设备兼容性概览

Android 适用于众多类型的设备，从手机到平板电脑和电视都能搭载使用。作为开发者，如此广泛的设备类型能为您的应用带来广大的潜在受众群体。为了能在所有这些设备上顺利运行，应用应该容许部分设备功能的变化，并提供可适应不同屏幕配置的灵活界面。

为了帮助您实现这一目标，Android 提供了一个动态应用框架，供您在静态文件中提供特定于配置的应用资源（/guide/topics/resources/overview）（例如针对不同屏幕尺寸的不同 XML 布局）。然后，Android 会根据当前设备配置加载适当的资源。因此，在对应用设计和一些额外的应用资源进行一些事先规划后，您可以发布单个应用软件包 (APK)，并在各种设备上提供优化的用户体验。

但是，您可以根据需要指定应用的功能要求，并控制哪些类型的设备可以通过 Google Play 商店安装您的应用。本页介绍了如何控制哪些设备可以访问您的应用，以及如何准备您的应用以确保它们覆盖合适的受众群体。如需详细了解如何让您的应用适应不同的设备，请参阅支持不同的设备 (/training/basics/supporting-devices)。

“兼容性”是什么意思？

随着您进一步阅读 Android 开发相关内容，您可能会在各种语境下遇到“兼容性”一词。兼容性有两种类型：设备兼容性和应用兼容性。

由于 Android 是一个开源项目，因此任何硬件制造商都可以制造搭载 Android 操作系统的设备。不过，**设备“兼容 Android”**的前提是它可以正常运行针对 Android 执行环境编写的应用。Android 执行环境的具体细节由 Android 兼容性计划（https://source.android.com/compatibility/overview.html)定义，每台设备都必须通过兼容性测试套件 (CTS) 测试才能被视为兼容。

作为应用开发者，您无需担心设备是否兼容 Android，因为只有与 Android 兼容的设备才会附带 Google Play 商店。因此，您可以放心，通过 Google Play 商店安装您的应用的用户使用的是 Android 兼容设备。

不过，您确实需要考虑您的**应用是否兼容**每一种可能的设备配置。由于 Android 以各种设备配置运行，因此部分功能并不适用于所有设备。例如，某些设备可能未配备罗盘传感器。如果应用的核心功能需要使用罗盘传感器，那么应用只能与带有罗盘传感器的设备兼容。

控制应用在设备上的可用性

应用可通过平台 API 利用 Android 支持的各种功能。有些功能基于硬件（例如罗盘传感器），有些功能基于软件（如应用窗口微件），有些功能则依赖于平台版本。并非每台设备都支持所有功能，因此您可能需要根据应用所需的功能控制应用在设备上的可用性。

要尽可能扩大应用的用户群，您应设法使用单个 APK 支持尽可能多的设备配置。在大多数情况下，要实现这一目标，您可以在运行时停用可选功能，并为应用资源 (/guide/topics/resources/providing-resources)提供针对不同配置的替代选项（例如针对不同屏幕尺寸的不同布局）。不过，如果需要，您可以根据以下设备特征，通过 Google Play 商店限制应用在设备上的可用性：

- 设备功能 (#Features)
- 平台版本 (#Version)
- 屏幕配置 (#Screens)

设备功能

为了让您根据设备功能管理应用的可用性，Android 为可能并不适用于所有设备的任何硬件或软件功能定义了功能 ID。例如，罗盘传感器的功能 ID 为 `FEATURE_SENSOR_COMPASS` (/reference/android/content/pm/PackageManager#FEATURE_SENSOR_COMPASS)，而应用微件的功能 ID 为 `FEATURE_APP_WIDGETS` (/reference/android/content/pm/PackageManager#FEATURE_APP_WIDGETS)。

根据需要，要在用户的设备不具备特定功能时阻止用户安装您的应用，您可以通过应用清单文件 (/guide/topics/manifest/manifest-intro)中的 `<uses-feature>` (/guide/topics/manifest/uses-feature-element) 元素声明这一点。

例如，如果您的应用在没有罗盘传感器的设备上没有意义，您可以使用以下清单标记声明需要罗盘传感器：

```
<manifest ... >
    <uses-feature android:name="android.hardware.sensor.compass"
                  android:required="true" />
    ...
```

```
</manifest>
```

Google Play 商店会将您的应用所需的功能与每个用户的设备上可用的功能进行比较，以确定您的应用是否与每台设备兼容。如果设备不具备您的应用所需的所有功能，则用户无法安装您的应用。

但是，如果应用的主要功能不需要某项设备功能，则应将 `required` (/guide/topics/manifest/uses-feature-element#required) 属性设置为 `"false"` 并在运行时检查是否有该设备功能。如果应用功能在当前设备上不可用，请适当降级相应的应用功能。例如，您可以通过调用 `hasSystemFeature()` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)) 来查询功能是否可用，如下所示：

KOTLIN (#KOTLIN)**JAVA**

```
PackageManager pm = getPackageManager();
if (!pm.hasSystemFeature(PackageManager.FEATURE_SENSOR_COMPASS)) {
    // This device does not have a compass, turn off the compass feature
    disableCompassFeature();
}
```

如需了解您可以用来通过 Google Play 商店控制应用对用户是否可用的所有过滤器，请参阅 Google Play 上的过滤器 (/google/play/filters)文档。

注意：一些系统权限 (/guide/topics/permissions)会隐式要求具备某项设备功能。例如，如果您的应用请求对 `BLUETOOTH` (/reference/android/Manifest.permission#BLUETOOTH) 的访问权限，这就隐式要求具备 `FEATURE_BLUETOOTH` (/reference/android/content/pm/PackageManager#FEATURE_BLUETOOTH) 设备功能。要停用基于此功能的过滤，并使您的应用可用于没有蓝牙的设备，您可以在 `<uses-feature>` (/guide/topics/manifest/uses-feature-element) 标记中将 `required` (/guide/topics/manifest/uses-feature-element#required) 属性设置为 `"false"`。如需详细了解隐式要求的设备功能，请参阅隐含功能要求的权限 (/guide/topics/manifest/uses-feature-element#permissions)。

平台版本

不同的设备可能会运行不同版本的 Android 平台，例如 Android 4.0 或 Android 4.4。每个后续的平台版本通常会添加之前版本中不可用的新 API。为表明可用的 API 集，每个平台版本都会指定 API 级别 (/guide/topics/manifest/uses-sdk-element#ApiLevels)。例如，Android 1.0 是 API 级别 1，而 Android 4.4 是 API 级别 19。

通过 API 级别，您可以使用 `<uses-sdk>` (/guide/topics/manifest/uses-sdk-element) 清单标记及其 `minSdkVersion` (/guide/topics/manifest/uses-sdk-element#min) 属性来声明应用兼容的最低版本。例如，Android 4.0（API 级别 14）中添加了日历提供程序 (/guide/topics/providers/calendar-provider) API。如果您的应用在没有这些 API 的情况下无法运行，您应将 API 级别 14 声明为应用的最低支持版本。

`minSdkVersion` (/guide/topics/manifest/uses-sdk-element#min) 属性声明应用兼容的最低版本，`targetSdkVersion` (/guide/topics/manifest/uses-sdk-element#target) 属性声明应用经过优化后适用的最高版本。

不过，请注意 `<uses-sdk>` (/guide/topics/manifest/uses-sdk-element) 元素中的属性会被替换为 `build.gradle` (/studio/build#build-files) 文件中的相应属性。因此，如果您使用的是 Android Studio，则必须在其中指定 `minSdkVersion` 和 `targetSdkVersion` 值：

```
android {
  defaultConfig {
    applicationId 'com.example.myapp'

    // Defines the minimum API level required to run the app.
    minSdkVersion 15

    // Specifies the API level used to test the app.
    targetSdkVersion 28

    ...
  }
}
```

要详细了解 `build.gradle` 文件，请参阅如何配置编译版本 (/studio/build)。

每个后续版本的 Android 都为使用之前平台版本的 API 构建的应用提供兼容性，因此您的应用应始终与未来版本的 Android 兼容，同时使用已记录的 Android API。

注意：`targetSdkVersion` (/guide/topics/manifest/uses-sdk-element#target) 属性不会阻止您的应用安装在高于指定值的平台版本上，但它很重要，因为它向系统指示您的应用是否应继承较新版本中的行为更改。如果您不将 `targetSdkVersion` (/guide/topics/manifest/uses-sdk-element#target) 更新到最新版本，则系统会认为您的应用在最新版本上运行时需要一些向后兼容性行为。例如，在 Android 4.4 中的行为更改 (/about/versions/android-4.4#Behaviors)中，使用 `AlarmManager` (/reference/android/app/AlarmManager) API 创建的闹钟现在默认不精确，因此系统可以批量处理应用闹钟并节省系统电量，但如果您的目标 API 级别低于“19”，则系统会为您的应用保留之前的 API 行为。

不过，如果您的应用使用的是较新平台版本中添加的 API，但其主要功能并不需要这些 API，则应在运行时检查 API 级别，并在 API 级别过低时适当降级相应的功能。在这种情况下，请将 `minSdkVersion` (/guide/topics/manifest/uses-sdk-element#min) 尽量设置为适用于应用主要功能的最低值，然后将当前系统的版本 `SDK_INT` (/reference/android/os/Build.VERSION#SDK_INT) 与 `Build.VERSION_CODES` (/reference/android/os/Build.VERSION_CODES) 中对应于您要检查的 API 级别的一个代号常量进行比较。例如：

KOTLIN (#KOTLIN)**JAVA**

```
if (Build.VERSION.SDK_INT < Build.VERSION_CODES.HONEYCOMB) {
    // Running on something older than API level 11, so disable
    // the drag/drop features that use <code><a href="/reference/android/content/ClipboardManager.html">Clipboar
    disableDragAndDrop();
}
```

屏幕配置

Android 可在各种尺寸的设备上运行，包括手机、平板电脑和电视。为了按照屏幕类型对设备进行分类，Android 为每种设备定义了两个特征：屏幕尺寸（屏幕的物理尺寸）和屏幕密度（屏幕上像素的物理密度，称为 DPI）。为了简化不同的配置，Android 将这些变体归纳成组，使它们更容易作为定位目标：

- 四种广义的尺寸：小、标准、大和特大。
- 还有几种广义的密度：mdpi（中）、hdpi（高）、xhdpi（超高）、xxhdpi（超超高）等。

默认情况下，您的应用会兼容所有屏幕尺寸和密度，因为系统会根据需要对各个屏幕的界面布局和图片资源进行相应的调整。不过，您应针对不同的屏幕尺寸添加专门的布局，针对常见的屏幕密度添加优化的位图图片，以优化每种屏幕配置的用户体验。

如需了解如何针对不同屏幕创建备用资源以及如何在必要时将应用限制为特定屏幕尺寸，请参阅支持不同屏幕 (/training/basics/supporting-devices/screens)。

出于业务原因控制应用的可用性

除了根据设备特征限制应用的可用性之外，您还可能需要出于业务或法律方面的原因限制应用的可用性。例如，显示伦敦地铁列车时刻表的应用不太可能适用于英国以外的用户。针对此类情况，Google Play 商店在 Play 管理中心提供了过滤选项，供您出于非技术原因（例如用户的语言区域或无线运营商）控制应用的可用性。

针对技术兼容性（例如必需的硬件组件）的过滤始终基于 APK 文件中包含的信息。但是，出于非技术原因（如地理语言区域）的过滤始终在 Google Play 管理中心 (/distribute/console)中处理。

继续阅读以下内容：

提供资源 (/guide/topics/resources/providing-resources)

介绍了 Android 应用如何采用将应用资源与应用代码分开的结构，包括如何为特定设备配置提供备用资源。

Google Play 上的过滤器 (/google/play/filters)

您可能还对以下内容感兴趣：

系统权限 (/guide/topics/permissions)

Android 如何通过权限系统要求应用在获得用户同意后才能使用特定 API，从而限制应用对这些 API 的访问权限。

介绍了 Google Play 商店阻止您的应用安装在不同设备上的不同方式。

下一页
屏幕兼容性概览 (/guide/practices/screens_support)

Last updated 2020-05-07 UTC.