

设备集成开发指导

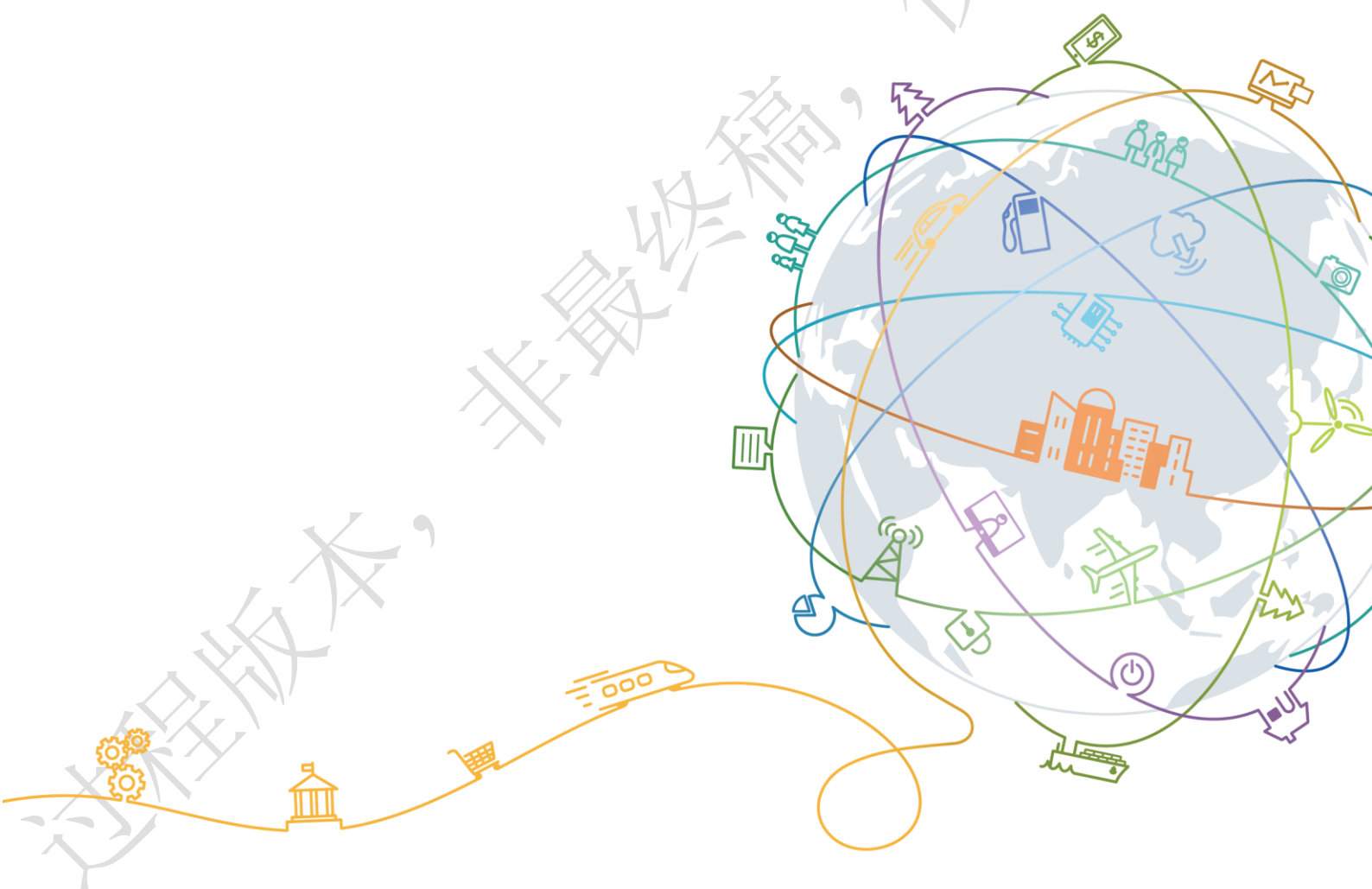
# 设备集成开发指导

文档版本

01

发布日期

2021-09-11



**版权所有 © 华为终端有限公司 2021。 保留一切权利。**

本材料所载内容受著作权法的保护，著作权由华为公司或其许可人拥有，但注明引用其他方的内容除外。未经华为公司或其许可人事先书面许可，任何人不得将本材料中的任何内容以任何方式进行复制、经销、翻印、播放、以超级链路连接或传送、存储于信息检索系统或者其他任何商业目的的使用。

## 商标声明



、华为，以上为华为公司的商标（非详尽清单），未经华为公司书面事先明示许可，任何第三方不得以任何形式使用。

## 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

<b>1 概述</b> .....	<b>1</b>
<b>2 准备工作</b> .....	<b>3</b>
<b>3 固件开发</b> .....	<b>5</b>
3.1 简介 .....	5
3.2 配置产品信息 .....	9
3.2.1 配置 parameter_common.c 中的参数信息 .....	9
3.2.2 配置 hal_sys_param.c 中的参数信息 .....	10
3.2.3 配置 hal_token.c 中的参数信息 .....	12
3.2.3.1 配置产品 ID 信息 .....	12
3.2.3.2 配置 AcKey 信息 .....	12
3.3 开发设备功能 .....	13
3.3.1 配置 hilink_device.h 中产品信息 .....	13
3.3.2 配置 hilink_device_sdk.c 中的服务信息 .....	14
3.3.3 实现 hilink_device_sdk.c 中的设备控制函数 .....	15
3.4 编译固件 .....	16
3.5 烧录固件 .....	16
<b>4 功能验证</b> .....	<b>17</b>
4.1 预置激活码 .....	17
4.2 测试配网和设备控制 .....	18
4.2.1 设置智慧生活 APP 测试环境 .....	18
4.2.2 测试设备配网与设备控制功能 .....	18
4.2.3 添加设备失败问题分析 .....	20

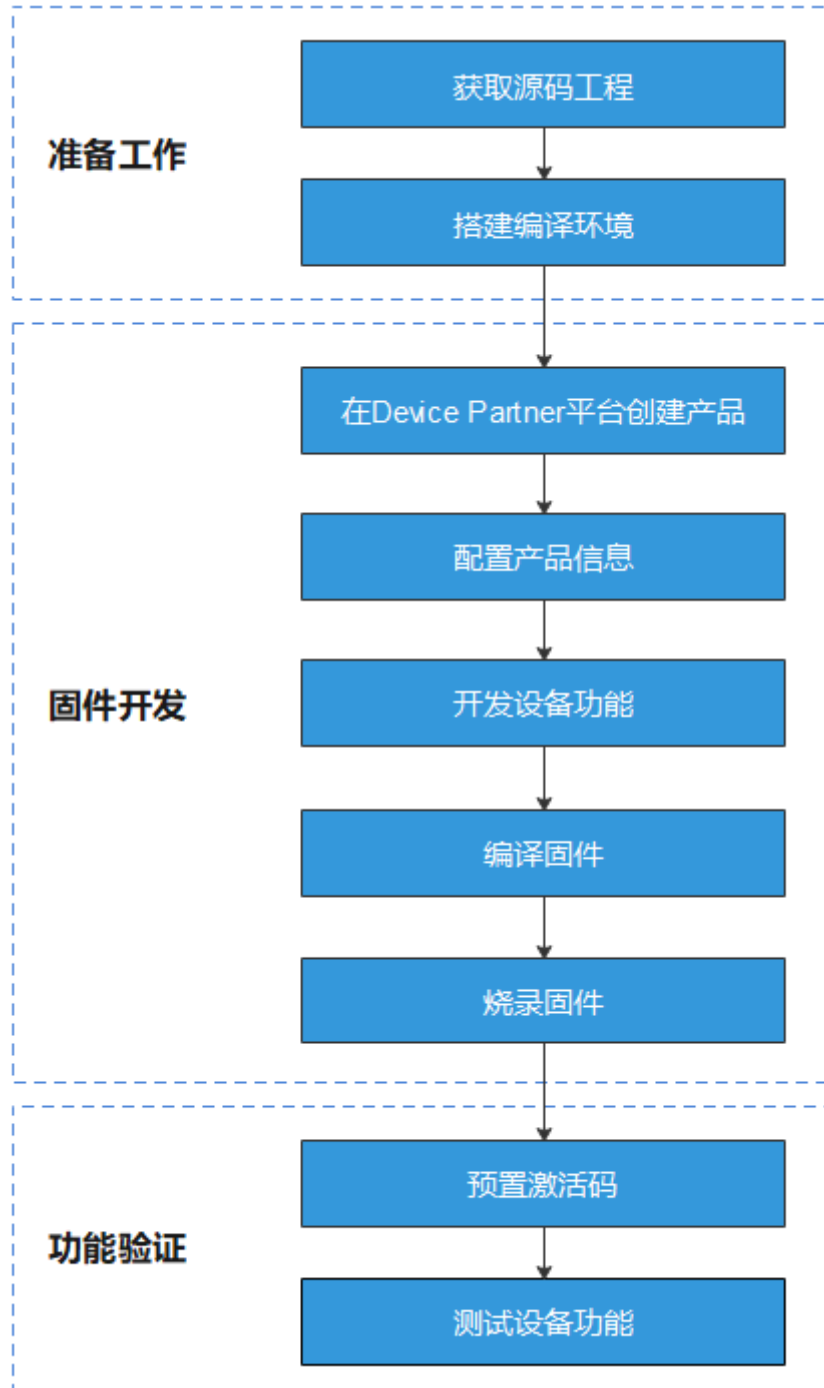
# 1 概述

## 简介

本文档为 HarmonyOS Connect 生态产品合作伙伴提供集成指导，旨在帮助伙伴快速熟悉开发流程，完成产品信息配置、配网和设备控制等功能开发，并基于智慧生活 APP 进行配网测试和设备控制测试。

## 开发流程

图1-1 设备集成开发流程



# 2 准备工作

步骤 1 联系模组商获取以下工具和信息。

表2-1 工具与信息

资料名称	说明
HarmonyOS Connect 服务包源码工程	用于开发固件。
编译工具链和使用指导	用于搭建编译环境。不同芯片采用的编译工具链不同，请联系模组商获取。
烧录工具	用于烧录固件。不同芯片采用的驱动和烧录工具不同，请联系模组商获取。
串口驱动	用于固件烧录过程中 PC 和设备的通信。不同芯片采用的串口驱动不同，请联系模组商获取。
SDK 集成路径	用于配置 HiLink SDK 和 Kit Framework 所需产品信息。
激活码预置方式	不同模组商采用的激活码预置方式不同（AT 指令写入或者烧录工具烧写），请联系模组商确认。

步骤 2 搭建编译环境。

步骤 3 安装开发板编译环境。

## 说明

根据开发板实际型号，参考对应的章节进行配置。

步骤 4 在 Device Partner 平台创建产品。

步骤 5 导出产品信息，用于固件开发中的信息配置。

在 Device Partner 平台的“产品开发”页面，单击“导出”可以获得产品基础信息，如图 2-1 所示。

图2-1 导出产品信息



----结束

# 3 固件开发

- 3.1 简介
- 3.2 配置产品信息
- 3.3 开发设备功能
- 3.4 编译固件
- 3.5 烧录固件

## 3.1 简介

### 工程配置项说明

为了保证设备配网、设备控制功能的实现，需要对 HarmonyOS Connect 服务包中的文件进行配置。具体文件及配置项的说明参见表 3-1。

表3-1 HarmonyOS Connect 服务包配置项说明

HarmonyOS Connect 组件名称	文件名称	配置项	用途
Kit Framework	parameter_common.c	操作系统名称、操作系统版本号以及软件版本号	用于产品认证过程中的兼容性测试。
	hal_sys_param.c	产品类型、厂商英文简称、品牌英文名、产品型号等	用于产品认证过程中的兼容性测试，以及设备添加过程中的设备信息校验。
	hal_token.c	产品 ID、AcKey 信息	用于设备添加过程中的设备激活码验证。
HiLink SDK	hilink_device.h	产品 ID、设备类型 ID、厂商 ID、设备类	用于设备配网过程中的 Wi-Fi 信息获取与设备注册。



HarmonyOS Connect 组件名称	文件名称	配置项	用途
		型名、厂商名称	
	hilink_device_sdk.c	服务信息	用于设备上报当前支持的服务列表。

## 附加参考：API 接口说明

表 3-2 提供了常用 API 接口的功能介绍，供开发者在固件开发时进行参考。

API 接口定义参见“base\startup\syspara\_lite\interfaces\kits\parameter.h”。

表3-2 API 接口说明

API 接口	API 返回值的名称	Device Partner 平台的对应字段	举例	说明和要求
GetDeviceType() / GetProductType()	设备类型	-	linkiot ipcamera	最大 32 字符。 具体见 <a href="#">设备类型</a> 定义章节。物联网设备取固定值 linkiot。
GetManufacture()	公司英文名简称	企业英文简称	HUAWEI	最大 32 字符。 Kit Framework 认证关键项，和单品激活码强关联。
GetBrand()	品牌英文名称	品牌英文名	HUAWEI	最大 32 字符。 Kit Framework 认证关键项，和单品激活码强关联。
GetMarketName()	外部产品系列名称	产品名称（传播名）	Mate 30	最大 32 字符。 用户可见，Kit Framework 认证关键项。
GetProductSeries()	产品系列英文名称	产品系列	TAS	最大 32 字符。 用以对不同产品类型进行分类管理。Kit Framework 认证关键项。
GetProductModel()	型号	产品型号	TAS-AL00	最大 32 字符。 设备关键信息之一，用以标识设备的类型，用户可见，通常打印在设备铭牌上，用以区分不同产品。该值也是进行

API 接口	API 返回值的名称	Device Partner 平台的对应字段	举例	说明和要求
				HarmonyOS 认证所需的关键数据，需要采用英文描述。 Kit Framework 认证关键项，和单品激活码强关联。
GetSoftwareModel()	内部软件子型号	-	TAS-AL00	最大 32 字符。 厂商软件型号，厂商自定义，多硬件共软件时区分软件分支。
GetHardwareModel()	硬件版本号	硬件设备版本号	TASAL00 CVN1	最大 32 字符。 厂商软件型号，厂商自定义。
GetHardwareProfile()	硬件支持配置	-	{RAM:352K,ROM:2M,WIFI:true}	最大 1000 字符。使用 json 格式字符串表示。根据芯片确定，例如 [RAM:352KB,ROM:288KB,wlan:true]。
GetSerial()	设备序列号	-	随设备差异	最大 64 字符。 SN 非配置，由厂家定义，并保证编号唯一。 Kit Framework 认证关键项，认证后，云端会记录此信息。
GetOsFullName() / GetOsName()	操作系统及版本号	操作系统版本号	HarmonyOS-2.0.1.27	最大 64 字符。 名称与版本号之间以 "-" 分隔。
GetDisplayVersion()	用户可见的软件版本号	软件版本号	1.0.0.6	最多 64 字符。 厂商定义填写内容。注意是整个系统的软件版本号而非 HarmonyOS 版本号。
GetBootloaderVersion()	Bootloader 版本号	-	u-boot-v2019.07	最多 64 字符。 实际情况填写，例如 XXX-bootloader-v2015.01.03。
GetSecurityPatchTag()	安全补丁标签	安全补丁级别	2021/1/1	最多 64 字符。 标识当前 OS 的安全补丁级别。按实际情况填写，例如 2020-12-01。
GetAbiList()	Native 接	-	1. riscv-	最多 64 字符。

API 接口	API 返回值的名称	Device Partner 平台的对应字段	举例	说明和要求
)	口列表		liteos 2. arma7_hard_neon-vfpv4-linux 3. arma7_soft-linux 4. arma7_softfp_neon-vfpv4-linux 5. arma7_hard_neon-vfpv4-liteos 6. arma7_soft-liteos 7. arma7_softfp_neon-vfpv4-liteos	用逗号隔开，仅有生态且生态中包含 native 应用的系统使用。 按实际情况填写，例如 riscv-liteos 这些字段对 kit-framework 没有影响，但是做认证的时候可能会检查，看认证团队的要求。
GetSdkApiVersion()	系统软件 API version	系统软件 API Level 系统 API Level	3	设备当前版本 API 级别，一般是整数，仅有生态的系统使用。 版本预置值，不需要修改。
GetFirstApiLevel()	设备首版本的系统软件 API level	-	3	一般是整数，仅有生态的系统使用。 版本预置值，不需要修改。
GetIncrementalVersion()	差异版本号	-	1.0.0.6	在设备型号确定的情况下，即在“设备类型+”公司+”品牌+”产品系列+”操作系统及版本号+”型号+”内部硬件子型号+”内部软件子型号”均相同的情况下，此值必须可以唯一标识软件版本。
GetVersionId()	版本 Id	版本 Id	-	最大 127 字符。

API 接口	API 返回值的名称	Device Partner 平台的对应字段	举例	说明和要求
				由多个字段拼接而成： \$(设备类型) + '/' + \$(公司英文名称简称) + '/' + \$(品牌英文名称) + '/' + \$(产品系列英文名称) + '/' + \$(操作系统及版本号) + '/' + \$(型号) + '/' + \$(内部软件子型号) + '/' + \$(系统软件 API level) + '/' + \$(差异版本号) + '/' + \$(构建类型) 在所有厂家的所有设备范围中，可以唯一标识版本。
GetBuildType()	构建类型	-	release:nolog	最多 32 字符。 同一基线代码的不同构建类型，比如 debug/release、log/nolog 可以用多个标识，分号分隔。
GetBuildUser()	构建 user	-	-	最多 32 字符。
GetBuildHost()	构建 host	-	-	最多 32 字符。
GetBuildTime()	构建时间	-	-	最多 32 字符。 Epoch Time，自 1970 年至今的秒数；例如 1294902266。
GetBuildRootHash()	版本 Hash	版本 Hash	-	默认为空串即可。

## 3.2 配置产品信息

### 3.2.1 配置 parameter\_common.c 中的参数信息

参考表 3-3，配置 parameter\_common.c 文件中的产品信息。

```
static char g_roBuildOs[] = {"OpenHarmony-release 1.1.0"}; // 系统名称和版本号，例如：OpenHarmony-release 1.1.0
static char g_roBuildVerShow[] = {"manufacture-software-version"}; // 用户可见软件版本号
```

表3-3 配置项说明

配置项	说明	示例
g_roBuildOs	系统名称和版本号，使用“-”连接。获取方式如下： 1. 在 Device Partner 平台的“产品开发”页面，选择对应产品。 2. 单击右上角的“详情”，在“产品信息”页签下，可以查看操作系统名称和版本号。	OpenHarmony-release 1.1.0
g_roBuildVerShow	用户可见软件版本号，厂商自定义。	-

### 3.2.2 配置 hal\_sys\_param.c 中的参数信息

参考表 3-4 配置产品详细信息，对 Kit Framework 认证结果有直接影响。企业英文名、品牌英文名、产品型号和认证过程强相关，信息不匹配会导致认证结果失败。

```
static const char OHOS_PRODUCT_TYPE[] = {"linkiot"}; // 固定值
static const char OHOS_MANUFACTURE[] = {"Test"}; // 企业英文名简称
static const char OHOS_BRAND[] = {"HiTest"}; // 品牌英文名
static const char OHOS_MARKET_NAME[] = {"SmartLight"}; // 产品名称（传播名），包含汉字时
// 建议使用 Unicode，避免乱码问题
static const char OHOS_PRODUCT_SERIES[] = {"Light"}; // 产品系列
static const char OHOS_PRODUCT_MODEL[] = {"HiTest0728"}; // 产品型号
static const char OHOS_SOFTWARE_MODEL[] = {"1.0.0"}; // 软件版本
static const char OHOS_HARDWARE_MODEL[] = {"1.0.0"}; // 硬件型号
static const char OHOS_HARDWARE_PROFILE[] = {"aout:true,display:true"}; // 系统能力，
// 参考产品信息介绍
static const char OHOS_BOOTLOADER_VERSION[] = {"bootloader"};
static const char OHOS_SECURITY_PATCH_TAG[] = {"2020-09-01"};
static const char OHOS_ABI_LIST[] = {"riscv-liteos"};
static const char OHOS_SERIAL[] = {"1234567890"}; // 厂商自定义，保持唯一
```

如果需要动态获取信息，比如从硬件存储或者 mcu 获取信息，可以修改 HalGet 开头的对应的方法。

表3-4 配置项说明

配置项	说明	示例
OHOS_PRODUCT_TYPE	设备类型，取固定值“linkiot”。	linkiot
OHOS_MANUFACTURE	公司英文简称，申请激活码后不可修改。获取方式如下： 1. 在 Device Partner 平台的“产品开发”页面，选择对应产品。 2. 单击右上角的“详情”，在	HUAWEI

配置项	说明	示例
	“产品信息”页签下，可以查看公司英文名。	
OHOS_BRAND	品牌英文名，申请激活码后不可修改。获取方式如下： 1. 在 Device Partner 平台的“产品开发”页面，选择对应产品。 2. 单击右上角的“详情”，在“产品信息”页签下，可以查看品牌英文名。	HUAWEI
OHOS_MARKET_NAME	外部产品系列名称，用户可见。获取方式如下： 1. 在 Device Partner 平台的“产品开发”页面，选择对应产品。 2. 单击右上角的“详情”，在“产品信息”页签下，可以查看品类。	Mate 30
OHOS_PRODUCT_SERIES	产品系列英文名称。获取方式如下： 1. 在 Device Partner 平台的“产品开发”页面，选择对应产品。 2. 单击右上角的“详情”，在“产品信息”页签下，可以查看产品系列。	TAS
OHOS_PRODUCT_MODEL	型号。获取方式如下： 1. 在 Device Partner 平台的“产品开发”页面，选择对应产品。 2. 单击右上角的“详情”，在“产品信息”页签下，可以查看产品型号。	TAS-AL00
OHOS_SOFTWARE_MODEL	内部软件子型号。厂商自定义。	TAS-AL00
OHOS_HARDWARE_MODEL	硬件版本号。厂商自定义。	TASAL00CVN1
OHOS_HARDWARE_PROFILE	硬件配置。厂商自定义，根据设备实际支持功能配置。	{RAM:352K,ROM:2M:WIFI:true}
OHOS_SECURITY_PATCH_TAG	安全补丁标签。厂商自定义。	2021/1/1
OHOS_ABI_LIST	Native 接口列表。取固定值“riscv-liteos”	riscv-liteos
OHOS_SERIAL	设备序列号。厂商自定义。	-

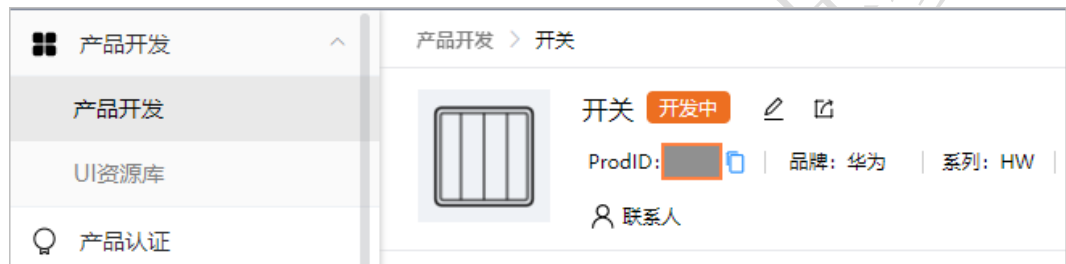
## 3.2.3 配置 hal\_token.c 中的参数信息

### 3.2.3.1 配置产品 ID 信息

步骤 1 获取产品 ID 信息。

在 Device Partner 平台的“产品开发”页面，选择对应产品。在产品开发界面，可以查看 ProdID。

图3-1 产品 ID 信息



步骤 2 修改产品 ID 信息。

在函数 OEMGetProdId 中，修改产品 ID 信息。

```
static int OEMGetProdId(char *productId, unsigned int len)
```

----结束

### 3.2.3.2 配置 AcKey 信息

步骤 1 获取 AcKey。具体方法参考步骤 5，在导出的信息可以查找到 AcKey 的取值。

步骤 2 配置 AcKey。

1. 调整十六进制文本，每个字节以 0x 开头。

例如，获取到的 AcKey 取值为

“112233445566778899AABBCCDDEEFF112233445566778899AABBCCDDEEFF112233445566778899AABBCCDDEEFF112233”，调整后的取值为：

```
0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xAA, 0xBB, 0xCC, 0xDD,
0xEE, 0xFF, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xAA, 0xBB,
0xCC, 0xDD, 0xEE, 0xFF, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99,
0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF, 0x11, 0x22, 0x33
```

2. 在函数中，修改 acKeyBuf，替换十六进制文本信息。

```
static int OEMGetAcKey(char *acKey, unsigned int len)
```

----结束

## 3.3 开发设备功能

### 3.3.1 配置 hilink\_device.h 中产品信息

配置产品基本信息，用于设备配网和设备注册。

```
#define PRODUCT_ID "9A8C" // 产品 ID，必须和产品真实信息一致
#define DEVICE_TYPE "046" // 产品类型 ID，必须和产品真实信息一致
#define MANUFACTURER "xxx" // 厂商 ID，必须和产品真实信息一致
#define DEVICE_MODEL "HiTest0728" // 产品型号，必须和产品真实信息一致
#define DEVICE_TYPE_NAME "HiLight" // 设备类型名，和“集成开发环节”ssid 信息中保持一致
#define MANUFACTURER_NAME "Huawei" // 厂商名称，和“集成开发环节”ssid 信息中保持一致
```

图3-2 SSID 配置

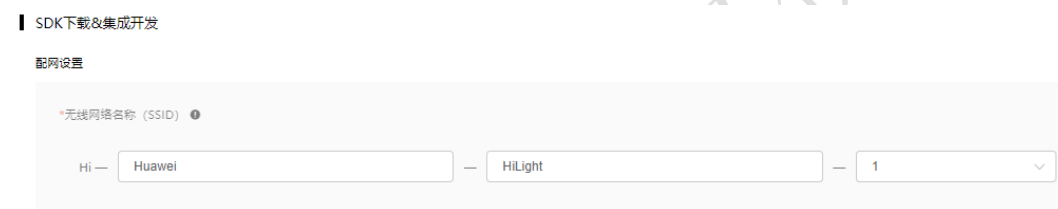


表3-5 配置项说明

配置项	说明	示例
PRODUCT_ID	产品 ID，参考准备工作步骤 5。	-
DEVICE_TYPE	产品类型 ID，参考准备工作步骤 5。	-
MANUFACTURER	厂商 ID。获取方式如下： 1. 在 Device Partner 平台的“产品开发”页面，选择对应产品。 2. 单击右上角的“详情”，在“产品信息”页签下，可以查看 ManufactureID。	-
DEVICE_MODEL	产品型号。获取方式如下： 1. 在 Device Partner 平台的“产品开发”页面，选择对应产品。 2. 单击右上角的“详情”，在“产品信息”页签下，可以查看产品	-



配置项	说明	示例
	型号。	
DEVICE_TYPE_NAME	设备类型名。参考图 3-2，需要和网站 SSID 保持一致。	Light
MANUFACTURER_NAME	产商名称。参考图 3-2，需要和网站 SSID 保持一致	HUAWEI

### 3.3.2 配置 hilink\_device\_sdk.c 中的服务信息

为了确保设备控制功能的正常使用，需要将 Profile 中定义的设备功能，配置在 hilink\_device\_sdk.c 中。Profile 中默认添加的功能（例如 ota、netinfo 等），无需在 hilink\_device\_sdk.c 文件中配置。

#### 须知

请确保信息配置正确，否则会导致设备信息校验失败，出现设备反复上线/下线的现象。

**步骤 1** 获取产品 Profile 文件。

1. 在 Device Partner 平台的“产品开发”页面，选择对应产品。
2. 在“交互设计 > 智慧生活 APP > 物模型定义”页面，单击右上角的“下载 Profile”。

**步骤 2** 在 hilink\_device\_sdk.c 文件中配置设备功能。

以门锁为例介绍如何配置设备功能，假定产品 profile 信息如表 3-6 所示，则对应的工程代码示例如下：

表3-6 产品 profile 信息（节选）

服务 sid	服务(中文)	服务类型 ServiceType
lockState	门在线/离线	state
lockMode	防护模式状态	mode

```
typedef struct {
    constchar* st;    // 服务类型
    constchar* svc id; // 服务id
} svc info t;

int gSvcNum = 2;
svc info tgSvcInfo[] =
{
```

```
{ "state", "lockState"}
{ "mode", "lockMode"} // 注意和结构体定义的顺序保持一致
};
```

----结束

### 3.3.3 实现 hilink\_device\_sdk.c 中的设备控制函数

本节介绍如何开发设备控制功能，需要使用 hilink\_device\_sdk.c 中的三个函数来实现。

表3-7 设备控制相关函数

函数名称	是否需要开发者实现	说明
hilink_put_char_state	是	云端下发控制指令后，会通 SDK 调用到这个函数。伙伴需要在对服务进行识别、分发和处理。
hilink_get_char_state	是	云端通过 HiLink SDK 获取设备状态信息，或者 HiLink SDK 主动调用接口获取设备状态信息。
hilink_report_char_state	否	HiLink SDK 报文上报接口，用于上报设备状态信息。

#### 步骤 1 配置设备服务总列表。

状态控制 **hilink\_put\_char\_state** 和状态查询 **hilink\_get\_char\_state** 都会调用 **getServiceIndex** 来查询服务，因此需要按照结构体 **SRV\_INFO\_S** 的定义，配置设备服务总列表 **services[SRV\_TOTAL\_COUNT]**。

```
// 服务属性结构定义
typedef struct{
    // 服务 ID, 服务的字符串标识
    char* id;
    //服务类型
    char* type;
    // 属性的数量
    int prop count;
    // 属性列表
    const SRV_PROPERTY S* props;
} SRV_INFO S;

// 设备服务的总数量
#define SRV_TOTAL_COUNT 2
// 设备服务总列表
const SRV_INFO_S services[SRV_TOTAL_COUNT] = {
    {"lockState", "state", SWITCH_PROP_COUNT, switch_props}
    {"lockMode", "mode", SWITCH_PROP_COUNT, switch_props}
};

// 获取指定 Service ID 对应的数组下标
```

```
int getServiceIndex(const char* svc_id) {
    int i = 0;
    for (i = 0; i < SRV_TOTAL_COUNT; i++) {
        if ((hilink_strlen(services[i].id) == hilink_strlen(svc_id)) &&
            (hilink_strncmp(svc_id, services[i].id, hilink_strlen(svc_id)) == 0)) {
            return i;
        }
    }
    return -1;
}
```

步骤 2 分发设备控制服务。

`hilink_put_char_state` 中增加 case 分支，对识别到的服务进行分发。

步骤 3 处理设备控制指令。

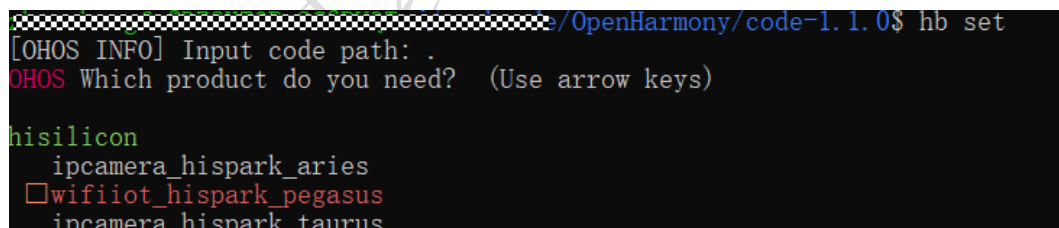
增加服务处理函数，例如：`handle_switch_cmd`、`handle_lock_state_cmd` 等。

----结束

## 3.4 编译固件

步骤 1 进入工程根目录，执行“hb set”、“.”，并选择需要构建的产品。

图3-3 构建设置示例



```
~/OpenHarmony/code-1.1.0$ hb set
[OHOS INFO] Input code path: .
OHOS Which product do you need? (Use arrow keys)

hisilicon
  ipcamera_hispark_aries
   wifiot_hispark_pegasus
  ipcamera_hispark_taurus
```

步骤 2 在工程根目录，执行“hb build -f”启动版本构建。

出现“xxxx build success”字样，则证明构建成功。

----结束

## 3.5 烧录固件

步骤 1 从模组商处获取串口驱动和烧录工具。

### 📖 说明

不同芯片使用的驱动和烧录工具均不同，建议联系模组商获取支撑。

步骤 2 按照模组商提供的指导文档安装驱动。

步骤 3 使用烧录工具烧写固件到模组上。

# 4 功能验证

## 4.1 预置激活码

### 4.2 测试配网和设备控制

## 4.1 预置激活码

### 说明

激活码是设备合法性认证的唯一标识，需要保持一机一码。认证成功后，原激活码不可再次使用。设备中预置正确的激活码是认证成功的必要条件。

#### 步骤 1 申请激活码。

参考[设备授权](#)申请激活码。申请时，激活码类型需要和 Kit Framework 版本对应。调测激活码对应 Kit Framework 为 debug 版本，商用激活码对应 Kit Framework 为 release 版本。

表4-1 激活码和 KitFramework 的关系

激活码类型	Kit Framework 版本	云端环境
调测	debug	认证环境
商用	release	商用环境

#### 步骤 2 预置激活码。

预置激活码一般有两种方法，一种是通过 AT 指令写入，另外一种是使用版本烧录工具烧写。不同模组商提供的写入方式不同，需要和模组商确认如何进行激活码预置。

### 须知

如果采用烧录工具烧写激活码，需要首先把激活码转换成可以烧录的文件。同时，需要提前与模组商确认烧录的位置和长度，以免覆盖到其他信息。

----结束

## 4.2 测试配网和设备控制

### 4.2.1 设置智慧生活 APP 测试环境

步骤 1 登录伙伴网站，在自测试环节下载 debug 版本智慧生活 APP。

图4-1 智慧生活下载方式



步骤 2 设置智慧生活 APP 测试环境。

打开智慧生活后，在“我的 -> 设置 -> 关于 -> 环境设置”，选择认证环境。环境设置请参考，4.1 预置激活码章节，步骤 1。

----结束

### 4.2.2 测试设备配网与设备控制功能

步骤 1 打开智慧生活，点击右上角“+”，选择“添加设备”，智慧生活会扫描附件所有处于待配网状态的设备。

图4-2 智慧生活首界面



步骤 2 选择需要配网的设备，点击“连接”开始配网。

步骤 3 配网成功后，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

## 4.2.3 添加设备失败问题分析

本节对添加设备过程进行拆解和说明，帮助伙伴了解添加设备的主要过程，以达成快速对问题定位定界的目的。

从执行顺序上看，添加设备过程依次经历以下三个过程阶段。

表4-2 添加设备过程介绍

阶段	作用	开始标志	结束标志	成功日志	失败日志
配网阶段	设备连接到 wifi 热点，具备联网能力	wait STA join AP	connect success	-	-
认证阶段	联网认证设备合法性，校验激活码和设备信息	INFO [KitFramework]: Device is in initialization mode	[KitFramework]: Response kit authentication status by executing callback	INFO [KitFramework]: Active symbol succeed	搜索关键字“ERROR [KitFramework]:”
注册阶段	注册设备信息，建立设备和账号的关联关系	set dev status [2]	set dev status [4]	set dev status [4]	未打印“set dev status [4]”，或者打印“set dev status [6]”。

### □ 说明

认证阶段日志信息较多，搜索关键字“INFO [KitFramework]:”可以查看认证的过程。

- 如果设备认证成功，会打印“INFO [KitFramework]: Active symbol succeed”日志。
- 如果设备认证失败，将不会打印“INFO [KitFramework]: Active symbol succeed”日志，需要搜索“ERROR [KitFramework]:”查看错误信息。