HarmonyOS 入门文档

V1.0

鸿蒙学堂 hmxt.org 整理

2020 年 9 月 10 日

目 录

1 HarmonyOS 概述

1.1 系统定义

HarmonyOS 是一款“面向未来”、面向全场景（移动办公、运动健康、社交通信、媒体娱乐等）的分布式操作系统。在传统的单设备系统能力的基础上，HarmonyOS 提出了基于同一套系统能力、适配多种终端形态的分布式理念，能够支持多种终端设备。

- 对消费者而言，HarmonyOS 能够将生活场景中的各类终端进行能力整合，形成一个“超级虚拟终端”，可以实现不同的终端设备之间的快速连接、能力互助、资源共享，匹配合适的设备、提供流畅的全场景体验。
- 对应用开发者而言，HarmonyOS 采用了多种分布式技术，使得应用程序的开发实现与不同终端设备的形态差异无关，降低了开发难度和成本。这能够让开发者聚焦上层业务逻辑，更加便捷、高效地开发应用。
- 对设备开发者而言，HarmonyOS 采用了组件化的设计方案，可以根据设备的资源能力和业务特征进行灵活裁剪，满足不同形态的终端设备对于操作系统的要求。

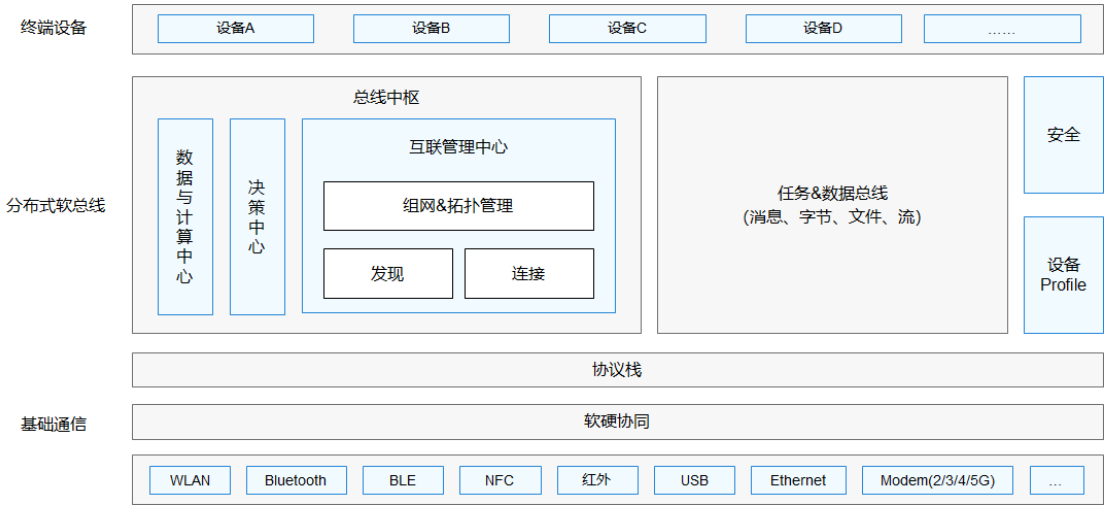
1.2 技术特性

1.2.1 硬件互助，资源共享

1.2.1.1 分布式软总线

分布式软总线是多种终端设备的统一基座，为设备之间的互联互通提供了统一的分布式通信能力，能够快速发现并连接设备，高效地分发任务和传输数据。分布式软总线示意图见图 1。

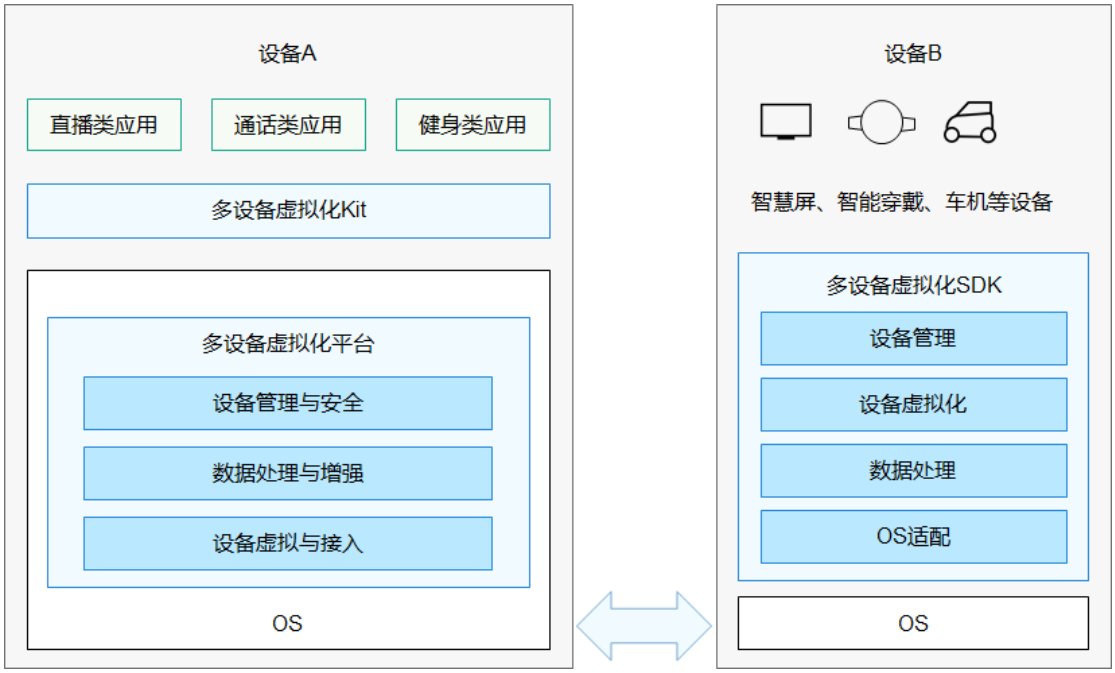
图 1 分布式软总线示意图



1.2.1.2 分布式设备虚拟化

分布式设备虚拟化平台可以实现不同设备的资源融合、设备管理、数据处理，多种设备共同形成一个超级虚拟终端。针对不同类型的任务，为用户匹配并选择能力合适的执行硬件，让业务连续地在不同设备间流转，充分发挥不同设备的资源优势。分布式设备虚拟化示意图见图 2。

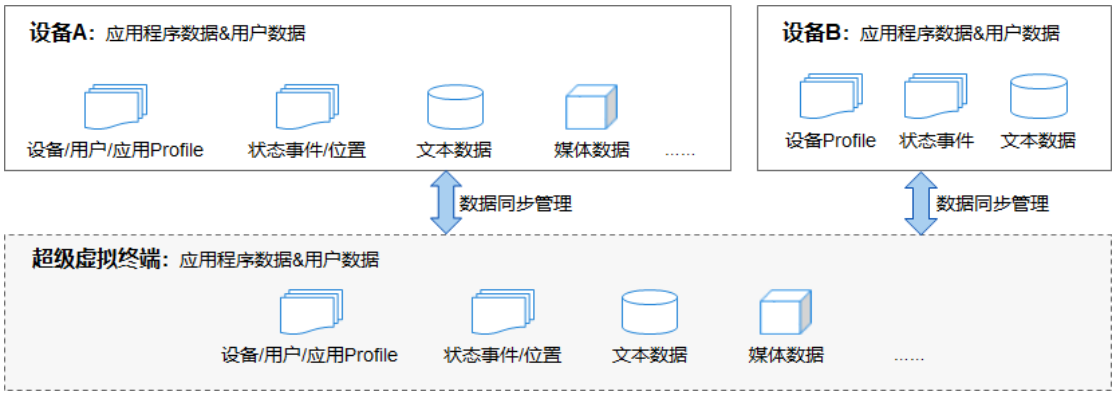
图 2 分布式设备虚拟化示意图



1.2.1.3 分布式数据管理

分布式数据管理基于分布式软总线的能力，实现应用程序数据和用户数据的分布式管理。用户数据不再与单一物理设备绑定，业务逻辑与数据存储分离，应用跨设备运行时数据无缝衔接，为打造一致、流畅的用户体验创造了基础条件。分布式数据管理示意图见图 3。

图 3 分布式数据管理示意图



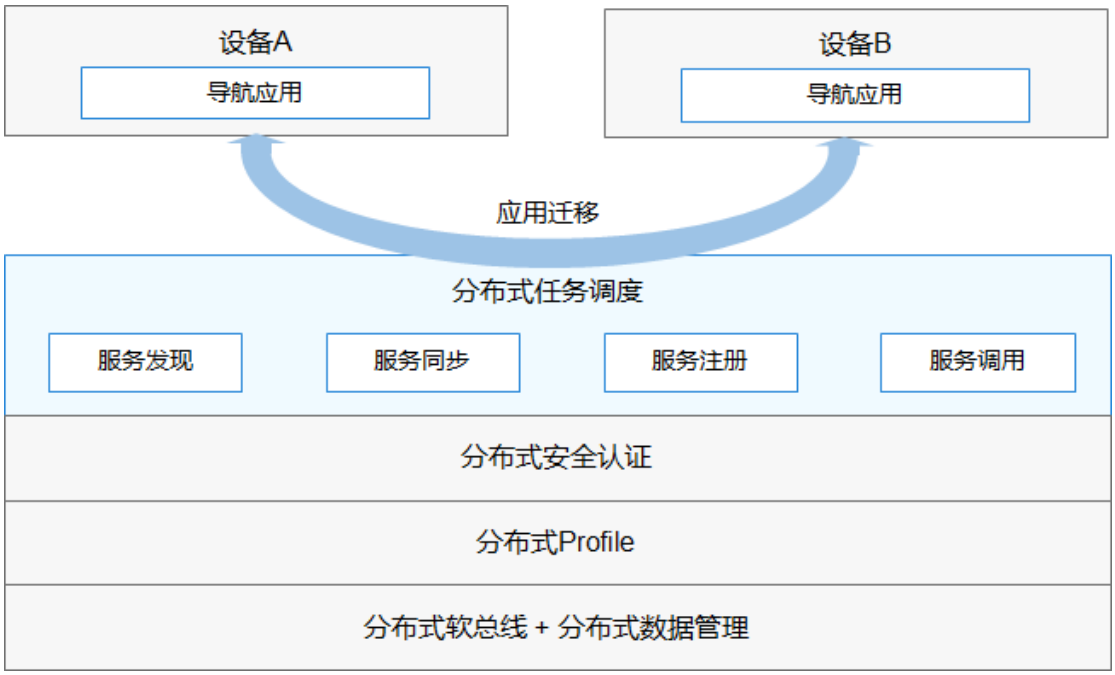
1.2.1.4 分布式任务调度

分布式任务调度基于分布式软总线、分布式数据管理、分布式 Profile 等技术特性，构建统一的分布式服务管理（发现、同步、注册、调用）机制，支持对跨设备的应用进行远程

启动、远程调用、远程连接以及迁移等操作，能够根据不同设备的能力、位置、业务运行状态、资源使用情况，以及用户的习惯和意图，选择合适的设备运行分布式任务。

图 4 以应用迁移为例，简要地展示了分布式任务调度能力。

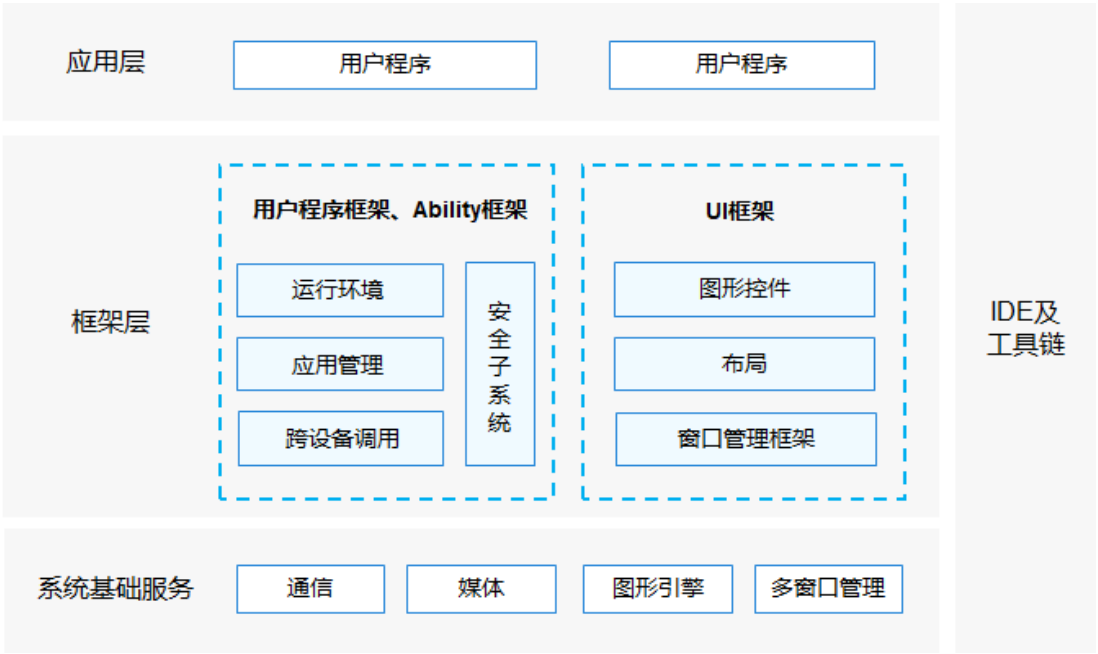
图 4 分布式任务调度示意图



1.2.2 一次开发，多端部署

HarmonyOS 提供了用户程序框架、Ability 框架以及 UI 框架，支持应用开发过程中多终端的业务逻辑和界面逻辑进行复用，能够实现应用的一次开发、多端部署，提升了跨设备应用的开发效率。一次开发、多端部署示意图见图 5。

图 5 一次开发、多端部署示意图



1.2.3 统一 OS，弹性部署

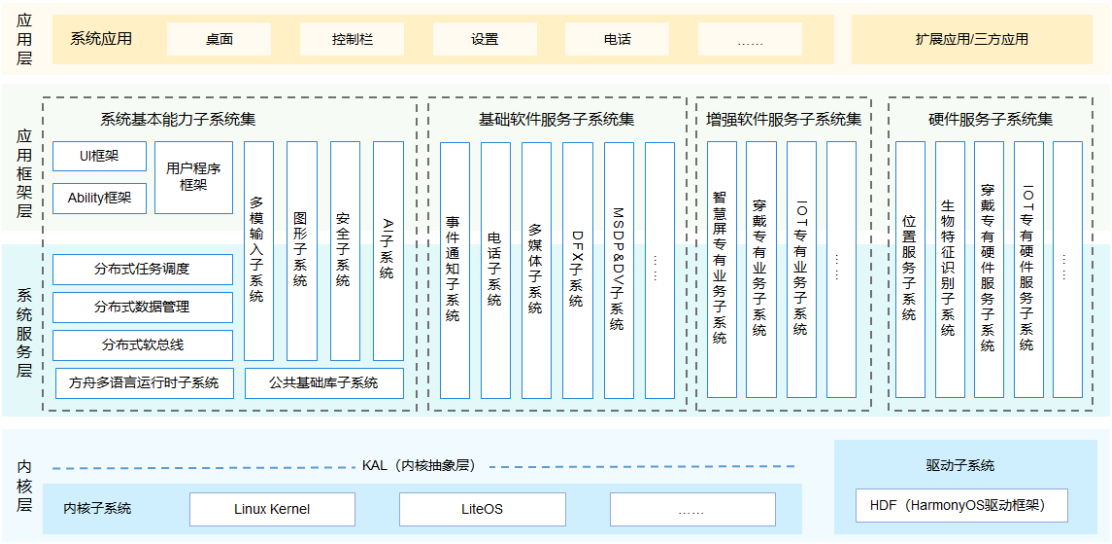
HarmonyOS 通过组件化和小型化等设计方法，支持多种终端设备按需弹性部署，能够适配不同类别的硬件资源和功能需求。支撑通过编译链关系去自动生成组件化的依赖关系，形成组件树依赖图，支撑产品系统的便捷开发，降低硬件设备的开发门槛。

- **支持各组件的选择（组件可有可无）**：根据硬件的形态和需求，可以选择所需的组件。
- **支持组件内功能集的配置（组件可大可小）**：根据硬件的资源情况和功能需求，可以选择配置组件中的功能集。例如，选择配置图形框架组件中的部分控件。
- **支持组件间依赖的关联（平台可大可小）**：根据编译链关系，可以自动生成组件化的依赖关系。例如，选择图形框架组件，将会自动选择依赖的图形引擎组件等。

1.3 技术架构

HarmonyOS 整体遵从分层设计，从下向上依次为：内核层、系统服务层、框架层和应用层。系统功能按照“系统 > 子系统 > 功能/模块”逐级展开，在多设备部署场景下，支持根据实际需求裁剪某些非必要的子系统或功能/模块。HarmonyOS 技术架构如图 1 所示。

图 1 技术架构



1.3.1 内核层

- **内核子系统**：HarmonyOS 采用多内核设计，支持针对不同资源受限设备选用适合的 OS 内核。内核抽象层（KAL，KernelAbstract Layer）通过屏蔽多内核差异，对上层提供基础的内核能力，包括进程/线程管理、内存管理、文件系统、网络管理和外设管理等。
- **驱动子系统**：HarmonyOS 驱动框架（HDF）是 HarmonyOS 硬件生态开放的基础，提供统一外设访问能力和驱动开发、管理框架。

1.3.2 系统服务层

系统服务层是 HarmonyOS 的核心能力集合，通过框架层对应用程序提供服务。该层包含以下几个部分：

- **系统基本能力子系统集**：为分布式应用在 HarmonyOS 多设备上的运行、调度、迁移等操作提供了基础能力，由分布式软总线、分布式数据管理、分布式任务调度、方舟多语言运行时、公共基础库、多模输入、图形、安全、AI 等子系统组成。其中，方舟运行时提供了

C/C++/JS 多语言运行时和基础的系统类库，也为使用方舟编译器静态化的 Java 程序（即应用程序或框架层中使用 Java 语言开发的部分）提供运行时。

- **基础软件服务子系统集**：为 HarmonyOS 提供公共的、通用的软件服务，由事件通知、电话、多媒体、DFX、MSDP&DV 等子系统组成。
- **增强软件服务子系统集**：为 HarmonyOS 提供针对不同设备的、差异化的能力增强型软件服务，由智慧屏专有业务、穿戴专有业务、IoT 专有业务等子系统组成。
- **硬件服务子系统集**：为 HarmonyOS 提供硬件服务，由位置服务、生物特征识别、穿戴专有硬件服务、IoT 专有硬件服务等子系统组成。

根据不同设备形态的部署环境，基础软件服务子系统集、增强软件服务子系统集、硬件服务子系统集内部可以按子系统粒度裁剪，每个子系统内部又可以按功能粒度裁剪。

1.3.3 框架层

框架层为 HarmonyOS 的应用程序提供了 Java/C/C++/JS 等多语言的用户程序框架和 Ability 框架，以及各种软硬件服务对外开放的多语言框架 API；同时为采用 HarmonyOS 的设备提供了 C/C++/JS 等多语言的框架 API，不同设备支持的 API 与系统的组件化裁剪程度相关。

1.3.4 应用层

应用层包括系统应用和第三方非系统应用。HarmonyOS 的应用由一个或多个 FA（Feature Ability）或 PA（Particle Ability）组成。其中，FA 有 UI 界面，提供与用户交互的能力；而 PA 无 UI 界面，提供后台运行任务的能力以及统一的数据访问抽象。基于

FA/PA 开发的应用，能够实现特定的业务功能，支持跨设备调度与分发，为用户提供一致、高效的应用体验。

1.4 系统安全

在搭载 HarmonyOS 的分布式终端上，可以保证“**正确的人，通过正确的设备，正确地使用数据**”。

- 通过“分布式多端协同身份认证”来保证“正确的人”。
- 通过“在分布式终端上构筑可信运行环境”来保证“正确的设备”。
- 通过“分布式数据在跨终端流动的过程中，对数据进行分类分级管理”来保证“正确地使用数据”。

1.4.1 正确的人

在分布式终端场景下，“正确的人”指通过身份认证的数据访问者和业务操作者。“正确的人”是确保用户数据不被非法访问、用户隐私不泄露的前提条件。HarmonyOS 通过以下三个方面来实现协同身份认证：

- **零信任模型**：HarmonyOS 基于零信任模型，实现对用户的认证和对数据的访问控制。当用户需要跨设备访问数据资源或者发起高安全等级的业务操作（例如，对安防设备的操作）时，HarmonyOS 会对用户进行身份认证，确保其身份的可靠性。
- **多因素融合认证**：HarmonyOS 通过用户身份管理，将不同设备上标识同一用户的认证凭据关联起来，用于标识一个用户，来提高认证的准确度。

- **协同互助认证**：HarmonyOS 通过将硬件和认证能力解耦（即信息采集和认证可以在不同的设备上完成），来实现不同设备的资源池化以及能力的互助与共享，让高安全等级的设备协助低安全等级的设备完成用户身份认证。

1.4.2 正确的设备

在分布式终端场景下，只有保证用户使用的设备是安全可靠的，才能保证用户数据在虚拟终端上得到有效保护，避免用户隐私泄露。

- **安全启动**

  确保源头每个虚拟设备运行的系统固件和应用程序是完整的、未经篡改的。通过安全启动，各个设备厂商的镜像包就不易被非法替换为恶意程序，从而保护用户的数据和隐私安全。

- **可信执行环境**

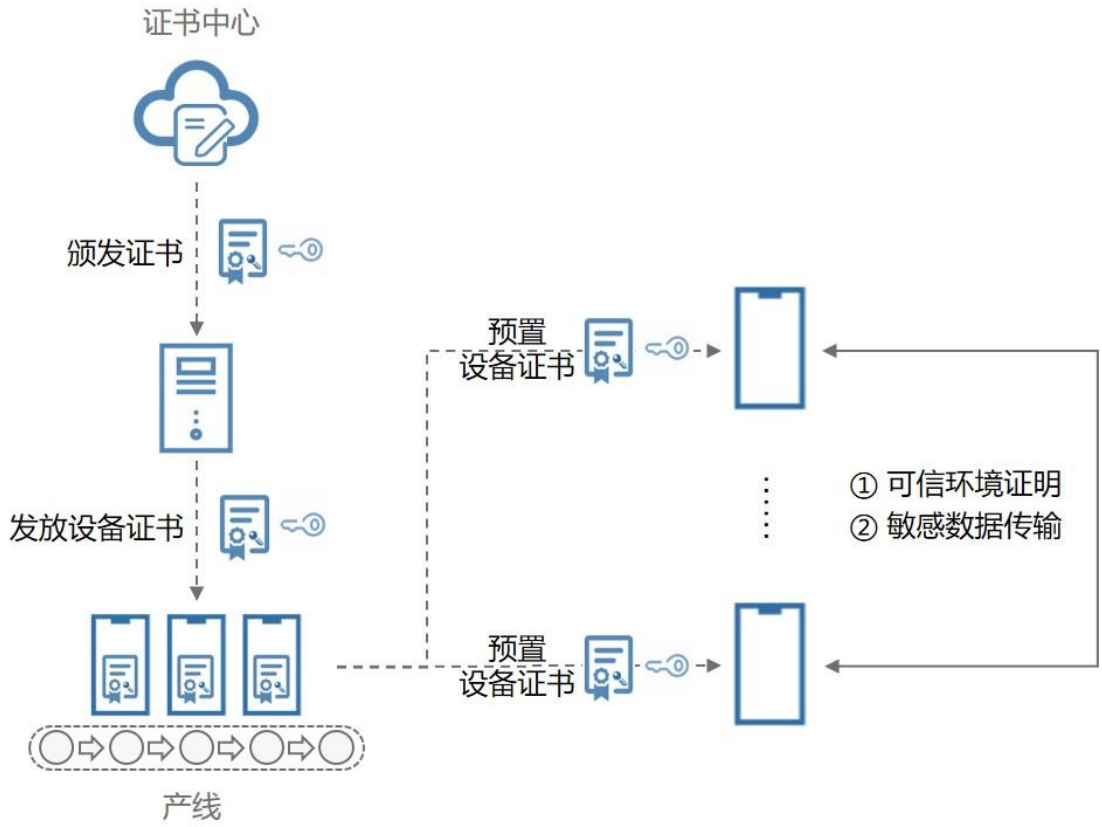
  提供了基于硬件的可信执行环境（TEE，Trusted Execution Environment）来保护用户的个人敏感数据的存储和处理，确保数据不泄露。由于分布式终端硬件的安全能力不同，对于用户的敏感个人数据，需要使用高安全等级的设备进行存储和处理。HarmonyOS 使用基于数学可证明的形式化开发和验证的 TEE 微内核，获得了商用 OS 内核 CC EAL5+的认证评级。

- **设备证书认证**

  支持为具备可信执行环境的设备预置设备证书，用于向其他虚拟终端证明自己的安全能力。对于有 TEE 环境的设备，通过预置 PKI（Public Key Infrastructure）设备证书给设备身份提供证明，确保设备是合法制造生产的。设备证书在产线进行预置，设备证书的私钥写入并安全保存在设备的 TEE 环境中，且只在 TEE 内进行使用。在必须传输用户的敏感

数据（例如密钥、加密的生物特征等信息）时，会在使用设备证书进行安全环境验证后，建立从一个设备的 TEE 到另一设备的 TEE 之间的安全通道，实现安全传输。如图 1 所示。

图 1 设备证书使用示意图



1.4.3 正确地使用数据

在分布式终端场景下，需要确保用户能够正确地使用数据。HarmonyOS 围绕数据的生成、存储、使用、传输以及销毁过程进行全生命周期的保护，从而保证个人数据与隐私、以及系统的机密数据（如密钥）不泄漏。

- **数据生成**：根据数据所在的国家或组织的法律法规与标准规范，对数据进行分类分级，并且根据分类设置相应的保护等级。每个保护等级的数据从生成开始，在其存储、使用、传输的整个生命周期都需要根据对应的安全策略提供不同强度的安全防护。虚拟超级终端的

访问控制系统支持依据标签的访问控制策略，保证数据只能在可以提供足够安全防护的虚拟终端之间存储、使用和传输。

- **数据存储**：HarmonyOS 通过区分数据的安全等级，存储到不同安全防护能力的分区，对数据进行安全保护，并提供密钥全生命周期的跨设备无缝流动和跨设备密钥访问控制能力，支撑分布式身份认证协同、分布式数据共享等业务。
- **数据使用**：HarmonyOS 通过硬件为设备提供可信执行环境。用户的个人敏感数据仅在分布式虚拟终端的可信执行环境中进行使用，确保用户数据的安全和隐私不泄露。
- **数据传输**：为了保证数据在虚拟超级终端之间安全流转，需要各设备是正确可信的，建立了信任关系（多个设备通过华为帐号建立配对关系），并能够在验证信任关系后，建立安全的连接通道，按照数据流动的规则，安全地传输数据。当设备之间进行通信时，需要基于设备的身份凭据对设备进行身份认证，并在此基础上，建立安全的加密传输通道。
- **数据销毁**：销毁密钥即销毁数据。数据在虚拟终端的存储，都建立在密钥的基础上。当销毁数据时，只需要销毁对应的密钥即完成了数据的销毁。

2 开发基础知识

2.1 应用基础知识

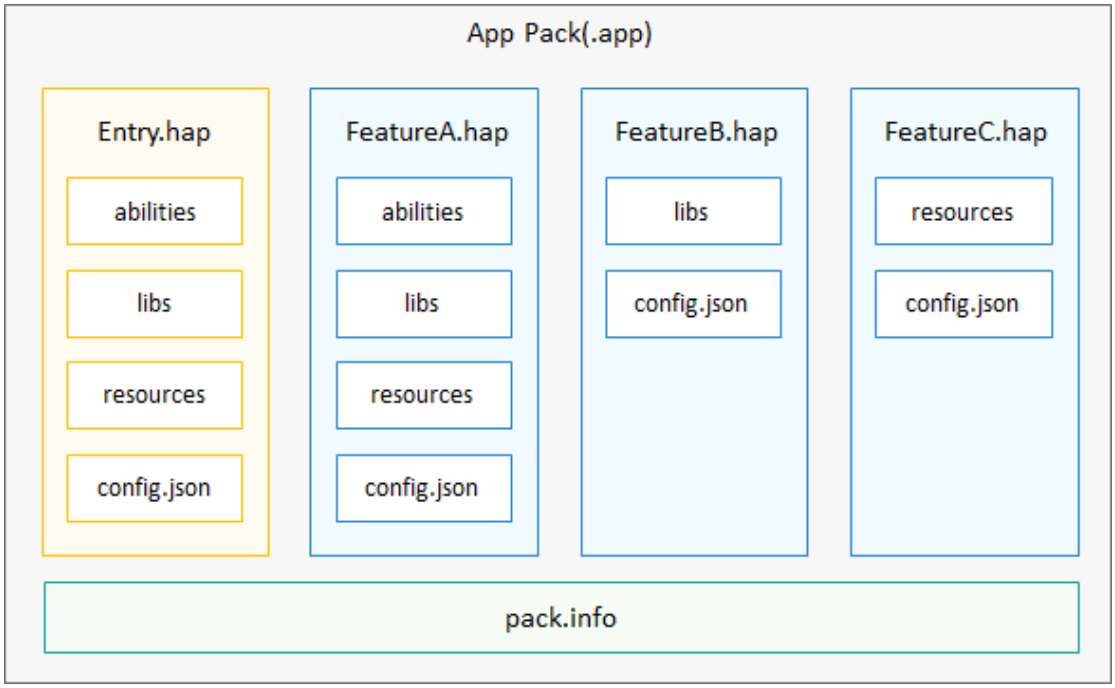
2.1.1 APP

HarmonyOS 的应用软件包以 **APP Pack**（Application Package）形式发布，它是由一个或多个 **HAP**（HarmonyOS Ability Package）以及描述每个 HAP 属性的 pack.info 组成。HAP 是 Ability 的部署包，HarmonyOS 应用代码围绕 Ability 组件展开。

一个 HAP 是由代码、资源、第三方库及应用配置文件组成的模块包，可分为 entry 和 feature 两种模块类型，如图 1 所示。

- **entry**：应用的主模块。一个 APP 中，对于同一设备类型必须有且只有一个 entry 类型的 HAP，可独立安装运行。
- **feature**：应用的动态特性模块。一个 APP 可以包含一个或多个 feature 类型的 HAP，也可以不含。只有包含 Ability 的 HAP 才能够独立运行。

图 1 APP 逻辑视图



2.1.2 Ability

Ability 是应用所具备的能力的抽象，一个应用可以包含一个或多个 Ability。Ability 分为两种类型：FA（Feature Ability）和 PA（Particle Ability）。FA/PA 是应用的基本组成单元，能够实现特定的业务功能。FA 有 UI 界面，而 PA 无 UI 界面。

2.1.3 库文件

库文件是应用依赖的第三方代码形式，存放在 libs 目录，是.so 文件。

2.1.4 资源文件

应用的资源文件（字符串、图片、音频等）存放于 resources 目录下，便于开发者使用和维护，详见资源文件分类。

2.1.5 配置文件

配置文件（config.json）是应用的 Ability 信息，用于声明应用的 Ability，以及应用所需权限等信息，详见应用配置文件。

2.1.6 pack.info

描述应用软件包中每个 HAP 的属性，由 IDE 编译生成，应用市场根据该文件进行拆包和 HAP 的分类存储。HAP 的具体属性包括：

- delivery-with-install：用于标识该 HAP 是否需要在主动安装时进行安装。
- name：HAP 文件名。

- module-type：模块类型，entry 或 feature。
- device-type：用于标识支持该 HAP 运行的设备类型。

2.2 应用配置文件

2.2.1 简介

应用的每个 HAP 的根目录下都存在一个“config.json”配置文件，主要涵盖以下三个方面：

- 应用的全局配置信息，包含应用的包名、生产厂商、版本号等基本信息。
- 应用在具体设备上的配置信息。
- HAP 包的配置信息，包含每个 Ability 必须定义的基本属性（如包名、类名、类型以及 Ability 提供的能力），以及应用访问系统或其他应用受保护部分所需的权限等。

2.2.1.1 文件约定

配置文件“config.json”采用 JSON 文件格式，由属性和值两部分构成：

- **属性**

  属性出现顺序不分先后，且每个属性最多只允许出现一次。

- **值**

  每个属性的值为 JSON 的基本数据类型（数值、字符串、布尔值、数组、对象或者 null 类型）。如果属性值需要引用资源文件，可参见资源文件。

2.2.2 配置文件的元素

此部分提供“config.json”文件中所有属性的详细解释。

2.2.2.1 配置文件的内部结构

应用的配置文件“config.json”中由“app”、“deviceConfig”和“module”三个部分组成，缺一不可。配置文件的内部结构说明参见表 1。

属性名称	含义	数据类型	是否可缺省
app	表示应用的全局配置信息。同一个应用的不同 HAP 包的“app”配置必须保持一致。	对象	否
deviceConfig	表示应用在具体设备上的配置信息。	对象	否
module	表示 HAP 包的配置信息。该标签下的配置只对当前 HAP 包生效。	对象	否

表 1 配置文件的内部结构说明

2.2.2.2 app 对象的内部结构

app 对象包含应用的全局配置信息，内部结构说明参见表 2。

属性名称	子属性名称	含义	数据类型	是否可缺省
bundleName	-	表示应用的包名，用于标识应用的唯一性。 采用反域名形式的字符串表示（例如，com.huawei.himusic）。建议第一级为域名后缀“com”，第二级为厂商/个人名，第三级为应用名，也可以采用多级。支持的字符串长度为 7~127 字节。	字符串	否

属性名称	子属性名称	含义	数据类型	是否可缺省
vendor	-	表示对应用开发厂商的描述。字符串长度不超过 255 字节。	字符串	可缺省，缺省值为空。
version	-	表示应用的版本信息。	对象	否
	code	表示应用的版本号，仅用于 HarmonyOS 管理该应用，对用户不可见。取值为大于零的整数。	数值	否
	name	表示应用的版本号，用于向用户呈现。取值可以自定义。	字符串	否
apiVersion	-	表示应用依赖的 HarmonyOS 的 API 版本。	对象	否
	compatible	表示应用运行需要的 API 最小版本。取值为大于零的整数。	数值	否
	target	表示应用运行需要的 API 目标版本。取值为大于零的整数。	数值	可缺省，缺省值为应用所在设备的当前 API 版本。

表 2 app 对象的内部结构说明

app 示例：

```
{
    "app": {
        "bundleName": "com.huawei.hiworld.example",
        "vendor": "huawei",
        "version": {
            "code": 2,
            "name": "2.0"
        }
        "apiVersion": {
            "compatible": 3,
```

```
            "target": 3
        }
    }
```

2.2.2.3 deviceConfig 对象的内部结构

deviceConfig 包含在具体设备上的应用配置信息，可以包含 default、car、tv、wearable、liteWearable、smartVision 等属性。default 标签内的配置是适用于所有设备通用，其他设备类型如果有特殊的需求，则需要在该设备类型的标签下进行配置。内部结构说明参见表 3。

属性名称	含义	数据类型	是否可缺省
default	表示所有设备通用的应用配置信息。	对象	否
car	表示车机特有的应用配置信息。	对象	可缺省，缺省为空。
tv	表示智慧屏特有的应用配置信息。	对象	可缺省，缺省为空。
wearable	表示智能穿戴特有的应用配置信息。	对象	可缺省，缺省为空。
liteWearable	表示轻量级智能穿戴特有的应用配置信息。	对象	可缺省，缺省为空。
smartVision	表示智能摄像头特有的应用配置信息。	对象	可缺省，缺省为空。

表 3 deviceConfig 对象的内部结构说明

default、car、tv、wearable、liteWearable、smartVision 等对象的内部结构说明，可参见表 4。

属性名称	含义	数据类型	是否可缺省
process	表示应用或者 Ability 的进程名。 如果在“deviceConfig”标签下配置了“process”标签，则该应用的所有 Ability 都运行在这个进程中。 如果在“abilities”标签下也为某个 Ability 配置了“process”标签，则该 Ability 就运行在这个进程中。 该标签仅适用于智慧屏、智能穿戴、车机。	字符串	可缺省，缺省为应用的软件包名。
directLaunch	表示应用是否支持在设备未解锁状态直接启动。如果配置为“true”，则表示应用支持在设备未解锁状态下启动。使用场景举例：应用支持在设备未解锁情况下接听来电。 该标签仅适用于智慧屏、智能穿戴、车机。	布尔类型	可缺省，缺省为 false。
supportBackup	表示应用是否支持备份和恢复。如果配置为“false”，则不支持为该应用执行备份或恢复操作。 该标签仅适用于智慧屏、智能穿戴、车机。	布尔类型	可缺省，缺省为 false。
compressNativeLibs	表示 libs 库是否以压缩存储的方式打包到 HAP 包。如果配置为“false”，则 libs 库以不压缩的方式存储，HAP 包在安装时无需解压 libs，运行时会直接从 HAP 内加载 libs 库。 该标签仅适用于智慧屏、智能穿戴、车机。	布尔类型	可缺省，缺省为 true。
network	表示网络安全性配置。该标签允许应用通过配置文件的安全声明来自定义其网络安全，无需修改应用代码。	对象	可缺省，缺省为空。

表 4 default/car/tv/wearable 等对象的内部结构说明

属性名称	含义	数据类型	是否可缺省
usesCleartext	表示是否允许应用使用明文网络流量（例如，明文 HTTP）。默认值为“false”。	布尔类型	可缺省，缺省为空。

属性名称	含义	数据类型	是否可缺省
	true：允许应用使用明文流量的请求。 false：拒绝应用使用明文流量的请求。		
securityConfig	表示应用的网络安全配置信息。	对象	可缺省，缺省为空。

表 5 network 对象的内部结构说明

属性名称	子属性名称	含义	数据类型	是否可缺省
domainSettings	-	表示自定义的网域范围的安全配置，支持多层嵌套，即一个 domainSettings 对象中允许嵌套更小网域范围的 domainSettings 对象。	对象	可缺省，缺省为空。
	cleartextPermitted	表示自定义的网域范围内是否允许明文流量传输。当 useCleartext 和 securityConfig 同时存在时，自定义网域是否允许明文流量传输以 cleartextPermitted 的取值为准。 true：允许明文流量传输。 false：拒绝明文流量传输。	布尔类型	否
	domains	表示域名配置信息，包含两个参数：subDomains 和 name。 subDomains（布尔类型）：表示是否包含子域名。如果为 “true”，此网域规则将与相应网域及所有子网域（包括子网域的子网域）匹配。否则，该规则仅适用于精确匹配项。	对象数组	否

属性名称	子属性名称	含义	数据类型	是否可缺省
		name（字符串）：表示域名名称。		

表 6 securityConfig 对象的内部结构说明

deviceConfig 示例：

```
"deviceConfig": {
    "default": {
        "process": "com.huawei.hiworld.example",
        "directLaunch": false,
        "supportBackup": false,
        "network": {
            "usesCleartext": true,
            "securityConfig": {
                "domainSettings": {
                    "cleartextPermitted": true,
                    "domains": [
                        {
                            "subDomains": true,
                            "name": "example.ohos.com"
                        }
                    ]
                }
            }
        }
    }
}
```

2.2.2.4 module 对象的内部结构

module 对象包含 HAP 包的配置信息，内部结构说明参见表 7。

属性名称	含义	数据类型	是否可缺省
package	表示 HAP 的包结构名称，在应用内应保证唯一性。采用反向域名格式（建议与 HAP 的工程目录保持一致）。字符串长度不超过 127 字节。 该标签仅适用于智慧屏、智能穿戴、车机。	字符串	否
name	表示 HAP 的类名。采用反向域名方式表示，前缀需要与同级的 package 标签指定的包名一致，也可采用“.”开头的命名方式。字符串长度不超过 255 字节。 该标签仅适用于智慧屏、智能穿戴、车机。	字符串	否
description	表示 HAP 的描述信息。字符串长度不超过 255 字节。如果字符串超出长度或者需要支持多语言，可以采用资源索引的方式添加描述内容。 该标签仅适用于智慧屏、智能穿戴、车机。	字符串	可缺省，缺省值为空。
supportedModes	表示应用支持的运行模式。当前只定义了驾驶模式（drive）。 该标签仅适用于车机。	字符串数组	可缺省，缺省值为空。
deviceType	表示允许 Ability 运行的设备类型。系统预定义的设备类型包括：tv（智慧屏）、car（车机）、wearable（智能穿戴）、liteWearable（轻量级智能穿戴）等。	字符串数组	否
distro	表示 HAP 发布的具体描述。 该标签仅适用于智慧屏、智能穿戴、车机。	对象	否
abilities	表示当前模块内的所有 Ability。采用对象数组格式，其中每个元素表示一个 Ability 对象。	对象数组	可缺省，缺省值为空。
js	表示基于 JS UI 框架开发的 JS 模块集合，其中的每个元素代表一个 JS 模块的信息。	对象	可缺省，缺省值为空。

属性名称	含义	数据类型	是否可缺省
shortcuts	表示应用的快捷方式信息。采用对象数组格式，其中的每个元素表示一个快捷方式对象。	对象数组	可缺省，缺省值为空。
defPermissions	表示应用定义的权限。应用调用者必须申请这些权限，才能正常调用该应用。	对象数组	可缺省，缺省值为空。
reqPermissions	表示应用运行时向系统申请的权限。	对象数组	可缺省，缺省值为空。

表 7 module 对象的内部结构说明

module 示例：

```
"module": {
    "package": "com.example.myapplication.entry",
    "name": ".MyOHOSAbilityPackage",
    "description": "$string:description_application",
    "supportedModes": [
        "drive"
    ],
    ],
    "deviceType": [
        "car"
    ],
    "distro": {
        "deliveryWithInstall": true,
        "moduleName": "ohos_entry",
        "moduleType": "entry"
    },
    "abilities": [
        ...
    ],
    "shortcuts": [
        ...
    ],
```

```
        "js": [
            ...
        ],
        "reqPermissions": [
            ...
        ],
        "defPermissions": [
            ...
        ]
    }
```

属性名称	含义	数据类型	是否可缺省
deliveryWithInstall	表示当前 HAP 是否在支持随应用安装。 true：支持随应用安装。 false：不支持随应用安装。	布尔类型	否
moduleName	表示当前 HAP 的名称。	字符串	否
moduleType	表示当前 HAP 的类型，包括两种类型：entry 和 feature。	字符串	否

表 8 distro 对象的内部结构说明

distro 示例：

```
    "distro": {
        "deliveryWithInstall": true,
        "moduleName": "ohos_entry",
        "moduleType": "entry"
    }
```

属性名称	含义	数据类型	是否可缺省
name	表示 Ability 名称。取值可采用反向域名方式表示，由包名和类名组成，如	字符串	否

属性名称	含义	数据类型	是否可缺省
	“com.example.myapplication.MainAbility”；也可采用“.”开头的类名方式表示，如“.MainAbility”。 该标签仅适用于智慧屏、智能穿戴、车机。		
description	表示对 Ability 的描述。取值可以是描述性内容，也可以是对描述性内容的资源索引，以支持多语言。	字符串	可缺省，缺省值为空。
icon	表示 Ability 图标资源文件的索引。取值示例：$media:ability_icon。 如果在该 Ability 的“skills”属性中，“actions”的取值包含 “action.system.home”，“entities”取值中包含“entity.system.home”，则该 Ability 的 icon 将同时作为应用的 icon。如果存在多个符合条件的 Ability，则取位置靠前的 Ability 的 icon 作为应用的 icon。	字符串	可缺省，缺省值为空。
label	表示 Ability 对用户显示的名称。取值可以是 Ability 名称，也可以是对该名称的资源索引，以支持多语言。 如果在该 Ability 的“skills”属性中，“actions”的取值包含 “action.system.home”，“entities”取值中包含“entity.system.home”，则该 Ability 的 label 将同时作为应用的 label。如果存在多个符合条件的 Ability，则取位置靠前的 Ability 的 label 作为应用的 label。	字符串	可缺省，缺省值为空。
uri	表示 Ability 的统一资源标识符。格式为 [scheme:][//authority][path][?query][#fragment]。	字符串	可缺省，对于 data 类型的 Ability 不可缺省。
launchType	表示 Ability 的启动模式，支持“standard”和“singleton”两种模式： standard：表示该 Ability 可以有多实例。“standard”模式适用于大多数应用场景。	字符串	可缺省，缺省值为 standard。

属性名称	含义	数据类型	是否可缺省
	singleton：表示该 Ability 只可以有一个实例。例如，具有全局唯一性的呼叫来电界面即采用“singleton”模式。 该标签仅适用于智慧屏、智能穿戴、车机。		
visible	表示 Ability 是否可以被其他应用调用。 true：可以被其他应用调用。 false：不能被其他应用调用。	布尔类型	可缺省，缺省值为 false。
permissions	表示其他应用的 Ability 调用此 Ability 时需要申请的权限。通常采用反向域名格式，取值可以是系统预定义的权限，也可以是开发者自定义的权限。如果是自定义权限，取值必须与“defPermissions”标签中定义的某个权限的“name”标签值一致。	字符串数组	可缺省，缺省值为空。
skills	表示 Ability 能够接收的 Intent 的特征。	对象数组	可缺省，缺省值为空。
deviceCapability	表示 Ability 运行时要求设备具有的能力，采用字符串数组的格式表示。	字符串数组	可缺省，缺省值为空。
type	表示 Ability 的类型。取值范围如下： page：表示基于 Page 模板开发的 FA，用于提供与用户交互的能力。 service：表示基于 Service 模板开发的 PA，用于提供后台运行任务的能力。 data：表示基于 Data 模板开发的 PA，用于对外部提供统一的数据访问抽象。	字符串	否
formEnabled	表示 FA 类型的 Ability 是否提供卡片（form）能力。该标签仅适用于 page 类型的 Ability。	布尔类型	可缺省，缺省值为 false。

属性名称	含义	数据类型	是否可缺省
	true：提供卡片能力。 false：不提供卡片能力。		
form	表示 AbilityForm 的属性。该标签仅当“formEnabled”为“true”时，才能生效。	对象	可缺省，缺省值为空。
orientation	表示该 Ability 的显示模式。该标签仅适用于 page 类型的 Ability。取值范围如下： unspecified：由系统自动判断显示方向。 landscape：横屏模式。 portrait：竖屏模式。 followRecent：跟随栈中最近的应用。	字符串	可缺省，缺省值为 unspecified。
backgroundModes	表示后台服务的类型，可以为一个服务配置多个后台服务类型。该标签仅适用于 service 类型的 Ability。取值范围如下： dataTransfer：通过网络/对端设备进行数据下载、备份、分享、传输等业务。 audioPlayback：音频输出业务。 audioRecording：音频输入业务。 pictureInPicture：画中画、小窗口播放视频业务。 voip：音视频电话、VOIP 业务。 location：定位、导航业务。 bluetoothInteraction：蓝牙扫描、连接、传输业务。 wifiInteraction：WLAN 扫描、连接、传输业务。 screenFetch：录屏、截屏业务。	字符串数组	可缺省，缺省值为空。

属性名称	含义	数据类型	是否可缺省
readPermission	表示读取 Ability 的数据所需的权限。该标签仅适用于 data 类型的 Ability。取值为长度不超过 255 字节的字符串。 该标签仅适用于智慧屏、智能穿戴、车机。	字符串	可缺省，缺省为空。
writePermission	表示向 Ability 写数据所需的权限。该标签仅适用于 data 类型的 Ability。取值为长度不超过 255 字节的字符串。 该标签仅适用于智慧屏、智能穿戴、车机。	字符串	可缺省，缺省为空。
directLaunch	表示 Ability 是否支持在设备未解锁状态直接启动。如果配置为“true”，则表示 Ability 支持在设备未解锁状态下启动。 如果“deviceConfig”和“abilities”中同时配置了“directLaunch”，则采用 Ability 对应的取值；如果同时未配置，则采用系统默认值。	布尔值	可缺省，缺省为 false。
configChanges	表示 Ability 关注的系统配置集合。当已关注的配置发生变更后，Ability 会收到 onConfigurationUpdated 回调。取值范围： locale：表示语言区域发生变更。 layout：表示屏幕布局发生变更。 fontSize：表示字号发生变更。 orientation：表示屏幕方向发生变更。 density：表示显示密度发生变更。	字符串数组	可缺省，缺省为空。
mission	表示 Ability 指定的任务栈。该标签仅适用于 page 类型的 Ability。默认情况下应用中所有 Ability 同属一个任务栈。 该标签仅适用于智慧屏、智能穿戴、车机。	字符串	可缺省，缺省为应用的包名。
targetAbility	表示当前 Ability 重用的目标 Ability。该标签仅适用于 page 类型的 Ability。如果配置了 targetAbility 属性，则当前 Ability（即别名 Ability）的属性中仅	字符串	可缺省，缺省值为空。表示当前 Ability 不是一

属性名称	含义	数据类型	是否可缺省
	“name”、“icon”、“label”、“visible”、“permissions”、“skills”生效，其它属性均沿用targetAbility 中的属性值。目标 Ability 必须与别名 Ability 在同一应用中，且在配置文件中目标 Ability 必须在别名之前进行声明。 该标签仅适用于智慧屏、智能穿戴、车机。		个别名 Ability。
multiUserShared	表示 Ability 是否支持多用户状态进行共享，该标签仅适用于 data 类型的 Ability。 配置为“true”时，表示在多用户下只有一份存储数据。需要注意的是，该属性会使 visible 属性失效。 该标签仅适用于智慧屏、智能穿戴、车机。	布尔类型	可缺省，缺省值为 false。
supportPipMode	表示 Ability 是否支持用户进入 PIP 模式（用于在在页面最上层悬浮小窗口，俗称“画中画”，常见于视频播放等场景）。该标签仅适用于 page 类型的 Ability。 该标签仅适用于智慧屏、智能穿戴、车机。	布尔类型	可缺省，缺省值为 false。

表 9 abilities 对象的内部结构说明

abilities 示例：

```
"abilities": [
    {
        "name": ".MainAbility",
        "description": "$string:description_main_ability",
        "icon": "$media:hiworld.png",
        "label": "HiMusic",
        "type": "page",
        "formEnabled": false,
        "launchType": "standard",
        "orientation": "unspecified",
        "permissions": [
        ],
        "visible": false,
        "skills": [
            {
```

```
                    "actions": [
                        "action.system.home"
                    ],
                    "entities": [
                        "entity.system.home"
                    ]
                }
            ],
            "configChanges": [
                "locale",
                "layout",
                "fontSize",
                "orientation"
            ],
            "directLaunch": false,
            "process": "string",
            "backgroundModes": [
                "dataTransfer",
                "audioPlayback",
                "audioRecording",
                "pictureInPicture",
                "voip",
                "location",
                "bluetoothInteraction",
                "wifiInteraction",
                "screenFetch"
            ],
        }
    ]
```

属性名称	子属性名称	含义	数据类型	是否可缺省
actions	-	表示能够接收的 Intent 的 action 值，可以包含一个或多个 action。 取值通常为系统预定义的 action 值，详见《API 参考》中的 ohos.aafwk.content.Intent 类。	字符串数组	可缺省，缺省值为空。

属性名称	子属性名称	含义	数据类型	是否可缺省
entities	-	表示能够接收的 Intent 的 Ability 的类别（如视频、桌面应用等），可以包含一个或多个 entity。 取值通常为系统预定义的类别，详见《API 参考》中的 ohos.aafwk.content.Intent 类，也可以自定义。	字符串数组	可缺省，缺省值为空。
uris	-	表示能够接收的 Intent 的 uri，可以包含一个或者多个 uri。	对象数组	可缺省，缺省值为空。
	scheme	表示 uri 的 scheme 值。	字符串	不可缺省。
	host	表示 uri 的 host 值。	字符串	可缺省，缺省值为空。
	port	表示 uri 的 port 值。	字符串	可缺省，缺省值为空。
	path	表示 uri 的 path 值。	字符串	可缺省，缺省值为空。
	type	表示 uri 的 type 值。	字符串	可缺省，缺省值为空。

表 10 skills 对象的内部结构说明

skills 示例：

```
1.    "skills": [
2.        {
3.            "actions": [
4.                "action.system.home"
5.            ],
6.            "entities": [
7.                "entity.system.home"
8.            ],
9.            "uris": [
```

```
                {
                    "scheme": "http",
                    "host": "www.xxx.com",
                    "port": "8080",
                    "path": "query/student/name",
                    "type": "text"
                }
            ]
        }
    ]
```

属性名称	含义	数据类型	是否可缺省
formEntity	表示 AbilityForm 支持的显示方式，当前支持的位置包括： homeScreen：以桌面图标显示。 searchbox：在全局搜索显示。	字符串数组	可缺省，缺省值为空。
minHeight	表示 AbilityForm 缩放时能达到的最小高度，单位：像素。	数值	可缺省，缺省值为 0。
defaultHeight	表示 AbilityForm 的默认高度，单位：像素。Form 使用方应当根据该值为 Form 申请相应高度的容器布局。	数值	可缺省，缺省值为 0。
minWidth	表示 AbilityForm 缩放时能达到的最小宽度，单位：像素。	数值	可缺省，缺省值为 0。
defaultWidth	表示 AbilityForm 的默认宽度，单位：像素。Form 使用方应当根据该值为 Form 申请相应宽度的容器布局。	数值	可缺省，缺省值为 0。

表 11 form 对象的内部结构说明

form 示例：

```
"form": {
    "formEntity": [
        "homeScreen",
        "searchbox"
    ],
    "minHeight": 100,
```

```
7.          "maxHeight": 200,
8.          "minWidth": 100,
9.          "maxWidth": 200
10.   }
```

属性名称	子属性名称	含义	数据类型	是否可缺省
name	-	表示 JS Module 的名字。该标签不可缺省，默认值为 default。	字符串	否
pages	-	表示 JS Module 的页面用于列举 JS Module 中每个页面的路由信息[页面路径+页面名称]。该标签不可缺省，取值为数组，数组第一个元素代表 JS FA 首页。	数组	否
window	-	用于定义与显示窗口相关的配置。 该标签仅适用于智慧屏、智能穿戴、车机。	对象	可缺省。
	designWidth	表示页面设计基准宽度。以此为基准，根据实际设备宽度来缩放元素大小。	数值	可缺省，缺省值为 750px。
	autoDesignWidth	表示页面设计基准宽度是否自动计算。当配置为 true 时，designWidth 将会被忽略，设计基准宽度由设备宽度与屏幕密度计算得出。	布尔类型	可缺省，缺省值为 false。

表 12 js 对象的内部结构说明

js 示例：

```
1.    "js": [
2.          {
3.                "name": "default",
4.                "pages": [
5.                      "pages/index/index",
```

```
6.                  "pages/detail/detail"
7.              ],
8.              "window": {
9.                  "designWidth": 750,
10.                 "autoDesignWidth": false
11.             }
12.         }
13.     ]
```

属性名称	子属性名称	含义	数据类型	是否可缺省
shortcutId	-	表示快捷方式的 ID。字符串的最大长度为 63 字节。	字符串	否
label	-	表示快捷方式的标签信息，即快捷方式对外显示的文字描述信息。取值可以是描述性内容，也可以是标识 label 的资源索引。字符串最大长度为 63 字节。	字符串	可缺省，缺省为空。
intents	-	表示快捷方式内定义的目标 intent 信息集合，每个 intent 可配置两个子标签，targetClass，targetBundle。	-	可缺省，缺省为空。
	targetClass	表示快捷方式目标类名。	字符串	可缺省，缺省值为空。
	targetBundle	表示快捷方式目标 Ability 所在应用的包名。	字符串	可缺省，缺省值为空。

表 13 shortcuts 对象的内部结构说明

示例：

```
1.    "shortcuts": [
2.        {
```

```
            "shortcutId": "id",
            "label": "$string:shortcut",
            "intents": [
                {
                    "targetBundle": "com.huawei.hiworld.himusic",
                    "targetClass": "com.huawei.hiworld.himusic.entry.MainAbility"
                }
            ]
        }
    ]
```

2.2.3 配置文件示例

以 JSON 文件为 config.json 的一个简单示例，该示例的应用声明为三个 Ability。

```
{
    "app": {
        "bundleName": "com.huawei.hiworld.himusic",
        "vendor": "huawei",
        "version": {
            "code": 2,
            "name": "2.0"
        }
        "apiVersion": {
            "compatible": 3,
            "target": 3
        }
    },
    "deviceConfig": {
        "default": {
        }
    },
    "module": {
        "package": "com.huawei.hiworld.himusic.entry",
        "name": ".MainApplication",
        "supportedModes": [
            "drive"
        ],
        "distro": {
            "moduleType": "entry",
            "deliveryWithInstall": true,
            "moduleName": "hap-car"
```

```
    },
    "deviceType": [
        "car"
    ],

    "abilities": [
        {
            "name": ".MainAbility",
            "description": "himusic main ability",
            "icon": "$media:ic_launcher",
            "label": "HiMusic",
            "launchType": "standard",
            "orientation": "unspecified",
            "visible": true,
            "skills": [
                {
                    "actions": [
                        "action.system.home"
                    ],
                    "entities": [
                        "entity.system.home"
                    ]
                }
            ],
            "type": "page",
            "formEnabled": false
        },
        {
            "name": ".PlayService",
            "description": "himusic play ability",
            "icon": "$media:ic_launcher",
            "label": "HiMusic",
            "launchType": "standard",
            "orientation": "unspecified",
            "visible": false,
            "skills": [
                {
                    "actions": [
                        "action.play.music",
                        "action.stop.music"
                    ],
                    "entities": [
                        "entity.audio"
                    ]
```

```
                        }
                    ],
                    "type": "service",
                    "formEnabled": false,
                    "backgroundModes": [
                        "audioPlayback"
                    ]
                },
                {
                    "name": ".UserADataAbility",
                    "type": "data",
                    "uri": "dataability://com.huawei.hiworld.himusic.UserADataAbility",
                    "visible": true
                }
            ],
            "reqPermissions": [{
                    "name": "ohos.permission.DISTRIBUTED_DATASYNC",
                    "reason": "",
                    "usedScene": {
                        "ability": [
                            "com.huawei.hiworld.himusic.entry.MainAbility",
                            "com.huawei.hiworld.himusic.entry.PlayService"
                        ],
                        "when": "inuse"
                    }
                }
            ]
        }
}
```

2.3 资源文件

2.3.1 资源文件分类

2.3.1.1 resources 目录

应用的资源文件（字符串、图片、音频等）统一存放于 resources 目录下，便于开发者使用和维护。resources 目录包括两大类目录，一类为 base 目录与限定词目录，另一类为 rawfile 目录，详见表 1。

资源目录示例：

```
resources
|---base   // 默认存在的目录
|    |---element
|    |    |---string.json
|    |---media
|    |    |---icon.png
|---en_GB-vertical-car-mdpi // 限定词目录示例，需要开发者自行创建
|    |---element
|    |    |---string.json
|    |---media
|    |    |---icon.png
|---rawfile   // 默认存在的目录
```

分类	base 目录与限定词目录	rawfile 目录
组织形式	按照两级目录形式来组织，目录命名必须符合规范，以便根据设备状态去匹配相应目录下的资源文件。 一级子目录为 **base 目录**和 **限定词目录**。 base 目录是默认存在的目录。当应用的 resources 资源目录中没有与设备状态匹配的限定词目录时，会自动引用该目录中的资源文件。	支持创建多层子目录，目录名称可以自定义，文件夹内可以自由放置各类资源文件。 rawfile 目录的文件不会根据设备状态去匹配不同的资源。

分类	base 目录与限定词目录	rawfile 目录
	限定词目录需要开发者自行创建。目录名称由一个或多个表征应用场景或设备特征的限定词组合而成，具体要求参见限定词目录。 二级子目录为**资源目录**，用于存放字符串、颜色、布尔值等基础元素，以及媒体、动画、布局等资源文件，具体要求参见资源组目录。	
编译方式	目录中的资源文件会被编译成二进制文件，并赋予资源文件ID。	目录中的资源文件会被直接打包进应用，不经过编译，也不会被赋予资源文件 ID。
引用方式	通过文件类型（type）和资源名称（name）的组合引用。 Java 文件采用：**ResourceTable.*type_name***。 特别地，如果引用的是系统资源，则采用：**ohos.global.systemres.ResourceTable.*type_name***。 XML 文件采用：**$*type:name***。 特别地，如果引用的是系统资源，则采用：**$ohos:*type:name***。	通过指定文件路径和文件名来引用。 例如： “resources/rawfile/example.js”。

表 1 resources 目录分类

2.3.1.2 限定词目录

限定词目录可以由一个或多个表征应用场景或设备特征的限定词组合而成，包括语言、文字、国家或地区、横竖屏、设备类型和屏幕密度等六个维度，限定词之间通过下划线（_）或者中划线（-）连接。开发者在创建限定词目录时，需要掌握限定词目录的命名要求以及与限定词目录与设备状态的匹配规则。

限定词目录的命名要求

- 限定词的组合顺序：*语言_文字_国家或地区-横竖屏-设备类型-屏幕密度*。开发者可以根据应用的使用场景和设备特征，选择其中的一类或几类限定词组成目录名称。
- 限定词的连接方式：语言、文字、国家或地区之间采用下划线（_）连接，除此之外的其他限定词之间均采用中划线（-）连接。例如：**zh_Hant_CN**、**zh_CN-car-ldpi**。
- 限定词的取值范围：每类限定词的取值必须符合表 2 中的条件，否则，将无法匹配目录中的资源文件。

限定词类型	含义与取值说明
语言	表示设备使用的语言类型，由 2 个小写字母组成。例如：zh 表示中文，en 表示英语。 详细取值范围，参见 **ISO 639-1**（ISO 制定的语言编码标准）。
文字	表示设备使用的文字类型，由 1 个大写字母（首字母）和 3 个小写字母组成。例如：Hans 表示简体中文，Hant 表示繁体中文。 详细取值范围，参见 **ISO 15924**（ISO 制定的文字编码标准）。
国家或地区	表示用户所在的国家或地区，由 2~3 个大写字母或者 3 个数字组成。例如：CN 表示中国，GB 表示英国。 详细取值范围，参见 **ISO 3166-1**（ISO 制定的国家和地区编码标准）。
横竖屏	表示设备的屏幕方向，取值如下： vertical：竖屏 horizontal：横屏
设备类型	表示设备的类型，取值如下： car：车机 tv：智慧屏 wearable：智能穿戴
屏幕密度	表示设备的屏幕密度（单位为 dpi），取值如下： sdpi：表示小规模的屏幕密度（Small-scale Dots Per Inch），适用于 120dpi 及以下的设备。

限定词类型	含义与取值说明
	mdpi：表示中规模的屏幕密度（Medium-scale Dots Per Inch），适用于 120dpi~160dpi 的设备。 ldpi：表示大规模的屏幕密度（Large-scale Dots Per Inch），适用于 160dpi~240dpi 的设备。 xldpi：表示特大规模的屏幕密度（Extra Large-scale Dots Per Inch），适用于 240dpi~320dpi 的设备。 xxldpi：表示超大规模的屏幕密度（Extra Extra Large-scale Dots Per Inch），适用于 320dpi~480dpi 的设备。 xxxldpi：表示超特大规模的屏幕密度（Extra Extra Extra Large-scale Dots Per Inch），适用于 480dpi~640dpi 的设备。

表 2 限定词取值要求

限定词目录与设备状态的匹配规则

- 在为设备匹配对应的资源文件时，限定词目录匹配的优先级从高到低依次为：区域（语言_文字_国家或地区）> 横竖屏 > 设备类型 > 屏幕密度。
- 如果限定词目录中包含**语言、文字、横竖屏、设备类型**限定词，则对应限定词的取值必须与当前的设备状态完全一致，该目录才能够参与设备的资源匹配。例如，限定词目录“zh_CN-car-ldpi”不能参与“en_US”设备的资源匹配。

2.3.1.3 资源组目录

base 目录与限定词目录下面可以创建资源组目录（包括 element、media、animation、layout、graphic、profile），用于存放特定类型的资源文件，详见表 3。

资源组目录	目录说明	资源文件
element	表示元素资源，以下每一类数据都采用相应的 JSON 文件来表征。 boolean，布尔型 color，颜色 float，浮点型 intarray，整型数组 integer，整型 pattern，样式 plural，复数形式 strarray，字符串数组 string，字符串	element 目录中的文件名称建议与下面的文件名保持一致。每个文件中只能包含同一类型的数据。 boolean.json color.json float.json intarray.json integer.json pattern.json plural.json strarray.json string.json
media	表示媒体资源，包括图片、音频、视频等非文本格式的文件。	文件名可自定义，例如：icon.png。
animation	表示动画资源，采用 XML 文件格式。	文件名可自定义，例如：zoom_in.xml。
layout	表示布局资源，采用 XML 文件格式。	文件名可自定义，例如：home_layout.xml。
graphic	表示可绘制资源，采用 XML 文件格式。	文件名可自定义，例如：notifications_dark.xml。
profile	表示其他类型文件，以原始文件形式保存。	文件名可自定义。

表 3 资源组目录 说明

2.3.1.4 系统资源文件

目前支持的系统资源文件详见表 4。

系统资源名称	含义	类型
ic_app	表示 HarmonyOS 应用的默认图标。	媒体
request_location_reminder_title	表示“请求使用设备定位功能”的提示标题。	字符串
request_location_reminder_content	表示“请求使用设备定位功能”的提示内容，即：请在下拉快捷栏打开"位置信息"开关。	字符串

表 4 系统资源文件说明

2.3.2 资源文件示例

2.3.2.1 boolean.json 示例

```
{
    "boolean":[
        {
            "name":"boolean_1",
            "value":true
        },
        {
            "name":"boolean_ref",
            "value":"$boolean:boolean_1"
        }
    ]
}
```

2.3.2.2 color.json 示例

```
{
```

```
    "color":[
        {
            "name":"red",
            "value":"#ff0000"
        },
        {
            "name":"red_ref",
            "value":"$color:red"
        }
    ]
}
```

float.json 示例

```
{
    "float":[
        {
            "name":"float_1",
            "value":"30.6"
        },
        {
            "name":"float_ref",
            "value":"$float:float_1"
        },
        {
            "name":"float_px",
            "value":"100px"
        }
    ]
}
```

intarray.json 示例

```
{
    "intarray":[
        {
            "name":"intarray_1",
            "value":[
                100,
                200,
                "$integer:integer_1"
            ]
        }
    ]
}
```

2.3.2.3 integer.json 示例

```
{
    "integer":[
        {
            "name":"integer_1",
            "value":100
        },
        {
            "name":"integer_ref",
            "value":"$integer:integer_1"
        }
    ]
}
```

2.3.2.4 pattern.json 示例

```
{
    "pattern":[
        {
            "name":"base",
            "value":[
                {
                    "name":"width",
                    "value":"100vp"
                },
                {
                    "name":"height",
                    "value":"100vp"
                },
                {
                    "name":"size",
                    "value":"25px"
                }
            ]
        },
        {
            "name":"child",
            "parent":"base",
            "value":[
                {
                    "name":"noTitile",
```

```
                "value":"Yes"
            }
        ]
    }
]
}
```

2.3.2.5 plural.json 示例

```
{
    "plural":[
        {
            "name":"eat_apple",
            "value":[
                {
                    "quantity":"one",
                    "value":"%d apple"
                },
                {
                    "quantity":"other",
                    "value":"%d apples"
                }
            ]
        }
    ]
}
```

2.3.2.6 strarray.json 示例

```
{
    "strarray":[
        {
            "name":"size",
            "value":[
                {
                    "value":"small"
                },
                {
                    "value":"$string:hello"
                },
                {
                    "value":"large"
                },
```

```
                {
                    "value":"extra large"
                }
            ]
        }
    ]
}
```

2.3.2.7 string.json 示例

```
{
    "string":[
        {
            "name":"hello",
            "value":"hello base"
        },
        {
            "name":"app_name",
            "value":"my application"
        },
        {
            "name":"app_name_ref",
            "value":"$string:app_name"
        },
        {
            "name":"app_sys_ref",
            "value":"$ohos:string:request_location_reminder_title"
        }
    ]
}
```

2.4 应用数据管理

HarmonyOS 应用数据管理支撑单设备的各种结构化数据的持久化，以及跨设备之间数据的同步、共享以及搜索功能。开发者通过应用数据管理，能够方便地完成应用程序数据在不同终端设备间的无缝衔接，满足用户跨设备使用数据的一致性体验。

2.4.1 本地应用数据管理

提供单设备上结构化数据的存储和访问能力。使用 SQLite 作为持久化存储引擎，提供了多种类型的本地数据库，分别是关系型数据库（Relational Database）和对象映射关系型数据库（Object Relational Mapping Database），此外还提供一种轻量级偏好数据库（Light Weight Preference Database），用以满足开发人员使用不同数据模型对应用数据进行持久化和访问的需求。

有关于本地应用数据管理的详细信息，请参阅关系型数据库、对象映射关系型数据库和轻量级偏好数据库。

2.4.2 分布式数据服务

分布式数据库支持用户数据跨设备相互同步，为用户提供在多种终端设备上一致的数据访问体验。通过调用分布式数据接口，应用可以将数据保存到分布式数据库中。通过结合帐号、应用唯一标识和数据库三元组，分布式数据库对属于不同应用的数据进行隔离。

有关于分布式数据库的详细信息，请参阅分布式数据服务。

2.4.3 分布式文件服务

在多个终端设备间为单个设备上应用程序创建的文件提供多终端的分布式共享能力。每台设备上都存储一份全量的文件元数据，应用程序通过文件元数据中的路径，可以实现同一应用文件的跨设备访问。

有关于分布式文件的详细信息，请参阅分布式文件服务。

2.4.4 数据搜索服务

在单个设备上，为应用程序提供搜索引擎级的全文索引管理、建立索引和搜索功能。

有关于数据搜索的详细信息，请参阅融合搜索。

2.4.5 数据存储管理

为应用开发者提供系统存储路径、存储设备列表，存储设备属性的查询和管理功能。

有关于数据存储的详细信息，请参阅数据存储管理。

2.5 应用权限管理

HarmonyOS 中所有的应用均在应用沙盒内运行。默认情况下，应用只能访问有限的系统资源，系统负责管理应用对资源的访问权限。

应用权限管理是由接口提供方（Ability）、接口使用方（应用）、系统（包括云侧和端侧）以及用户等多方共同参与的整个流程，保证受限接口是在约定好的规则下被正常使用，避免接口被滥用而导致用户、应用和设备受损。

2.5.1 权限声明

- 应用需要在 config.json 中使用“reqPermissions”属性对需要的权限逐个进行声明。
- 若使用到的三方库也涉及权限使用，也需统一在应用的 config.json 中逐个声明。
- 没有在 config.json 中声明的权限，应用就无法获得此权限的授权。

2.5.2 动态申请敏感权限

动态申请敏感权限基于用户可知可控的原则，需要应用在运行时主动调用系统动态申请权限的接口，系统弹框由用户授权，用户结合应用运行场景的上下文，识别出应用申请相应敏感权限的合理性，从而做出正确的选择。

即使用户向应用授予了请求的权限，应用在调用受此权限管控的接口前，也应该先检查自己有无此权限，而不能把之前授予的状态持久化，因为用户在动态授予后还可以通过设置取消应用的权限。

有关于应用动态申请敏感权限的详细信息，请参阅动态申请权限。

2.5.3 自定义权限

HarmonyOS 为了保证应用对外提供的接口不被恶意调用，需要对调用接口的调用者进行鉴权。

大多情况下，系统已定义的权限满足了应用的基本需要，若有特殊的访问控制需要，应用可在 config.json 中以"defPermissions": []属性来定义新的权限，并通过“availableScope”和“grantMode”两个属性分别确定权限的开放范围和授权方式，使得权限定义更加灵活且易于理解。有关 HarmonyOS 权限开放范围和授权方式详细的描述，请参阅权限授予方式字段说明和权限限制范围字段说明。

为了避免应用自定义新权限出现重名的情况，建议应用对新权限的命名以包名的前两个字段开头，这样可以防止不同开发者的应用间出现自定义权限重名的情况。

2.5.4 权限保护方法

- 保护 Ability：通过在 config.json 里对应的 Ability 中配置"permissions": ["*权限名*"]属性，即可实现保护整个 Ability 的目的，无指定权限的应用不能访问此 Ability。
- 保护 API：若 Ability 对外提供的数据或能力有多种，且开放范围或保护级别也不同，可以针对不同的数据或能力在接口代码实现中通过 verifyPermission(String permissionName, int pid, int uid)来对 uid 标识的调用者进行鉴权。

2.5.5 权限使用原则

- 权限申请最小化。跟用户提供的功能无关的权限，不要申请；尽量采用其他无需权限的操作来实现相应功能（如：通过 intent 拉起系统 UI 界面由用户交互、应用自己生成 uuid 代替设备 ID 等）。
- 权限申请完整。应用所需权限（包括应用调用到的三方库依赖的权限）都要逐个在应用的 config.json 中按格式声明。
- 满足用户可知。应用申请的敏感权限的目的需要真实准确告知用户。
- 权限就近申请。应用在用户触发相关业务功能时，就近提示用户授予实现此功能所需的权限。

- 权限不扩散。在用户未授权的情况下，不允许提供给其他应用使用。
- 应用自定义权限防止重名。建议以包名为前缀来命名权限，防止跟系统定义的权限重名。

2.6 应用隐私保护

随着移动终端及其相关业务（如移动支付、终端云等）的普及，用户隐私保护的重要性愈发突出。应用开发者在产品设计阶段就需要考虑保护的用户隐私，提高应用的安全性。HarmonyOS 应用开发需要遵从其隐私保护规则，在应用上架应用市场时，应用市场会根据规则进行校验，如不满足条件则无法上架。

2.6.1 数据收集及使用公开透明

应用采集个人数据时，应清晰、明确地告知用户，并确保告知用户的个人信息将被如何使用。

- 应用申请操作系统受限权限和敏感权限时，需要明确告知用户权限申请的目的和用途，并获取用户的同意。受限权限 API 使用方案请参考权限章节。详细的 UX 设计方案请参考 UX 设计隐私方案。

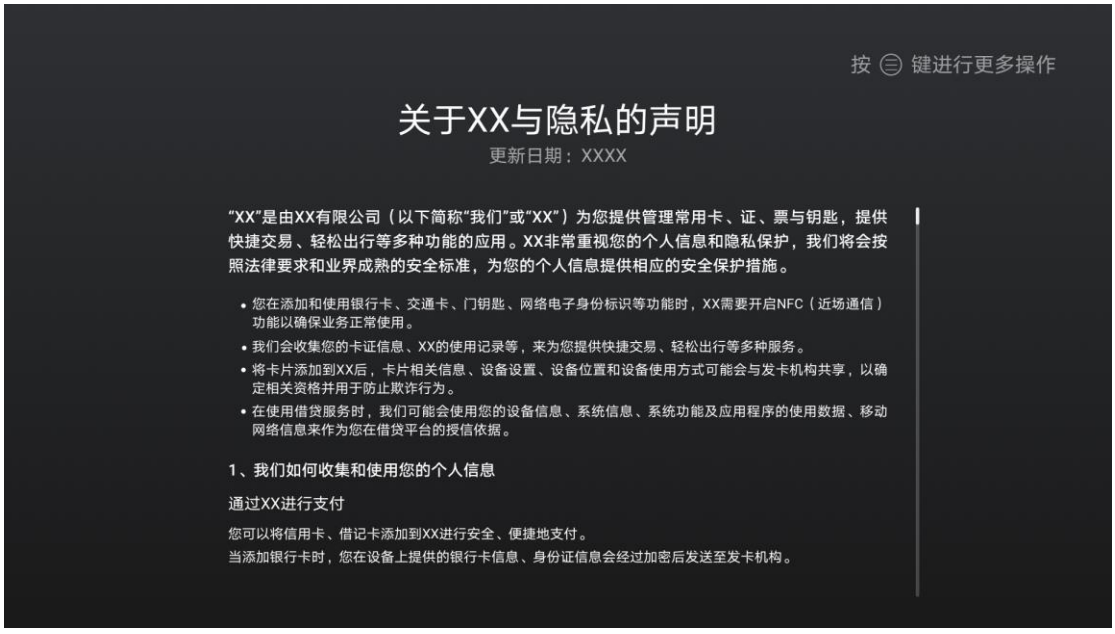
  图 1 敏感权限获取弹框示例

- 开发者应制定并遵从适当的隐私政策，在收集、使用留存和第三方分享用户数据时需要符合所有适用法律、政策和规定。需充分告知用户处理个人数据的种类、目的、处理方式、保留期限等，满足数据主体权利等要求。

  根据以上原则，我们设计了示例以供参考。隐私通知/声明的参考示例如下：

图 2 应用隐私通知示例图

图 3 应用隐私声明示例图



- 个人数据应当基于具体、明确、合法的目的收集，不应以与此目的不相符的方式作进一步处理。对于收集目的变更和用户撤销同意后再次使用的场景都需要用户重新同意。隐私声明变更示例图，隐私声明撤销同意示例图所示。

  图 4 隐私声明变更示例图


隐私声明变更通知
为提供更好的服务，我们更新了隐私声明。
详情内容请参阅最新的 关于XX与隐私的声明
取消
同意

图 5 撤销同意示例图


按 ☰ 键进行更多操作
关于XX与隐私的声明
更新日期：XXXX
"XX"是由XX有限公司（以下简称"我们"或"XX"）为您提供管理常用卡、证、票与钥匙，提供快捷交易、轻松出行等多种功能的应用。XX非常重视您的个人信息和隐私保护，我们将会按照法律要求和业界成熟
护措施。
查看更多
不同意本声明
• 您在添加和使用银行
要开启NFC（近场通信）功能以确保业务正常使用
• 我们会收集您的卡证信
等多种服务。
• 将卡片添加到XX后，
会与发卡机构共享，以确定相关资格并用于防止欺诈行为。
• 在使用借贷服务时，我们可能会使用您的设备信息、系统信息、系统功能及应用程序的使用数据、移动网络信息来作为您在借贷平台的授信依据。
1、我们如何收集和使用您的个人信息
通过XX进行支付
您可以将信用卡、借记卡添加到XX进行安全、便捷地支付。
当添加银行卡时，您在设备上提供的银行卡信息、身份证信息会经过加密后发送至发卡机构。

- 应用的隐私声明应覆盖本应用所有收集的个人数据。
- 有 UI 的 Ability 运行时需要在明显位置展示 Ability 的功能名称及开发者名称/logo。
- 应用的隐私声明应在应用首次启动时通过弹框等明显的方式展示给用户，并提供用户查看隐私声明的入口。
- 调用第三方 Ability 时，需要明确调用方与被调用方履行的隐私责任，并在声明弹框中告知数据主体相关隐私权责。
- 调用第三方 Ability 时，如涉及个人数据的分享，调用方需在隐私声明中说明分享的数据类型和数据接收者的类型。

2.6.2 数据收集及使用最小化

应用个人数据收集应与数据处理目的相关，且是适当、必要的。开发者应尽可能对个人数据进行匿名或化名，降低数据主体的风险。仅可收集和处理与特定目的相关且必需的个人数据，不能对数据做出与特定目的不相关的进一步处理。

- 敏感权限申请的时候要满足权限最小化的要求，在进行权限申请时，只申请获取必需的信息或资源所需要的权限。
- 应用针对数据的收集要满足最小化要求，不收集与应用提供服务无关联的数据。
- 数据使用的功能要求能够使用户受益，收集的数据不能用于与用户正常使用无关的功能。

2.6.3 数据处理选择和控制

对个人数据处理必须要征得用户的同意，用户对其个人数据要有充分的控制权。

- 应用申请使用系统权限：应用弹窗提醒，向用户呈现应用需要获取的权限和权限使用目的、应用需要收集的数据和使用目的等，通过用户点击“确认”的方式完成用户授权，让用户对应用权限的授予和使用透明、可知、可控。
- 用户可以修改、取消授予应用的权限：当用户不同意某一权限或者数据收集时，应当允许用户使用与这部分权限和数据收集不相关的功能。
- 在进入应用的主界面之前不建议直接弹窗申请敏感权限，仅在用户使用功能时才请求对应的权限。
- 系统对于用户的敏感数据和系统关键资源的获取设置了对应的权限，应用访问这些数据时需要申请对应的权限。相关权限列表请参考应用权限列表章节。

2.6.4 数据安全

从技术上保证数据处理活动的安全性，包括个人数据的加密存储、安全传输等安全机制，

系统应默认开启或采取安全保护措施。

- **数据存储**

1. 应用产生的密钥以及用户的敏感数据需要存储在应用的私有目录下，敏感数据定义可参考数据分类分级标准。
2. 应用可以调用系统提供的本地数据库 RdbStore 的加密接口对敏感数据进行加密存储。接口详见关系型数据库章节。
3. 应用产生的分布式数据可以调用系统的分布式数据库进行存储，对于敏感数据需要采用分布式数据库提供的加密接口进行加密，接口详见分布式数据服务章节。

- **安全传输**

需要分别针对本地传输和远程传输采取不同的安全保护措施。

本地传输：

1. 应用通过 intent 跨应用传输数据时避免包含敏感数据，**intent scheme url 协议**使用过程中加入安全限制，防止 UXSS 等安全问题。
2. 应用内组件调用应采用安全方式，避免通过隐式方式进行调用组件，防止组件劫持。
3. 避免使用 socket 方式进行本地通信，如需使用，localhost 端口号随机生成，并对端口连接对象进行身份认证和鉴权。

4. 本地 IPC 通信安全：作为服务提供方需要校验服务使用方的身份和访问权限，防止服务使用方进行身份仿冒或者权限绕过。

   远程传输：

5. 使用 https 代替 http 进行通信，并对 https 证书进行严格校验。
6. 避免进行远程端口进行通信，如需使用，需要对端口连接对象进行身份认证和鉴权。
7. 应用进行跨设备通信时，需要校验被访问设备和应用的身份信息，防止被访问方的设备和应用进行身份仿冒。
8. 应用进行跨设备通信时，作为服务提供方需要校验服务使用方的身份和权限，防止服务使用方进行身份仿冒或者权限绕过。

2.6.5 本地化处理

应用开发的数据优先在本地进行处理，对于本地无法处理的数据上传云服务要满足最小化的原则，不能默认选择上传云服务。

2.6.6 未成年人数据保护要求

如果应用是针对未成年人设计的，或者应用通过收集的用户年龄数据识别出用户是未成年人，开发者应该结合目标市场国家的相关法律，专门分析未成年人个人数据保护的问题。

收集未成年人数据前需要征得监护人的同意。

3 快速入门

3.1 简介

本文档适用于 HarmonyOS 应用开发的初学者。编写两个简单的页面，实现在第一个页面点击按钮跳转到第二个页面。

说明

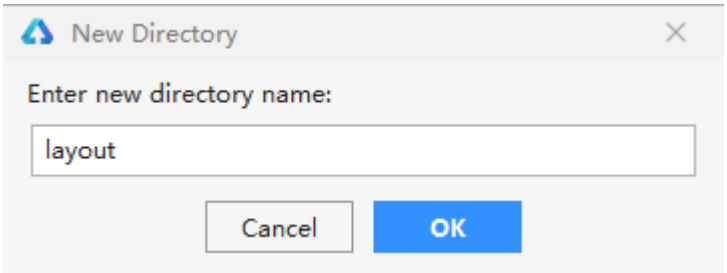
开始前，请参考 DevEco Studio 快速开始完成环境搭建、创建并运行一个项目。

3.2 编写第一个页面

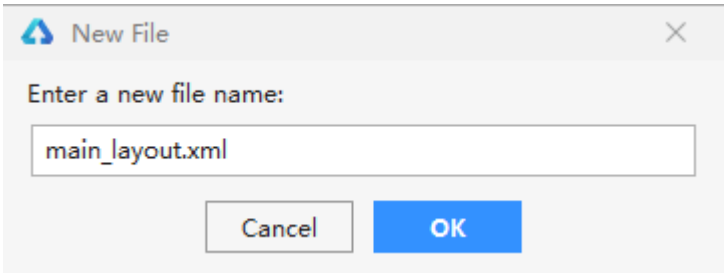
在 Java UI 框架中，提供了两种编写布局的方式：在 XML 中声明 UI 布局和在代码中创建布局。这两种方式创建出的布局没有本质差别，为了熟悉两种方式，我们将通过 XML 的方式编写第一个页面，通过代码的方式编写第二个页面。

3.2.1 XML 编写页面

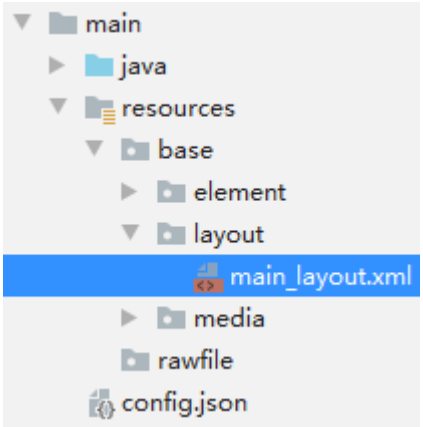
1. 在“Project”窗口，打开“entry > src > main > resources > base”，右键点击“base”文件夹，选择“New > Directory”，命名为“layout”。



2. 右键点击“layout”文件夹，选择“New > File”，命名为“main_layout.xml”。



在“layout”文件夹下可以看到新增了“main_layout.xml”文件。

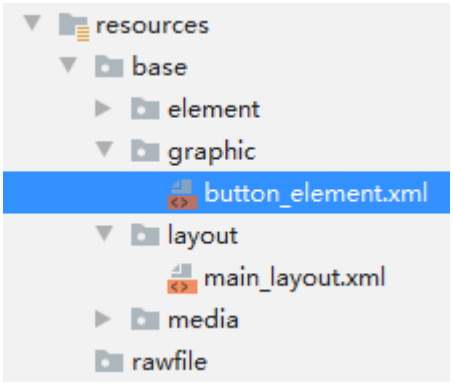


3. 打开“main_layout.xml”文件，添加一个文本和一个按钮，示例代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<DependentLayout
        xmlns:ohos="http://schemas.huawei.com/res/ohos"
        ohos:width="match_parent"
        ohos:height="match_parent"
        ohos:background_element="#000000">
    <Text
            ohos:id="$+id:text"
            ohos:width="match_content"
            ohos:height="match_content"
            ohos:center_in_parent="true"
            ohos:text="Hello World"
            ohos:text_color="white"
            ohos:text_size="32fp"/>
    <Button
            ohos:id="$+id:button"
            ohos:width="match_content"
            ohos:height="match_content"
            ohos:text_size="19fp"
            ohos:text="Next"
            ohos:top_padding="8vp"
            ohos:bottom_padding="8vp"
            ohos:right_padding="80vp"
```

```
24.                 ohos:left_padding="80vp"
25.                 ohos:text_color="white"
26.                 ohos:background_element="$graphic:button_element"
27.                 ohos:center_in_parent="true"
28.                 ohos:align_parent_bottom="true"/>
29. </DependentLayout>
```

4. 上述按钮的背景是通过“button_element”来显示的，需要在“base”目录下创建“graphic”文件夹，在“graphic”文件夹中新建一个“button_element.xml”文件。



“button_element.xml”的示例代码如下：

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <shape
3.          xmlns:ohos="http://schemas.huawei.com/res/ohos"
4.          ohos:shape="oval">
5.      <solid
6.              ohos:color="#007DFF"/>
7.  </shape>
```

3.2.2 加载 XML 布局

1. 在“Project”窗口中，选择“entry > src > main > java > com.example.helloworld > slice”，打开“MainAbilitySlice.java”文件。
2. 重写 onStart()方法加载 XML 布局，示例代码如下：

```
1.  package com.example.myapplication.slice;
2.
3.  import com.example.myapplication.ResourceTable;
4.  import ohos.aafwk.ability.AbilitySlice;
5.  import ohos.aafwk.content.Intent;
6.
7.  public class MainAbilitySlice extends AbilitySlice {
8.
9.      @Override
```

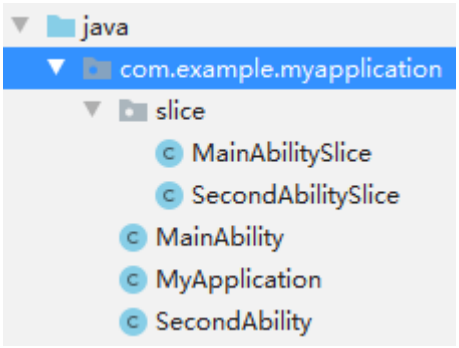
```
10.         public void onStart(Intent intent) {
11.             super.onStart(intent);
12.             super.setUIContent(ResourceTable.Layout_main_layout); //  加载 XML 布局
13.         }
14.
15.         @Override
16.         public void onActive() {
17.             super.onActive();
18.         }
19.
20.         @Override
21.         public void onForeground(Intent intent) {
22.             super.onForeground(intent);
23.         }
24.     }
```

3. 请参考应用运行，效果如图所示：

3.3 创建另一个页面

3.3.1 创建 Feature Ability

1. 在“Project”窗口，打开“entry > src > main > java”，右键点击“com.example.myapplication”文件夹，选择“New > Ability > Empty Feature Ability(Java)”。
2. 配置 Ability 时，将“Page Name”设置为“SecondAbility”，点击“Finish”。创建完成后，可以看到新增了“SecondAbility”和“SecondAbilitySlice”文件。



3.3.2 代码编写界面

在上一节中，我们用 XML 的方式编写了一个包含文本和按钮的页面。为了帮助开发者熟悉在代码中创建布局的方式，接下来我们使用此方式编写第二个页面。

打开 “SecondAbilitySlice.java”文件，添加一个文本，示例代码如下：

```
1.  package com.example.myapplication.slice;
2.
3.  import ohos.aafwk.ability.AbilitySlice;
4.  import ohos.aafwk.content.Intent;
5.  import ohos.agp.colors.RgbColor;
6.  import ohos.agp.components.DependentLayout;
7.  import ohos.agp.components.DependentLayout.LayoutConfig;
8.  import ohos.agp.components.Text;
9.  import ohos.agp.components.element.ShapeElement;
10. import ohos.agp.utils.Color;
11.
12. import static ohos.agp.components.ComponentContainer.LayoutConfig.MATCH_PARENT;
13. import static ohos.agp.components.ComponentContainer.LayoutConfig.MATCH_CONTENT;
14.
15. public class SecondAbilitySlice extends AbilitySlice {
```

```
16.
17.         @Override
18.         public void onStart(Intent intent) {
19.             super.onStart(intent);
20.             // 声明布局
21.             DependentLayout myLayout = new DependentLayout(this);
22.             // 设置布局大小
23.             myLayout.setWidth(MATCH_PARENT);
24.             myLayout.setHeight(MATCH_PARENT);
25.             ShapeElement element = new ShapeElement();
26.             element.setRgbColor(new RgbColor(0, 0, 0));
27.             myLayout.setBackground(element);
28.
29.             // 创建一个文本
30.             Text text = new Text(this);
31.             text.setText("Nice to meet you.");
32.             text.setWidth(MATCH_PARENT);
33.             text.setTextSize(55);
34.             text.setTextColor(Color.WHITE);
35.             // 设置文本的布局
36.             DependentLayout.LayoutConfig textConfig = new
    DependentLayout.LayoutConfig(MATCH_CONTENT,MATCH_CONTENT);
37.             textConfig.addRule(LayoutConfig.CENTER_IN_PARENT);
38.             text.setLayoutConfig(textConfig);
39.             myLayout.addComponent(text);
40.
41.             super.setUIContent(myLayout);
42.         }
43.
44.         @Override
45.         public void onActive() {
46.             super.onActive();
47.         }
48.
49.         @Override
50.         public void onForeground(Intent intent) {
51.             super.onForeground(intent);
52.         }
53.     }
```

3.4 实现页面跳转

1. 打开第一个页面的“MainAbilitySlice.java”文件，重写 onStart()方法添加按钮的响应逻辑，实现点击按钮跳转到下一页，示例代码如下：

```
package com.example.myapplication.slice;

import com.example.myapplication.ResourceTable;
import ohos.aafwk.ability.AbilitySlice;
import ohos.aafwk.content.Intent;
import ohos.aafwk.content.Operation;
import ohos.agp.components.*;

public class MainAbilitySlice extends AbilitySlice {

    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
        super.setUIContent(ResourceTable.Layout_main_layout);
        Button button = (Button) findComponentById(ResourceTable.Id_button);

        if (button != null) {
            // 为按钮设置点击回调
            button.setClickedListener(new Component.ClickedListener() {
                @Override
                public void onClick(Component component) {
                Intent secondIntent = new Intent();
                // 指定待启动 FA 的 bundleName 和 abilityName
                Operation operation = new Intent.OperationBuilder()
                        .withDeviceId("")
                        .withBundleName("com.example.myapplication")
                        .withAbilityName("com.example.myapplication.SecondAbility")
                        .build();
                secondIntent.setOperation(operation);
                startAbility(secondIntent); // 通过 AbilitySlice 的 startAbility 接口实现启动另一个页面
                }
            });
        }
    }

    @Override
    public void onActive() {
        super.onActive();
```

```
39.         }
40.
41.         @Override
42.         public void onForeground(Intent intent) {
43.             super.onForeground(intent);
44.         }
45.     }
```

2. 再次运行项目，效果如图所示：