ChatGPT

HarmonyOS 5 手机应用开发迁移指南

1. HarmonyOS 5 应用开发核心规范

ArkTS 编程语言

HarmonyOS 5 主要推行 **ArkTS（Ark TypeScript）** 语言作为应用开发的首选语言。ArkTS 基于 TypeScript 进行了扩展，具备静态类型检查和动态类型的灵活性，增加了声明式 UI 构建等新特性，让开发者能更简单直观地创建界面 [1]。与 Swift/Objective-C 不同，ArkTS 由方舟引擎运行，采用自动垃圾回收而非引用计数，对 iOS 开发者来说需要适应 JS/TS 风格的语法和异步编程模型。

Stage 应用模型

HarmonyOS 在 API 9+（HarmonyOS 3.1 起）引入了 **Stage 模型**，并将其作为主推应用模型。与早期的 FA（Feature Ability）模型不同，Stage 模型下所有应用组件共享同一个 ArkTS 引擎实例，减少内存占用并支持组件间状态共享 [2]。Stage 模型采用 **面向对象** 的设计，引入 `AbilityStage`、`WindowStage` 等类管理应用生命周期和窗口，使多窗口多设备场景下应用管理更灵活 [3]。Stage 模型仅支持 ArkTS 开发，是 HarmonyOS 5 及未来版本的推荐架构。开发者需编写入口 `ArkUI` 组件（通常继承自 `AbilityStage`），在应用启动时由系统加载。

ArkUI 用户界面框架

ArkUI 是 HarmonyOS 5 提供的全新声明式 UI 框架，配合 ArkTS 使用。ArkUI 采用类似 SwiftUI 的声明式范式，开发者通过描述 UI 状态来构建界面，框架自动根据状态变化刷新界面 [4]。ArkUI 提供丰富的 **内置组件**（文本、按钮、列表、布局容器等）和响应式状态管理（如 `@State` 装饰器）机制。**示例：** 以下 ArkUI 示例定义了一个简单页面，包含文字和按钮，点击按钮时输出日志：

```
// ArkUI 声明式 UI 示例（ArkTS）
@Entry @Component
struct MainPage {
  build() {
    Column() {
      Text("欢迎来到 HarmonyOS")
      Button("点击我").onClick(() => {
        console.info("按钮点击")
      })
    }
  }
}
```

ArkUI 的范式与 SwiftUI 十分相似，例如使用 `@State` 管理组件状态、使用链式语法设置样式等，熟悉 SwiftUI 的开发者可以较快上手 ArkUI。

Hvigor 构建工具

HarmonyOS 5 使用 **DevEco Hvigor** 作为项目构建工具。Hvigor 是基于 TypeScript 实现的构建任务编排工具，提供任务注册、工程配置、构建流程管理等功能 [5]。DevEco Studio 集成 Hvigor 自动执行构建任务，实现 HAP 模块与 APP 应用包的编译、打包、签名流程 [6]。Hvigor 将工程抽象为树状结构：根节点是项目，叶节点是各功能模块 [7]。构建分为初始化、编译、打包、签名等任务，Hvigor 通过插件和 `hvigorfile.ts` 脚本定义任务及其依赖关系，形成 DAG 有序执行 [8]。开发者既可在 DevEco Studio GUI 下构建，也可使用命令行 `hvigor assemble` 等指令，在不同环境下得到一致的构建结果 [9]。总体来说，Hvigor 类似于 Gradle，但以 TypeScript 脚本形式提供更灵活的扩展能力。

OHPM 包管理工具

HarmonyOS 5 提供 **OHPM（OpenHarmony Package Manager）** 来管理第三方库依赖。OHPM CLI 用于发布、安装和依赖管理 OpenHarmony 共享包（HAR/HSP 等）[10]。OHPM 的配置文件为 **oh-package.json5**，功能类似 Node.js 的 package.json，用于定义项目依赖项、脚本命令、项目信息等 [11] [12]。开发者可通过 `ohpm init` 初始化生成 oh-package.json5 [13]，在其中声明 `dependencies`（运行时依赖）、`devDependencies`（开发期依赖）等键值对来管理库版本 [14]。运行 `ohpm install` 时，OHPM 会解析项目级和各模块下的 oh-package.json5，下载安装所需依赖包及其**精确版本锁定**文件，确保不同环境构建一致性 [15]。OHPM 支持私有仓库（ohpm-repo）的搭建，可在本地或私有服务器部署共享库仓库，并通过 `ohpm config set registry <仓库地址>` 指定安装源 [16]。对于 iOS 开发者而言，OHPM 的作用类似于 CocoaPods/Swift Package Manager：统一管理依赖库，并简化集成过程。

应用签名流程

HarmonyOS 要求应用在构建时完成签名，以保证应用完整性和来源可靠 [17]。在 Stage 模型下，签名配置位于**工程级 build-profile.json5** 中，通过 `signingConfigs` 字段指定签名证书路径、签名材料等 [18]。DevEco Studio 提供自动签名选项，可一键生成调试证书并写入 build-profile.json5（每个开发者需本地生成）[19]。团队协作时建议**手动配置统一签名**：如将签名证书（.p12）和 profile 文件（.p7b）放在项目目录，build-profile.json5 中引用相对路径 [18]。HarmonyOS 的签名机制类似于 Android：开发者自备证书签名应用，上架应用市场时需提供签名信息用于校验。而 iOS 由 Apple 统一签名发行。需要注意，HarmonyOS Next 签名配置文件应纳入版本管理，避免团队成员签名路径不一致导致构建失败 [20] [18]。总的来说，iOS 开发者需从 Apple Provisioning Profile 模式转变为 **本地证书签名** 模式，并在项目配置中维护签名信息。

关键配置文件（app.json5、module.json5、build-profile.json5 等）

HarmonyOS 5 Stage 模型使用 JSON5 配置文件定义应用的元数据和构建信息：

- **app.json5：** 应用全局配置文件，包括应用包名（bundleName）、版本号、应用权限范围（如系统权限声明）、应用图标、应用名称 label 等全局信息。Stage 模型下，每个应用有且仅有一个 app.json5，描述应用级别属性。
- **module.json5：** 模块配置文件，相当于应用的子模块/子包描述。每个模块（例如入口模块 entry 或扩展能力模块）都有自己的 module.json5，包含模块名称、模块类型（如 ability 或 shell）、支持的设备类型列表、模块内 Ability 组件的定义、以及模块所需的权限声明等 [21] [22]。Stage 模型下 module.json5 是代替 FA 模型 config.json 的配置单元 [21]。开发者在其中的 `requestPermissions` 列表声明**需要申请的权限**（区分用户授权权限与系统权限）[23]。
- **build-profile.json5：** 构建配置文件，可有工程级和模块级两个层次。工程级 build-profile.json5 定义项目包含的模块列表（`modules` 字段）、构建选项（如编译器参数 arkOptions）、**签名配置 signingConfigs**、产品产物配置等 [24] [25]。模块级 build-profile.json5 可覆盖工程级的一些设置，常用于多模块分别配置。Build-profile.json5 是 Hvigor 构建的重要输入，Hvigor 初始化时读取它构建项目结构 [26]。

- **oh-package.json5：** 前述 OHPM 包依赖配置文件，列出项目级依赖的第三方 HAR 库版本、项目自身信息（name、version、description 等）以及脚本命令等 [27]。
- **其他配置：** 如 `local.properties`（本地环境配置，例如 SDK 路径）、`obfuscation-rules.txt`（代码混淆规则）等 [28]。如果使用了分布式数据或 FA 模型，可能还有 config.json 等，但在 Stage ArkTS 开发中主要使用上述 JSON5 文件。

iOS 开发者需要注意，HarmonyOS 的应用配置**分散在多个 JSON5 文件**中：`app.json5/module.json5` 的作用类似于 iOS 的 Info.plist（声明应用和模块的元数据），而 build-profile.json5 有些类似 Xcode 的 Build Settings + scheme 配置的综合，用于控制编译打包行为。

2. HarmonyOS 5 全部可用 Kit 和 API 模块

HarmonyOS 5 提供了丰富的系统 **Kit（套件）** 和对应 API 模块，涵盖 UI、网络、存储、设备、媒体、定位、传感器、通知、账号、支付、隐私安全等各方面能力。下面按功能分类列出主要 Kit 的用途、代表 API 示例（以 ArkTS 为主），并标明支持的最低 API Level。

UI Kit（ArkUI 图形界面）

功能与用途： UI Kit 提供构建手机界面的全部UI组件和接口，包括基础组件（文本、图像、按钮、列表等）、复杂组件（如地图组件、Web组件）、动画和布局能力，以及多窗口、多设备自适应布局等。ArkUI 支持声明式范式，开发者通过定义组件结构和状态来渲染界面。UI Kit 还包含样式和主题管理、国际化、本地化布局适配等功能，为应用提供一致的 UX 体验。

代表 API 示例（ArkTS）： 使用 ArkUI 框架构建界面。例如创建一个带有文本和按钮的页面：

```
// ArkUI 创建界面示例
@Entry @Component
struct HelloPage {
  @State count: number = 0;
  build() {
    Column({ alignItems: Alignment.Center }) {
      Text(`点击次数: ${this.count}`)
        .fontSize(20).margin(10)
      Button("点我一下").onClick(() => {
        this.count += 1;
      })
    }
  }
}
```

上述代码使用 ArkUI 的 `Column` 布局垂直排列 Text 和 Button，并通过 `@State` 管理点击次数状态，实现按钮点击时文本更新，体现了 ArkUI 声明式和响应式的特点。

支持 API Level： ArkUI（Stage 模型）在 HarmonyOS 3.1 (API 9) 引入 [29] 并持续增强，HarmonyOS 5 进一步丰富了组件库。UI 基础能力在 API 9+ 均受支持，更高版本提供了一些新组件和特性（如 API 17+ 引入新的图形能力等）。

网络 Kit（网络通信服务）

功能与用途： 网络 Kit 提供设备联网和通信的相关能力，包括 HTTP/HTTPS 网络请求、套接字通信、Wifi 和蜂窝网络状态管理等。开发者可以使用网络 Kit 的 API 访问 Web 服务、上传下载数据，以及监听网络连接变化。HarmonyOS 5 的网络模块支持常用协议及 TLS 安全通信，满足应用联网需求。

代表 API 示例（ArkTS）： 通过 HTTP 请求获取网络数据。有两种常用方式：一是使用全局 `fetch` 接口（类似 Web 标准），二是使用更底层的 `HttpRequest` API。下面示例展示使用 **HttpRequest** 进行 GET 请求：

```
import http from '@ohos.net.http';

let httpRequest = http.createHttp(); // 创建 HttpRequest 实例
httpRequest.request("https://api.example.com/data", {
  method: http.RequestMethod.GET
}).then(response => {
  if (response.responseCode === 200) {
    console.info("收到数据: " + response.result);
  }
}).catch(err => {
  console.error("请求失败: " + JSON.stringify(err));
});
```

在 ArkTS 中，通过 `@ohos.net.http` 模块的 `createHttp()` 获取 HttpRequest 对象，然后调用 `.request(url, options)` 即可发起异步请求 [30] [31]。该方法返回 Promise，可使用 `then/catch` 获取结果。HarmonyOS 还支持更高级的网络库封装，如将常用 GET/POST 封装成工具方法 [32]。**注意：**应用需要在 `module.json5` 中声明网络权限 `ohos.permission.INTERNET`，否则无法访问网络 [33]。

支持 API Level： 网络请求基础能力自 HarmonyOS 1.0 起就支持（API 3+），ArkTS 环境下 `@ohos.net.http` 模块在 API 7+ 可用。多数网络接口在 HarmonyOS 5 (API 15+) 上表现更完善，支持 HTTP2 等特性。开发者应参考对应 API 版本的文档确保兼容性。

存储 Kit（数据存储服务）

功能与用途： 存储 Kit 提供设备本地和分布式的数据存储能力，包括文件系统访问、数据库读写、键值存储、偏好设置等。开发者可以使用存储接口对应用沙盒目录中的文件进行**创建、读取、写入、删除**等操作 [34]。对于轻量数据，HarmonyOS 提供 **Preferences 用户首选项** 用于键值对存储，类似 iOS 的 UserDefaults，用于保存配置、状态等 [35]。此外，存储 Kit 下还有关系型数据库（`ohos.data.rdb`）接口，方便存储结构化数据。HarmonyOS 5 在存储安全方面也增强，例如提供 **加密存储** 和 **关键数据保护** 功能（见隐私安全部分）。

代表 API 示例（ArkTS）： 1. **文件存储：** 使用文件管理API读取文件。例如读取应用文档目录下的文本文件：

```
import file from '@ohos.file.fs'; // 文件系统基础API
const fs = file.openSync('/data/storage/base/file.txt', file.OpenMode.READ);
let content = fs.readSync(); // 读取文件内容
fs.closeSync();
console.info("文件内容: " + new TextDecoder().decode(content));
```

上述代码通过 `@ohos.file.fs` 模块提供的文件系统接口打开、读取文件 [36]。

1. **用户首选项（Preferences）：** 用于保存简单的键值数据。例如保存用户设置：

```
import preferences from '@ohos.data.preferences';
const pref = preferences.getPreferences("/data/storage/base/myPrefs.preferences");
pref.put("username", "Alice");
pref.flush(); // 刷新将数据写入磁盘
let user = pref.get("username", "<default>");
console.info("读取用户名: " + user);
```

这里使用 `ohos.data.preferences` 模块获取 Preferences 实例，对其调用 `put`、`get` 方法存取键值对。`flush()` 用于确保数据持久化到文件。

支持 API Level： 基础文件操作在 API 3+ 即受支持 [37]。Preferences 首选项存储在 API 7 起提供（HarmonyOS 2+），在 API 18 引入了新的 GSKV 存储引擎可选 [38]。分布式数据管理（如分布式 KV 数据库）需要更高 API 版本和设备支持。HarmonyOS 5 对存储API的支持已经相当成熟，在手机上本地存储相关API均可直接使用。

设备 Kit（设备信息与管理）

功能与用途： 设备 Kit 提供查询和管理设备属性的能力，包括设备基本信息（型号、设备ID、操作系统版本等）、系统状态（电量、电池健康、存储容量）、设备设置控制等。开发者可以通过设备信息 API 获取终端设备的各种标识和状态参数，用于优化应用在不同硬件上的表现或实现设备相关的功能。此外，Device Kit 也涵盖了一些设备控制接口，例如调节屏幕亮度、获取网络类型、监听设备方向改变等（部分功能可能在其他子模块，如传感器或系统界面 kit 中提供）。

代表 API 示例（ArkTS）： 获取设备基本信息：

```
import deviceInfo from '@ohos.deviceInfo';
console.info("设备型号: " + deviceInfo.deviceModel);
console.info("产品名称: " + deviceInfo.productBrand);
console.info("HarmonyOS 版本: " + deviceInfo.osFullName);
```

上述代码使用 `@ohos.deviceInfo` 模块直接读取全局导出的设备信息属性，如 `deviceModel`（型号）、`productBrand`（品牌）、`osFullName`（操作系统名称和版本）等 [39]。这些属性由系统提供，应用可只读访问。值得注意的是，新接口 `ohos.deviceInfo` 自 API 6 起可用，替代了早期的 `system.device` 接口 [40]。

支持 API Level： 设备信息查询接口自 API 6（HarmonyOS 2.0）开始支持 [39]。HarmonyOS 5 (API 15+) 提供更丰富的设备参数和统一的设备管理接口。大多数设备属性接口在手机上都是向后兼容的，但部分新增字段可能需要较新 API。开发者应根据应用最低平台选择相应的接口进行调用。

媒体 Kit（多媒体服务）

功能与用途： 媒体 Kit 包含音频、视频和相机等多媒体相关的功能接口。主要包括：**音视频播放**（通过 AVPlayer 等播放引擎播放本地或流媒体内容）、**音频录制与播放**（麦克风录音、媒体文件录制，音频管理）、**摄像头**（拍照、录像、相机参数控制）、**媒体编解码**（如果需要处理媒体流）等。HarmonyOS 5 提供的媒体框架允许开发者轻松地在应用中加入多媒体功能，比如音乐播放、视频播放、扫码摄像、拍照上传等。

代表 API 示例（ArkTS）：

- **音视频播放（AVPlayer）：** Harmony 提供了 AVPlayer 类用于完整的媒体播放功能 [41]。例如播放网络视频：

```
import media from '@ohos.multimedia.avplayer';
let player = media.createAVPlayer();
player.src = "https://example.com/video.mp4"; // 设置媒体源
player.on('stateChange', (state) => {
 console.info("播放状态: " + state);
});
player.play(); // 开始播放视频
```

在上述代码中，首先通过 `createAVPlayer()` 创建播放器实例，设置 `src` 属性为视频URL，监听播放状态变化，然后调用 `play()` 播放。AVPlayer 支持本地文件路径或流媒体 URL，提供状态、进度等事件回调，功能类似于 iOS 的 AVPlayer [42]。

- **相机拍照（Camera Kit）：** HarmonyOS 提供 Camera Kit 接口控制相机硬件。基本流程是：获取相机设备列表，创建相机会话，设置预览组件，捕获图像。简单示例：

```
import camera from '@ohos.multimedia.camera';
const cameraManager = camera.getCameraManager();
let cameras = cameraManager.getCameras(); // 获取可用相机
let cameraDevice = cameras[0];
cameraDevice.openCamera((err, cameraObj) => {
 if (!err) {
  cameraObj.takePhoto({
   quality: 100
  }, (error, image) => {
   if (!error) {
    console.info("拍照成功，图片路径: " + image.uri);
   }
  });
 }
});
```

以上伪代码展示了打开第一个摄像头并拍照，将照片保存的过程（实际需指定输出文件等参数）。Camera Kit 还可控制录像、对焦、变焦等。

支持 API Level： 多媒体能力在不同 API 版本陆续引入： - AVPlayer 播放器 API 在 API 9+ (HarmonyOS 3) 开始提供全面支持，HarmonyOS 5 对其进行了性能优化，支持外挂字幕、倍速播放等（部分特性在 API 15+ 提供）。 - Camera Kit 在 API 5+ 即有基础支持（HarmonyOS 2.0 相机），在 API 9 之后提供 ArkTS 接口的相机服务能力 [43]。 - 音频播放与录制在 API 3+ 就存在（媒体基础能力），API 9+ ArkTS 增加了对这些接口的封装。
总体而言，HarmonyOS 5 的媒体Kit已经非常成熟，大部分手机多媒体功能在 API 15 及以上都可用。

定位 Kit（位置服务）

功能与用途： 定位 Kit 提供设备的定位能力，包括获取**实时的精确位置**、获取**最近位置记录**、订阅位置变化、地理围栏等功能 [44]。应用可通过位置服务获取经纬度，用于地图导航、位置分享、附近服务等场景。HarmonyOS 的定位服务支持 GPS、网络定位等，并可设置定位精度、场景（导航、高精度、节能等）以平衡耗电和精度 [45]。还提供地理围栏接口，当设备进入或离开特定区域时触发通知 [46]。

代表 API 示例（ArkTS）： 获取当前设备位置：

```
import geolocation from '@ohos.location';

// 请求当前位置（异步返回 Promise）
geolocation.getCurrentLocation().then(loc => {
  console.info(`当前位置: 纬度${loc.latitude}, 经度${loc.longitude}`);
}).catch(err => {
  console.error("定位失败: " + err);
});

// 或订阅位置变化
geolocation.subscribe({
    scenario: geolocation.LocationRequestScenario.NORMAL,
    priority: geolocation.LocationRequestPriority.ACCURACY_FIRST
  },
  (location) => {
    console.info("位置更新: lat=" + location.latitude + ", lon=" + location.longitude);
  }
);
```

此示例使用 `@ohos.location` 模块的 `getCurrentLocation()` 方法获取一次当前位置，以及 `subscribe()` 方法持续监听位置变化 [45]。可以指定定位场景（如 NORMAL 普通、NAVIGATION 导航场景）和优先级（高精度优先或低功耗优先）等参数 [45]。在调用前需确保在 `module.json5` 中声明了相应的定位权限（`ohos.permission.LOCATION` 等），并在运行时请求用户授权。

支持 API Level： 位置服务基础功能自 API 3+ 就存在（HarmonyOS 内核集成了定位服务），ArkTS 接口在 API 9 起提供更易用的方法。API 10+ 增加了定位场景和优先级设置，API 15 以后性能和精度进一步提升。一般获取单次位置在所有版本都支持，但例如地理围栏等高级功能可能需要 API 13+。开发者应参考各 API 接口的版本注明。

传感器 Kit（传感器服务）

功能与用途： 传感器 Kit 提供对设备各种传感器的访问接口，包括加速度计、陀螺仪、磁力计、陀螺仪、光传感器、距离传感器、气压计等。开发者可以查询设备支持的传感器列表，并**订阅传感器数据**来获取连续读数，或进行一次性读取 [47]。传感器服务允许设置数据采样频率（上报周期）、获取传感器动态变更（如传感器上下线）等。常见的应用场景如计步器（订阅加速度传感器）、指南针（使用磁力传感器和陀螺仪）、摇一摇等，都可以通过传感器 Kit 实现。

代表 API 示例（ArkTS）： 持续监听加速度传感器数据：

```
import { sensor } from '@kit.SensorServiceKit';

sensor.getSensorList((err, sensors) => {
  if (!err) {
    console.info("设备支持的传感器: " + sensors.map(s => s.name).join(", "));
  }
});

// 持续订阅加速度计数据，100ms 上报一次
sensor.on(sensor.SensorId.ACCELEROMETER, (data: sensor.AccelerometerResponse) => {
  console.info(`加速度: x=${data.x.toFixed(2)}, y=${data.y.toFixed(2)}, z=${data.z.toFixed(2)}`);
}, { interval: 100000000 }); // 纳秒为单位，这里100ms
```

首先通过 `sensor.getSensorList` 获取传感器列表 [48]。然后调用 `sensor.on(sensorId, callback, options)` 订阅某传感器数据 [49]。上述传入 `sensor.SensorId.ACCELEROMETER` 以及 `interval` 参数表示以100毫秒周期获取加速度数据。数据通过回调异步返回。若只需获取一次数据，可使用 `sensor.once` 方法 [50]。停止监听则调用 `sensor.off`。使用传感器前需在应用配置中声明相应权限（如运动健康类传感器可能需要 `BODY_SENSOR` 权限），并在运行时确保设备支持该传感器。传感器 Kit 还涵盖马达振动功能，可通过 vibrator 模块控制设备振动 [51]。

支持 API Level： 传感器订阅接口自 API 6 开始提供 [39]。HarmonyOS 5 (API 15+) 的传感器服务更完善，支持更多类型传感器以及跨设备获取（在分布式场景下）。加速度、陀螺仪等常见传感器在早期版本即可使用，但例如环境光、气压等在旧设备/旧版本上可能不支持。总体来说，API 7+ 的设备都支持主要传感器接口。

通知 Kit（用户通知服务）

功能与用途： 通知 Kit 提供应用在系统通知中心发布通知消息的能力。包括**本地通知**的发布、更新和移除，通知渠道的创建和管理、通知授权状态查询等 [52]。开发者可以使用通知服务在状态栏向用户推送提醒信息，例如聊天消息通知、待办事项提醒等。HarmonyOS 5 的通知 Kit 允许丰富的通知样式（大文本、图片、动作按钮等）以及与系统**通知栏的交互**。需要注意的是，从隐私出发用户需要对应用通知授予权限后才能展示通知。

代表 API 示例（ArkTS）： 发布本地通知流程：

```
import notification from '@ohos.notification';

// 1. 申请通知发布权限（弹框请求用户授权）
notification.requestEnableNotification().then(granted => {
  if (!granted) {
    console.warn("用户未授权通知权限");
  }
});

// 2. 创建通知请求并发布
let noticeRequest = {
  content: {
    contentType: notification.ContentType.NOTIFICATION_CONTENT_BASIC_TEXT,
    normal: {
      title: "提醒", text: "您有一条新消息"
    }
```

```
  },
  slotType: notification.SlotType.SERVICE_INFORMATION
};
notification.publish(noticeRequest, (err) => {
  if (!err) {
    console.info("通知已发布");
  }
});
```

首先调用 `requestEnableNotification()` 接口，请求用户授权发送通知 [53]。该接口会触发系统弹窗询问用户同意与否 [54]。之后构造通知内容并调用 `notification.publish()` 发布通知。发布时可以指定通知 `slotType`（类别）以应用不同的重要级别、声音等设置（类似 iOS 的通知分组和设置）。通知内容支持富文本、长文本、大图等。开发者也可通过 `notification.subscribe` 订阅通知点击事件，将用户点击通知与应用内部页面跳转关联起来 [55]。

支持 API Level： 通知框架在 API 3+ 就已存在。API 8 引入通知槽（Channel）机制，API 10+ 优化了通知权限管理，需要应用显式请求授权后才能发布通知 [53]。HarmonyOS 5 对通知进行了增强，API 15+ 支持更加丰富的通知样式以及**通知扩展服务**（允许后台处理通知交互，例如通知按钮点击等）。一般的通知发布接口在 API 7+ 即可使用，但为了兼容新版权限机制，建议应用最低兼容 API 10 及以上的通知授权模型。

账号 Kit（华为账号服务）

功能与用途： 账号 Kit 提供华为账号登录及相关用户信息获取的能力。开发者可以集成 Account Kit 来让用户使用华为帐号一键登录应用，并获取经过用户授权的个人信息（昵称、头像等）。Account Kit 简化了第三方登录流程，提供现成的 **登录按钮组件** 和 **登录面板组件** 供应用使用 [56]。此外，Account Kit 还能获取 Access Token、Refresh Token 以调用其他华为服务，以及支持未成年人账号保护模式等 [57]。

代表 API 示例： 使用内置的华为帐号登录按钮组件进行登录：

```
import { LoginPanel, loginComponentManager } from '@ohos.account.accountKit';

// 在ArkUI界面中直接使用 LoginPanel 组件（假设在ArkUI的build函数中）
LoginPanel({ appId: 'YOUR_APP_ID' });

// 或通过代码触发登录流程
loginComponentManager.login(() => {
  console.info("登录成功");
}, (errCode) => {
  console.error("登录失败，错误码: " + errCode);
});
```

上面代码展示了两种方式：一种是在界面中嵌入华为账号的 Panel 登录界面组件或 Button 组件 [56]，用户点击后会调起华为帐号登录授权；另一种是通过 `loginComponentManager.login()` 以编程方式启动登录流程并处理结果回调。使用 Account Kit 前需要在华为开发者联盟申请 AppID 并配置 OAuth 信息，然后在应用中引用 Account Kit 模块。成功登录后可通过 Account Kit API 获取 `AccountInfo`（包含用户名、OpenID等）。

支持 API Level： Account Kit 属于应用服务层的能力，一般在 HarmonyOS 2.x 起就通过 HMS Core 提供支持。在 HarmonyOS 4.0+ 集成到了系统应用服务中。ArkTS Account Kit 组件在 API 10+ 可用。需要注意

Account Kit 实际依赖于 HMS Core 服务环境，确保设备装有 HMS 服务或在鸿蒙手机上开启华为帐号服务。对于大部分 HarmonyOS 5 手机，该服务默认可用，API 11+ 上使用体验最佳。

支付 Kit（支付服务）

功能与用途： 支付 Kit（鸿蒙支付服务）提供在应用中集成华为支付能力的接口，包括**基础支付**（安全调用收银台完成支付）、**签约代扣**（如订阅服务定期扣款）、**合单支付**（多个商品合并支付）等 [58]。通过支付 Kit，应用（通常是商户类应用）可以快速调用系统级的支付界面，处理用户的付款流程，而无需从零实现支付协议。HarmonyOS 5 的支付服务支持银行卡、Huawei Pay、数字人民币等多种支付方式，确保支付过程安全便捷。

代表 API 示例（ArkTS）： 发起基础支付流程：

```
import { paymentService } from '@ohos.payment'; // 引入支付服务模块

let orderInfo = {
 // 假设已从服务端获取的订单信息字符串
 orderAmount: "9.99", productDesc: "高级会员月卡", orderId: "123456"
};
paymentService.pay(orderInfo, (err, result) => {
 if (err) {
  console.error("支付失败: " + err.code);
 } else {
  console.info("支付成功，流水号: " + result.tradeId);
 }
});
```

以上代码使用 `paymentService.pay` 接口，传入订单信息和回调，即可拉起华为安全支付界面让用户完成支付 [58]。支付 Kit 自动处理支付结果，回调中 `result` 包含交易流水等信息。若需使用数字人民币等，还可调用 `ecnyPaymentService` 模块提供的专用接口 [59]。应用接入支付 Kit 前需在华为 AppGallery Connect 配置支付能力并获取参数（如商户ID），代码中订单信息通常由服务端签名生成确保安全。

支持 API Level： Payment Kit 在 HarmonyOS 4.1.0(API 11) 引入 [58] 并在之后版本扩展。API 11+ 的设备支持基础支付功能，API 16 起新增 Reader Kit（电子书阅读支付相关）等特定场景能力 [60]。使用支付 Kit 时，也需要设备上 HMS Core 支持（支付服务属于 HMS 功能的一部分）。一般HarmonyOS 5手机（API 15+）上支付 Kit 均可正常使用。

隐私与安全 Kit（隐私与设备安全）

功能与用途： HarmonyOS 非常重视用户隐私和应用安全。隐私与安全相关的 Kit 包括**权限管理**、**安全数据存储**、**设备认证**等模块：

- **权限管理：** 提供 API 查询应用权限授予状态、请求权限，以及在配置文件中声明权限的机制。HarmonyOS 权限分为「系统授权权限」和「用户授权权限」两类：系统权限（普通权限）只需在 module.json5 的 `requestPermissions` 列表声明即可自动授予，不需用户确认；用户权限（敏感权限，如定位、相机）则必须在**运行时调用接口请求用户授权** [23]。开发者可以通过 `requestPermissions([])` 方法弹框请求用户许可 [61]。权限管理 Kit 确保未经用户同意，应用无法获取敏感数据。

- **安全存储 (Asset Store Kit)：** 提供保护关键数据的存储服务，例如**密钥、密码**等敏感信息的加密存储。Asset Store Kit 支持创建**关键资产**（如私钥）、更新和删除等操作，并可结合可信执行环境确保数据安全 [62]。开发者使用 ArkTS 接口如 `addAsset(key, data, options)` 来存放数据，这些数据存储于系统安全区域，防止未经授权访问。

- **设备安全与认证 (Device Security Kit)：** 包含诸如 **TrustedAppService**（可信应用证明）[63] 等接口，为应用提供设备可信状态证明、应用完整性校验等功能。例如 TrustedAppService 可以创建和销毁证明密钥、初始化证明会话并获取设备的安全位置信息，用于一些对安全要求极高的场景（如数字版权、金融交易）。

代表 API 示例： 在应用启动时检查并请求用户敏感权限，例如定位：

```
if (!device.hasLocationPermission()) {
 // 假设有自定义封装的权限检测方法
 const permission = 'ohos.permission.LOCATION';
 const result = await permissionRequestor.requestPermissions([permission]);
 console.info("定位权限授权结果: " + result);
}
```

在 `module.json5` 中需提前声明：

```
"requestPermissions": [
 {
  "name": "ohos.permission.LOCATION",
  "reason": "获取位置信息用于导航",
  "usedScene": { "ability": ["com.example.MyAbility"] }
 }
]
```

上述 JSON5 片段声明了需要 **用户授权** 的定位权限及用途。在运行时，应用调用请求权限的 API 弹窗让用户同意，正如上面的伪代码所示 [22]。

另一个示例，使用 Asset Store Kit 保存密码：

```
import asset from '@ohos.security.asset';

let handle = asset.openAsset("mySecret", true); // 打开或创建名为"mySecret"的安全存储
asset.add(handle, "password", new TextEncoder().encode("123456")); // 保存敏感数据
let data = asset.get(handle, "password");
console.info("读取密码密文长度: " + data.byteLength);
asset.close(handle);
```

Asset Store Kit 会将“password”键对应的数据加密保存在受信区域，确保即使应用被逆向也难以提取明文。

支持 API Level： 权限管理机制在 API 7+ 开始区分普通/敏感权限，API 10 之后要求敏感权限必须动态申请 [23]。HarmonyOS 5 进一步完善了权限细粒度和用户提示。Asset Store Kit 是较新的能力，在 HarmonyOS 3.2(API 10) 及以上提供，HarmonyOS 5 对其功能完善并整合到安全中心。可信应用证明等高级安

全服务一般在 API 9+ 才引入。整体而言，隐私安全相关Kit需要较新的系统支持，建议在 API 10 及以上的设备上使用，以获取完备的安全特性。

3. iOS 开发者迁移 HarmonyOS 5 需要适应的内容

从 iOS 转向 HarmonyOS 开发，涉及语言、框架、工具链等多方面的改变。下面总结了 iOS 开发者需要学习和适应的重点：

编程语言迁移：Swift/Obj-C → ArkTS

语言范式差异： iOS 主要使用 Swift（或部分遗留项目用 Objective-C），而 HarmonyOS Stage 模型采用 ArkTS。Swift 是强类型编译语言，语法简洁、安全（有 Optionals、ARC 内存管理等）；ArkTS 则是动态语言 JavaScript/TypeScript 的超集，拥有灵活类型和动态特性，运行在 Ark 引擎上自动垃圾回收。开发者需要适应从 Swift 的面向协议/值语义，转变为 ArkTS 的原型继承与函数式风格。

语法与特性： 虽然 ArkTS 增强了静态检查，但毕竟基于 TypeScript，函数、类、模块的定义和 Swift 存在显著区别。例如，ArkTS 使用 `function` / `=>` 定义函数、用 `Promise` / `async-await` 处理异步，而 Swift 使用 `func` 关键字和 `async-await`（Swift 5.5+）。ArkTS 没有 Swift 那样严格的访问控制和值类型/引用类型区分，这可能会改变开发者在架构设计时对数据不可变性的考虑。Objective-C 开发者需要适应没有头文件、没有显式消息发送 `[ ]` 语法，转而使用 TS/JS 的模块化和调用方式。

内存管理： Swift/Obj-C 依赖 ARC/引用计数，开发者需要注意强引用循环，而 ArkTS 由 VM 自动垃圾回收，无需手动管理内存，但也意味着需要留心避免产生无法GC的闭包引用。从手动/半自动管理内存转向完全依赖 GC，对习惯优化内存的 iOS 开发者来说是个观念转变（大部分情况下 GC 已足够高效，但内存敏感场景需注意释放引用避免内存泄漏）。

多线程与并发： iOS 使用 GCD、Operation 等进行并发编程，而 ArkTS 由于单线程事件循环模型，更多使用 Promise、async-await 实现异步，以及 Web Worker 类似的 Worker 机制实现多线程。ArkTS 没有直接等价于 GCD 的全局队列概念，但可以通过创建 Worker 线程并与主线程消息通信。迁移时需要重新思考并发逻辑：例如网络请求在 ArkTS 中通常使用 Promise，UI 更新也是事件驱动，避免了 iOS 中需要切换回主线程更新 UI 的情形（ArkUI 默认在主线程运行 UI逻辑）。

总之，开发者需要重新熟悉 ArkTS 语言的**语法、类型系统、标准库**。好在 ArkTS 与 TypeScript 高度相似，如果有 JS/TS 基础将更易掌握。也可以利用 VS Code 等工具的语言服务来辅助 ArkTS 编码。华为官方文档提供了“从 TypeScript 到 ArkTS”的详细差异说明，建议深入学习 [64]。

UI 框架迁移：UIKit/SwiftUI → ArkUI

Imperative vs Declarative： UIKit 属于命令式 UI 框架，开发者需要直接操作 UIView 的层次关系、frame布局；SwiftUI 则是声明式，与 ArkUI 在理念上相近。对于熟悉 SwiftUI 的开发者，ArkUI 的 @State、@Component（类似 SwiftUI View）、组合子视图的方式会感觉熟悉，主要差异在语法和组件库。举例来说，SwiftUI 用 Struct 和 `View` 协议描述界面，ArkUI 用 TS 的 struct 类 + `@Component` 装饰器实现组件，二者都强调**状态驱动UI**。SwiftUI 的 `@State`、`@Binding` 等有对应的 ArkUI `@State`、`@Link`（双向绑定）等机制，学习成本不高。

组件与布局： UIKit 提供UIButton、UILabel等视图组件，SwiftUI 提供Text、Button等，ArkUI 也有类似的 Text、Button 组件。ArkUI 的布局容器如 `Row`（水平布局）、`Column`（垂直布局）、`Flex`（弹性布局）等，分别对应 SwiftUI 的 `HStack`、`VStack` 和灵活布局。ArkUI 还支持基于 CSS Flexbox 的布局属性，对于精通 Auto Layout 或 SwiftUI 布局的开发者，需要习惯 ArkUI 特有的布局属性（如 `.margin()`、`.padding()` 链

式调用）。**样式**方面，ArkUI 组件支持丰富的样式方法，相当于把 UIKit 的属性和 CSS 样式结合，例如 `.backgroundColor()`、`.fontSize()`，开发体验接近 SwiftUI 的修饰符链条。

导航与路由： UIKit 使用 UINavigationController 管理页面栈，SwiftUI 则有 NavigationView 和 NavigationLink。ArkUI 在 Stage 模型下，通过 `Ability` 实现页面导航，每个页面对应一个 UIAbility，可以使用 `router.push({ url: "XXX" })` 在页面间跳转（ArkUI 也提供类似 Router 的接口）。在设计思维上，HarmonyOS 将页面和能力解耦，通过 Ability 来导航，可以视为类似 iOS Storyboard Segue 或 SwiftUI NavigationLink 的功能，但需要习惯在 ArkTS 层配置路由信息（通常在模块的 `module.json5` 声明各 Ability 的路由名称）。

生命周期： iOS ViewController 有 `viewDidLoad`、`viewWillAppear` 等生命周期，SwiftUI 则弱化了这些概念（但有 onAppear 修饰符）。ArkUI 组件同样有生命周期回调，如 `aboutToAppear`（组件即将显示）和 `aboutToDisappear` [65]。Ability 本身也有应用级生命周期（如 Ability 的 `onCreate`、`onDestroy`）。iOS 开发者需要在 HarmonyOS 中留意**Ability 的生命周期**与 UI 组件生命周期的区别：Ability 更像 UIApplicationDelegate+UIViewController 的合体，管理应用/页面的开启和关闭，而 ArkUI 组件生命周期更多用于页面UI的状态管理。比如，类似 `viewDidAppear` 的逻辑可以放在 Ability 的 onForeground 或 ArkUI 组件的 aboutToAppear 钩子中实现。

小结： UIKit 转 ArkUI 转变较大，需要习惯声明式UI思路；SwiftUI 转 ArkUI 则相对平滑，只需掌握 ArkUI 特定语法和组件用法。建议多参考 ArkUI 官方示例代码，熟悉常用组件的使用方法。

数据与网络能力迁移

本地数据存储： iOS 常用 **UserDefaults** 保存简单配置，**Keychain** 保存敏感信息，**Core Data**/SQLite 存储结构化数据。HarmonyOS 对应有 **Preferences**（用户首选项）用于键值对数据，**KV 存储/文件**用于简单数据持久化，安全信息可借助 **Asset Store Kit** 加密存储（或自行实现加密后存在 Preferences）。对于复杂数据模型，HarmonyOS 提供 **DataAbility** 或关系型数据库（`ohos.data.rdb`）类似 CoreData 的作用，但实现上更接近Android的ContentProvider思想。如果应用在iOS用到了文件沙盒（读取沙盒路径文件）、在Harmony也可以通过文件系统API读取应用自身目录。

迁移时，需要将**UserDefaults调用替换为 Preferences API**：例如 `UserDefaults.standard.set(x, forKey:y)` 替换为 `preferences.put(key, value)` 并 `flush()`；Keychain 可通过 Asset Store Kit 或使用 ArkTS 的加解密库手动实现存储。CoreData 则需要重新设计，比如使用 SQLite 的 ArkTS 封装库（OpenHarmony社区有基于 sqlite 的数据Ability封装）或直接采用 DataAbility + RDB 接口。总的来说，HarmonyOS 数据存储接口更类似 Android，有明确的读写权限控制和分布式选项，这与 iOS 封闭沙盒有区别，需要开发者了解**数据沙盒路径**和**分布式存储**是否启用（大多数情况下手机应用只用本地存储）。

网络通信： iOS 使用 URLSession 或第三方库（如 Alamofire）进行 HTTP 请求。HarmonyOS 提供内置 **HttpRequest** 类完成同样功能，支持 HTTP/HTTPS 和双向认证等。迁移非常直接：将 URLSession 的 dataTask 替换为 HttpRequest 的 `request` 调用，并用 Promise 或 async-await 处理结果。例如： - Swift URLSession:

```
URLSession.shared.dataTask(with: url) { data, response, error in
  // 处理结果
}.resume()
```

- ArkTS HttpRequest:

```
httpRequest.request(url, { method: "GET" })
  .then(res => { /* 处理结果 */ });
```

二者都是异步获取网络数据，但 ArkTS 更偏向 Promise 模式，没有 iOS Delegate 回调。iOS 开发者需注意，在 ArkTS 中处理 JSON 非常方便（因JS天然支持JSON），不用像 Swift 那样序列化/反序列化 Data -> JSON。对于 WebSocket、BLE 等通信，HarmonyOS 也有对应模块（如 @ohos.communication.socket 等），用法类似 Web API 或 Android API，需要重新学习对应接口调用方式。总的来说，**HTTP请求**迁移成本低，但**网络监听/底层通信**可能接口差异较大，要参考HarmonyOS文档。

多媒体与后台服务： 如果 iOS 应用涉及多媒体播放（AVPlayer）或后台任务（如 Background Fetch、Push Notifications），在 HarmonyOS 上也需调整。多媒体播放可使用 Harmony 的 AVPlayer；后台任务在 HarmonyOS 中通过 **ExtensionAbility**（如 Service Extension）实现，或利用系统调度（如定时任务 Ability）。iOS 的Push通知在Harmony对应为 Push Kit，需要集成HMS的推送服务。数据和服务能力迁移需要通盘考虑：**哪些 iOS Framework 在 HarmonyOS 有类似Kit**？常见的有： - 定位：CoreLocation -> Harmony Location Kit - 地图：MapKit/GoogleMaps -> 调用高德/百度地图 SDK（ArkTS 可集成第三方 JS SDK 或 HMS Map Kit） - 支付：Apple Pay/IAP -> HMS Core支付/In-App Purchases - 云存储：iCloud -> 华为云空间或者 OneHop传播，需寻找替代方案

因此，迁移时除了语言API，还需**替换所依赖的服务**。比如 Apple 提供的服务（登录、支付、地图）通常在 Harmony要换成华为或第三方服务，这涉及业务层面的调整。

项目结构与工程配置差异

文件结构： iOS 项目通常以 Xcode 工程管理，拥有 `*.xcodeproj` 和源代码、资源文件组成。HarmonyOS 工程使用 DevEco Studio（基于 IntelliJ）管理，采用 Gradle 类似的文件组织。Stage 模型 ArkTS 工程下，一般有一个工程目录，里面一个或多个模块文件夹（如 `entry` 模块）。每个模块下有 `src` 目录，里面分 `main/ets`(存放 ArkTS 源码)、`main/resources`（资源文件，如图片、布局等），类似 Android 工程结构。相对而言，iOS 的资源（Assets.xcassets）、代码混在一个 target 下，而 Harmony 默认 entry/feature 分模块清晰，**需要开发者熟悉多模块的概念**。简单应用通常所有UI放在 entry 模块，但大型应用可拆分功能模块，每个模块对应独立 hap 包。

配置文件： iOS Info.plist 包含应用名称、图标文件名、权限用途描述等配置；HarmonyOS 将这类信息分别写在 app.json5 和 module.json5。例如应用名称、图标在 app.json5 的 `app.name` 和 `app.icon` 字段定义，权限用途在 module.json5 的 `requestPermissions.reason` 描述，URL Scheme 类似的能力则通过配置 `intents` 来实现。不同之处在于，HarmonyOS 强调**显式声明**应用能力和权限，没有 iOS 那种弱依赖 Info.plist 键值的做法。例如，使用相机权限，iOS 要在 Info.plist 添加 NSCameraUsageDescription 文本，而 HarmonyOS 要在 module.json5 声明权限名+用途，并在代码中调用请求授权。

依赖与构建： iOS 使用 CocoaPods/SPM 管理依赖库，引入第三方库需要添加 pods 或 Swift Package，并由 Xcode 构建整合。HarmonyOS 则使用 OHPM，开发者需要在 oh-package.json5 里声明依赖，再通过 DevEco Studio 或命令行同步安装库。依赖库形式方面，iOS 常见的是静态库、framework、Swift Package，而 HarmonyOS 则是 HAR 包或者 npm 类 JS包。构建过程中，iOS 链接库由 Xcode 自动处理，HarmonyOS 则由 Hvigor 根据 oh-package.json5 下载的 HAR，在编译阶段合并进应用 HAP。工程配置上，iOS 大部分配置在 Xcode可视化设置（Build Settings、Capabilities）中完成，比如启用 Push、Background Modes。而 HarmonyOS 则在配置文件或Gradle脚本完成，例如在 build-profile.json5 或 hvigor 插件中配置签名、压缩、混淆、打包选项 [66]。这种显式脚本配置对iOS开发者来说不太直观，需要一些学习，但好处是**配置即代码**，方便版本管理。

调试方式差异

本地调试： iOS 调试主要通过 Xcode，将应用跑在真机或Simulator，使用断点、调试控制台、View Debugging 等工具。HarmonyOS 开发使用 DevEco Studio 调试 ArkTS 应用。调试方式与 Android 类似，可以通过 USB 或 HDB（Huawei Debug Bridge）连接真机，在 DevEco 中一键运行调试。ArkTS 支持断点调试，开发者可在 ArkTS 代码中打断点，调试器能够逐步执行、查看变量，就像调试 JavaScript 或 Java 一样。HarmonyOS 5 甚至支持**跨语言调试**：如果 ArkTS 调用了 C/C++ 模块，可以在 ArkTS 断点处跳转到本地代码继续调试 [67] ——这一点类似于 Xcode 可以调试 Swift 和 C 同时进行。需要注意的是，ArkUI 的声明式框架调试有别于命令式UI，一些 UI 刷新并非顺序执行，调试时可借助日志或 DevEco 提供的 Inspector 查看UI层次。相对于 iOS Simulator，HarmonyOS DevEco 提供的模拟器还不够完善（主要还是使用真机调试为主），这点需要适应。

日志与分析： iOS 使用 NSLog 或 OSLog 打印日志，在 Xcode 控制台查看；HarmonyOS 则使用 `console.info/debug/error` 或 Hilog 调用打印日志，可在 DevEco Studio Logcat 或 IDE 控制台查看输出。性能分析方面，iOS 有 Instruments 工具检查内存、CPU、Leaks，HarmonyOS DevEco Studio 也提供性能分析插件，但目前生态下可能不如 Instruments 丰富。可以使用 ArkUI 自带的 Inspector 调试UI布局和渲染性能 [65]，以及 profiler 分析应用启动、响应等。在迁移初期，开发者可能会怀念 Instruments 等强大的分析工具，需要寻找 HarmonyOS 下对应的替代方案或第三方工具。

打包签名与发布流程差异

打包产物： iOS 打包产物为 `.ipa` 文件（实质为应用 .app 包的 Zip），通过 App Store 分发。HarmonyOS 应用产物是 `.hap` 和 `.app`：每个模块会打包成一个 HAP (Harmony Ability Package)，如果应用只有一个模块，那么最终发布一个包含单个 HAP 的 APP Pack（类似IPA）。多个 HAP 的应用会生成一个 APP 文件，内含多个 HAP（有点类似 Android 多模块拆成多个 APK Bundle）。iOS 开发者需要了解 **HAP 等同于可执行模块**，可以单独安装调试；APP 则是用于应用市场发布的载体格式。

签名证书： Apple 要求开发者使用从 Apple 获取的证书和 provisioning profile 签名应用，由 Xcode 自动完成，大多细节由苹果托管。而 HarmonyOS 开发者需要自己生成签名证书（可以使用 keytool 或 OpenSSL 生成 .p12 私钥证书，以及 .p7b 公钥证书链），然后在华为开发者后台注册该证书信息。DevEco Studio 提供生成调试证书的功能，但正式发布需要**手动配置 release 签名**。签名信息填写在 build-profile.json5 的 signingConfigs 中，包括证书路径、别名、密码等 [18]。发布流程上，Android/Harmony 采取“开发者本地签名-华为后台校验”的模式，相当于开发者拥有更大掌控权，但也多了一步配置任务。iOS 转 Harmony，要习惯**管理自己的签名证书**，包括保管私钥、防止泄露（因为一旦泄露，恶意者可签名伪造应用）。

发布审核： iOS 应用提交 App Store Connect，经过苹果审核后发布。HarmonyOS 应用通过华为 **AppGallery Connect** 提交审核发布。两者流程类似，需要填写应用信息、上传截图、设置分发地区等。差异在于： - 华为 AppGallery 对应用包体的检查项与苹果略有不同，例如对隐私权限合规也有要求但策略可能不完全相同，需要阅读华为的审核指南（总体上遵循中国的法规和华为的生态标准）。 - 发布周期：苹果审核通常较严格、时间不定；华为 AppGallery 审核相对迅速一些，一般数天内可完成审核（在中国区，华为应用商店审核速度较快）。 - 付费政策：苹果抽成 30%（小开发者 15%），华为商店也有类似抽成比例（一般也是 30%），不过具体政策可能随地区和活动有所调整。若应用内含 IAP，需要使用各自平台的支付 SDK（苹果内购 vs HMS IAP Kit），这一点前面提到过，需要开发者分别实现。

渠道发布： iOS 只有 App Store 官方渠道（中国有 TestFlight等测试渠道，但正式发布只能 App Store）。HarmonyOS（Android APK 同理）在国内有多家应用商店，**AppGallery** 是华为官方渠道，也是 HarmonyOS 上应用分发的主要渠道。但开发者也可以选择在其他安卓市场发布 APK/HAP（前提应用兼容那些设备）。然而，对于纯Harmony能力的应用（ArkTS Stage应用），大概率只能在华为系设备运行，建议聚焦 AppGallery 平台发布。对于面向中国市场的应用，可能还需考虑通过腾讯应用宝等渠道获取用户 —— 这和iOS独家渠道的策略很不同，需要产品运营层面决定。

简而言之，发布方面 iOS 开发者需要学习使用 AppGallery Connect：注册开发者（个人或企业实名认证）、创建应用、提交包体、填写资料、以及后续的更新管理、用户反馈等。华为的应用市场后台和苹果的类似，但界面和细节不同，需花时间熟悉。

权限与系统能力申请方式差异

权限模型区别： iOS 的权限（相机、麦克风、定位等）在 Info.plist 用键声明用途描述，并在代码第一次使用相关API时由系统弹窗请求授权。HarmonyOS 则要求**在配置文件和代码双方都声明**：在 module.json5 的 `requestPermissions` 列表中声明需要申请的权限及用途 [68]，同时对于用户授权权限，应用必须在运行时调用 `requestPermissions()` 触发弹窗 [23]。也就是说，在 HarmonyOS 中权限请求是**前置显式**行为。如果忘记在配置文件声明，哪怕代码请求权限，系统也不会授予；反之亦然。如果权限是系统自动授予的（如普通权限），则只需在配置声明，不弹窗。

实例比较： 以获取定位为例： - iOS: 在 Info.plist 加 `NSLocationWhenInUseUsageDescription` 描述，然后：

```
locationManager.requestWhenInUseAuthorization()
```

系统弹窗用户允许后，回调 `didChangeAuthorization` 通知应用。 - HarmonyOS: 在 module.json5:

```
"requestPermissions": [ { "name": "ohos.permission.LOCATION", "reason": "需要获取位置提供导航" } ]
```

然后代码中：

```
const result = await permission.requestPermissions(['ohos.permission.LOCATION']);
if(result) { ... }
```

系统弹窗让用户允许，结果通过 Promise 返回。

二者逻辑类似，但 HarmonyOS 强制在配置中注明用途（虽然这一用途字符串目前主要用于审核，并不会自动弹给用户）。同时，HarmonyOS 把权限粒度划分得与 Android 接近，一些权限在安装时自动授予（如互联网权限），另一些要运行时申请。此外，HarmonyOS 提供 `PermissionKit` 可查询权限状态、跳转设置页等，这点和 iOS `CLAuthorizationStatus` 等查询类似。

系统能力开关： iOS 的某些系统能力（Background Modes、Push Notifications）需要在 Xcode Capabilities 中开启，对应修改 Entitlements 文件。HarmonyOS 则在配置文件的 `module` 字段或 AppGallery Connect 配置开启。例如**Push**服务需要集成 Push Kit SDK，并在 AGC 上为应用打开“消息推送”服务，无需像 iOS 那样在本地做 entitlement 声明（Harmony 的权限体系已涵盖大部分场景）。**后台任务**方面，iOS 有 Background Fetch，需要申请相应模式；HarmonyOS 则有后台 Ability（ServiceExtensionAbility）可在应用不在前台时运行，但第三方应用对此有所限制 [69]，通常建议使用 WorkScheduler（类似Android的JobScheduler）处理后台定时任务。

隐私合规： iOS 要求开发者在提交应用时填写隐私表单说明收集哪些数据。华为 AppGallery 也有类似的**个人信息收集清单**需要填写，并遵守《鸿蒙隐私规范》。从技术上，Harmony 提供了**隐私标签**功能，应用可以调用 API 来弹出华为统一的隐私弹窗等。如果你的 iOS 应用已有隐私合规流程，迁移到 HarmonyOS 时也要遵守当地法律和华为的合规要求，及时在应用中提示用户并获取同意。

总的来说，权限与系统能力在 HarmonyOS 上更加**声明式和显式**。iOS 开发者需要从“事后请求授权”的思路转变为“事前声明+事中请求”的双重把关，并且熟悉每个权限名（很多与 Android 相同，如 CAMERA、RECORD_AUDIO 等）的用法。官方文档和示例代码对权限的声明和请求都有范例，可以参考套用，以确保应用正常运行且通过商店审核。

上线发布渠道对比：App Store vs AppGallery Connect

开发者账户： Apple Developer 需要付年费订阅，华为 AppGallery Connect 注册开发者目前**免费**（个人开发者需要实名认证，企业开发者提交资质）。这降低了初学者试水 HarmonyOS 的门槛。

应用审核侧重： Apple Store 审核以全球标准执行，对应用内容、功能合规、UI/UX都有严格要求；华为 AppGallery 主要面向华为设备用户，审核重点包括**隐私合规**（需要提供隐私政策、符合数据最小化原则等）、**应用稳定性**、**本地法律合规**（比如不得有违法违规内容）等。相对而言，AppGallery 对 UI 创新没有苹果那么挑剔，但对在中国运营的内容（比如防沉迷、广告合规）要求高。迁移时如果应用包含社交、游戏等，要注意集成华为的防沉迷、实名认证 SDK 以满足监管要求。

生态服务： 苹果生态封闭，应用如果需要推送、地图等服务只能用苹果或自行实现。华为生态开放程度高一些，AppGallery Connect 提供多种服务（类似 Firebase），如远程推送、分析、A/B 测试、短信、授权登录等，开发者可以选择性集成。此外，HarmonyOS 具备**分布式能力**（比如多设备协同），如果应用希望利用这些，可以借助鸿蒙特点实现创新功能。这些是 iOS 没有的生态特性，也是迁移后可以探索的增量功能。

用户和市场： iOS 面向全球高端市场，App Store 覆盖面广；HarmonyOS 当前主要用户集中在中国（搭载 HarmonyOS 的华为设备用户）。上架 AppGallery 意味着主要触达华为手机用户群。对于商业应用，可能需要调整变现策略——例如华为渠道常见预装、推广等合作模式不同于苹果的纯商店分发。另外，如果原应用有订阅/支付，需要迁移到华为 IAP，并注意价格、合规要求。上线后，AppGallery 提供的运营位资源可以争取，比如参与华为的活动获得推荐，这是国内安卓生态常见的运营手段，iOS开发者也需了解这些以便最大化获取用户。

更新机制： iOS 应用更新需要通过 App Store，用户设备自动检查。HarmonyOS 除了商店更新，也允许应用内检查更新（如果托管在 AppGallery，则也有类似 API 可以提示用户更新）。鉴于 HarmonyOS 也兼容 Android App，这里值得一提：如果你的应用也考虑 Android 平台用户，可能需要在其他安卓市场发布APK。HarmonyOS 5 对 APK 兼容，但 ArkTS 应用需要在鸿蒙环境运行，暂时覆盖面有限。因此发布策略上，可能需要维护**两个代码版本**（iOS 和 Harmony/Android），通过不同商店发布。

总结而言，上线发布从 iOS 转 HarmonyOS 主要是熟悉新平台的**规则和机会**。建议通读 AppGallery Connect 发布指南，了解从提交审核到上线各步骤（与苹果类似，但细节不同）。对于有现成 iOS 应用的团队，可以考虑在 AppGallery 先发布 Android 版本（如果有），再逐步推出 HarmonyOS 专属版本，借助鸿蒙特性提升华为用户的体验。后续还可以探索上架 HarmonyOS 原子服务（免安装小卡片）等渠道，扩大应用影响力。

4. 迁移对比速查表与示例代码对照

概念与功能对比总览

对比维度	**iOS 开发** (Apple iOS)	**HarmonyOS 开发** (Huawei Harmony)
应用架构模型	MVC / MVVM（UIViewController 结构）；SwiftUI 声明式视图	Stage 模型 (UIAbility + ExtensionAbility); ArkUI 声明式界面

对比维度	**iOS 开发** (Apple iOS)	**HarmonyOS 开发** (Huawei Harmony)
主要编程语言	Swift（静态强类型, ARC 内存管理）; Objective-C (引用计数)	ArkTS（基于 TypeScript，静态类型+动态特性，自动GC）; 也支持 JS、C/C++ 等
UI框架	UIKit (命令式, Storyboard/XIB) / SwiftUI (声明式)	ArkUI (声明式 UI 框架，组件丰富，状态驱动界面)
UI组件生命周期	UIViewController 的 `viewDidLoad/Appear` 等, SwiftUI 的 onAppear/onDisappear	UIAbility 的 onForeground/ onBackground; ArkUI 组件 aboutToAppear/aboutToDisappear 钩子
布局机制	Auto Layout（约束）, SwiftUI 布局（HStack/ VStack etc）	Flex布局、Grid、自适应布局（基于 ArkUI 容器，如 Row, Column）
路由导航	UINavigationController / segue; SwiftUI NavigationLink	Ability 路由（FA：Page/Router，Stage：AbilityLoader 跳转）; ArkUI Router.push/ replace
本地存储	UserDefaults, Keychain, CoreData, 文件读写	Preferences（首选项）, Asset Store（安全存储）, DataAbility/RDB, 文件读写
网络通信	URLSession (GET/POST), NSURLSessionWebSocket, 第三方 Alamofire	@ohos.net.http (HttpRequest 请求), WebSocket 模块, 第三方 JS 网络库
多线程并发	GCD (DispatchQueue), NSOperationQueue, async/await	Promise, async/await; Worker 线程 (Web Workers 类似)
开发IDE	Xcode (Interface Builder, Instruments)	DevEco Studio (基于 IntelliJ, 内置模拟器/调试器)
构建工具	Xcode Build (clang, LLVM), CocoaPods, SwiftPM 集成	DevEco Hvigor (TS脚本构建打包), OHPM 依赖管理
项目配置	Info.plist, *.xcconfig, Build Settings (Xcode GUI 配置)	app.json5 / module.json5 / build-profile.json5 (JSON5 配置文件)
调试测试	真机调试 + iOS Simulator; XCTest 单元测试; TestFlight 预发布	真机调试 + DevEco模拟器; XCTest 类似框架 (DevEco UT 框架); DFX 调试工具
签名机制	Apple 签名（开发/发布证书+Provisioning Profile），IPA 分发	自签名（本地 keystore .p12 签名 HAP），AppGallery 复核签名
权限声明	Info.plist 声明用途字符串 + 首次自动弹窗授权	module.json5 声明权限 + 运行时调用 requestPermissions 弹窗
通知推送	UserNotifications.framework + APNS	@ohos.notification 本地通知 + Push Kit (需 HMS Core 支持)
应用发布渠道	Apple App Store (独家官方商店)	Huawei AppGallery (主渠道，亦可通过APK 在其他市场发布)
发布审核	严格人工审核（内容、隐私、性能）; 上架需付费开发者账号	人工+自动审核（内容、安全）；开发者账号免费，需实名认证

对比维度	**iOS 开发** (Apple iOS)	**HarmonyOS 开发** (Huawei Harmony)
生态服务	Apple 专有服务（Sign in with Apple, Apple Pay, iCloud 等）	华为/HMS 服务（Account Kit, IAP Kit, Cloud Storage 等）；可选第三方替代

示例代码对照

以下通过简单的代码片段对比 iOS 与 HarmonyOS 在开发实现上的区别。

UI 界面构建示例对比

假设要实现一个界面，包含一个文本标签和一个按钮，点击按钮时更新文本。
在 iOS (SwiftUI) 中，可以这样实现：

```
// iOS SwiftUI 实现
import SwiftUI

struct ContentView: View {
  @State private var count: Int = 0

  var body: some View {
    VStack {
      Text("点击次数: \(count)")
        .font(.title)
        .padding()
      Button(action: {
        count += 1
      }) {
        Text("点我一下")
      }
      .padding()
    }
  }
}
```

SwiftUI 使用 `@State` 管理点击计数，界面由 VStack 包含 Text 和 Button，按钮点击修改状态。

在 HarmonyOS (ArkUI) 中，实现方式如下：

```
// HarmonyOS ArkUI (ArkTS) 实现
@Entry @Component
struct CounterPage {
 @State count: number = 0;

 build() {
  Column({ alignItems: Alignment.Center }) {
   Text(`点击次数: ${this.count}`)
    .fontSize(24).margin(8)
```

```
    Button("点我一下")
      .onClick(() => {
        this.count += 1
      })
      .margin(8).backgroundColor("#EEEEEE")
  }
 }
}
```

ArkUI 也通过 `@State` 保存状态，使用 Column 布局堆叠 Text 和 Button。点击事件用 `onClick` 注册，修改状态后框架自动刷新绑定该状态的 Text 显示。可以看到，两者在代码结构上非常相似：SwiftUI 用闭包更新状态，ArkUI 用箭头函数；界面声明上一个用 Swift 语法，一个用 TS 语法，但思想都是**声明 UI = f(State)**。

网络请求示例对比

比较在两个平台上进行 HTTP 请求并处理响应的代码。

iOS (Swift): 使用 URLSession 发起 GET 请求：

```
// iOS 使用 URLSession GET 请求
import Foundation

guard let url = URL(string: "https://api.example.com/data") else { return }
let task = URLSession.shared.dataTask(with: url) { data, response, error in
  if let error = error {
    print("请求失败: \(error)")
    return
  }
  if let data = data,
    let resultStr = String(data: data, encoding: .utf8) {
    print("收到数据: \(resultStr)")
  }
}
task.resume()
```

这段 Swift 代码创建了一个 dataTask，异步获取数据后将结果打印。

HarmonyOS (ArkTS): 使用 @ohos.net.http 模块的 HttpRequest：

```
// HarmonyOS 使用 HttpRequest GET 请求
import http from '@ohos.net.http';

const url: string = "https://api.example.com/data";
let httpRequest = http.createHttp();
httpRequest.request(url, { method: http.RequestMethod.GET })
 .then(response => {
  if (response.responseCode === 200) {
   console.info("收到数据: " + response.result);
  } else {
```

```
    console.error("请求失败，状态码: " + response.responseCode);
  }
 })
 .catch(err => {
  console.error("请求错误: " + JSON.stringify(err));
 });
```

两者对比如下：iOS 通过闭包获取结果，在主线程外执行，需要注意线程切换；ArkTS 用 Promise 处理异步，直接在 then 中更新（ArkUI应用中Promise完成后若要更新UI，框架会保证在UI线程执行，开发者无需额外处理）。ArkTS 获取的 `response.result` 已是字符串，不需要像 Swift 那样手动将 Data 转为文本。需要提醒的是，在 ArkTS 中使用网络前，应确保已在配置文件声明网络权限，否则请求可能会失败而无明确错误信息。

权限申请示例对比

以请求相机权限为例，对比两端代码。

iOS: 在 Info.plist 添加键 `NSCameraUsageDescription`，代码中：

```
import AVFoundation

switch AVCaptureDevice.authorizationStatus(for: .video) {
case .authorized:
  // 已授权，直接使用相机
case .notDetermined:
  // 第一次请求
  AVCaptureDevice.requestAccess(for: .video) { granted in
    if granted {
      print("用户同意相机权限")
    } else {
      print("用户拒绝相机权限")
    }
  }
case .denied, .restricted:
  // 用户已拒绝或家长控制，提示去设置开启
default:
  break
}
```

HarmonyOS: 在 module.json5 声明：

```
"requestPermissions": [
 {
  "name": "ohos.permission.CAMERA",
  "reason": "拍照需要摄像头权限"
 }
]
```

ArkTS 代码中：

```
import ohos.permission as permission;

// 检查并请求摄像头权限
let hasCamera = await permission.hasPermission("ohos.permission.CAMERA");
if (!hasCamera) {
  const grantResult = await permission.requestPermissions(["ohos.permission.CAMERA"]);
  if (grantResult) {
    console.info("用户同意相机权限");
  } else {
    console.warn("用户拒绝相机权限");
  }
}
```

可以看到，iOS 通过系统 API 检查权限状态并请求授权；HarmonyOS 则需要先在配置中声明，再用 `permission.requestPermissions` 触发授权弹窗。iOS 授权结果通过闭包异步返回，HarmonyOS 使用 Promise `await` 更加直观。HarmonyOS 还提供 `hasPermission` 快速检查当前状态。总体差异在于**配置声明**和**API 调用方式**不同，但逻辑类似。iOS 开发者需注意 HarmonyOS 权限名通常以 `ohos.permission.X` 命名，且不要忘记 module.json5 的声明，否则代码请求会直接返回失败而不弹窗。

以上对照和示例旨在帮助 iOS 开发者快速理解从 Apple 生态迁移到 HarmonyOS 5 生态在开发上的异同。通过结构化的指南和实例代码，相信可以加速您的迁移学习过程，利用好 HarmonyOS 5 提供的新特性，为用户带来更优质的跨端应用体验。

【参考资料：HarmonyOS官方文档、CSDN 博客等，详见以上引用】 2 6 10 58 39 47 53 23

---

1 4 ArkTS简介与代码示例_30天血压arkts代码-CSDN博客
https://blog.csdn.net/qq_40698086/article/details/145105501

2 3 面试必问！鸿蒙开发中的FA模型和Stage模型是什么？他们分别有什么区别？_ak 模型与 stage 的区别-CSDN博客
https://blog.csdn.net/maniuT/article/details/137524488

5 6 7 8 9 15 17 18 19 20 21 24 25 26 28 HarmonyOS NEXT开发进阶（十二）：build-profile.json5 文件解析-CSDN博客
https://blog.csdn.net/sunhuaqiang1/article/details/146278059

10 16 HarmonyOS：ohpm使用指导 - 为敢技术 - 博客园
https://www.cnblogs.com/strengthen/p/18508503

11 12 13 14 27 深入理解HarmonyOS中的oh-package.json5配置-CSDN博客
https://blog.csdn.net/qq_14863717/article/details/139525819

22 23 在HarmonyOS鸿蒙NEXT中ArkTS开发的应用出现“权限申请失败”如何 ...
https://bbs.itying.com/topic/67e0cc55687c4e0048a84d78

29 《鸿蒙HarmonyOS应用开发从入门到精通（第2版）》学习笔记
https://developer.aliyun.com/article/1648128

30 31 32 HarmonyOS鸿蒙ArkTS/ArkUI项目，封装http网络请求，封装公共API以及调用请求的过程实现。_arkts api类-CSDN博客
https://blog.csdn.net/qq_36034945/article/details/135385736

33 【HarmonyOS】【ArkTS】如何使用HTTP网络请求获取动态数据刷新 ...
https://www.cnblogs.com/mayism123/p/17462738.html

34 （二一）ArkTS 数据存储与管理 - CSDN博客
https://blog.csdn.net/m0_71441912/article/details/145981774

35 HarmonyOS轻量级数据持久化-Preferences使用 - 稀土掘金
https://juejin.cn/post/7343139078488195126

36 37 ohos.file.fs (文件管理)-ArkTS API-Core File Kit（文件基础服务）-应用 ...
https://developer.huawei.com/consumer/cn/doc/harmonyos-references/js-apis-file-fs

38 通过用户首选项实现数据持久化(ArkTS) - Huawei
https://developer.huawei.com/consumer/cn/doc/harmonyos-guides/data-persistence-by-preferences

39 ohos.deviceInfo (设备信息)-设备管理-ArkTS API-Basic Services Kit ...
https://developer.huawei.com/consumer/cn/doc/harmonyos-references/js-apis-device-info

40 system.device (设备信息)-已停止维护的接口-ArkTS API-Basic ...
https://developer.huawei.com/consumer/cn/doc/harmonyos-references/js-apis-system-device

41 42 harmony 鸿蒙Using AVPlayer to Play Videos (ArkTS) - seaxiang
https://www.seaxiang.com/blog/zS31ct

43 Camera Kit（相机服务）-媒体- 华为HarmonyOS开发者 - Huawei
https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V5/camera-dev-arkts-V5

44 获取设备的位置信息开发指导（ArkTS） - Huawei Developer
https://developer.huawei.com/consumer/cn/doc/harmonyos-guides/location-guidelines

45 HarmonyOS鸿蒙Next ArkTS中获取位置服务 - IT营社区
https://bbs.itying.com/topic/688447d4babc5b0092d1fc8a

46 地理围栏开发指导(ArkTS)-Location Kit（位置服务）-应用服务
https://developer.huawei.com/consumer/cn/doc/harmonyos-guides/geofence-guidelines

47 ArkTS API-Sensor Service Kit（传感器服务）-硬件-系统
https://developer.huawei.com/consumer/cn/doc/harmonyos-references/js-apis-system-sensor

48 49 50 鸿蒙next教程：传感器开发指导(ArkTS)_鸿蒙传感器-CSDN博客
https://blog.csdn.net/qq_39652397/article/details/149174763

51 ArkTS API-Sensor Service Kit（传感器服务）-硬件-系统
https://developer.huawei.com/consumer/cn/doc/harmonyos-references/js-apis-system-vibrate

52 ArkTS API-Notification Kit（用户通知服务）-应用服务- 华为 ...
https://developer.huawei.com/consumer/cn/doc/harmonyos-references/js-apis-notificationmanager

53 54 请求通知授权-Notification Kit（用户通知服务）-应用服务 - Huawei
https://developer.huawei.com/consumer/cn/doc/harmonyos-guides/notification-enable

55 鸿蒙HarmonyOS实战-ArkTS语言基础类库（通知） - 博客园
https://www.cnblogs.com/shudaoshan/p/18237225

56 ArkTS组件-Account Kit（华为账号服务）-应用服务 - Huawei Developer
https://developer.huawei.com/consumer/cn/doc/harmonyos-references/account-arkts-component

57 Account Kit（华为账号服务）-应用服务- 华为HarmonyOS开发者
https://developer.huawei.com/consumer/cn/doc/harmonyos-guides/account-faq-10

58 ArkTS API-Payment Kit（鸿蒙支付服务）-应用服务
https://developer.huawei.com/consumer/cn/doc/harmonyos-references/payment-paymentservice

59 ecnyPaymentService (数字人民币服务)-ArkTS API-Payment Kit ...
https://developer.huawei.com/consumer/cn/doc/harmonyos-references/payment-digitalcnyservice

60 Huawei HarmonyOS 5.0.4 is rolling out with API 16 and new features
https://www.huaweicentral.com/huawei-harmonyos-5-0-4-is-rolling-out-with-api-16-and-new-features/

61 HarmonyOS NEXT 鸿蒙ArkTS 权限控制-录音播放功能实现 - 稀土掘金
https://juejin.cn/post/7476492363030003762

62 Asset Store Kit（关键资产存储服务）-安全-系统 - Huawei
https://developer.huawei.com/consumer/cn/doc/harmonyos-guides/asset-arkts

63 TrustedAppService（可信应用服务）-ArkTS API-Device Security Kit ...
https://developer.huawei.com/consumer/cn/doc/harmonyos-references/devicesecurity-taas-api

64 69 如何在Stage模型中创建后台任务 - Huawei
https://developer.huawei.com/consumer/cn/doc/harmonyos-faqs/faqs-background-tasks-1

65 Component如何监听应用前后台切换-方舟UI框架（ArkUI） - Huawei
https://developer.huawei.com/consumer/cn/doc/harmonyos-faqs/faqs-arkui-230

66 新增和增强特性-DevEco Studio-HarmonyOS NEXT Developer Beta1 ...
https://developer.huawei.com/consumer/cn/doc/harmonyos-releases/deveco-studio-new-features-db1

67 Adding a Component-UI Development (ArkTS-based Declarative ...
https://device.harmonyos.com/en/docs/apiref/harmonyos-guides/arkts-common-components-video-player

68 OpenHarmony应用访问控制权限申请开发范例 - Huawei
https://developer.huawei.com/consumer/cn/forum/topic/0207126523008620185