HarmonyOS 应用开发实验手册

华为技术有限公司

商标声明

 和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： http://e.huawei.com

目录

1 实验介绍

1.1 实验说明

1.1.1 关于本实验

本实验通过在 Windows 电脑上搭建 HarmonyOS 开发环境，并以三个应用场景实践作为演练，辅助理解掌握 HarmonyOS 应用的开发过程。

1.1.2 实验目的

- 掌握 HarmonyOS 应用开发环境搭建
- 掌握 HarmonyOS 应用开发基础知识
- 了解 HarmonyOS 分布式技术在实际场景中的应用。

1.1.3 实验规划

1. 开发环境配置。
2. 演练场景一：第一个 HarmonyOS 程序。

核心要点：环境配置，工程创建，编译运行等。

3. 演练场景二：基于分布式的简易备忘录程序。

核心要点：UI 开发，权限使用，数据库的使用，页面的分布式迁移

4. 演练场景三：基于分布式的多端协同程序。

核心要点：UI 开发，权限使用，事件通知，分布式任务调度，图像编解码

1.2 开发工具简介

HUAWEI DevEco Studio 是基于 IntelliJ IDEA Community 开源版本打造，面向华为终端全场景多设备的一站式集成开发环境（IDE），为开发者提供工程模板创建、开发、编译、调试、发布等 E2E 的 HarmonyOS 应用开发服务。通过使用 DevEco Studio，开发者可以更高效的开发具备 HarmonyOS 分布式能力的应用，进而提升创新效率。

作为一款开发工具，除了具有基本的代码开发、编译构建及调测等功能外，DevEco Studio 还具有如下特点：

图1-1 DevEco Studio 能力图

- 多设备统一开发环境：支持多种 HarmonyOS 设备的应用开发，包括手机（Phone）、平板（Tablet）、车机（Car）、智慧屏（TV）、智能穿戴（Wearable），轻量级智能穿戴（LiteWearable）和智慧视觉（Smart Vision）设备。
- 支持多语言的代码开发和调试：包括 Java、XML（Extensible Markup Language）、C/C++ 、 JS（JavaScript）、CSS（Cascading Style Sheets）和 HML（HarmonyOS Markup Language）。
- 支持 FA（Feature Ability）和 PA（Particle Ability）快速开发：通过工程向导快速创建 FA/PA 工程模板，一键式打包成 HAP（HarmonyOS Ability Package）。
- 支持分布式多端应用开发：一个工程和一份代码可跨设备运行，支持不同设备界面的实时预览和差异化开发，实现代码的最大化重用。
- 支持多设备模拟器：提供多设备的模拟器资源，包括手机、平板、车机、智慧屏、智能穿戴设备的模拟器，方便开发者高效调试。
- 支持多设备预览器：提供 JS 和 Java 预览器功能，可以实时查看应用的布局效果，支持实时预览和动态预览；同时还支持多设备同时预览，查看同一个布局文件在不同设备上的呈现效果。

1.3 实验设备介绍

1.3.1 模拟器

DevEco Studio 提供模拟器供开发者运行和调试 HarmonyOS 应用，对于 Phone、Tablet、Car、TV 和 Wearable 可以使用 Remote Emulator 运行应用。

（备注：需要登录华为账号认证）

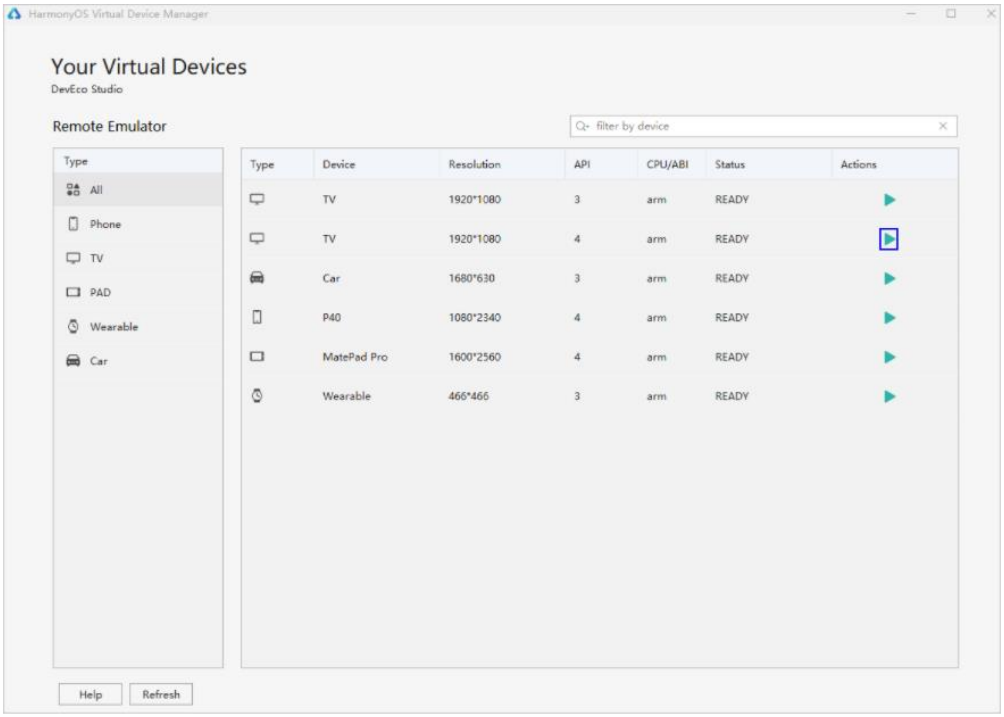

图1-2 模拟器示意图

1.3.2 真机

搭建 HarmongyOS 的设备。

本实验中为华为 P40。

1.4 实验准备

1.4.1 模拟器准备

表1-1 实验设备模拟器要求

预置条件	要求
华为帐号	已注册且完成实名认证
DevEco Studio	已安装且环境已配置完成

1.4.2 真机准备

表1-2 实验设备真机要求

预置条件	要求
华为账号	已注册且完成实名认证

DevEco Studio	已安装且环境已配置完成
真机设备	1、2台HarmonyOS设备，并获取到设备的UDID hdc shell bm get –u可获取UDID 2、完成组网 所有设备接入同一网络； 所有设备登录相同华为帐号； 所有设备上开启"设置->更多连接->多设备协同 "。
证书&Profile	1、密钥和证书请求文件。 2、AGC项目下已创建的HarmonyOS应用包名。 3、AGC中已申请的证书文件。 4、AGC中已添加设备列表（对应真机的UDID）。 5、AGC中已申请对应应用的Profile文件。

可参考：

HarmonyOS应用开发环境.docx

2 HarmonyOS 应用开发

2.1 工程创建

菜单栏选择 File > New > New Project 来创建一个新工程。

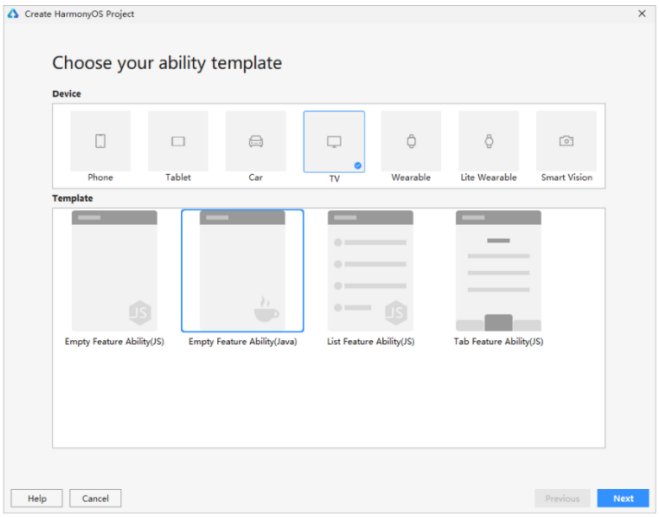

图2-1 新建工程

点击 Next，进入到工程配置阶段，需要根据向导配置工程的基本信息。

Project Name：工程的名称，可以自定义。

Package Name：软件包名称，默认情况下，应用 ID 也会使用该名称，应用发布时，应用 ID 需要唯一。

Save Location：工程文件本地存储路径。

Compatible API Version：兼容的 SDK 最低版本。

点击 Finish，工具会自动生成示例代码和相关资源，等待工程创建完成。

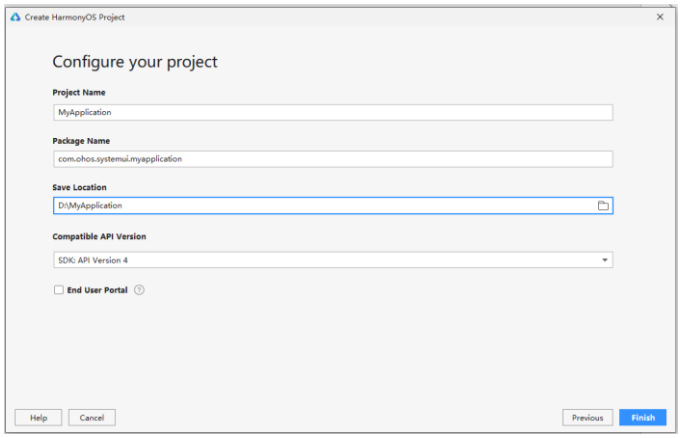

图2-2 工程配置

2.2 创建 Ability

选中对应的模块，点击鼠标右键，选择 New > Ability ，然后选择对应 Ability 模板。

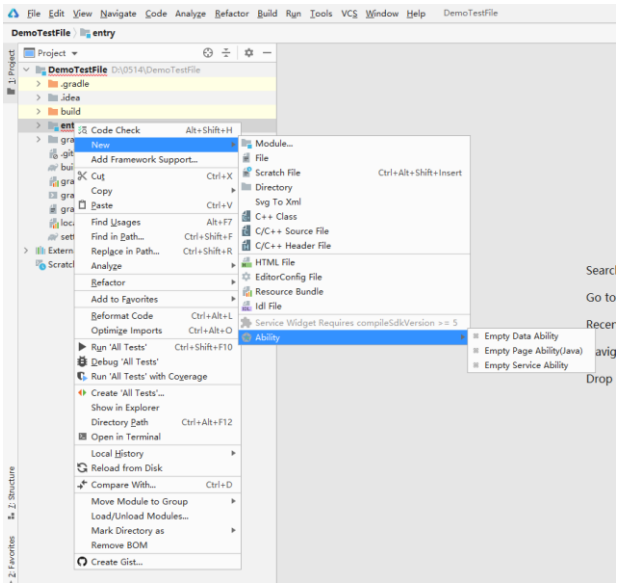

图2-3 Ability 创建

2.3 编译构建

2.3.1 不带签名信息的构建

注意：不带签名信息的 Hap 只能运行在模拟器上。

打开左下角的 OhosBuild Variants，检查并设置模块的编译构建类型（可选择 debug/Release ）。

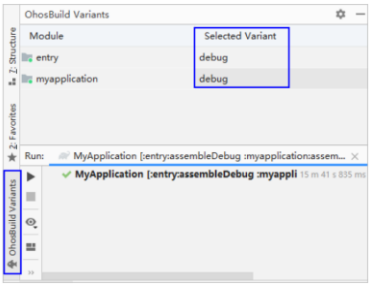

图2-4 构建配置

在主菜单栏，点击 Build > Build APP(s)/Hap(s) > Build Hap(s)，生成不带签名的 HAP。

2.3.2 带签名信息的构建

打开 File>Project Structure，在 Modules>entry（模块名称）>Signing Configs 窗口中 ，配置指定模块的签名信息。

Store File：选择密钥库文件，文件后缀为.p12。

Store Password：输入密钥库密码。

Key Alias：输入密钥的别名信息。

Key Password：输入密钥的密码。

SignAlg：签名算法，固定为 SHA256withECDSA。

Profile File：选择申请的 Profile 文件，文件后缀为.p7b。

Certpath File：选择申请的数字证书文件，文件后缀为.cer。

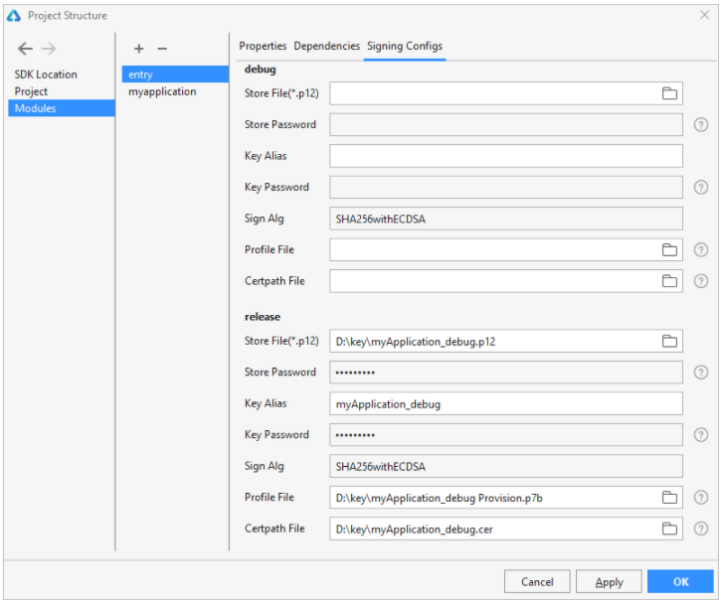

图2-5 签名配置

打开左下角的 OhosBuild Variants，检查并设置模块的编译构建类型（debug/release）。

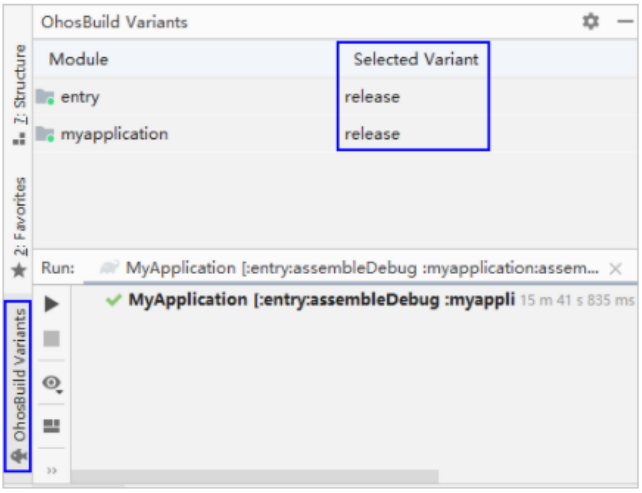

图2-6 Release 构建配置

在主菜单栏，点击 Build > Build APP(s)/Hap(s) > Build Hap(s)，生成已签名的 HAP。

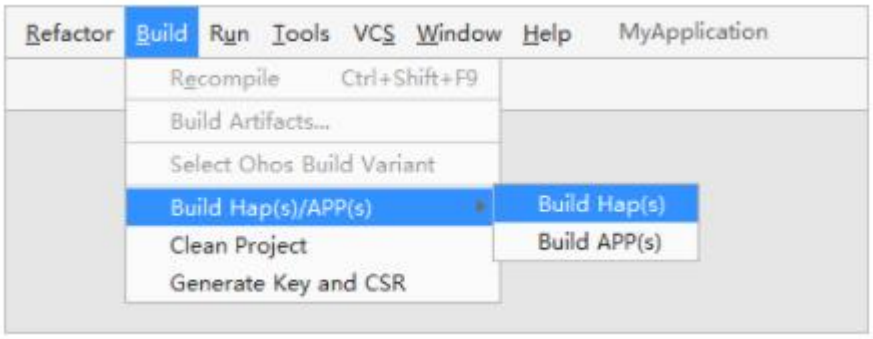

图2-7 编译构建

2.4 安装运行

2.4.1 模拟器安装运行

在 DevEco Studio 菜单栏，点击 Tools > HVD Manager。

在浏览器中弹出华为开发者联盟帐号登录界面，请输入已实名认证的华为开发者联盟帐号的用户名和密码进行登录。

登录后，请点击界面的允许按钮进行授权。

图2-8 模拟器登录

点击已经连接的 Remote Emulator 设备运行按钮，启动远程模拟设备（同一时间只能启动一个设备）。

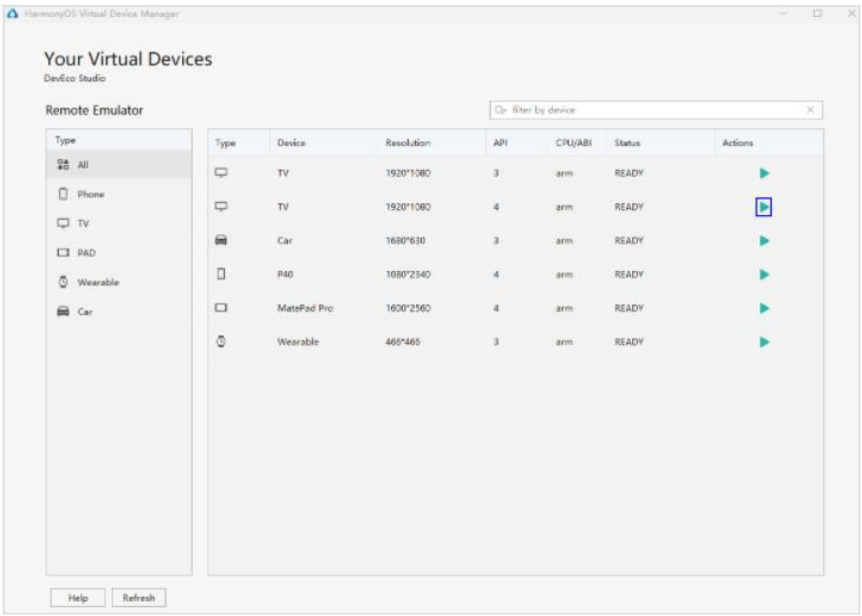

图2-9 选择模拟器运行

点击 DevEco Studio 的 Run > Run'模块名称'或使用默认快捷键 Shift+F10。

2.4.2 真机安装运行

通过 DevEco Studio 在真机设备上运行 HarmonyOS 应用时，需要在 AppGallery Connect 中申请调试证书和 Profile 文件，并对 HAP 进行签名后才能在真机设备上运行。

前提条件：

1. 需要提前打包带签名信息的 HAP。

2. 打开“开发者模式”，可在设置 > 关于手机/关于平板中，连续多次点击“版本号”，直到提示“您正处于开发者模式”即可。

操作步骤：

1. 使用 USB 方式，将 Phone 或者 Tablet 与 PC 端进行连接。

2. 在 Phone 或者 Tablet 中，USB 连接方式选择“传输文件”。

3. 在 Phone 或者 Tablet 中，打开设置 > 系统和更新 > 开发人员选项，打开“USB 调试”开关。

4. 在菜单栏中，点击 Run>Run'模块名称'或使用默认快捷键 Shift+F10 运行应用。

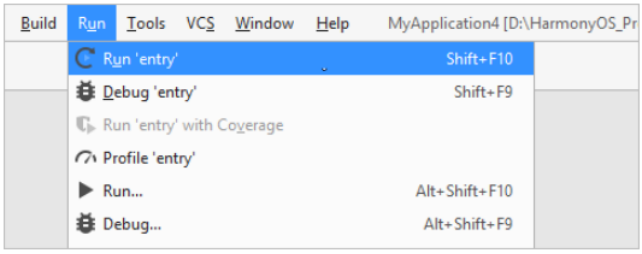

图2-10 模块运行

5. DevEco Studio 启动 HAP 的编译构建和安装。安装成功后，Phone 或者 Tablet 会自动运行安装的 HarmonyOS 应用。

3 第一个 HarmonyOS 程序

3.1 任务说明

运行第一个 HarmonyOS 程序。

在模拟器/真机上运行 Hello World 程序。

图3-1 任务一示意图

3.2 开发思路

先搭建好环境，创建 Demo 工程，在模拟器上运行正常后，给 Hap 包签名，再安装到真机上运行。

3.3 开发步骤

步骤 1 创建工程

1. 打开 DevEco Studio，在欢迎页点击 Create HarmonyOS Project，创建一个新工程。

2. 选择设备类型和模板，以 Phone 为例，选择 Empty Feature Ability(Java)，点击 Next。

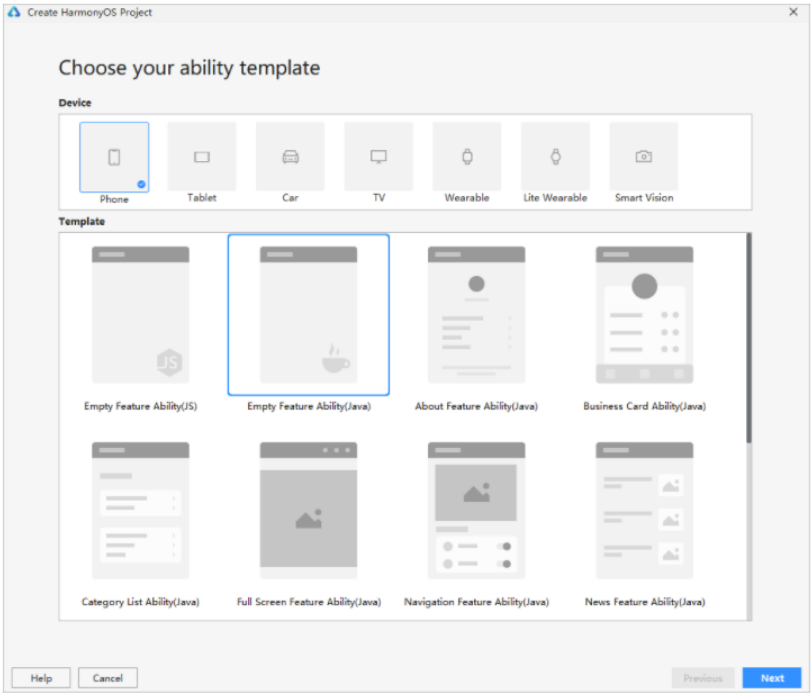

图3-2 任务一创建工程

3. 填写项目相关信息，保持默认值即可，点击 Finish。
4. 工程创建完成后，DevEco Studio 会自动进行工程的同步，同步成功如下图所示。

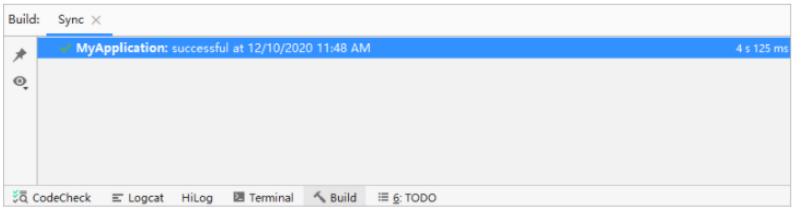

图3-3 任务一工程同步

步骤 2 编写界面

使用默认模板，不需要改动。

步骤 3 编译构建

参考 2.3

3.4 调试运行

调试运行（模拟器）

签名后在真机运行

参考 2.4

3.5 结果验证

查看应用显示，界面出现 Hello World

图3-4 运行结果

4 基于分布式的简易备忘录程序

4.1 任务说明

开发一个简易备忘录程序，包含以下参考界面及功能：（界面可不一样，功能需覆盖）

功能 1：首界面展示备忘录列表，界面可新建备忘录事项，点击已有单条备忘录可进入编辑界面进行编辑修改，长按单条可提示进行删除。

图4-1 任务二主界面示意图

#功能 2：编辑界面可修改内容区，点击”保存”可保存，点击“迁移”可选择目标设备，先把设备后可将编辑界面迁移到该设备继续编辑，完成后可迁回本地界面进行保存。

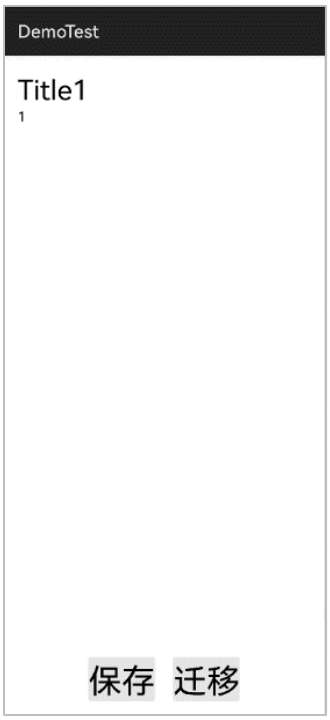

图4-2 任务二编辑界面示意图

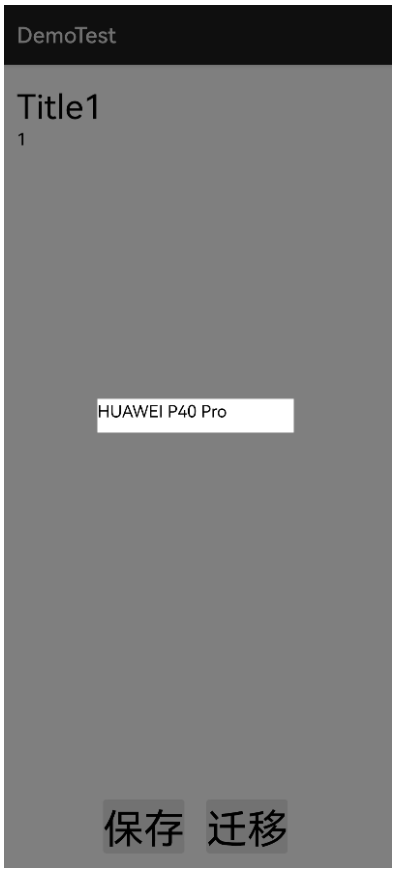

图4-3 任务二设备选择界面示意图

4.2 开发思路

1. 界面分析和设计

主界面由一个标题文本和一个 list 构成，list 每个 item 分标题和内容两行显示，列表与数据做绑定。

编辑界面由标题和内容构成，标题是一个普通文本控件，内容是一个可编辑文本，底部两个按钮。

2. 跨设备迁移

编辑界面可做迁移，因此该界面需要实现 Ability 相关的迁移接口，完成数据的迁移和生命周期管理。

跨设备需要申请对应的权限。

3. 数据处理

数据列表和每条数据可通过数据库保存及获取。

4.3 开发步骤

步骤 1 创建工程

1. 打开 DevEco Studio，在欢迎页点击 Create HarmonyOS Project，创建一个新工程。
2. 选择设备类型和模板，选择 Empty Feature Ability(Java)，点击 Next。

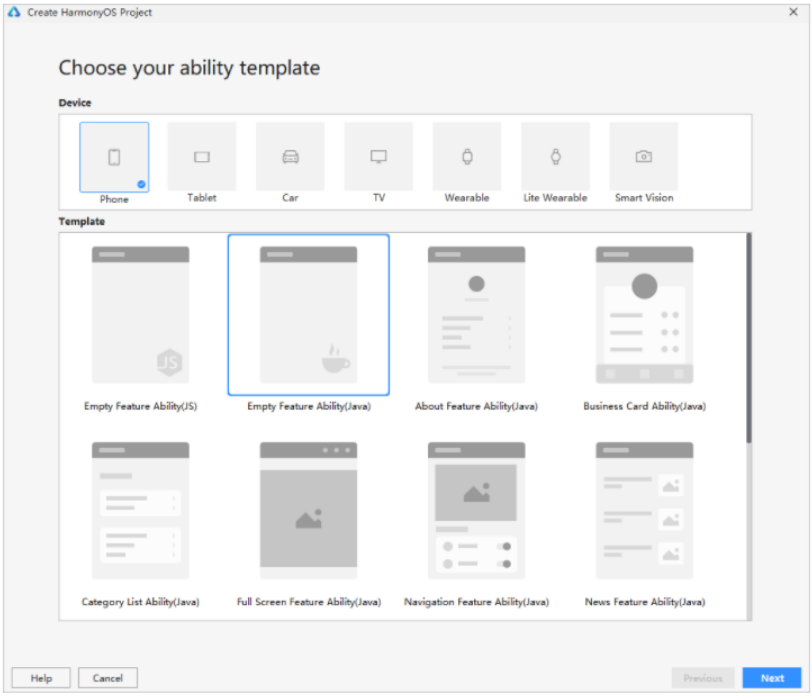

图4-4 新建工程

3. 填写项目相关信息，包名取名为 AGC 上申请的应用包名，点击 Finish。
4. 工程创建完成后，等待 DevEco Studio 自动进行工程的同步。

步骤 2 代码编写

新建两个界面 PageAbility 分别表示主界面和编辑界面

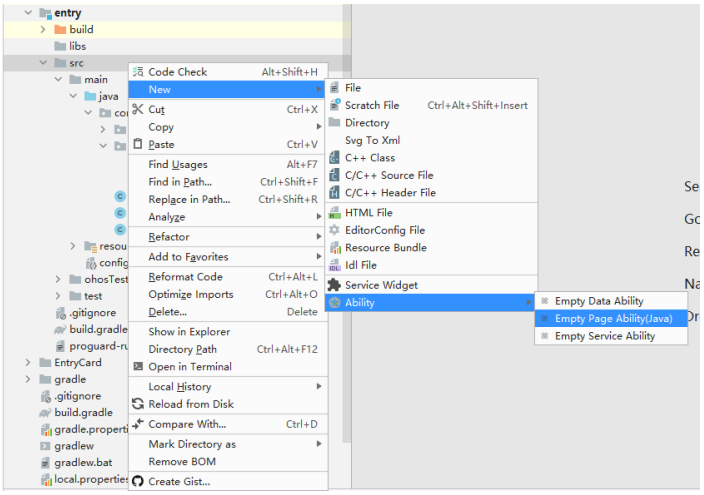

图4-5 新建 Aibility

配置文件中声明必要权限、配置界面信息

```
"reqPermissions": [
  {
    "name": "ohos.permission.GET_DISTRIBUTED_DEVICE_INFO"
  },
  {
    "name": "ohos.permission.DISTRIBUTED_DATASYNC"
  },
  {
    "name": "ohos.permission.DISTRIBUTED_DEVICE_STATE_CHANGE"
  },
  {
    "name": "ohos.permission.GET_BUNDLE_INFO"
  }
]
```

图4-6 权限配置

完成逻辑编写（界面实现、查询连接设备、迁移逻辑、数据管理逻辑等）

- 主界面

主要技术点：列表控件及响应、列表与数据绑定、Ability 跳转

```
private void initListContainer() {
    //主界面 list
    mlistContainer = (ListContainer) findComponentById(ResourceTable.Id_list_container);
    // 列表数据
    List<SampleItem> list = getData();
    //数据与列表绑定
    SampleItemProvider sampleItemProvider = new SampleItemProvider(list, this);
```

```
        mlistContainer.setItemProvider(sampleItemProvider);
        mlistContainer.setItemClickedListener(new ListContainer.ItemClickedListener() {
            @Override
            public void onItemClicked(ListContainer listContainer, Component component, int i, long l)
{
                //列表每个 item 点击事件
                SampleItem item = (SampleItem) listContainer.getItemProvider().getItem(i);
                startAbility(item.getmTitle());
            }
        });
        mlistContainer.setItemLongClickedListener(new ListContainer.ItemLongClickedListener() {
            @Override
            public boolean onItemLongClicked(ListContainer listContainer, Component component, int
i, long l) {
                //列表每个 item 长按事件
                SampleItem item = (SampleItem) listContainer.getItemProvider().getItem(i);
                showDeleteDialog(item.getmTitle());
                return true;
            }
        });
        mBtn = (Button) findComponentById(ResourceTable.Id_new_record);
        mBtn.setClickedListener(new Component.ClickedListener() {
            @Override
            public void onClick(Component component) {
                //新建按钮响应事件，Ability 跳转
                startAbility("");
            }
        });
    }
```

```
    @Override
    public Component getComponent(int i, Component component, ComponentContainer
componentContainer) {
        //每条 item 显示
        final Component cpt;
        if (component == null) {
            cpt = LayoutScatter.getInstance(mSlice).parse(ResourceTable.Layout_item_sample, null,
false);
        } else {
            cpt = component;
        }
        SampleItem sampleItem = mList.get(i);
        Text text = (Text) cpt.findComponentById(ResourceTable.Id_title);
        text.setText(sampleItem.getmTitle());

        Text content = (Text) cpt.findComponentById(ResourceTable.Id_content);
        content.setText(sampleItem.getmString());
```

```
        return cpt;
    }
```

```
    public void refreshList(List<SampleItem> list) {
        mList = list;
        //列表数据刷新
        notifyDataChanged();
    }
```

- 编辑界面

主要技术点：控件使用、设备获取、跨设备迁移、数据保存与恢复、轻量级偏好数据库数据管理。

```
    private void checkDevice() {
        // 通过 FLAG_GET_ONLINE_DEVICE 标记获得在线设备列表
        List<DeviceInfo> deviceInfoList =
DeviceManager.getDeviceList(DeviceInfo.FLAG_GET_ONLINE_DEVICE);
        if (deviceInfoList.size() < 1) {
            showTip(this, "无在网设备");
        } else {
            showDeviceChooser(deviceInfoList);
        }
    }
```

```
    private void showDeviceChooser(List<DeviceInfo> deviceInfoList) {
        // 设备 List 弹框，与设备列表绑定
        ListDialog dialog = new ListDialog(this);
        String[] names = new String[deviceInfoList.size()];
        for(int i = 0; i<deviceInfoList.size();i++) {
            names[i] = deviceInfoList.get(i).getDeviceName();
        }
        dialog.setItems(names);
        dialog.setOnSingleSelectListener(new IDialog.ClickedListener() {
            @Override
            public void onClick(IDialog iDialog, int i) {
                DeviceInfo info = deviceInfoList.get(i);
                try {
                    // 开始任务迁移
                    continueAbility();
                } catch (IllegalStateException | UnsupportedOperationException e) {
                }
                dialog.hide();
```

```
    @Override
    public boolean onSaveData(IntentParams intentParams) {
        // 任务迁移数据保存
        intentParams.setParam("title", mTitle.getText());
        intentParams.setParam("content", mDetail.getText());
        return true;
    }

    @Override
    public boolean onRestoreData(IntentParams intentParams) {
        // 任务迁移数据恢复
        mCacheKey = getIntentString(intentParams, "title");
        mCacheContent = getIntentString(intentParams, "content");
        return true;
    }

    private String getIntentString(IntentParams intentParams, String key) {
        Object value = intentParams.getParam(key);
        if ((value != null) && (value instanceof String)) {
            return (String) value;
        }
        return null;
    }
```

```
    private void saveRecord() {
        //轻量级偏好数据库保存数据
        Preferences preferences =
PreferencesHelper.getInstance().getPreference(getApplicationContext());
        preferences.putString(mTitle.getText(), mDetail.getText());
        preferences.flushSync();
        terminateAbility();
    }
```

3）编译打包

参考 2.3

步骤 3 编译构建

参考 2.3

4.4 调试运行

调试运行（模拟器）

签名后在真机运行

参考 2.4

4.5 结果验证

在应用中进行相关用例操作，完成 4.1 任务功能。

备注：场景二可参考相关 demo 代码 DemoTest.zip。

5 基于分布式的多端协同程序

5.1 任务说明

开发一个多端协同程序，类似以下界面：

图5-1 显示端主界面

图5-2 控制端主界面

显示端功能

1.展示一个图片列表。

2.可选择连接控制器设备。

3.连上后，支持由控制端切换选中的图片，选中的图片实时在底部大图区放大显示。

4.控制器选择 OK 后将图片文件保存。

控制端功能

1.可通过上下左右四个按钮切换选择显示端的图片列表。

2. 点击刷新后将选择的图片展示在对端新界面中。

5.2 开发思路

5.2.1 显示端

1. 界面分析和设计

界面或划分为 3 块：可滚动的列表，底部固定图片展示区，按钮区。

滚动列表可由 ScrollView 组件内部套表格布局完成。

2. 查找、连接、绑定控制端

3. 响应控制端的处理

4. 数据处理

控制端选择确认后保存文件。

5.2.2 控制端

1. 界面分析和设计

控制区和展示区，控制区显示功能按钮，展示区显示保存后的图片。

2. 查找、连接、绑定显示端服务
3. 向显示端发送指令
4. 数据处理。

5.2.3 服务绑定：

1. 显示端暴露服务 Service 用于控制端绑定。
2. 控制端绑定服务，持有服务代理 proxy。
3. 指令数据通过 proxy 传递。

内部使用事件通知。

5.3 开发步骤

步骤 1 创建工程

1. 打开 DevEco Studio，在欢迎页点击 Create HarmonyOS Project，创建一个新工程。
2. 选择设备类型和模板，选择 Empty Feature Ability(Java)，点击 **Next**。

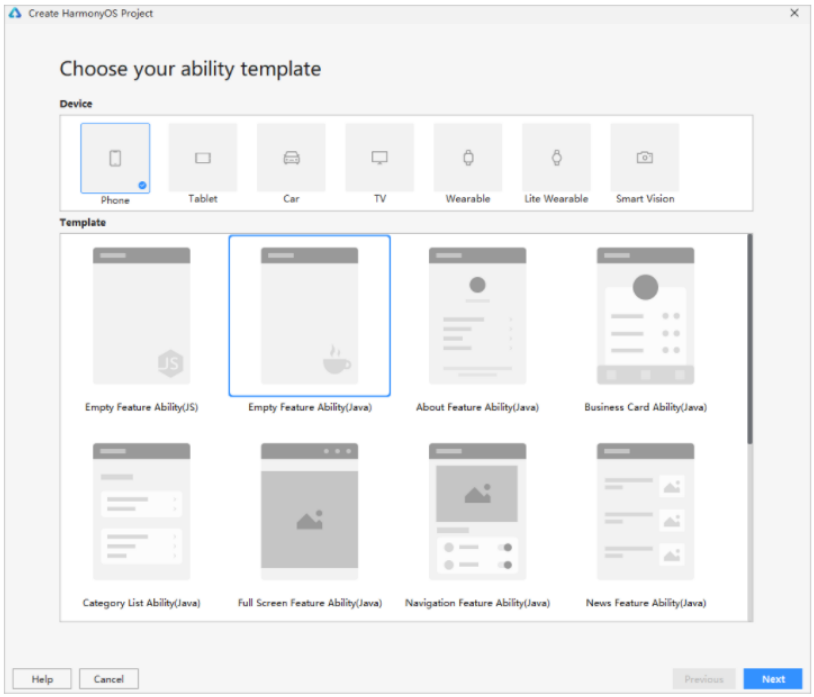

图5-3 新建工程

3. 填写项目相关信息，包名取名为 AGC 上申请的应用包名，点击 Finish。

4. 工程创建完成后，等待 DevEco Studio 自动进行工程的同步。

步骤 2 代码编写

1. 新建两个界面 PageAbility 分别表示显示界面和控制界面，新建一个 ServiceAbility 用于服务。

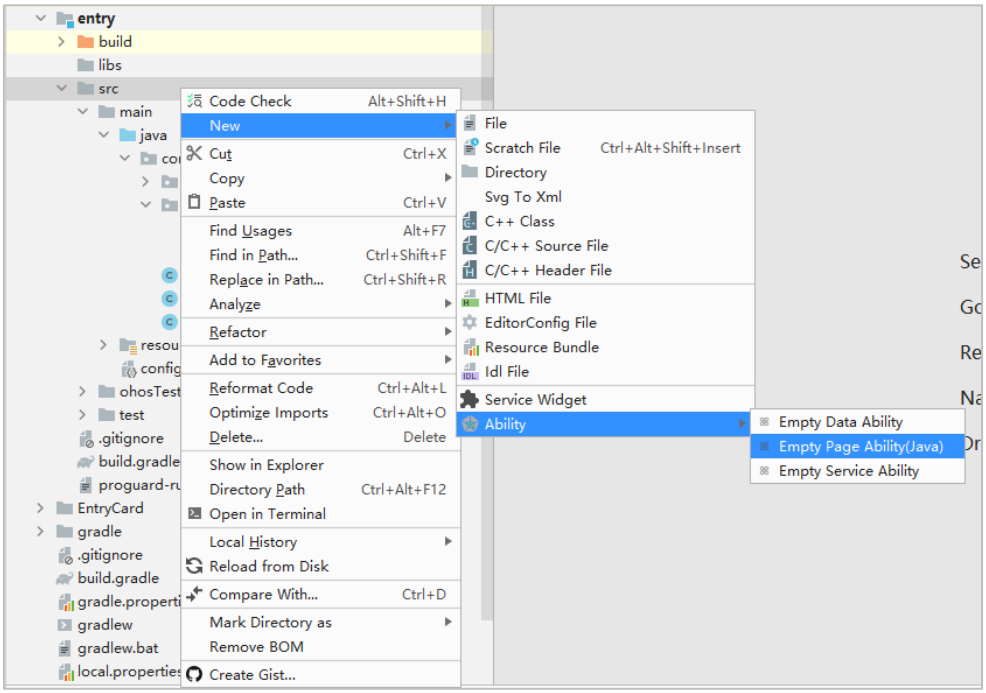

图5-4 创建 Ability

2. 配置文件中声明必要权限、配置界面信息

```
"reqPermissions": [
  {
    "name": "ohos.permission.DISTRIBUTED_DATASYNC",
    "reason": "多设备协同",
    "usedScene": {
      "ability": [
        ".MainAbility",
        ".ControlAbility",
        ".ServiceAbility"
      ],
      "when": "inuse"
    }
  },
  {
    "name": "ohos.permission.DISTRIBUTED_DEVICE_STATE_CHANGE"
    "reason": "获取设备状态变化",
    "usedScene": {
      "ability": [
        ".MainAbility",
        ".ControlAbility",
        ".ServiceAbility"
      ],
      "when": "inuse"
    }
  },
  {
    "name": "ohos.permission.GET_DISTRIBUTED_DEVICE_INFO",
    "reason": "获取设备基本信息",
    "usedScene": {
      "ability": [
        ".MainAbility",
        ".ControlAbility",
        ".ServiceAbility"
      ],
      "when": "inuse"
    }
  },
  {
    "name": "ohos.permission.GRT_BUNDLE_INFO",
```

图5-5 权限声明

3. 完成功能逻辑编写（界面实现、查询连接设备、绑定服务、指令传输、文件数据等）

显示端

主要技术点：控件使用、事件通知、图像编码解码、跨设备启动 FA

```
// 可滚动布局
    <ScrollView
        ohos:id="$+id:scrollview"
        ohos:height="0"
        ohos:weight="6"
        ohos:width="match_content">
// 表格控件
        <TableLayout
            ohos:id="$+id:tableLayout"
            ohos:height="match_content"
            ohos:width="match_content"
            ohos:row_count="1"
            ohos:column_count="2"
            ohos:orientation="horizontal">
// 填充表格元素
            <DirectionalLayout
                ohos:id="$+id:layout_movie_one"
```

```
                ohos:height="match_content"
                ohos:width="match_content"
                ohos:orientation="vertical"
                ohos:background_element="red"
                ohos:padding="5vp">

                <Image
                    ohos:height="match_content"
                    ohos:width="match_content"
                    ohos:id="$+id:image_one"
                    ohos:image_src="$media:wuxianerji"/>

                <Text
                    ohos:height="match_content"
                    ohos:width="match_content"
                    ohos:id="$+id:text_one"
                    ohos:text="无线耳机"
                    ohos:text_size="15fp"
                    ohos:layout_alignment="horizontal_center"/>

            </DirectionalLayout>
```

```
private void subscribe() {
//监听事件通知
    MatchingSkills matchingSkills = new MatchingSkills();
    matchingSkills.addEvent(Constants.UPDATE_EVENT);
    CommonEventSubscribeInfo subscribeInfo = new CommonEventSubscribeInfo(matchingSkills);
    mSubscriber = new MyCommonEventSubscriber(subscribeInfo);
    try {
        CommonEventManager.subscribeCommonEvent(mSubscriber);
    } catch (RemoteException e) {
        HiLog.error(LABEL_LOG, TAG, "subscribeCommonEvent occur exception.");
    }
}

private void unSubscribe() {
//取消事件通知监听
    try {
        CommonEventManager.unsubscribeCommonEvent(mSubscriber);
    } catch (RemoteException e) {
        HiLog.error(LABEL_LOG, "unSubscribe Exception");
    }
}

class MyCommonEventSubscriber extends CommonEventSubscriber {
    MyCommonEventSubscriber(CommonEventSubscribeInfo info) {
```

```
            super(info);
        }

        @Override
        public void onReceiveEvent(CommonEventData commonEventData) {
            //收到事件通知后的处理
            Intent intent = commonEventData.getIntent();
            int requestType = intent.getIntParam(Constants.REQUEST_TYPE_KEY, -1);
            HiLog.info(LABEL_LOG, "onReceiveEvent requestType = " + requestType);
            switch (requestType) {
                case Constants.REQUEST_MOVE:
                    int direction = intent.getIntParam(Constants.DATA_KEY, 0);
                    handleMove(direction);
                    break;
                case Constants.REQUEST_SAVE:
                    getContext().getGlobalTaskDispatcher(TaskPriority.DEFAULT)
                            .asyncDispatch(() -> handleSave());
                    break;
                default:
                    break;
            }
        }
    }
```

```
    private void handleSave() {
        PixelMap src = mPreviewImg.getPixelMap();
        // 文件存储
        String sdcardPath =
getApplicationContext().getExternalFilesDir(Environment.DIRECTORY_PICTURES)+File.separator +
Constants.FILE_NAME;;
        HiLog.info(LABEL_LOG, "handleSave to "+ sdcardPath);
        // 图像编码
        Util.writDataToFile(src, sdcardPath);
    }
```

```
    public void openRemoteAbility(String deviceId, String bundleName, String abilityName) {
        // 跨设备启动 FA
        Intent intent = new Intent();
        String localDeviceId = KvManagerFactory.getInstance()
                .createKvManager(new KvManagerConfig(this)).getLocalDeviceInfo().getId();
        intent.setParam("localDeviceId", localDeviceId);
        Operation operation = new Intent.OperationBuilder()
                // 设备目标设备、目标应用的 bundle 和 ability 名字
                .withDeviceId(deviceId)
                .withBundleName(bundleName)
                .withAbilityName(abilityName)
```

```
                .withFlags(Intent.FLAG_ABILITYSLICE_MULTI_DEVICE)
                .build();
        intent.setOperation(operation);
        startAbility(intent);
    }
```

控制端

主要技术点：跨设备服务绑定、图像解码

```
    @Override
    public void connectPa(Context context, String deviceId) {
        // 跨设备服务绑定
        if (deviceId != null && !deviceId.trim().isEmpty()) {
            Intent connectPaIntent = new Intent();
            Operation operation = new Intent.OperationBuilder()
                // 设备目标设备、目标应用的 bundle 和 ability 名字
                    .withDeviceId(deviceId)
                    .withBundleName(context.getBundleName())
                    .withAbilityName(ServiceAbility.class.getName())
                    .withFlags(Intent.FLAG_ABILITYSLICE_MULTI_DEVICE)
                    .build();
            connectPaIntent.setOperation(operation);
            conn = new IAbilityConnection() {
                @Override
                public void onAbilityConnectDone(ElementName elementName, IRemoteObject
remote, int resultCode) {
                    HiLog.info(LABEL_LOG,   "===connectRemoteAbility done");
                    proxy = new MyRemoteProxy(remote);
                }

                @Override
                public void onAbilityDisconnectDone(ElementName elementName, int resultCode) {
                    HiLog.info(LABEL_LOG, TAG, "onAbilityDisconnectDone.....");
                    proxy = null;
                }
            };
            context.connectAbility(connectPaIntent, conn);
        }
    }
```

```
        private void loadPreview() {
        String sdcardPath =
getApplicationContext().getExternalFilesDir(Environment.DIRECTORY_PICTURES)+File.separator +
Constants.FILE_NAME;;
        HiLog.info(LABEL_LOG, "updatePreview from " + sdcardPath + ",");
        // 图像解码
        PixelMap pixelMap = Util.readDataFromFile(sdcardPath);
        getUITaskDispatcher().asyncDispatch(() -> {
```

```
            mPreviewImg.setPixelMap(pixelMap);
        });
    }
```

服务绑定

主要技术点：服务 Ability、事件通知、RPC 通信

```
    private void sendEvent(int requestType, int direction) {
        HiLog.info(LABEL_LOG,   "sendEvent.....");
        try {
            Intent intent = new Intent();
            Operation operation = new Intent.OperationBuilder()
                    .withAction(Constants.UPDATE_EVENT)
                    .build();
            intent.setOperation(operation);
            if (requestType == Constants.REQUEST_MOVE) {
                // 指令传输
                intent.setParam(Constants.DATA_KEY, direction);
            }
            intent.setParam(Constants.REQUEST_TYPE_KEY, requestType);
            CommonEventData eventData = new CommonEventData(intent);
            // 事件通知
            CommonEventManager.publishCommonEvent(eventData);
        } catch (RemoteException e) {
            HiLog.error(LABEL_LOG,   "publishCommonEvent occur exception.");
        }
    }
```

```
    public int senDataToRemote(int requestType, Map paramMap) {
        MessageParcel data = MessageParcel.obtain();
        MessageParcel reply = MessageParcel.obtain();
        MessageOption option = new MessageOption(MessageOption.TF_SYNC);
        int ec = 1;

        try {
        // RPC 通信序列化数据
            if (requestType == Constants.REQUEST_SAVE) {
                data.writeInt(requestType);
                remote.sendRequest(requestType, data, reply, option);
            } else if (paramMap.get(Constants.DATA_KEY) instanceof Integer) {
                int direction = (int) paramMap.get(Constants.DATA_KEY);
                data.writeInt(requestType);
                data.writeInt(direction);
                remote.sendRequest(requestType, data, reply, option);
            }

            ec = reply.readInt();
```

```
            if (ec != ERR_OK) {
                HiLog.error(LABEL_LOG, TAG, "RemoteException:");
            }
        } catch (RemoteException e) {
            HiLog.error(LABEL_LOG, TAG, "RemoteException:");
        } finally {
            ec = ERR_OK;
            data.reclaim();
            reply.reclaim();
        }
        return ec;
    }
```

步骤 3 编译构建

参考 2.3

5.4 调试运行

调试运行（模拟器）

签名后在真机运行

参考 2.4

5.5 结果验证

在应用中进行相关用例操作，完成 5.1 任务功能。

备注：场景三可参考相关 demo 代码 DemoTestFile.zip。