_TAG_LEN,                 //tag 长度 16Byte
    tag);                        //tag，拼接在加密数据后面
memcpy_s(output + inputLen, GCM_TAG_LEN, tag, sizeof(tag))  //将 tag 添加到数据尾部
```

样例数据：

```
需要加密数据：{"errCodeDemo":0}
IV: 0x62,0x30,0xfc,0x88,0x79,0x87,0x4d,0xde,0x59,0x53,0xb9,0xdc
sessionKey:
0x4b,0x2f,0x68,0x3b,0xa3,0x6e,0x8e,0xfa,0x4d,0xb2,0x0e,0xf2,0x5b,0xc2,0x9e,0x4b
```



```
输出加密数据：
0x13,0x08,0x72,0x7d,0xed,0x69,0x84,0x9c,0x7c,0x8f,0x6a,0xd8,0x2b,0xff,0x68,0x65,0x1
5
输出 tag：
0xdf,0xd0,0x60,0x9b,0x6e,0x7d,0xbe,0x14,0x18,0x5f,0xb7,0xd1,0x8d,0x38,0x13,0x15
拼接成的 Encrydata：
0x13,0x08,0x72,0x7d,0xed,0x69,0x84,0x9c,0x7c,0x8f,0x6a,0xd8,0x2b,0xff,0x68,0x65,0x1
5,0xdf,0xd0,0x60,0x9b,0x6e,0x7d,0xbe,0x14,0x18,0x5f,0xb7,0xd1,0x8d,0x38,0x13,0x15
```

步骤 4 打包 payload 数据包

按照如下顺序，拼接负载数据

```
Rev + Service Length + Service Name + Body Length + Body( IV+EncryData+SessionId)
```

计算 Hmac 校验值

```
const mbedtls_md_info_t *mdInfo = NULL;
mdInfo = mbedtls_md_info_from_type(MBEDTLS_MD_SHA256);
mbedtls_md_hmac(mdInfo,
    hmacKey,          //hmacKey。
    keyLen,           //hmacKey 长度，32Byte
    input,            //打包的 payload 数据
    inputLen,         //payload 数据
    output            //Hmac 校验值
);
```

样例数据：

```
Rev : 0x11
Service Length: 0x0d
Service Name: 0x63,0x75,0x73,0x74,0x6f,0x6d,0x53,0x65,0x63,0x44,0x61,0x74,0x61
Body Length: 0x4d,0x00
IV: 0x62,0x30,0xfc,0x88,0x79,0x87,0x4d,0xde,0x59,0x53,0xb9,0xdc
EncryData:
0x13,0x08,0x72,0x7d,0xed,0x69,0x84,0x9c,0x7c,0x8f,0x6a,0xd8,0x2b,0xff,0x68,0x65,0x1
5,0xdf,0xd0,0x60,0x9b,0x6e,0x7d,0xbe,0x14,0x18,0x5f,0xb7,0xd1,0x8d,0x38,0x13,0x15
SessionId:
0x30,0x30,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x31,0x32,0x33,0x3
4,0x35,0x36,0x37,0x38,0x93,0x3,0x13,0x23,0x33,0x43,0x56,0x37,0x38,0x39,0xef
打包的负载数据：
0x11,0x0d,0x63,0x75,0x73,0x74,0x6f,0x6d,0x53,0x65,0x63,0x44,0x61,0x74,0x61,0x4d,0x0
0,0x62,0x30,0xfc,0x88,0x79,0x87,0x4d,0xde,0x59,0x53,0xb9,0xdc,0x13,0x8,0x72,0x7d,0x
ed,0x69,0x84,0x9c,0x7c,0x8f,0x6a,0xd8,0x2b,0xff,0x68,0x65,0x15,0xdf,0xd0,0x60,0x9b,
0x6e,0x7d,0xbe,0x14,0x18,0x5f,0xb7,0xd1,0x8d,0x38,0x13,0x15,0x30,0x30,0x30,0x31,0x3
2,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0
x93,0x3,0x13,0x23,0x33,0x43,0x56,0x37,0x38,0x39,0xef
HmacKey:
0x13,0xf4,0xdb,0x60,0xff,0xba,0x91,0xbd,0x3e,0x36,0xe2,0x1,0x1c,0x94,0xaa,0xf4,0x54
,0xbf,0xb5,0x65,0xf6,0x7,0xd8,0x47,0xe,0x7c,0x50,0x81,0x52,0x2a,0xf0,0x82
输出 Hmac：
0x5e,0x74,0x58,0x64,0x5d,0xa0,0x1,0xcd,0x78,0x6e,0xdd,0xb8,0x3,0x70,0xc1,0xdd,0x17,
0x24,0xc8,0x41,0xb5,0xed,0xca,0x5d,0xc0,0xa8,0x7f,0x9b,0x51,0x62,0xb2,0x28
payload 数据：
0x11,0x0d,0x63,0x75,0x73,0x74,0x6f,0x6d,0x53,0x65,0x63,0x44,0x61,0x74,0x61,0x4d,0x0
0,0x62,0x30,0xfc,0x88,0x79,0x87,0x4d,0xde,0x59,0x53,0xb9,0xdc,0x13,0x8,0x72,0x7d,0x
ed,0x69,0x84,0x9c,0x7c,0x8f,0x6a,0xd8,0x2b,0xff,0x68,0x65,0x15,0xdf,0xd0,0x60,0x9b,
```



```
0x6e,0x7d,0xbe,0x14,0x18,0x5f,0xb7,0xd1,0x8d,0x38,0x13,0x15,0x30,0x30,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x93,0x3,0x13,0x23,0x33,0x43,0x56,0x37,0x38,0x39,0xef,0x5e,0x74,0x58,0x64,0x5d,0xa0,0x1,0xcd,0x78,0x6e,0xdd,0xb8,0x3,0x70,0xc1,0xdd,0x17,0x24,0xc8,0x41,0xb5,0xed,0xca,0x5d,0xc0,0xa8,0x7f,0x9b,0x51,0x62,0xb2,0x28
```

步骤 5 打包蓝牙数据

将 header 和 payload 拼接在一起就是最终发送应用侧的数据包，第一个字段：响应数据为 0x01，主动上报数据为 0x02。

最终数据参考：

```
0x01,0x09,0x01,0x00,0x00,0x03,0x00,0x11,0x0d,0x63,0x75,0x73,0x74,0x6f,0x6d,0x53,0x65,0x63,0x44,0x61,0x74,0x61,0x4d,0x00,0x62,0x30,0xfc,0x88,0x79,0x87,0x4d,0xde,0x59,0x53,0xb9,0xdc,0x13,0x8,0x72,0x7d,0xed,0x69,0x84,0x9c,0x7c,0x8f,0x6a,0xd8,0x2b,0xff,0x68,0x65,0x15,0xdf,0xd0,0x60,0x9b,0x6e,0x7d,0xbe,0x14,0x18,0x5f,0xb7,0xd1,0x8d,0x38,0x13,0x15,0x30,0x30,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x93,0x3,0x13,0x23,0x33,0x43,0x56,0x37,0x38,0x39,0xef,0x5e,0x74,0x58,0x64,0x5d,0xa0,0x1,0xcd,0x78,0x6e,0xdd,0xb8,0x3,0x70,0xc1,0xdd,0x17,0x24,0xc8,0x41,0xb5,0xed,0xca,0x5d,0xc0,0xa8,0x7f,0x9b,0x51,0x62,0xb2,0x28
```

----结束

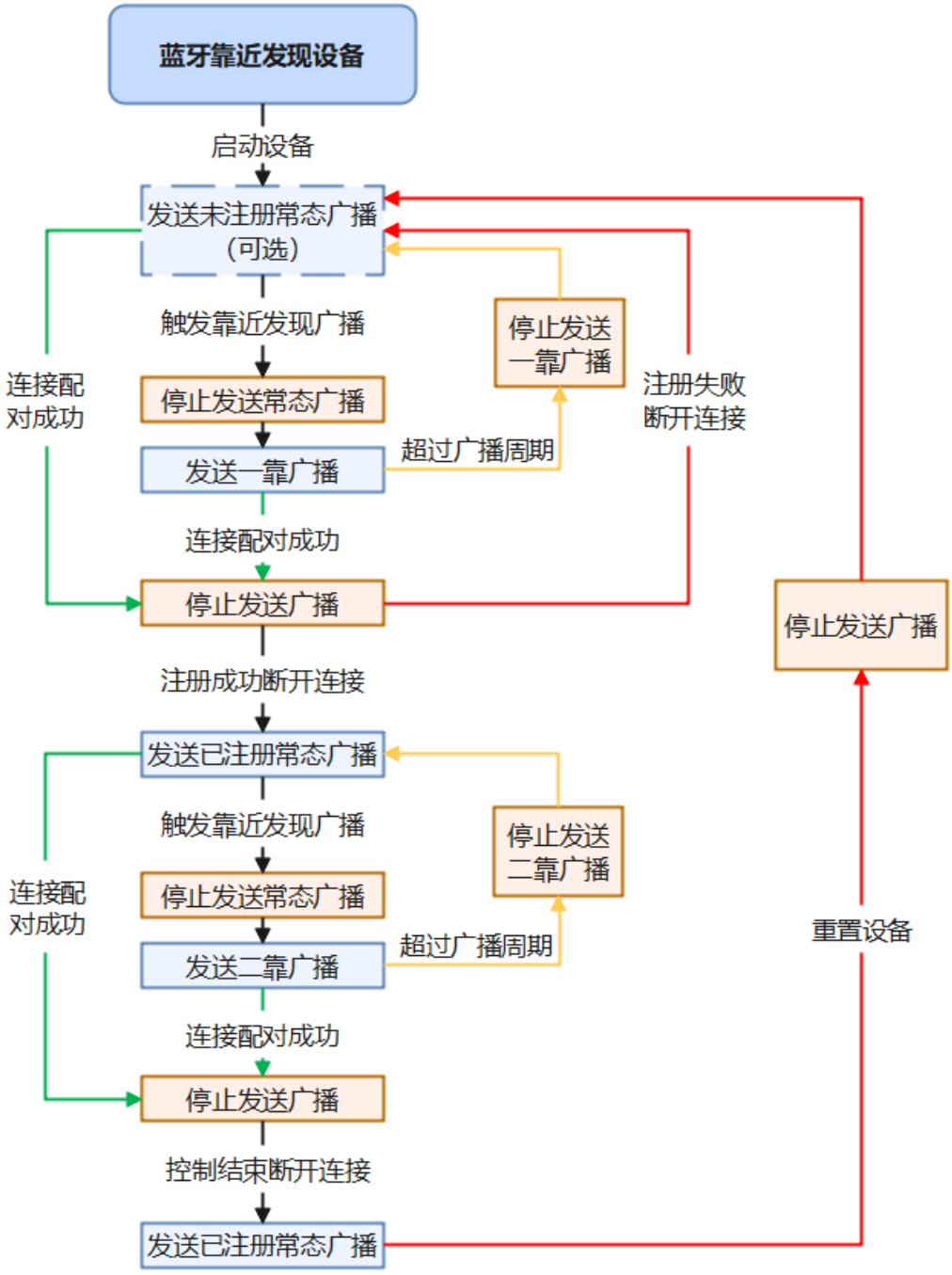
3.6 设备广播实例

3.6.1 蓝牙靠近发现设备广播实例

蓝牙靠近发现设备广播切换

蓝牙靠近发现的设备需要发送 4 种广播：一靠广播、二靠广播、未注册常态广播和已注册常态广播。为了用户使用设备时有良好的体验，我们需要根据业务实际发送响应的广播，既能够让用户快速体验到服务，又不会收到骚扰，广播切换实例如下：

图3-7 广播切换实例（蓝牙靠近发现设备）



- 发送未注册常态广播

  设备在未注册时，启动时默认发送未注册常态广播，使用智慧生活 App 可以扫描添加设备。

- 发送一靠广播

  设备在未注册时，手动触发（开机/上电/双击/使用等）的靠近发现广播。手机亮屏靠近时，可以触发拉起 FA，进入设备注册流程；使用智慧生活 App 也可以扫描添加设备。

- 发送已注册常态广播

设备在已注册时，非靠近发现场景下，默认发送的常态广播。用于打开设备卡片时连接设备，或者设备连接过程中蓝牙连接断开时重新连接。

- 发送二靠广播

  设备在已注册时，手动触发（开机/上电/双击/使用等）的靠近发现广播。有注册关系的用户手机亮屏靠近时，可以触发拉起 FA，进入设备控制流程；其他用户手机靠近时无感知。使用智慧生活 App 也可以扫描添加设备。

- 停止发送一靠/二靠广播

  靠近发现广播每次触发后发送 1 分钟，超过 1 分钟后切换成常态广播。

- 重置设备

  重置设备时，需要将广播切换成未注册常态广播和一靠广播，设备注册信息不需要清除，下次注册时直接覆盖就行。

- 停止发送广播

  连接配对成功后，需要停止发送广播，并且连接断开后，需要发送常态广播。

蓝牙靠近发现设备广播实例

- 一靠广播

  一靠广播需要手动触发发送，ProtocolId 0x17 字段置为 0x15，具体字段参考蓝牙靠近发现广播规范，参考代码如下：

```
//AAA，由产品品牌名与设备名称组成，厂商自定义，1~10 位。

unsignedcharmyadvData_0001[30] = {
0x2, 0x1, 0x6, 0x1A, 0x16, 0xEE, 0xFD, 0x01, 0x01, 0x0D, 0x04, 0x00, 0x11, 0xF8,
0x12,
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
0xFF, 0x17, 0x01, 0x15, 0x14, 0x02, demoDevInfo->sn[10], demoDevInfo->sn[11],
0x91, 0x01, 0x01
    };
unsignedcharmyrspData_0001[] = {
0x13, 0x9,  'H', 'i', '-', 'A', 'A', 'A', '-', 0x31,
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
demoDevInfo->subProductId[0], demoDevInfo->subProductId[1],
demoDevInfo->sn[8], demoDevInfo->sn[9], demoDevInfo->sn[10],demoDevInfo->sn[11]
    };
```

- 二靠广播

  二靠广播需要手动触发发送，ProtocolId 0x17 字段置为 0x00/0x01，具体字段参考蓝牙靠近发现广播规范，参考代码如下：

```
//AAA，由产品品牌名与设备名称组成，厂商自定义，1~10 位。

unsignedcharmyadvData_0011[30] = {
0x2, 0x1, 0x6, 0x1A, 0x16, 0xEE, 0xFD, 0x01, 0x01, 0x0D, 0x04, 0x00, 0x11, 0xF8,
0x12,
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
0xFF, 0x17, 0x01, 0x01, 0x14, 0x02, demoDevInfo->sn[10], demoDevInfo->sn[11],
0x91, 0x01, 0x01
    };
```



```
unsignedcharmyrspData_0011[] = {
0x14, 0x9,  'H', 'I', '-', 'A', 'A', 'A', '-', 0x31,
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
demoDevInfo->sn[8], demoDevInfo->sn[9], demoDevInfo->sn[10],demoDevInfo->sn[11],
MM,PP[0],PP[1]
    };
```

- 未注册常态广播

  当设备未注册时，智慧生活 App 可以扫描广播添加设备，需要发送未注册常态广播。未注册常态广播的蓝牙广播响应数据(myRspData)具体字段参考表 3-3，广播内容(myAdvData)厂家自定义即可。参考代码如下：

```
//AAA，由产品品牌名与设备名称组成，伙伴自定义，1~10 位。
unsignedcharmyadvData_0010[] = {
0x2, 0x1, 0x6
    };

unsignedcharmyrspData_0001[] = {
0x13, 0x9,  'H', 'i', '-', 'A', 'A', 'A', '-', 0x31,
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
demoDevInfo->subProductId[0], demoDevInfo->subProductId[1],
demoDevInfo->sn[8], demoDevInfo->sn[9], demoDevInfo->sn[10],demoDevInfo->sn[11]
    };
```

- 已注册常态广播

  当蓝牙断开连接时，为了保持设备能够被重连，或者通过 H5 连接设备，需要发送常态广播。已注册常态广播的蓝牙广播响应数据(myRspData)具体字段参考表 3-3，广播内容(myAdvData)厂家自定义即可。参考代码如下：

```
//AAA，由产品品牌名与设备名称组成，伙伴自定义，1~10 位。
unsignedcharmyadvData_0010[] = {
0x2, 0x1, 0x6
    };
unsignedcharmyrspData_0010[] = {
0x14, 0x9,  'H', 'I', '-', 'A', 'A', 'A', '-', 0x31,
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
demoDevInfo->sn[8], demoDevInfo->sn[9], demoDevInfo->sn[10],demoDevInfo->sn[11],
MM,PP[0],PP[1]
    };
```

3.6.2 蓝牙碰一碰设备广播实例

蓝牙碰一碰切换设备状态

蓝牙碰一碰的设备需要发送 2 种广播：一碰广播和二碰广播。

- 当设备未注册时，发送一碰广播。
- 当设备已注册时，发送二碰广播。

设备重置时，需要将广播切换成一碰广播，设备注册信息不需要清除，下次注册时直接覆盖即可。

蓝牙碰一碰设备广播实例

- 一碰广播

  设备未注册时发送一碰广播，ProtocolId 0x17 字段置为 0x15，具体字段参考蓝牙碰一碰广播规范，参考代码如下：

```
//AAA，由产品品牌名与设备名称组成，厂商自定义，1~10 位。
unsigned char myadvData_1000[29] = {
0x2, 0x1, 0x6, 0x19, 0x16, 0xEE, 0xFD, 0x01, 0x06, 0x0, 0x17, 0x0, 0xA, 0x17,
0x01, 0x15, 0x14, 0x02,
demoDevInfo->sn[10], demoDevInfo->sn[11], 0x91, 0x01, 0x56, demoDevInfo-
>productId[0],demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3], 0x0, 0xF8
    };
unsigned char myrspData_1000[] = {
0x13, 0x9, 'H', 'i', '-', 'A', 'A', 'A', '-', 0x31,
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
demoDevInfo->subProductId[0], demoDevInfo->subProductId[1],
demoDevInfo->sn[8], demoDevInfo->sn[9], demoDevInfo->sn[10], demoDevInfo->sn[11]
    };
```

- 二碰广播

  设备已注册时发送二碰广播，ProtocolId 0x17 字段置为 0x00/0x01，具体字段参考蓝牙碰一碰广播规范，参考代码如下：

```
//AAA，由产品品牌名与设备名称组成，厂商自定义，1~10 位。
unsignedcharmyadvData_1010[29] = {
0x2, 0x1, 0x6, 0x19, 0x16, 0xEE, 0xFD, 0x01, 0x06, 0x0, 0x17, 0x0, 0xA, 0x17,
0x01, 0x01, 0x14, 0x02,
demoDevInfo->sn[10], demoDevInfo->sn[11], 0x91, 0x01, 0x56, demoDevInfo-
>productId[0],
demoDevInfo->productId[1], demoDevInfo->productId[2], demoDevInfo->productId[3],
0x0, 0xF8
    };
unsignedcharmyrspData_1010[] = {
0x14, 0x9, 'H', 'I', '-', 'A', 'A', 'A', '-', 0x31,
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
demoDevInfo->sn[8], demoDevInfo->sn[9], demoDevInfo->sn[10], demoDevInfo-
>sn[11],
MM,PP[0],PP[1]
    };
```

3.6.3 NFC 碰一碰设备广播实例

NFC 碰一碰切换设备状态

NFC 碰一碰的设备需要发送 2 种广播：未注册常态广播和已注册常态广播。

- 当设备未注册时，发送未注册常态广播。
- 当设备已注册时，发送已注册常态广播。

设备重置时，需要将广播切换成未注册常态广播，设备注册信息不需要清除，下次注册时直接覆盖即可。

NFC 碰一碰设备广播实例

- 未注册常态广播

  当设备未注册时，智慧生活 App 可以扫描广播添加设备，需要发送未注册常态广播。未注册常态广播的蓝牙广播响应数据(myRspData)具体字段参考表 3-3，广播内容(myAdvData)厂家自定义即可。参考代码如下：

```
//AAA，由产品品牌名与设备名称组成，伙伴自定义，1~10 位。
unsignedcharmyadvData_0010[] = {
0x2, 0x1, 0x6
    };

unsignedcharmyrspData_0001[] = {
0x13, 0x9,  'H', 'i', '-', 'A', 'A', 'A', '-', 0x31,
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
demoDevInfo->subProductId[0], demoDevInfo->subProductId[1],
demoDevInfo->sn[8], demoDevInfo->sn[9], demoDevInfo->sn[10],demoDevInfo->sn[11]
    };
```

- 已注册常态广播

  当蓝牙断开连接时，为了保持设备能够被重连，或者通过 H5 连接设备，需要发送常态广播。已注册常态广播的蓝牙广播响应数据(myRspData)具体字段参考表 3-3，广播内容(myAdvData)厂家自定义即可。参考代码如下：

```
//AAA，由产品品牌名与设备名称组成，伙伴自定义，1~10 位。
unsignedcharmyadvData_0010[] = {
0x2, 0x1, 0x6
    };
unsignedcharmyrspData_0010[] = {
0x14, 0x9,  'H', 'I', '-', 'A', 'A', 'A', '-', 0x31,
demoDevInfo->productId[0], demoDevInfo->productId[1], demoDevInfo->productId[2],
demoDevInfo->productId[3],
demoDevInfo->sn[8], demoDevInfo->sn[9], demoDevInfo->sn[10],demoDevInfo->sn[11],
MM,PP[0],PP[1]
    };
```


4 功能验证

4.1 测试蓝牙设备发现

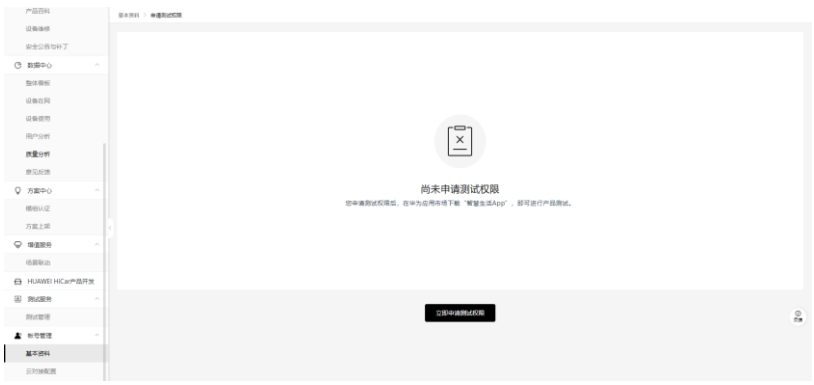
下载 HW Air Link 工具，使用工具查看设备发送的广播，对比 BLE 设备广播规范，检查广播是否符合协议规范。

4.2 测试设备注册和设备控制

4.2.1 配置测试环境

- 方式一：针对调测手机登录的华为帐号申请测试权限。
  a. 使用调测手机登录的华为帐号登录 Device Partner 平台，进入管理中心。
  b. 选择“帐号管理 > 基本资料”，单击右上角的“申请测试权限”。
  c. 单击“立刻申请测试权限”申请测试权限。

图4-1 申请测试权限



d. 验证测试权限是否申请成功。

在测试手机上登录华为帐号，打开智慧生活 App，进入“我的 > 连接三方平台”。如果三方平台名称后面出现“认证”字样，表明申请成功。

图4-2 验证测试权限是否申请成功

⚠ 注意

由于智慧生活基础服务没有 debug 版本，在进行 FA 测试时请使用方式一，每个测试帐号都需要登录申请测试权限。

- 方式二：下载 debug 版本智慧生活 App。

  a. 登录伙伴网站，在自测试环节下载 debug 版本智慧生活 App。

  图4-3 智慧生活 App 下载方式

  b. 设置智慧生活 App 测试环境。

  打开智慧生活 App 后，在 “我的 > 设置 > 关于 > 环境设置”，选择“认证沙箱”环境。

4.2.2 测试设备注册与设备控制功能

步骤 1 打开智慧生活 App，点击右上角“+”，选择“添加设备”。

图4-4 智慧生活 App 首界面

步骤 2 智慧生活 App 会扫描附近所有处于待连接的设备。选择需要连接的设备，点击“连接”。

步骤 3 连接成功后，设置设备位置信息。如卧室、阳台等。

步骤 4 打开设备卡片，进入设备控制界面。

设备控制界面为交互设计环节部署的 H5 界面，展示了设备状态和功能控制服务等。

说明

调测阶段，因为产品还没有提交认证，所以会有警告窗口，点击“继续”即可。

步骤 5 点击设备控制按钮，如开关等，设备侧会收到相关指令。

----结束

5 参考

mbedtls 加密库下载地址：https://github.com/ARMmbed/mbedtls/releases/tag/v2.16.10

- 华为智能硬件合作伙伴 > 原子化服务开发
- 华为智能硬件合作伙伴 > 常见问题

