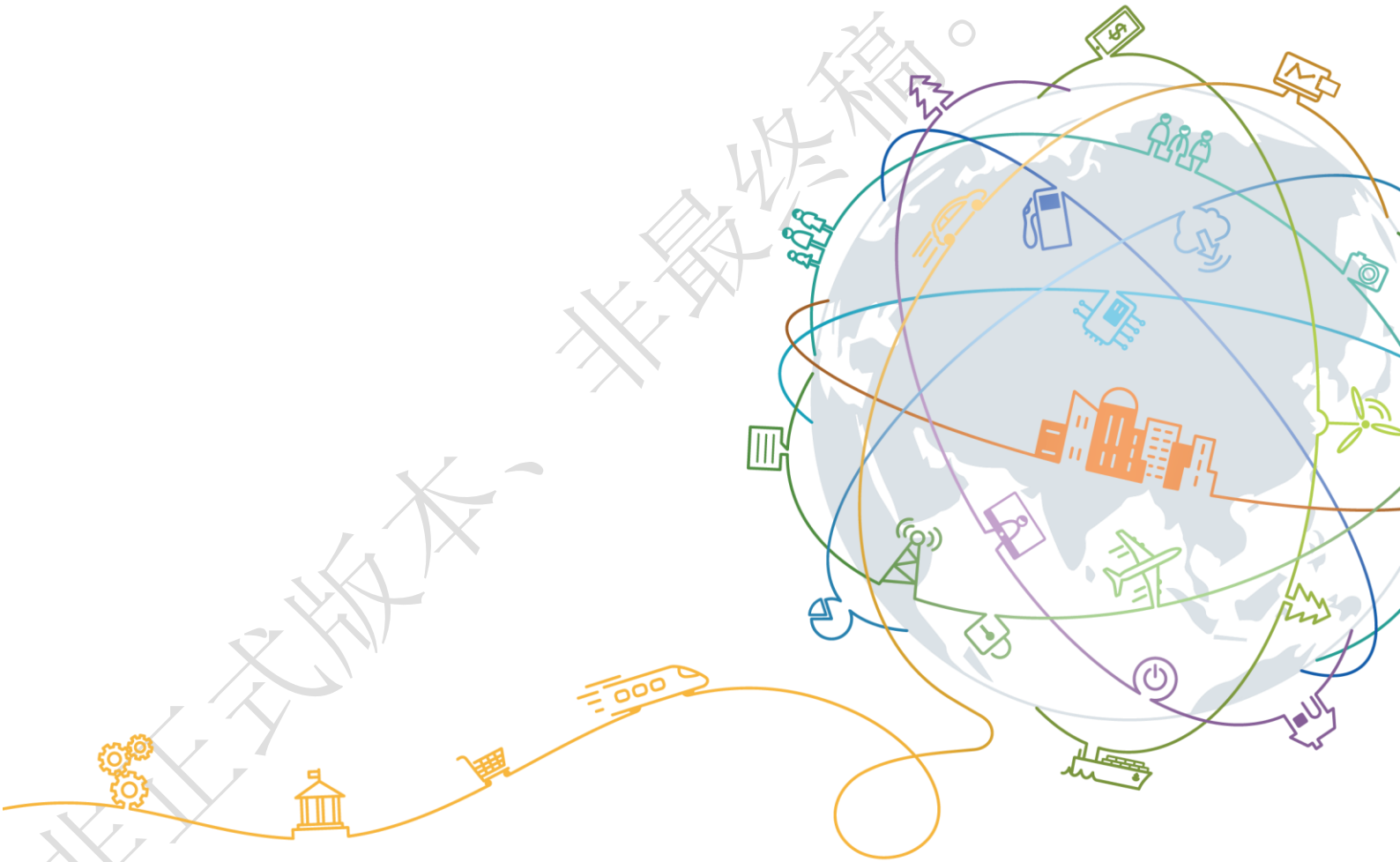
原子化服务开发指南

文档版本 01

发布日期 2021-08-25



华为终端有限公司

商标声明


、HUAWEI、HUAWEI、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

1 原子化服务概述

简介

原子化服务是一种基于 HarmonyOS API 开发，支持运行在“1+8+N”设备上，具有独立入口、免安装的用户应用程序形态。

本手册旨在指导 HarmonyOS Connect 合作伙伴，快速构建出适用于手机的、具有连接配对和设备控制等能力的原子化服务，为用户提供无障碍激活设备、一键登录、快速绑定、便捷交互等体验。

拉起方式

用户可以通过手机碰一碰、靠近发现等方式，快速拉起手机上的原子化服务，实现 HarmonyOS Connect 设备的连接和控制。两种拉起方式的简介见表 1-1。

表1-1 原子化服务的两种拉起方式

拉起方式	碰一碰	靠近发现
能力介绍	碰一碰能力依托 NFC 短距通信协议，通过触碰交互将手机和全场景设备连接起来，为用户提供手机到周边设备多种业务无缝切换的良好体验，解决了应用跨设备接续难、设备配网难和传输难等问题。	靠近发现是华为基于蓝牙通信面向生态合作伙伴开放的一项核心能力，可以助力全场景设备更便捷地与华为产品连接，共同打造核心竞争力。
使用方式	设备具有 NFC 标签，手机通过“碰一碰” 识别设备上的 NFC 标签，即可运行相应的原子化服务。	设备具有蓝牙靠近发现功能，手机通过“靠近发现”识别设备发出的蓝牙广播，并运行相应的原子化服务。
使用示例	用户拿出华为手机“碰一碰”蒸烤料理机，手机将自动跳转到相应的原子化服务页面，显示蒸烤料理机的配网界面或者控制界面。用户在手机侧即可操作设备，比如在控制界面选择系统推荐菜谱，体验自动烹饪等功能。	用户拿出手机靠近一台已开启蓝牙功能的智能风扇，手机上自动弹出原子化服务，显示风扇的状态信息和控制界面。用户在手机上即可调控风扇。

应用场景

- **设备连接**

  “碰一碰”是用户触达设备控制服务的开端。

  首次使用时，手机“碰一碰”未连接的设备，用户需要先完成设备连接，连接完成后拉起控制面板。

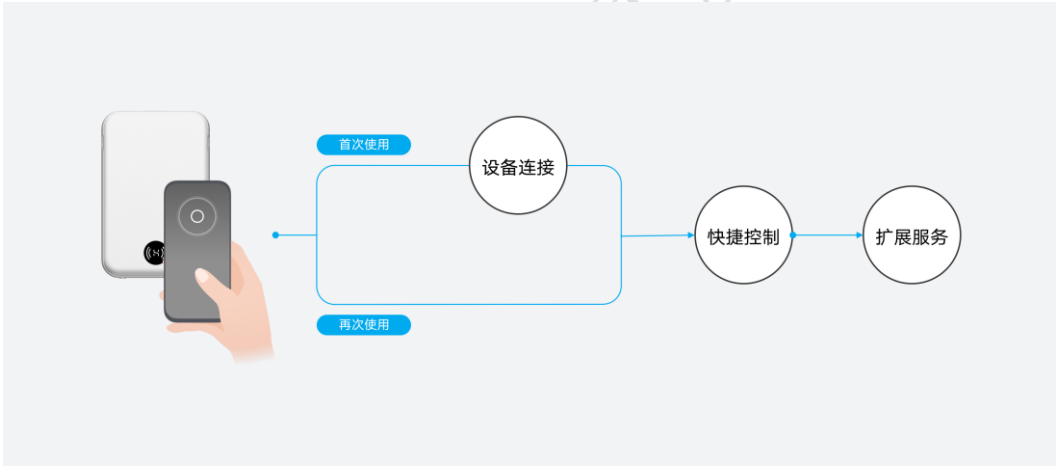
  再次使用时，手机“碰一碰”已连接设备，直接拉起控制面板进行快捷控制，用户无需再次进行设备连接。

- **快捷控制**

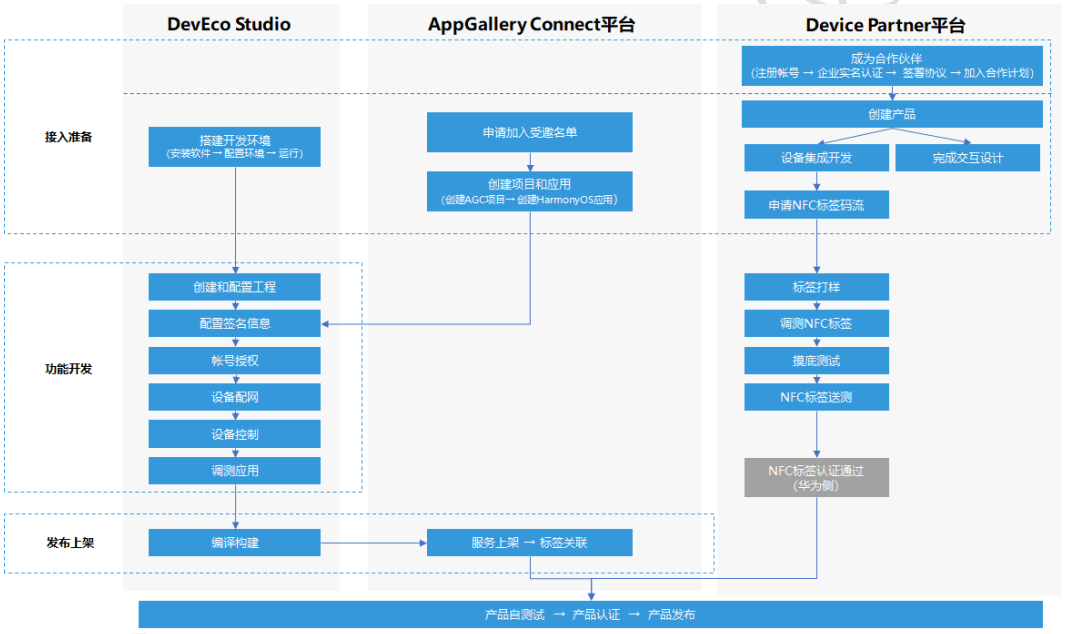
  快捷控制是日常使用的核心功能，大多数的控制功能通过控制面板界面来实现。

- **扩展服务**

  当用户需要了解更丰富的扩展服务时，可以进入到全屏界面完成。全屏页面可以展示丰富的内容，如设备的服务，如玩法、食谱、场景等，适用于复杂的功能使用场景。



2 开发整体流程



3 接入准备

3.1 前提条件

- **成为华为智能硬件合作伙伴。**详细操作参考成为合作伙伴，包括注册华为帐号、完成企业实名认证、签署协议、加入合作计划等操作。

  经过实名认证的的企业帐号可以用于：在 Device Partner 平台创建和管理产品、使用 DevEco Studio 开发原子化服务、以及在 AppGallery Connect 上架原子化服务等。
- **准备用于集成开发的模组。**
- **准备一部用于原子化服务调测的华为手机**，需要满足如下三个条件：
  - 确认华为手机的操作系统已升级为 HarmonyOS 系统，且系统版本号为 2.0.0.116 及以上（可通过手机的“设置 > 关于手机 > 版本号”查看）。

说明

当前支持 HarmonyOS 系统的手机型号包括：Mate X2、Mate40、Mate40E、Mate 40 Pro、Mate 40 Pro+、Mate 40 RS 保时捷设计、P40、P40 Pro、P40 Pro+、Mate 30 4G、Mate 30 Pro 4G、Mate 30 5G、Mate 30 Pro 5G、Mate 30 RS 保时捷设计、Mate 30E Pro 5G 等。

  - 确认手机中已安装表 3-1 中的应用，且应用版本号满足要求。检查方法如下：
    i. 在手机上选择“设置 > 应用和服务 > 应用管理”。
    ii. 点击右上角的⋮，选择“显示系统程序”。
    iii. 在搜索框中输入**应用名称**，在搜索结果中单击应用，即可查看版本号。

    如果搜索不到应用，或版本号不满足要求，请联系华为技术支持人员获取相应版本。

表3-1 手机应用要求

应用名称	用途	版本号要求
服务中心	用于碰一碰拉起原子化服务。	12.0.0.310 及以上
智慧生活基础服务	用于设备配网和设备控制。	12.0.1.308 及以上
智慧生活	用于设备房间号更改、设备删除、设备分享等功能。	12.0.0.306 及以上

3.2 DevEco Studio 相关准备工作

通过 DevEco Studio 工具，开发者可以更高效地开发、编译、调试、发布原子化服务。本节介绍如何准备搭建 DevEco Studio 开发环境，为原子化服务的功能开发做准备。

搭建 DevEco Studio 开发环境

步骤 1 下载并安装 DevEco Studio 软件。

说明

DevEco Studio 软件版本号应为 V2.1 Release 及以上。

步骤 2 配置 DevEco Studio 开发环境。

步骤 3 创建和运行 Hello World，验证环境设置是否正确。

----结束

3.3 AppGallery Connect 平台相关准备工作

申请加入受邀名单

当合作伙伴加入受邀名单之后，才能使用应用分发和应用上架等功能。

如果合作伙伴不在受邀名单中，请将“开发者名称”、“申请背景”、“支持设备类型”及“Developer ID”发送至 agconnect@huawei.com，华为运营人员将在 1-3 个工作日内为安排对接人员。Developer ID 查询方法请参见查询开发者帐号 ID 及项目 ID。

待加入受邀名单中，再继续执行后续操作。

创建项目和应用

在 AppGallery Connect 网站创建应用之后，才能使用 AppGallery Connect 提供的各类服务、并在华为应用市场发布自己的应用。

步骤 1 在 AppGallery Connect 网站中创建 AGC 项目。请参考：创建项目。

步骤 2 在 AppGallery Connect 网站中创建一个 HarmonyOS 应用。请参考：创建 HarmonyOS 应用。

步骤 3 建议您在本地进行应用调试，以查看和验证应用运行效果，减少发布过程中可能遇到的问题。请参考：调试 HarmonyOS 应用。

----结束

须知

- 在创建 HarmonyOS 应用的时候“应用包名”要唯一，建议为“com.公司名.模块名”。在 DevEco Studio 创建碰一碰模板工程时也需要使用该应用包名。
- 如果采用自动化签名的调试方式，会自动在 AGC 中创建用于调试的数字证书和 Profile 文件，由于当前 AGC 调试证书最多仅支持 2 个，即最多同时只支持为两个应用进行自动化调试。如已达到上限，需要在“用户与访问 > 证书管理”页面中“废除”多余的调试证书文件。
- 无论选择了**自动化签名**或者**手动签名**方式，在后续操作中（如申请签名指纹证书等），务必保证“***.p12”文件的唯一性。

3.4 Device Partner 平台相关准备工作

创建产品

开发者需要先在智能硬件合作伙伴平台创建产品信息。具体操作参考：创建产品。

设备集成开发

华为智能硬件合作伙伴平台为产品开发提供适配的解决方案，包括模组、开放能力 SDK、App 开发工具、设备开发工具以及测试工具等。

集成生态服务包后，可将工程编译成 bin 文件，然后将 bin 文件烧录到模组芯片。烧录成功后，芯片便可模拟成需要测试的设备。合作伙伴可以联系华为技术支持人员获取《设备集成指导》。

NFC 标签相关

步骤 1 申请 NFC 标签码流。

申请标签码流时，需要配置表 3-2 的字段值。合作伙伴需要要根据配网方式的不同，获取对应设备的 Wi-Fi/BLE Mac 地址和 SN 号。

表3-2 标签字段配置说明

字段	获取方式	示例
ProductID	从 Device Partner 平台获取，需要转成 ASCII 码表示。	ProdID：1234 ASCII 码：31323334
Mac 地址	• **Wi-Fi 地址查看方式：** 1. 打开手机，进入“设置 ”功能。 2. 搜索并进入“开发人员选项”。 说明 华为手机打开“开发人员选项”的方法如下： 1. 打开手机，进入设置功能，搜索“关于手	11 22 33 44 55 66

字段	获取方式	示例
	机”选项。 2. 快速点击“版本号”的项目多次，直到提示开发者模式已打开，退出该页面。 3. 开启“WLAN 详细日志记录”。 4. 在 Wi-Fi 列表找到对应设备，查看 Mac 地址。 • **BLE 地址查看方式：**联系华为技术接口人获取工具查看。	
SN 号	1. 在智慧生活找到该设备，进入当前设备详情页。 2. 点击右上角更多按钮，选择“设备信息”。 3. 查看序列号（SN）。	11 22 33 44 55 66 11 22 33 44 55 66

生成的完整的标签码流示例如下：

- 极速配网（NAN）和常规配网（SoftAP）

  D2022D6877200100480031323334008108005520068504141704061122334455661 40C1122334455661122334455661701 0A

- 蓝牙辅助配网（BLE）

  D2022D6877200100480031323334008108005520068503141703061122334455661 40C1122334455661122334455661701 09

说明

蓝色部分为写标签内容时需要更改的内容，请根据具体读取的设备参数信息进行更改。

步骤 2 标签打样。

步骤 3 调试 NFC 标签。

步骤 4 摸底测试。

步骤 5 NFC 标签送测。

步骤 6 等待 NFC 标签认证通过。

----结束

交互设计

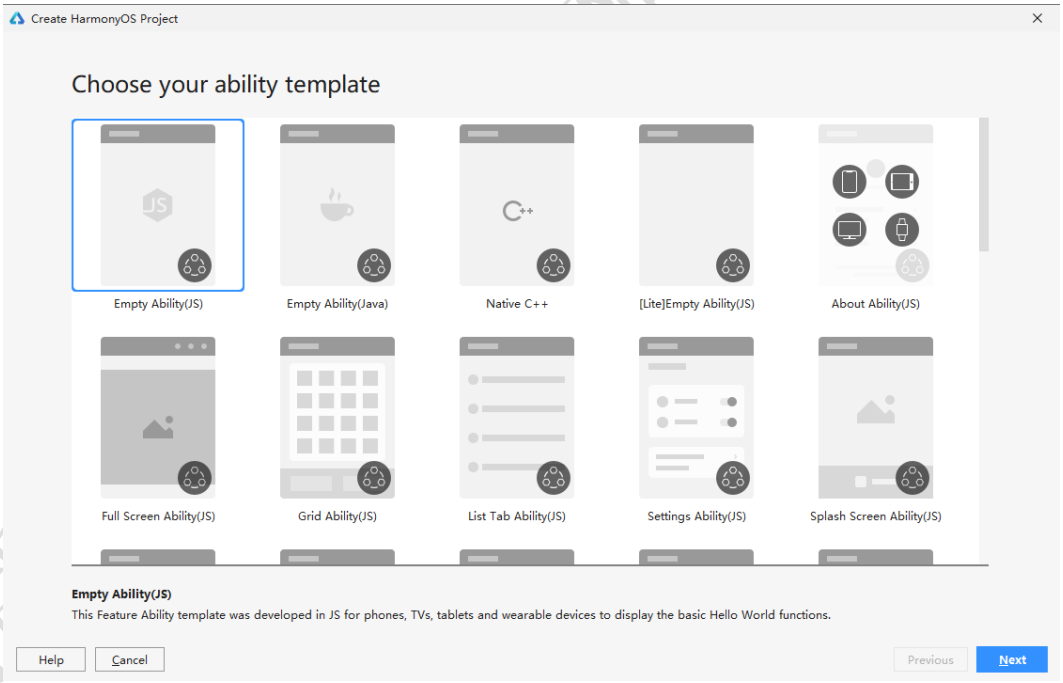
1. 按照设备控制设计规范，完成 FA 相关的 UX/UI 设计稿。
2. 对照设计自检表，自检 UX/UI 设计稿，确保设计稿符合要求。
3. 参考 FA 服务中的步骤 4~步骤 7，完成 UX/UI 设计稿的审核。

4 功能开发

4.1 创建和配置工程

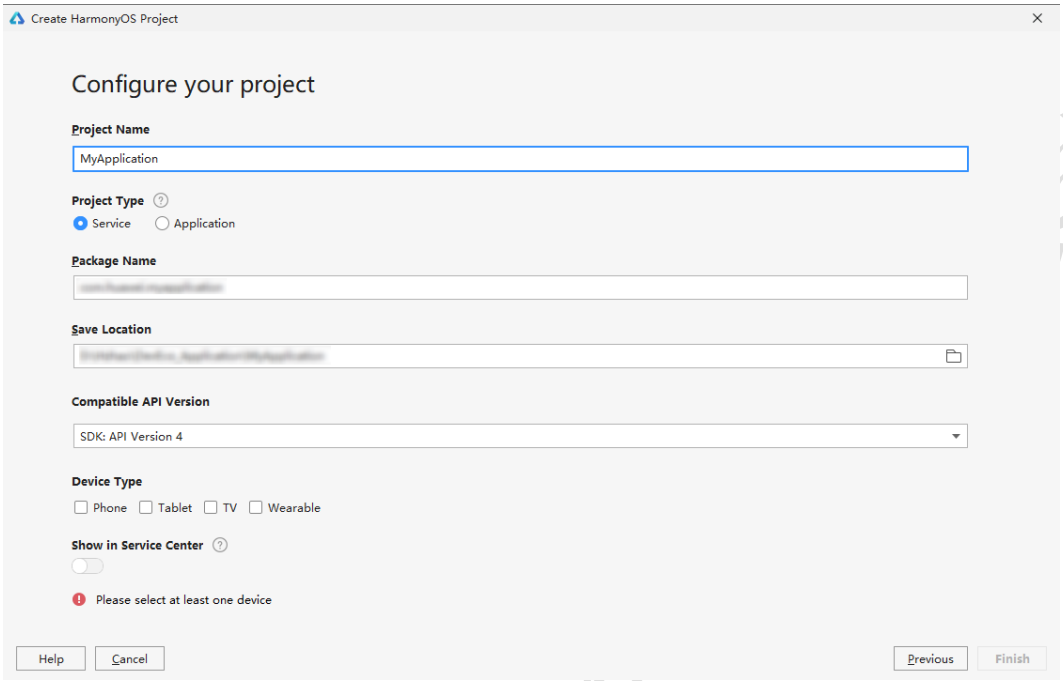
步骤 1 启动 DevEco Studio，选择“Empty Ability(JS)”模板，并点击 Next。

图4-1 选择模板



步骤 2 在 Project Type 中选择 Service，Package Name 填写与在 AppGallery Connect 平台创建的 HarmonyOS 应用一样的包名。在填写好其它信息之后，点击 Finish，会自动生成基于“Empty Ability(JS)”的模板工程代码。

图4-2 配置工程并创建



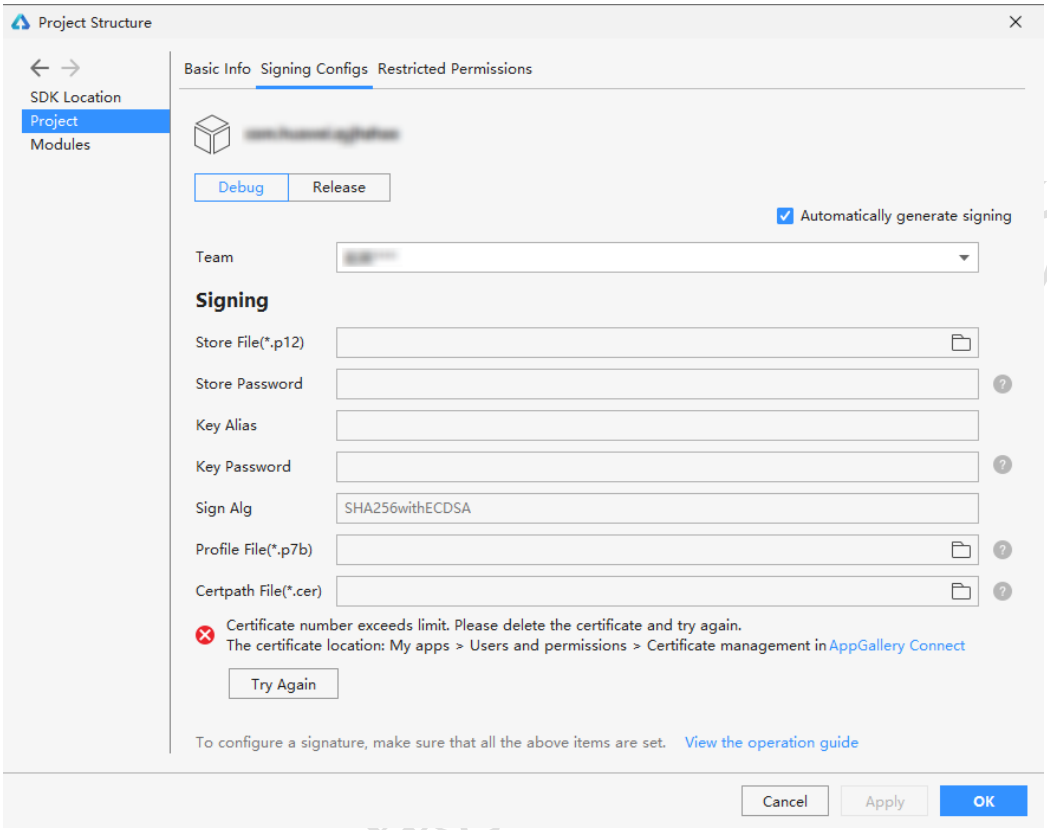
----结束

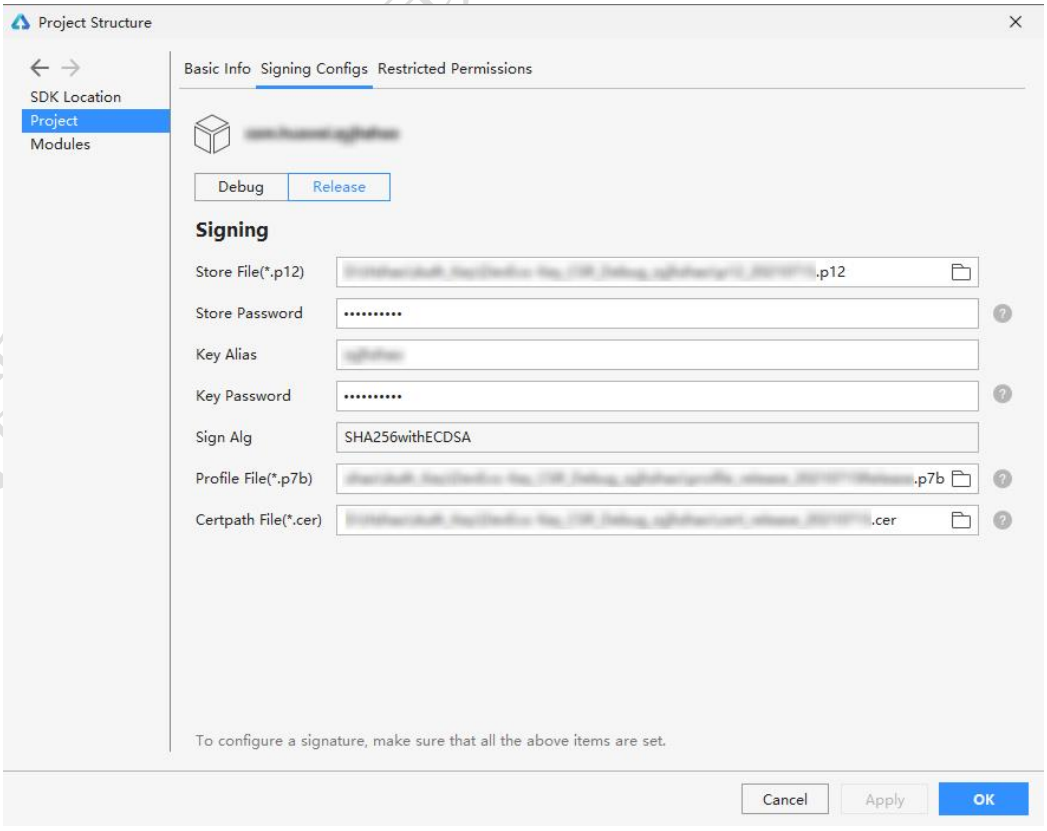
4.2 配置签名信息

签名配置

将生成签名证书的签名文件对工程进行配置。

在 DevEco Studio 工具界面，选择“File > Project Structure”，对 Project 分别配置 Debug 和 Release 签名（Release 签名获取参考：准备签名文件）。



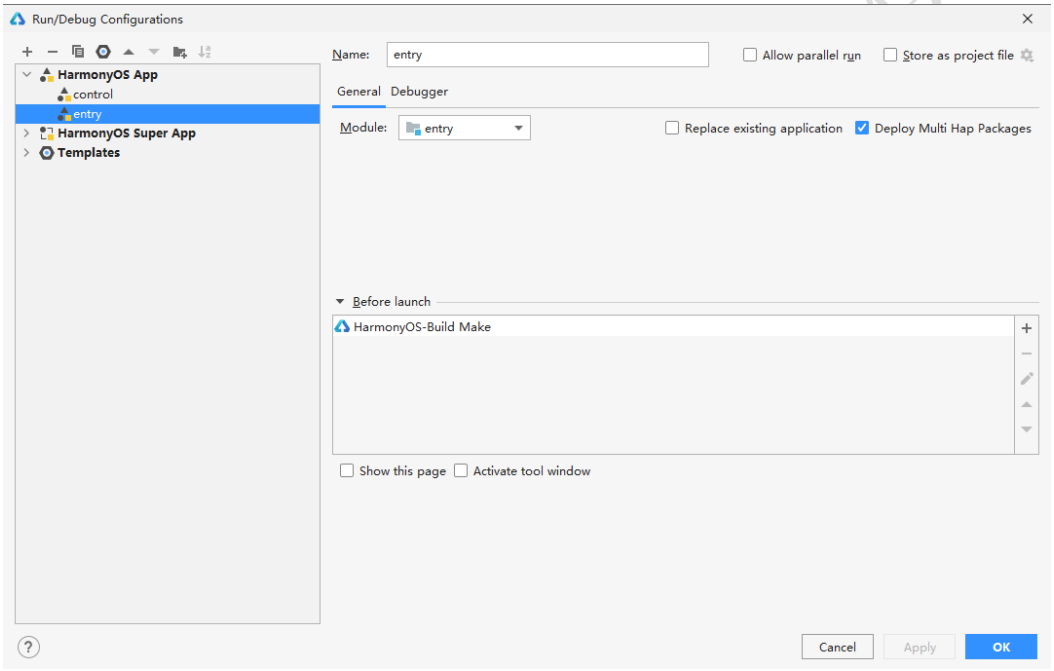


说明

在 DevEco Studio 2.1 Release 及更高版本中支持仅需要配置 Project 中的 Debug 和 Release 签名即可（Debug 签名支持自动化签名和手动签名两种方式）。

设置 Hap 包安装方式

当前工程中一个设备存在多个模块（entry 和 control 模块），且存在模块间的调用，在调试阶段需要同时安装多个模块的 hap 包到设备中。此时，需要在待调试模块的设置项（DevEco Studio，File > Project Structure）中勾选“Deploy Multi Hap Packages”。例如 entry 模块调用 control 模块，在调试 entry 模块时，需要同时安装 control 模块，应该在 entry 模块的调试设置项中勾选“Deploy Multi Hap Packages”后再启动调试。



4.3 帐号授权

生成签名证书指纹

签名证书指纹用于校验应用的真实性。

说明

在生成签名证书指纹前需要满足以下条件：

JDK 已经配置环境变量（样例为：DevEco Studio 安装在 D:\Program Files\Huawei 目录下），配置系统环境变量：D:\Program Files\Huawei\DevEco Studio 2.1.0.501\tools\openjdk\bin

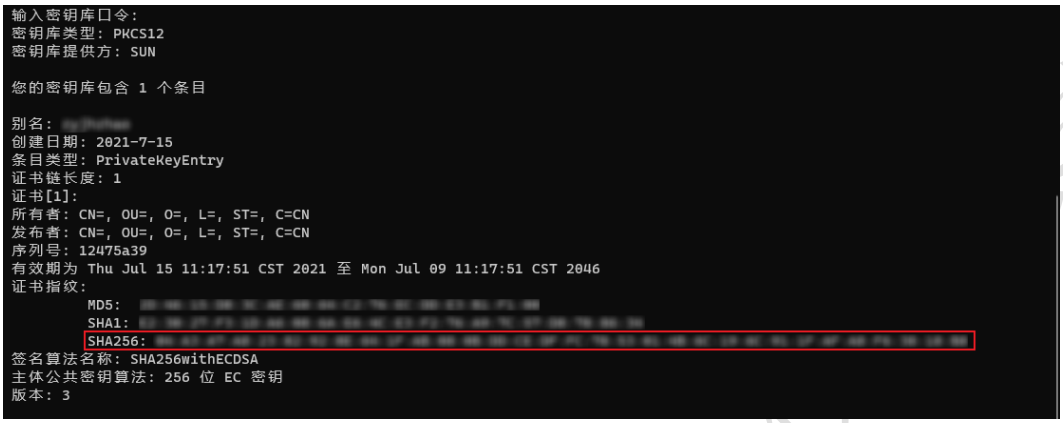
操作步骤如下：

步骤 1 在签名文件目录下（样例为：D:\Auth_Key\）打开 CMD 命令工具，执行命令：keytool -list -v －keystore ***.p12， 其中***.p12 为申请调试证书时申请的 Key Store 文件。

例如：

```
keytool -list -v -keystore ***.p12
```

步骤 2 根据结果获取对应的 SHA256 指纹。



----结束

申请 Oauth2.0 客户端 ID

华为帐号服务为您提供了简单、安全的登录授权功能，方便用户快捷登录。用户不必输入帐号、密码和繁琐验证，就可以通过“华为帐号登录”快速登录，即刻使用您的App。

目前支持基于 OAuth 2.0 协议标准接入华为帐号，您需要先获取凭证，申请申请Oauth2.0 客户端 ID。

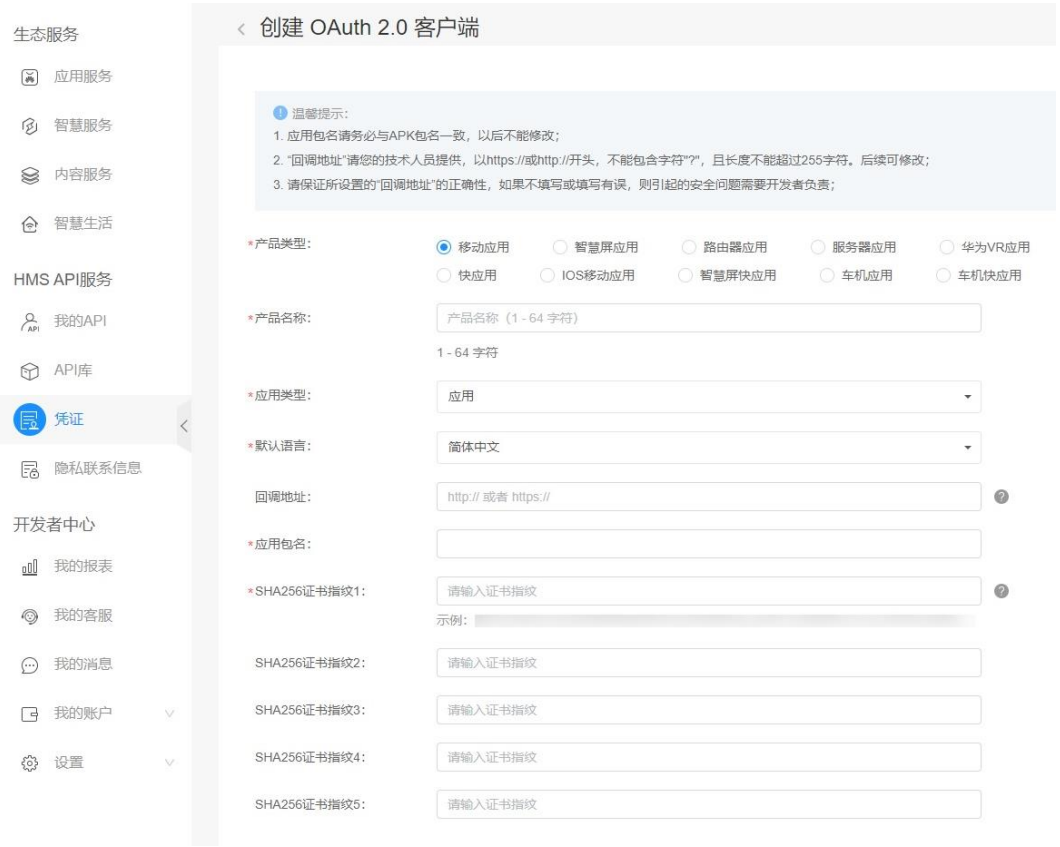
步骤 1 打开操作平台：开发者联盟，进入“管理中心”，点击左侧菜单“凭证”。

步骤 2 创建 OAuth 2.0 客户端。填写相关信息，点击“创建”。

表 1 创建 OAuth 2.0 客户端

名称	状态	操作
产品类型	必选，提交后不能修改	选择“移动应用”。
产品名称	必选，提交后不能修改	**产品名称会显示在向用户申请授权时的页面中，**并且产品名称需要与上述创建 HarmonyOS 应用的应用名称一致。

名称	状态	操作
应用类型	必选，提交后不能修改	选择“应用”。
默认语言	必选，提交后不能修改	选择“简体中文”。
回调地址	可选	由您的技术人员提供，以 https://或 http://开头，不能包含字符"?"，且长度不能超过 255 字符。
应用包名	必选，提交后不能修改	建议为“com.公司名.模块名”，注意包名需与创建的 HarmonyOS 应用的应用包名保持一致。
SHA256 证书指纹	必选，提交后可以修改	上述生成秘钥和证书请求文件。



----结束

帐号授权配置

帐号授权开发之前需要配置以下选项，否则无法进行帐号授权登录开发。帐号授权配置步骤如下：

步骤 1 在项目的工程目录，打开命令号窗口，执行以下命令，安装帐号授权依赖的 NPM 包。

```
npm install @hmscore/hms-js-account

# 如有报错，执行以下命令后，再重新执行上述命令
npm init -f
npm install formidable --save
```

步骤 2 在项目的工程目录，找到\entry\build.gradle 文件，增加 jsb-ohos-adapter 的配置配置依赖。

```
dependencies {
    ...
    implementation(group: 'com.huawei.hms', name: 'jsb-ohos-adapter', version:
'5.3.0.303', ext: 'har')
}
```

步骤 3 根据上述申请的 OAuth 2.0 客户端 ID 和秘钥，找到项目的\entry\src\main\config.json 文件中，添加相应的 APP ID 配置信息：

```
// 修改 customizeData 属性
"metaData": {
    "customizeData": [
        {
            "name": "com.huawei.hms.client.appid",
            "value": "123456789" // 申请的 OAuth 2.0 客户端 ID
        }
    ]
}
```

步骤 4 配置组件代理，在\entry\src\main\config.json 文件中，添加配置："allowComponentsProxy": true，例如：

```
"deviceConfig": {
    "default": {
        "allowComponentsProxy": true
    }
}
```

----结束

帐号授权开发

说明

华为帐号体系属于 HMS 服务，需要参考 HMSCore JS API 授权相关功能。如果业务中需要使用登录的其他功能，可参考开源库指导：hms-js-account

步骤 1 entry 目录下的 MainAbility.java`文件，注册 HMS 服务。

```
import com.huawei.hms.jsb.adapter.har.bridge.HmsBridge;

public class MainAbility extends AceAbility {
    // ...

    @Override
    public void onStart(Intent intent) {
        intent.setParam("window modal", 3);
        HmsBridge.getInstance().initBridge(this); // Add
```



```
        // ...

        super.onStart(intent);
    }

    @Override
    public void onStop() {
        HmsBridge.getInstance().destoryBridge(); // Add
        super.onStop();
    }

    // ...
}
```

步骤 2 entry 目录下的 index.js（首页）文件。

1. 增加导入和公共参数。

```
import HMSAccount from "@hmscore/hms-js-account";

const permissions = [
HMSAccount.Permission.ACCESSTOKEN
];

const scopes = [
HMSAccount.Scope.OPENID, // 关联华为帐号
HMSAccount.Scope.PROFILE // 获取您的头像和昵称
];

export default {
    data: {
        // ...
        openIdSaved: "" // Add
    },
}
```

2. 华为帐号登录方法 hwLoginClick()中调用 HMS 的 HMSAccount.signIn()接口。

```
import storage from '@system.storage';

const STORAGE_KEY_OPEN_ID = 'open_id';

//hw authorization icon click event
hwLoginClick() {
    HMSAccount.signIn(permissions, scopes).then((res) => {
        this.accessToken = res.data.accessToken;
        this.openId = res.data.openId;
        this.saveOpenId(this.openId);
    }).catch((err) => {
        console.error("User info. Invoke `HMSAccount.signIn()` failed, error
info: " + JSON.stringify(err));
    });
},
// Save openId
saveOpenId(openId) {
    storage.set({
        key: STORAGE KEY OPEN ID,
        value: openId,
```

```
        success: function () {
        },
        fail: function (data, code) {
         console.error("User info. Parameter of `openId` save failed, data: " +
JSON.stringify(data) + ", code: " + code);
        }
    });
},
```

3. 初始静默登录校验，在拉起 FA 首页时候，会进行静默登录校验。

```
import app from '@system.app';

async silentSignIn() {
    this.openIdSaved = await this.getOpenId(STORAGE KEY OPEN ID);

    if (this.openIdSaved != "") {
        HMSAccount.silentSignIn(permissions, scopes).then((res) => {
            let openId = res.data.openId; // 华为帐号的唯一标识
            if (openId && (openId == this.openIdSaved)) {
                // openId 相同，确定华为帐号无变化
                // 如果设备未配网，进入网络设置页面，如果已经配网，进入设备控制页
                let netConfigFlag = false; // 是否有获取到 deviceId
                if (netConfigFlag) {
                    // TODO: go to control page
                } else {
                    // TODO: go to network configuration page
                }
            }
        }).catch((err) => {
            console.error("User info. Invoke `HMSAccount.silentSignIn()` failed,
error info: " + JSON.stringify(err));
        });
    }
},
// Get openId
getOpenId(key) {
    return new Promise((resolve, reject) => {
        storage.get({
            key: key,
            success: function (data) {
                resolve(data);
            },
            fail: function (data, code) {
                console.error("User info. Parameter of `openId` get failed, data:
" + JSON.stringify(data) + ", code: " + code);
                reject();
            }
        });
    });
},
```

4. 初始化中静默登录的方法调用。

```
onInit() {
    // ...

    this.getOpenId(STORAGE_KEY_OPEN_ID).then((data) => {
```



```
        if (data && data != "") {
            this.silentSignIn();
        }
    });
},
```

----结束

帐号授权取消

说明

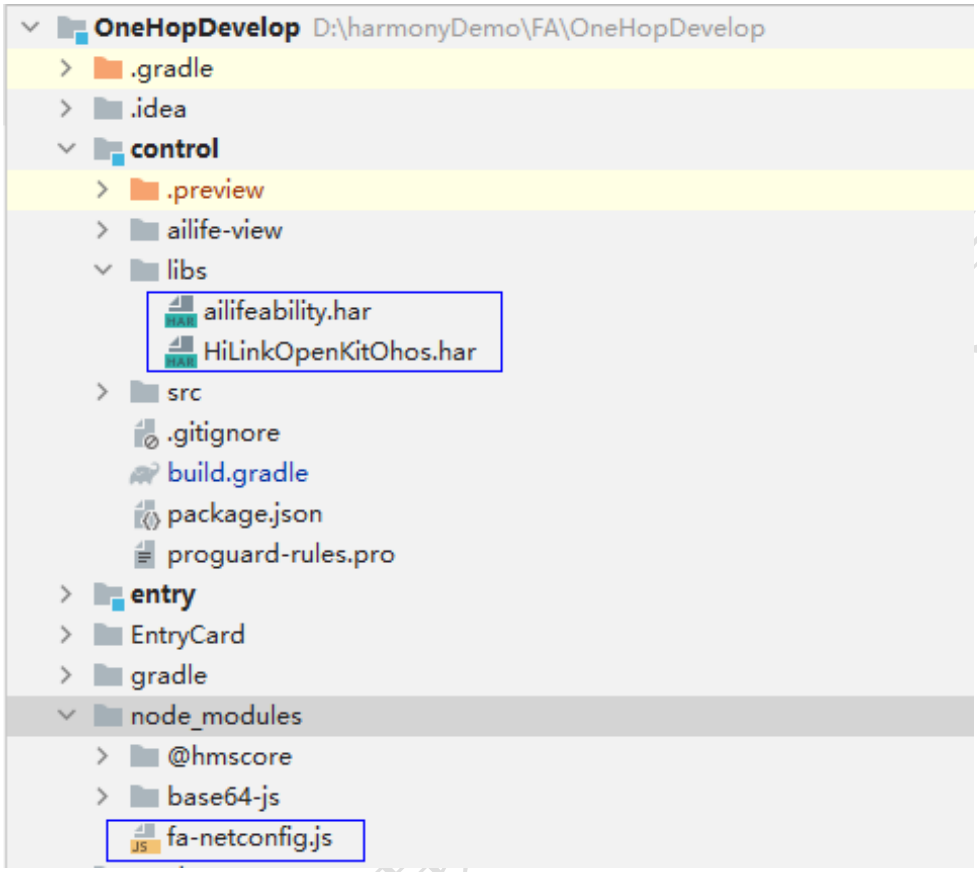
如需要取消帐号授权，可以通过“华为帐号 > 隐私中心 > 控制对您帐号的访问”，选择到相应的应用，取消授权即可。

4.4 设备配网

添加依赖模块

联系华为技术接口人获取 ailifeability.har、HiLinkOpenKitOhos.har 和 fa-netconfig.js 文件，并将 har 包放置到 control/libs 目录下，将 fa-netconfig.js 文件放到根目录 node_modules 文件夹下。

图4-3 添加 har 包依赖和 fa-netconfig.js 文件



须知

放置相关 Har 包文件之后，需要在 DevEco Studio 项目中同步工程，点击“File > Sync Project with Gradle Files”。

入参说明

碰一碰根据设备的配网状态分为首碰和二碰。

- 首碰表示未配网，此时手机与 NFC 标签碰一碰后，将启动配网流程。
- 二碰表示已配网，此时手机与 NFC 标签碰一碰后，将启动控制流程。

当进行碰一碰操作时，由 HiLink Service 拉起 HarmonyOS 服务。程序从 5.2 标签关联的 Ability 开始执行，需要在 Ability 的 onStart 方法中对 intent 携带的参数进行解析，据此判断是启动配网流程还是启动控制流程。

- 首碰（未配网）HiLink Service 传递参数如下所示。

```
{
    "empty": false,
    "params": {
        "ProtocolID": 10,
        "abilityName": "{your package name}.MainAbility",
```

```
      "apMac": "************",
      "bundleName": "{your package name}",
      "callBackActivityName":
"com.huawei.hilink.framework.deviceaddui.activity.WifiSettingDialogActivity",
      "callBackPackageName": "com.huawei.hilink.framework",
      "customData": {
         "empty": false,
         "params": {
            "91": "************************",
         },
         "unpacked": true
      },
      "icon": "{pic path}.png",
      "init": true,
      "isBound": false,
      "label": "{label}",
      "nanSessionId": "********-****-****-****-************",
      "ohos.extra.param.key.INSTALL_ON_DEMAND": false,
      "prodId": "{product id}",
      "sn": "************",
      "subModelId": "00",
      "uuid": "********-****-****-****-************",
      "window_modal": 3
   },
   "unpacked": true
}
```

未配网状态，intent 不会携带 deviceId 信息，据此判断启动配网流程。

表4-1 参数说明

参数名称	说明
apMac	设备 Wi-Fi MAC 地址
0X91	厂商自定义信息
deviceId	设备 ID
init	判断是从 HiLink Service 配网界面重新拉起 FA，还是从通过 NFC 标签碰一碰拉起 FA 场景
nanSessionId	极速配网（NAN）通道 ID
prodId	设备型号
subModelId	设备子型号
uuid	回调传入的随机数

- 二碰（已配网）HiLink Service 传递参数如下所示。

```
{
   "empty": false,
```

```
   "params": {
      "ProtocolID": 10,
      "apMac": "************",
      "callBackActivityName":
"com.huawei.hilink.framework.deviceaddui.activity.WifiSettingDialogActivity",
      "callBackPackageName": "com.huawei.hilink.framework",
      "deviceId": "********-****-****-****-************",
      "init": true,
      "isBound": false,
      "nanSessionId": "NAN_DEVICE_NOT_FOUND",
      "ohos.extra.param.key.INSTALL_ON_DEMAND": false,
      "sn": "************",
      "uuid": "********-****-****-****-************",
      "window_modal": 3
   },
   "unpacked": true
}
```

已配网状态，intent 会携带 deviceId 信息，据此判断启动控制流程。

说明

可以通过 nanSessionId 入参的值，查看当前设备是否处于 NAN 待配网状态。nanSessionId 为"NAN_DEVICE_NOT_FOUND"时，表示未发现 NAN 待配网设备；nanSessionId 为"********-****-****-****-************"时，表示当前发现了 NAN 待配网设备。

设备配网

设备配网分为极速配网（NAN）和常规配网（SoftAP）两种方式，其中极速配网（NAN）方式，在配网的过程中就可以拉起设备控制界面，当前只有 Hi3861 芯片支持。

不同的的配网方式要传递的参数不一样、跳转的页面也不一样，需要先对 HiLink Service 传递过来的参数进行解析，再根据解析出来的字段判断是极速配网（NAN）还是常规配网（SoftAP）。示例步骤如下：

步骤 1 在意图配置的 Ability 的 onStart 方法中对 intent 携带的参数进行解析，代码示例如下：

```
@Override
public void onStart(Intent intent) {
    intent.setParam("window modal", WINDOW MODAL);
    boolean isInit = intent.getBooleanParam("init", false);
    String deviceId = intent.getStringParam("deviceId");
    String prodId = intent.getStringParam("prodId");
    String nanSessionId = intent.getStringParam("nanSessionId");
    String uuid = intent.getStringParam("uuid");

    if (!isInit && !EntryUtil.isSessionIdInvalid(nanSessionId)) {
     // TODO: 已经从 HiLink Service 配网界面重新拉起 FA，设备正在进行 NAN 配网，跳转半模态秒控页
面
    } else if (!isInit && !EntryUtil.isEmpty(deviceId)) {
     // TODO: 已经从 HiLink Service 配网界面重新拉起 FA，设备已经配网完成，进入半模态控制界面
    } else {
     // TODO: 通过 NFC 标签碰一碰拉起 FA 场景，包括：帐号授权、设备配网等过程
        super.onStart(intent);
    }
}
```

步骤 2 在 4.3 帐号授权之后，根据解析的参数数据进行跳转判断逻辑，代码示例如下：

```
private static final String HILINK NET CONFIG URI =
"hilink://hilinksvc.huawei.com/device?action=deviceAdd&prodId=%s&fromApp=%s";

/**
 * 跳转配网或控制界面判断
 */
private void goNetConfigOrDevicePage() {
    if (EntryUtil.isEmpty(deviceId)) {
        // 设备未配网，进入 HiLink Service 配网界面
     openHiLinkNetConfig();
    } else {
        // 设备已配网，进入设备控制界面
    }
}

/**
 * 打开配网界面
 */
private void openHiLinkNetConfig() {
    mContext.getUITaskDispatcher().asyncDispatch(() -> {
        String uriStr = String.format(Locale.ROOT, HILINK NET CONFIG URI, productId,
mContext.getBundleName());
        LogUtil.info(TAG, "openHiLinkNetConfig: uriStr = " + uriStr);
        Intent intent = new Intent();
        intent.setParam("uuid", uuid);
        Operation operation = new Intent.OperationBuilder()
                .withDeviceId("")
                .withFlags(Intent.FLAG ABILITY NEW MISSION |
Intent.FLAG NOT OHOS COMPONENT)
                .withAction("android.intent.action.VIEW")
                .withUri(Uri.parse(uriStr))
                .build();
        intent.setOperation(operation);
        mContext.startAbility(intent, -1);
        mContext.terminateAbility(); // 拉起 网络设置页面 的同时，关闭当前界面
    });
}
```

----结束

4.5 设备控制

设备控制分类

设备控制分为两种方式，极速配网（NAN）控制和常规设备控制。极速配网（NAN）控制是在配网的过程中即可对设备进行控制，当前只有 Hi3861 芯片支持。另一种常规设备控制，只有在设备配网成功之后才能进行设备控制。

跳转到设备控制界面

无论是进入到配网界面，还是进入到设备控制界面，都是首先由 HiLink Service 拉起意图里配置好的 Ability，再在 onStart 函数里面，解析传递过来的参数。关于参数解析，请参考设备配网的步骤 1。解析完成之后，再根据传递的参数不同，进行不同的跳转。请参考设备配网的步骤 2。

关于跳转到设备控制界面，代码示例如下：

```
// 跳转到设备控制页示例代码
private static final String BUNDLE NAME = "{your package name}";
private static final String DEVICE ABILITY NAME = "{your package
name}.ControlMainAbility";
private void openDevicePage() {
    Intent intent = new Intent();
    intent.setParam("device id", deviceId); // 如果配网方式是常规配网（SoftAP）或者蓝牙辅助配网（BLE），跳转控制页时需要传 deviceId
    // intent.setParam("session id", sessionId); // 如果配网方式是极速配网（NAN），跳转控制页时需要传 sessionId
    ElementName elementName = new ElementName("", BUNDLE NAME, DEVICE ABILITY NAME);
    intent.setElement(elementName);
    intent.addFlags(Intent.FLAG ABILITY NEW MISSION);
    intent.addFlags(Intent.FLAG INSTALL ON DEMAND);
    mContext.startAbility(intent, 0);
}
```

NAN 配网控制

步骤 1 进入到控制界面后，在 onStart 函数里解析传递过来的参数，并进行 NAN 控制的注册，代码示例如下：

```
@Override
public void onStart(Intent intent) {
    intent.setParam("window modal", HALF MODAL);
    String sessionId = intent.getStringParam("sessionId");
    NetConfigAbility.register(this, sessionId);
    setPageParams("", intent.getParams());
    super.onStart(intent);
}
```

步骤 2 注册监听设备侧消息回调和监听 NAN 配网回调（由于该示例代码中所调用接口是 js 的，因此需要在该控制 ability 对应的 js 的 onInit 函数中进行注册），代码示例如下：

```
onInit() {
    setTimeout(() => {
        this.registerNanControlCallback();
        this.registerMsgReceive();
    }, 500);
},
// 注册监听设备侧消息回调：在该回调中可以获取设备状态信息，对设备对应的状态进行刷新（如开关状态等）
registerMsgReceive() {
    let commonInfo = {
        sessionId: this.sessionId,
        nanDataType: 1 // 1 代表是 nan 类型
    };
    this.$app.$def.NetConfig.registerMsgReceive(commonInfo, (value) => {
```



```
        // 监听获取设备状态信息，对设备对应的状态进行刷新（如开关状态等）
    }).then((ret) => {
    });
},
// 监听 NAN 配网回调：当 NAN 配网成功后，可以获取到该设备的 deviceId，这时用户可以选择通过
deviceId 进行设备控制
registerNanControlCallback() {
    const commonInfo = {
        sessionId: this.sessionId
    };
    this.$app.$def.NetConfig.registerNanNetConfigStatusCallback(commonInfo, (result)
=> {
        if (result.code == 0) {
            // 配网成功，获取到该设备的 deviceId，这时用户可以选择通过 deviceId 进行设备控制
            this.deviceId = result.data;
        } else {
            // 配网失败
        }
    }).then((ret) => {
    });
}
```

步骤 3 下发设备控制命令，代码示例如下：

```
sendCommand() {
    let commonInfo = {
        sessionId: this.sessionId,
        nanDataType: 1, // 1 代表是 nan 类型
        serviceId: "switch", //要控制的设备服务的 ID（如：当前该设备的开关按钮的服务 ID 是：
"switch")
        hiLinkControlData: { // 控制的参数(如：当前该设备的开关按钮，开的控制参数是：(on：1))
            on: 1
        }
    }
    getApp(this).NetConfig.sendMessage(commonInfo, "", (value) => {
        // 下发控制命令成功后，可以在步骤 2 的 registerMsgReceive 中获取到该设备的状态变化信息，进
行页面刷新
    }).then((ret) => {
    });
}
```

----结束

常规设备控制

步骤 1 在 control/src/main/java/{your package name}下，找到 MyApplication 类文件，在 onInitialize 函数中初始化应用。示例代码如下：

```
@Override
public void onInitialize() {
    super.onInitialize();
    AiLifeServiceHelper.initApplication(this);
}
```

步骤 2 进入到控制界面后，在 onStart 函数里解析传递过来的参数，并且需要添加连接设备管理服务的操作。由于连接该服务属于耗时操作，需要在子线程中调用。示例代码如下：


```
@Override
public void onStart(Intent intent) {
    deviceId = intent.getStringParam("device_id"); //进行设备管理的相关操作时需要该参数（如：设备信息变化订阅、获取设备信息、设备控制等）
    getGlobalTaskDispatcher(TaskPriority.DEFAULT).asyncDispatch(new Runnable() {
        @Override
        public void run() {
            // 连接服务(只当 result 返回值>0 时，才能调用设备管理的相关接口)
            int result = AiLifeServiceHelper.connect(abilityContext);
            if (result < 0) {// 如果 result<0 说明连接服务失败，直接退出应用
                terminateAbility();
            } else {
                // TODO: 如果 result>0，说明连接服务成功，这时候可以做一些设备信息变化订阅和获取设备信息的操作
            }
        }
    });
}
```

步骤 3 获取设备的相关信息，示例代码如下：

```
private HiLinkDevice mHiLinkDevice;

private final DeviceManager deviceManager;

/**
 * 获取设备信息
 */
public void getHiLinkDevice() {
    AiLifeServiceParamBuilder builder = new AiLifeServiceParamBuilder();
    builder.addScope(ApiParameter.Scope.FEATURE REQUEST CLOUD)
         .addScope(ApiParameter.Scope.FEATURE CLOUD CONTROL);
    PacMapEx parameters = builder.createParameters();
    deviceManager = (DeviceManager)
AiLifeServiceHelper.getService(AiLifeServiceHelper.DEVICE MANAGER SERVICE,
parameters);

    deviceManager.getHiLinkDevice(ApiParameter.Source.FROM CLOUD, deviceId, new
DataCallback<HiLinkDevice>() {
         @Override
         public void onSuccess(HiLinkDevice hiLinkDevice) {
              // TODO: 获取到设备的名称、图片、在线/离线状态、位置等信息，通知 JS 端刷新相关界面
              mHiLinkDevice = hiLinkDevice;
              getProfileData();
         }

         @Override
         public void onFailure(int i, String s) {
              // TODO: 数据获取失败，通知 JS 端弹出相关的错误提示信息
         }
    });
}

/**
 * 获取设备状态
 */
```

```
private void getProfileData() {
    mHiLinkDevice.getProfileData(ApiParameter.Source.FROM_CLOUD, new ArrayList<>(),
new DataCallback<List<ServiceEntity>>() {
        @Override
        public void onSuccess(List<ServiceEntity> serviceEntities) {
            // TODO: 获取到设备的状态，比如开关、档位等状态信息，通知 JS 端刷新相关界面
        }

        @Override
        public void onFailure(int i, String s) {
            // TODO: 数据获取失败，通知 JS 端弹出相关的错误提示信息
        }
    });
}
```

步骤 4 在订阅设备信息事件中可以监听到该设备状态信息返回值，从而可以刷新设备对应状态（如设备开关状态等），示例代码如下：

```
public void subscribeDeviceEvent() {
   List<String> listParam = new ArrayList<>();
   listParam.add(deviceId);
   Objects.requireNonNull(deviceManager).subscribeDeviceEvent(new
DeviceListener.Builder()
          .addDeviceDeleteListener(s -> {
              // TODO: 监听设备删除变化，进行页面刷新
          })
          .addDeviceInfoListener(listParam, (s, hiLinkDevice) -> {
              // TODO: 监听设备基本信息变化，比如在线/离线等改变，进行页面刷新
          })
          .addProfileDataListener(deviceId, profileDataChangeEntity -> {
              // TODO: 监听设备状态变化，进行页面刷新
          })
          .build());
}
```

步骤 5 下发设备控制命令，示例代码如下：

```
private HiLinkDevice hiLinkDevice = null;
private CommandParam commandParam = null;
private void sendCommand(String serviceId, String type, int value) {
    try {
        commandParam = new CommandParam.Builder()
            .type(ApiParameter.CommandType.SERVICE ID) // 要控制的设备服务的类型
            .serviceId(serviceId) // 要控制的设备服务的 ID（如：当前该设备的开关按钮的服务
ID 是："switch"）
            .addCharastic(type, value) // 控制的参数(如：当前该设备的开关按钮，开的控制参
数是：("on", 1))
            .mode(ApiParameter.CommandType.SERVICE ID) // 控制方式
                  .build();
        hiLinkDevice = new HiLinkDevice();
        hiLinkDevice.setDeviceId(deviceId); // 单个设备控制需要设置该设备的 deviceId
        hiLinkDevice.sendCommand(commandParam, new DataCallback<String>() {
            @Override
            public void onSuccess(String s) { // 下发命令成功
            }
```



```
                @Override
                public void onFailure(int i, String s) {// 下发命令失败
                }
            });
        } catch (Exception e) {
            LogUtil.error(TAG + "sendCommand Exception:" + e.getMessage());
        }
}
```

----结束

4.6 调试应用

4.6.1 前提条件

1. 在正式环境上架之前，需要先部署镜像环境进行调试和验证，具体请联系华为技术接口人。
2. 已申请 HiLink Service 配网和控制权限。具体操作请联系华为技术接口人，同时需要提供如下信息。
   - **包名：**如 com.huawei.demo。
   - **SHA 256 指纹证书：**由 ***.p12 文件生成。
   - **prodId、deviceTypeId 和 manufacturerId：**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 4-4 所示。

图4-4 导出产品信息

4.6.2 环境准备

表4-2 测试环境准备

设备	版本/规格要求	相关操作
手机	手机的 HarmonyOS 系统已升级至 2.0.0.116 及以上版本。详情请参考 3.1 前提条件。	打开手机 NFC 开关，正常连接网络。
设备	烧录集成产品包版本，设备侧贴好已写好信息的 NFC 标签。	设备正常上电，处于待配网状态。
网络条件	2.4G Wi-Fi 网络，非通过短信验证码方式登录的网络。	-

4.6.3 测试步骤

步骤 1 碰一碰拉起 HarmonyOS 服务（FA）。

表4-3 拉起 HarmonyOS 服务

测试编号	预置条件	测试步骤	预期结果
测试用例 1	无	打开手机的 NFC 开关，连接 Wi-Fi 网络，手机碰一碰设备侧 NFC 标签。	手机侧正常拉起服务。

步骤 2 碰一碰，拉起设备配网。

- **极速配网（NAN）**

表4-4 设备配网（极速配网（NAN））

测试编号	预置条件	测试步骤	预期结果
测试用例 2	设备处于 NAN 待配网状态，且标签和手机距离模组 30CM 以内。	手机碰一碰设备侧 NFC 标签，拉起配网服务。进入配网界面，选择网络，输入 Wi-Fi 密码后进入配网流程。	手机侧 NAN 配网过程中进入控制界面。 设备侧正常联网。

- **常规配网（SoftAP）**

表4-5 设备配网（常规配网（SoftAP））

测试编号	预置条件	测试步骤	预期结果
测试用例 2	设备处于待配网状态。 当未发现 NAN 设备，或者 NAN 配网失败时候，则会走 SoftAP 配网。	手机碰一碰设备侧 NFC 标签，拉起配网服务。进入配网界面，选择网络之后，输入 Wi-Fi 密码后进入配网流程。	手机侧 SoftAP 配网成功，进入控制界面。 设备侧正常联网。

步骤 3 碰一碰，业务控制功能测试。

表4-6 业务控制功能测试

编号	预置条件	测试步骤	预期结果

编号	预置条件	测试步骤	预期结果
测试用例 3	手机侧已通过碰一碰拉起服务。 设备侧网络已配置。	手机二碰拉起控制界面，进行控制功能测试。	手机侧控制界面，可正常下发指令到设备。并且可以订阅设备状态信息返回值。

----结束

5 发布上架

5.1 服务上架

开发者完成 HarmonyOS 应用开发后，需要将应用打包成 APP，用于发布到华为应用市场。具体操作请参考应用发布。

5.2 标签关联

碰一碰场景，需要通过手机碰 NFC 标签，来拉起对应的 FA 服务。需要进行意图配置，将 FA 服务与 NFC 标签进行关联。

前提条件

服务已上架 AppGallery Connect。

操作步骤

步骤 1 用 AppGallery Connect 帐号登录华为开发者联盟。

步骤 2 在左侧导航中选择“智慧服务”，并单击“华为快服务智慧平台”。

页面将显示当前已在 AppGallery Connect 上架的原子化服务。

图5-1 快服务平台

步骤 3 在服务列表中，找到需要关联的原子化服务，单击“编辑”。

图5-2 HarmonyOS 服务列表

步骤 4 选择“配置 > 设备标签”，单击“添加”，添加设备标签。

图5-3 设备标签

步骤 5 在弹窗中，勾选需要的设备大类后，并单击“确定”。

图5-4 选择设备

步骤 6 参照表 5-1，配置标签名称、产品型号和子型号信息，并单击“保存”。

图5-5 填写标签信息

表5-1 参数说明

参数	必填/选填	说明
标签名称	必填	不超过 32 个字符。
产品型号	必填	第三方厂商在 Device partner 平台申请接入的设备产品 ID，限制 4 个字符。
产品子型号	选填	第三方厂商在 Device partner 平台申请接入的设备产品子 ID，限制两位十六进制数字。

步骤 7 单击“发布”，待人工审核通过后，可通过手机触碰 NFC 标签的形式实现免安装拉起原子化服务。

----结束

6 参考

- 交互流程
- JS FA 开发指导
- JS API 参考
- FA 开发指导
- Java API 参考