<uses-feature>

本文内容

语法：

```
<uses-feature
  android:name="string"
  android:required=["true" | "false"]
  android:glEsVersion="integer" />
```

包含它的文件：

`<manifest>`

说明：

声明应用使用的单一硬件或软件功能。

`<uses-feature>` 声明的用途是将您的应用依赖的硬件和软件功能集通知任何外部实体。 该元素提供的 `required` 属性可让您指定您的应用是需要声明的功能并且没有该功能便无法正常工作，还是使用该功能只是一种优先选择，没有它仍然可以正常工作。 由于功能支持可能因 Android 设备而异，`<uses-feature>` 元素发挥的重要作用是，能够让应用描述其使用的随设备而变化的功能。

您的应用声明的可用功能集对应于 Android `PackageManager` 提供的功能常量集，方便起见，本文末尾的功能参考资料部分列出了这些功能常量。

您必须在单独的 `<uses-feature>` 元素中指定每个功能，因此如果您的应用需要多个功能，就需要声明多个 `<uses-feature>` 元素。例如，要求设备同时具有蓝牙和相机功能的应用会声明以下这两个元素：

```
<uses-feature android:name="android.hardware.bluetooth" />
<uses-feature android:name="android.hardware.camera" />
```

一般而言，您始终都应确保为应用需要的所有功能声明 `<uses-feature>` 元素。

声明的 `<uses-feature>` 元素仅供参考，这意味着 Android 系统本身在安装应用前不会检查设备是否提供相应的功能支持。 不过，其他服务（如 Google Play）或应用可能会在处理您的应用或与其交互的过程中检查它的 `<uses-feature>` 声明。 因此，声明您的应用使用的所有功能（见下文列表）非常重要。

对某些功能而言，您或许可以通过特定属性定义功能的版本，例如所用 Open GL 的版本（使用 `glEsVersion` 声明）。 设备具有或不具有的其他功能（如相机）使用 `name` 属性进行声明。

尽管只为运行 API 级别 4 或更高版本系统的设备激活了 `<uses-feature>` 元素，我们仍然建议为所有应用加入这些元素，即使

Google Play 过滤

Google Play 利用在您的应用清单中声明的 `<uses-feature>` 元素从不符合应用的硬件和软件功能要求的设备上将应用滤除。

通过指定您的应用要求的功能，可以让 Google Play 只向设备符合应用功能要求的用户而不是所有用户提供您的应用。

如需了解有关 Google Play 如何以功能为依据进行过滤的重要信息，请阅读下面的 Google Play 和根据功能进行过滤。

`minSdkVersion` 为“3”或更低也要这样做。 运行较低版本平台的设备会直接忽略该元素。

> **注**：声明功能时，切记您还必须视情况请求权限。 例如，您仍须请求 `CAMERA` 权限，才能让您的应用访问 Camera API。 请求权限可授予应用对相应硬件和软件的访问权，而声明应用使用的功能则可确保设备兼容性。

属性：

`android:name`

以描述符字符串形式指定应用使用的单一硬件或软件功能。 硬件功能和软件功能部分列出了有效的属性值。 这些属性值区分大小写。

`android:required`

表示应用是否需要 `android:name` 中所指定功能的布尔值。

- 当您为某项功能声明 `android:required="true"` 时，即是规定当设备不具有该指定功能时，应用*无法正常工作，或设计为无法正常工作*。
- 当您为某项功能声明 `android:required="false"` 时，则意味着如果设备具有该功能，应用会在必要时*优先使用该功能*，但应用*设计为不使用该指定功能也可正常工作*。

如果未予声明，`android:required` 的默认值为 `"true"`。

`android:glEsVersion`

应用需要的 OpenGL ES 版本。高 16 位表示主版本号，低 16 位表示次版本号。 例如，要指定 OpenGL ES 2.0 版，您需要将其值设置为“0x00020000”；要指定 OpenGL ES 3.2，则需将其值设置为“0x00030002”。

应用应在其清单中至多指定一个 `android:glEsVersion`。 如果指定不止一个，将使用数值最高的 `android:glEsVersion`，任何其他值都会被忽略。

如果应用不指定 `android:glEsVersion` 属性，则系统假定应用只需要 OpenGL ES 1.0，即所有 Android 设备都支持的版本。

应用可以假定，如果平台支持给定 OpenGL ES 版本，也同样支持所有数值更低的 OpenGL ES 版本。 因此，同时需要 OpenGL ES 1.0 和 OpenGL ES 2.0 的应用必须指定它需要 OpenGL ES 2.0。

能够使用几种 OpenGL ES 版本中任一版本的应用只应指定它需要的数值最低的 OpenGL ES 版本。 （它可以在运行时检查是否有更高版本的 OpenGL ES 可用）。

如需了解有关如何使用 OpenGL ES（包括如何在运行时检查支持的 OpenGL ES 版本）的详细信息，请参阅 OpenGL ES API 指南。

引入的版本：

API 级别 4

另请参阅：

- `PackageManager`
- `FeatureInfo`
- `ConfigurationInfo`
- `<uses-permission>`
- Google Play 上的过滤器

Google Play 和根据功能进行过滤

Google Play 会过滤对用户可见的应用，让用户只能看到和下载与其设备兼容的应用。 它过滤应用的其中一种方法是按功能兼容性进行过滤。

为确定应用与给定用户设备的功能兼容性，Google Play 会比较：

- 应用需要的功能 — 应用在其清单内的 `<uses-feature>` 元素中声明这些功能
  与...
- 设备上提供的硬件或软件功能 — 设备以只读系统属性形式报告其支持的功能。

为确保功能比较的准确性，Android 软件包管理器提供了一个共享的功能常量集，应用和设备均利用该集来声明功能要求和支持。 本文末尾的功能参考资料部分以及 `PackageManager` 的类文档中列出了这些可用的功能常量。

当用户启动 Google Play 时，应用通过调用 `getSystemAvailableFeatures()` 在软件包管理器中查询设备上提供的功能列表。 然后，Google 商店应用会在为用户建立会话时将功能列表向上传递给 Google Play。

您每次向 Google Play Developer Console 上传应用时，Google Play 都会扫描应用的清单文件。 它会寻找 `<uses-feature>` 元素并在某些情况下结合其他元素（例如 `<uses-sdk>` 和 `<uses-permission>` 元素）对其进行评估。 在建立应用所需的功能集之后，它会在内部将该列表存储为与应用 `.apk` 和应用版本关联的元数据。

当用户利用 Google Play 应用搜索或浏览应用时，该服务会将各应用需要的功能与用户设备上提供的功能进行比较。 如果设备提供了应用所需的全部功能，则 Google Play 允许用户看到该应用并可能允许用户下载该应用。 如果设备不支持任何所需功能，Google Play 会滤除该应用，令其对用户不可见，也无法供用户下载。

由于您在 `<uses-feature>` 元素中声明的功能会直接影响 Google Play 对您的应用的过滤方式，因此必须了解 Google Play 如何评估应用的清单以及如何建立所需功能集。 下文提供了详细信息。

根据显式声明的功能进行过滤

显式声明的功能是指您的应用在 `<uses-feature>` 元素中声明的功能。 功能声明可包括 `android:required=["true" | "false"]` 属性（如果您编译的应用面向 API 级别 5 或更高版本），您可以通过它指定应用是绝对需要该功能，没有它便无法正常工作（设置为 `"true"` 时），还是应用会在提供了该功能时予以优先使用，但应用本身设计为不使用它也能正常运行（设置为 `"false"` 时）。

Google Play 按以下方式处理显式声明的功能：

- 如果一项功能被显式声明为所需功能，Google Play 会将该功能添加到应用的所需功能列表。 然后，它会从不提供该功能的设备上滤除该应用，让用户无法看到。例如：

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

- 如果一项功能被显式声明为*并非*所需功能，Google Play *不会*将该功能添加到所需功能列表。 因此，在过滤应用时，从不会考虑显式声明的非所需功能。 即使设备不提供声明的功能，Google Play 仍会考虑与设备兼容的应用并将其显示给用户，除非有其他过滤规则适用。 例如：

  ```
  <uses-feature android:name="android.hardware.camera" android:required="false" />
  ```

- 如果显式声明了某个功能，但未加入 `android:required` 属性，则 Google Play 会假定该功能是所需功能，并对其设置过滤。

一般而言，如果您的应用设计为在 Android 1.6 及更低版本上运行，API 中不提供 `android:required` 属性，Google Play 会假定任何以及所有 `<uses-feature>` 声明都是所需功能。

> **注**：通过显式声明某项功能并加入 `android:required="false"` 属性，您可以在 Google Play 上有效停用所有针对指定功能的过滤。

根据隐含功能进行过滤

*隐含*功能是指应用正常工作所必需，但*未*在清单文件的 `<uses-feature>` 元素中进行声明的功能。 严格地讲，每个应用都应*始终*声明其使用或需要的所有功能，因此应将缺少应用所使用功能的声明视为错误。 不过，作为给用户和开发者提供的一种保障措施，Google Play 会寻找每个应用中的隐含功能并设置针对这些功能的过滤，就像它为显式声明功能所做的那样。

应用可能需要某项功能却不会声明它，这是因为：

- 应用的编译目标是旧版本的 Android 内容库（Android 1.5 或更低版本），并且 `<uses-feature>` 元素不可用。
- 开发者错误地认为所有设备都提供该功能，没必要进行声明。

- 开发者无意中遗漏了功能声明。
- 开发者显式声明了功能，但声明无效。 例如，`<uses-feature>` 元素名称拼写错误或无法识别的 `android:name` 属性字符串值都会令功能声明无效。

为将以上情况考虑在内，Google Play 会尝试通过检查清单文件中声明的*其他元素*（具体地讲，`<uses-permission>` 元素）来发现应用的隐含功能要求。

如果应用请求硬件相关权限，Google Play *假定该应用使用基础的硬件功能，并因此需要这些功能*，即使可能没有对应的 `<uses-feature>` 声明。 对于此类权限，Google Play 会向它为应用存储的元数据添加这些基础的硬件功能，并设置针对它们的过滤。

例如，如果应用请求 `CAMERA` 权限，但没有针对 `android.hardware.camera` 声明 `<uses-feature>` 元素，Google Play 会认为该应用需要相机，并且不应向没有相机的设备用户显示。

如果您不想让 Google Play 根据特定隐含功能进行过滤，可以停用该行为。 要执行此操作，请在一个 `<uses-feature>` 元素中显式声明该功能，并加入一个 `android:required="false"` 属性。 例如，要停用源自 `CAMERA` 权限的过滤，您需要声明该功能，如下所示。

```
<uses-feature android:name="android.hardware.camera" android:required="false" />
```

> 必须了解的是，您在 `<uses-permission>` 元素中请求的权限可能会直接影响 Google Play 对您的应用的过滤方式。 参考资料部分隐含功能要求的权限完整列出了一组隐含功能要求并因此触发过滤的权限。

针对蓝牙功能的特殊处理

Google Play 在决定针对蓝牙的过滤时应用的规则与上述略有差异。

如果某个应用在 `<uses-permission>` 元素中声明了蓝牙权限，但未在 `<uses-feature>` 元素中显式声明蓝牙功能，则 Google Play 会按照 `<uses-sdk>` 元素中的指定检查作为应用设计运行目标的 Android 平台版本。

如下表所示，只有在应用将其最低或目标平台声明为 Android 2.0（API 级别 5）或更高级别时，Google Play 才会启用针对蓝牙功能的过滤。但请注意，当应用在 `<uses-feature>` 元素中显式声明蓝牙功能时，Google Play 会应用正常过滤规则。

表 1. Google Play 如何确定请求蓝牙权限却未在 `<uses-feature>` 元素中声明蓝牙功能的应用的蓝牙功能要求。

如果 `minSdkVersion`...	或 `targetSdkVersion`	结果
<=4（或未声明 uses-sdk）	<=4	Google Play *将不会*根据设备所报告的对 `android.hardware.bluetooth` 功能的支持情况从任何设备中滤除该应用。
<=4	>=5	Google Play 会从任何不支持 `android.hardware.bluetooth` 功能（包括旧版本）的设备中滤除该应用。
>=5	>=5	

以下示例根据 Google Play 对蓝牙功能的处理方式说明了不同的过滤效果。

在第一个示例中，一个设计为在旧版 API 级别上运行的应用声明了蓝牙权限，但未在 `<uses-feature>` 元素中声明蓝牙功能。

> *结果*：Google Play 不会从任何设备中滤除该应用。

```
<manifest ...>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-sdk android:minSdkVersion="3" />
    ...
</manifest>
```

在下面的第二个示例中，同一应用还声明了目标 API 级别“5”。

> *结果*：Google Play 现在认为该功能是所需功能，将从所有不报告蓝牙支持的设备（包括运行旧版本平台的设备）上滤除该应用。

```
<manifest ...>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-sdk android:minSdkVersion="3" android:targetSdkVersion="5" />
    ...
</manifest>
```

在本例中，同一应用现在明确声明了蓝牙功能。

结果：与上例完全相同（应用了过滤）。

```
<manifest ...>
    <uses-feature android:name="android.hardware.bluetooth" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-sdk android:minSdkVersion="3" android:targetSdkVersion="5" />
    ...
</manifest>
```

最后，在以下示例中，同一应用添加了一个 `android:required="false"` 属性。

结果：Google Play 针对所有设备停用根据蓝牙功能支持进行过滤。

```
<manifest ...>
    <uses-feature android:name="android.hardware.bluetooth" android:required="false" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-sdk android:minSdkVersion="3" android:targetSdkVersion="5" />
    ...
</manifest>
```

测试应用需要的功能

您可以利用 Android SDK 中包括的 `aapt` 工具来确定 Google Play 如何根据应用声明的功能和权限对其进行过滤。 要执行该操作，请使用 `dump badging` 命令运行 `aapt`。这会使 `aapt` 解析您的应用的清单，并应用与 Google Play 所用相同的规则来确定您的应用需要的功能。

要使用该工具，请按以下步骤操作：

1. 首先，生成您的应用并将其导出为未签署的 `.apk`。如果您是在 Android Studio 中进行开发，请使用 Gradle 生成您的应用：
   a. 打开项目并选择 **Run > Edit Configurations**。
   b. 选择靠近 **Run/Debug Configurations** 窗口左上角的加号。
   c. 选择 **Gradle**。
   d. 在 **Name** 中输入 `Unsigned APK`。
   e. 从 **Gradle project** 部分选择您的模块。
   f. 在 **Tasks** 中输入 `assemble`。
   g. 选择 **OK** 完成新配置。
   h. 确保在工具栏中选择了 **Unsigned APK** 运行配置，并选择 **Run > Run 'Unsigned APK'**。

   您可以在 *`<ProjectName>`*`/app/build/outputs/apk/` 目录中找到您的未签署 `.apk`。
2. 接下来，如果 `aapt` 工具尚未出现在您的 PATH 中，定位该工具。如果您使用的是 SDK Tools r8 或更高版本，可以在 *`<SDK>`*`/platform-tools/` 目录中找到 `aapt`。

   > **注**：您必须使用为可用的最新 Platform-Tools 组件提供的 `aapt` 版本。 如果您没有最新的 Platform-Tools 组件，请使用 Android SDK 管理器下载它。
3. 使用以下语法运行 `aapt`：

```
$ aapt dump badging <path_to_exported_.apk>
```

以下是针对上面第二个蓝牙示例的命令输出示例：

```
$ ./aapt dump badging BTExample.apk
package: name='com.example.android.btexample' versionCode='' versionName=''
uses-permission:'android.permission.BLUETOOTH_ADMIN'
uses-feature:'android.hardware.bluetooth'
sdkVersion:'3'
targetSdkVersion:'5'
application: label='BT Example' icon='res/drawable/app_bt_ex.png'
launchable activity name='com.example.android.btexample.MyActivity'label='' icon=''
uses-feature:'android.hardware.touchscreen'
main
supports-screens: 'small' 'normal' 'large'
locales: '--_--'
densities: '160'
```

功能参考资料

下文提供有关硬件功能、软件功能以及隐含具体功能要求的权限集的参考信息。

硬件功能

此部分介绍最新平台版本支持的硬件功能。 要指出您的应用使用或需要某项硬件功能，请在 `android:name` 属性中声明相应的值（以 `"android.hardware"` 开头）。 请在您每次声明一项硬件功能时使用单独的 `<uses-feature>` 元素。

音频硬件功能

`android.hardware.audio.low_latency`

应用使用设备的低延迟时间音频管道，该管道可以减少处理声音输入或输出时的滞后和延迟。

`android.hardware.audio.output`

应用使用设备的音响设备、音频耳机插孔、蓝牙流式传输能力或类似机制传输声音。

`android.hardware.audio.pro`

应用使用设备的高端音频功能和性能能力。

`android.hardware.microphone`

应用使用设备的麦克风记录音频。

蓝牙硬件功能

`android.hardware.bluetooth`

应用使用设备的蓝牙功能，通常是为了与其他支持蓝牙的设备通信。

`android.hardware.bluetooth_le`

应用使用设备的低功耗蓝牙无线电功能。

相机硬件功能

`android.hardware.camera`

应用使用设备的后置相机。只有前置相机的设备不会列出该功能，因此如果您的应用可与任何朝向的相机通信，请改用

android.hardware.camera.any 功能。

android.hardware.camera.any

应用使用设备的其中一个相机或用户为设备连接的外置相机。 如果您的应用不要求相机必须是后置式，请使用此值来替代 android.hardware.camera。

android.hardware.camera.autofocus

应用使用设备相机支持的自动对焦功能。

应用通过使用该功能暗示其还使用 android.hardware.camera 功能，除非这个父功能在声明时使用了 android:required="false"。

android.hardware.camera.capability.manual_post_processing

应用使用设备相机支持的 MANUAL_POST_PROCESSING 功能。

您的应用可以通过该功能替换相机的自动白平衡功能。 使用 android.colorCorrection.transform、android.colorCorrection.gains 以及 TRANSFORM_MATRIX 的 android.colorCorrection.mode。

android.hardware.camera.capability.manual_sensor

应用使用设备相机支持的 MANUAL_SENSOR 功能。

该功能隐含对自动曝光锁定 (android.control.aeLock) 的支持，该支持可以让相机的曝光时间和灵敏度一直固定在特定值。

android.hardware.camera.capability.raw

应用使用设备相机支持的 RAW 功能。

该功能暗示设备可以保存 DNG（原始）文件，并且设备的相机提供您的应用直接处理这些原始图像所需的 DNG 相关元数据。

android.hardware.camera.external

应用与用户为设备连接的外置相机通信。 但该功能不能保证外置相机可供您的应用使用。

android.hardware.camera.flash

应用使用设备相机支持的闪光功能。

应用通过使用该功能暗示其还使用 android.hardware.camera 功能，除非这个父功能在声明时使用了 android:required="false"。

android.hardware.camera.front

应用使用设备的前置相机。

应用通过使用该功能暗示其还使用 android.hardware.camera 功能，除非这个父功能在声明时使用了 android:required="false"。

android.hardware.camera.level.full

应用使用设备的至少一个相机提供的 FULL 级图像捕捉支持。 提供 FULL 支持的相机可提供快速捕捉功能、逐帧控制和手动后期处理控制。

设备 UI 硬件功能

android.hardware.type.automotive

应用设计为在车辆内的一组屏幕上显示其 UI。 用户使用硬按钮、触摸、旋转控制器以及类鼠标界面与应用进行交互。 车辆的屏幕通常

出现在车辆的中控台或仪表板中。 这些屏幕的尺寸和分辨率通常有限。

> **注：**切记，由于用户是在驾车时使用这类应用 UI，应用必须尽量不要让驾驶员分心。

`android.hardware.type.television`

（已弃用；请改用 `android.software.leanback`。）

应用设计为在电视上显示其 UI。该功能将“电视”定义为一种典型的起居室电视体验：显示在大屏幕上，用户坐在远处，主流输入形式是类似方向键的东西，一般不使用鼠标、指示器或触摸设备。

`android.hardware.type.watch`

应用设计为在手表上显示其 UI。手表佩戴在身体（例如手腕）上。 用户在很近的距离与设备互动。

指纹硬件功能

`android.hardware.fingerprint`

应用使用设备的生物识别硬件读取指纹。

手柄硬件功能

`android.hardware.gamepad`

应用捕获来自设备本身或其连接的手柄的游戏控制器输入。

红外线硬件功能

`android.hardware.consumerir`

应用使用设备的红外线 (IR) 功能，通常是为了与其他消费 IR 设备通信。

定位硬件功能

`android.hardware.location`

应用使用设备上的一项或多项功能来确定位置，例如 GPS 位置、网络位置或基站位置。

`android.hardware.location.gps`

应用使用从设备上的全球定位系统 (GPS) 接收器获得的精确位置坐标。

应用通过使用该功能暗示其还使用 `android.hardware.location` 功能，除非这个父功能在声明时使用了属性 `android:required="false"`。

`android.hardware.location.network`

应用使用从设备上支持的基于网络的地理定位系统获得的粗略位置坐标。

应用通过使用该功能暗示其还使用 `android.hardware.location` 功能，除非这个父功能在声明时使用了属性 `android:required="false"`。

NFC 硬件功能

`android.hardware.nfc`

应用使用设备的近距离无线通信 (NFC) 功能。

`android.hardware.nfc.hce`

（已弃用。）

应用使用设备上托管的 NFC 卡模拟。

OpenGL ES 硬件功能

`android.hardware.opengles.aep`

应用使用设备上安装的 OpenGL ES Android 扩展包。

传感器硬件功能

`android.hardware.sensor.accelerometer`

应用使用从设备的加速计读取的运动信息来检测设备的当前方向。 例如，应用可以使用加速计读数来确定何时在纵向与横向方向之间切换。

`android.hardware.sensor.ambient_temperature`

应用使用设备的外界（环境）温度传感器。例如，天气应用可以报告室内或室外温度。

`android.hardware.sensor.barometer`

应用使用设备的气压计。例如，天气应用可以报告气压。

`android.hardware.sensor.compass`

应用使用设备的磁力计（罗盘）。例如，导航应用可以用户当前面朝的方向。

`android.hardware.sensor.gyroscope`

应用使用设备的陀螺仪来检测旋转和倾斜，从而形成一个六轴方向系统。 通过使用该传感器，应用可以更顺利地检测其是否需要在纵向与横向方向之间切换。

`android.hardware.sensor.hifi_sensors`

应用使用设备的高保真 (Hi-Fi) 传感器。例如，游戏应用可以检测用户的高精度移动。

`android.hardware.sensor.heartrate`

应用使用设备的心率监测器。例如，健身应用可以报告用户心率随时间的变化趋势。

`android.hardware.sensor.heartrate.ecg`

应用使用设备的超声波心动图 (ECG) 心率传感器。例如，健身应用可以报告有关用户心率的更详细信息。

`android.hardware.sensor.light`

应用使用设备的光传感器。例如，应用可以根据环境光照条件显示两种不同配色方案中的一种。

`android.hardware.sensor.proximity`

应用使用设备的近程传感器。例如，电话应用可以在其检测到用户握持的设备贴近身体时关闭设备的屏幕。

`android.hardware.sensor.relative_humidity`

应用使用设备的相对湿度传感器。例如，天气应用可以利用湿度来计算和报告当前露点。

`android.hardware.sensor.stepcounter`

应用使用设备的计步器。例如，健身应用可以报告用户需要走多少步才能达到每天的计步目标。

`android.hardware.sensor.stepdetector`

应用使用设备的步测器。例如，健身应用可以利用每步的间隔时间来推测用户正在进行的锻炼类型。

屏幕硬件功能

`android.hardware.screen.landscape`

`android.hardware.screen.portrait`

应用要求设备使用纵向或横向方向。 如果您的应用同时支持这两种方向，则无需声明任一功能。

例如，如果您的应用要求纵向方向，则应声明以下功能，使得只有支持纵向方向（始终或由用户选择）的设备才能运行您的应用：

```
<uses-feature android:name="android.hardware.screen.portrait" />
```

默认情况下假定两种方向均非要求的方向，这样您的应用就可以安装在支持一种或同时支持两种方向的设备上。 不过，如果应用的任何 Activity 利用 `android:screenOrientation` 属性请求在特定方向下运行，则此声明意味着您的应用要求该方向。 例如，如果您使用 `"landscape"`、`"reverseLandscape"` 或 `"sensorLandscape"` 声明 `android:screenOrientation`，则您的应用将只能安装在支持横向方向的设备上。

最佳做法是，您仍应使用 `<uses-feature>` 元素来声明对该方向的要求。 如果您使用 `android:screenOrientation` 为 Activity 声明了某个方向，但实际并无此要求，可通过使用 `<uses-feature>` 元素并加入 `android:required="false"` 声明该方向来停用这一要求。

为实现后向兼容性，任何运行 Android 3.1（API 级别 12）或更低版本的设备都同时支持横向和纵向方向。

电话硬件功能

`android.hardware.telephony`

应用使用设备的电话功能，例如提供数据通信服务的无线电话。

`android.hardware.telephony.cdma`

应用使用码分多址接入 (CDMA) 无线电话系统。

应用通过使用该功能暗示其还使用 `android.hardware.telephony` 功能，除非这个父功能在声明时使用了 `android:required="false"`。

`android.hardware.telephony.gsm`

应用使用全球移动通信系统 (GSM) 无线电话系统。

应用通过使用该功能暗示其还使用 `android.hardware.telephony` 功能，除非这个父功能在声明时使用了 `android:required="false"`。

触摸屏硬件功能

`android.hardware.faketouch`

应用使用基本的触摸交互事件，例如点按和拖动。

声明为必需时，此功能表示应用只兼容模拟触摸屏（“假触摸”界面）或实际具有触摸屏的设备。

带有假触摸界面的设备为用户提供模拟部分触摸屏功能的用户输入系统。 例如，鼠标或遥控器可以驱动屏幕光标。 如果您的应用需要基本的点击式交互（换言之，它在只使用方向键控制器的情况下无法正常工作），则应声明该功能。 由于这是最低水平的触摸交互，因此您还可以在提供更复杂触摸界面的设备上使用声明该功能的应用。

> **注**：默认情况下应用需要 `android.hardware.touchscreen`。 如果您希望自己的应用可供提供假触摸界面的设备使用，则必须按如下方式显式声明不要求提供触摸屏：

```
<uses-feature android:name="android.hardware.touchscreen" android:required="false" />
```

`android.hardware.faketouch.multitouch.distinct`

应用在假触摸界面上区分两个或更多个“手指”的触摸轨迹。 这是 `android.hardware.faketouch` 功能的一个超集。 声明为必需时，此功能表示应用只兼容模拟区分两个或更多个手指的触摸轨迹或实际具有触摸屏的设备。

不同于 `android.hardware.touchscreen.multitouch.distinct` 所定义的区分式多点触摸，通过假触摸界面支持区分式多点触摸的输入设备并不支持所有双指手势，因为输入会转换成屏幕上的光标移动。 也就是说，在此类设备上的单指手势移动光标，双指划动触发单指触摸事件，而其他双指手势则触发相应的双指触摸事件。

提供双指触摸触控板进行光标移动的设备可支持该功能。

`android.hardware.faketouch.multitouch.jazzhand`

应用在假触摸界面上区分五个或更多个“手指”的触摸轨迹。 这是 `android.hardware.faketouch` 功能的一个超集。 声明为必需时，此功能表示应用只兼容模拟区分五个或更多个手指的触摸轨迹或实际具有触摸屏的设备。

不同于 `android.hardware.touchscreen.multitouch.jazzhand` 所定义的区分式多点触摸，通过假触摸界面支持单手多点触摸的输入设备并不支持所有五指手势，因为输入会转换成屏幕上的光标移动。 也就是说，在此类设备上的单指手势移动光标，多指手势触发单指触摸事件，而其他多指手势则触发相应的多指触摸事件。

提供五指触摸触控板进行光标移动的设备可支持该功能。

`android.hardware.touchscreen`

应用利用设备的触摸屏功能来实现比基本触摸事件交互性更强的手势，例如滑屏。 这是 `android.hardware.faketouch` 功能的一个超集。

默认情况下，您的应用需要该功能。因此，您的应用不可供默认情况下只提供模拟触摸界面（“假触摸”）的设备使用。 如果您希望自己的应用可供提供假触摸界面的设备（甚至只提供方向键控制器的设备）使用，则必须通过在声明 `android.hardware.touchscreen` 时加入 `android:required="false"` 来显式声明不要求提供触摸屏。 如果您的应用使用（但并不需要）真触摸屏界面，则应添加此声明。

如果您的应用确实需要触摸界面（以便执行滑屏之类的更高级触摸手势），则您无需声明任何触摸界面功能，因为它们在默认情况下是必需功能。 不过，最好还是显式声明您的应用使用的所有功能。

如果您需要进行更复杂的触摸交互（例如多指手势），则应声明您的应用使用高级触摸屏功能。

`android.hardware.touchscreen.multitouch`

应用使用设备的基本两点式多点触摸功能（例如实现张合手势的功能），但应用不需要独立追踪触摸轨迹。 这是 `android.hardware.touchscreen` 功能的一个超集。

应用通过使用该功能暗示其还使用 `android.hardware.touchscreen` 功能，除非这个父功能在声明时使用了 `android:required="false"`。

`android.hardware.touchscreen.multitouch.distinct`

应用使用设备的高级多点触摸功能来独立追踪两点或更多点的轨迹。 该功能是 `android.hardware.touchscreen.multitouch` 功能的一个超集。

应用通过使用该功能暗示其还使用 `android.hardware.touchscreen.multitouch` 功能，除非这个父功能在声明时使用了 `android:required="false"`。

`android.hardware.touchscreen.multitouch.jazzhand`

应用使用设备的高级多点触摸功能来独立追踪五点或更多点的轨迹。 该功能是 `android.hardware.touchscreen.multitouch` 功能的一个超集。

应用通过使用该功能暗示其还使用 `android.hardware.touchscreen.multitouch` 功能，除非这个父功能在声明时使用了

`android:required="false"`。

USB 硬件功能

`android.hardware.usb.accessory`

应用表现为 USB 设备，与 USB 主机相连。

`android.hardware.usb.host`

应用使用与设备相连的 USB 附件。设备充当 USB 主机。

Wi-Fi 硬件功能

`android.hardware.wifi`

应用使用设备上的 802.11 网络 (Wi-Fi) 功能。

`android.hardware.wifi.direct`

应用使用设备上的 Wi-Fi Direct 网络功能。

软件功能

此部分介绍最新平台版本支持的软件功能。 要指出您的应用使用或需要某项软件功能，请在 `android:name` 属性中声明相应的值（以 `"android.software"` 开头）。 请在您每次声明一项软件功能时使用单独的 `<uses-feature>` 元素。

通信软件功能

`android.software.sip`

应用使用会话发起协议 (SIP) 服务。通过使用 SIP，应用可以支持互联网电话操作，例如视频会议和即时消息传递。

`android.software.sip.voip`

应用使用基于 SIP 的互联网语音协议 (VoIP) 服务。通过使用 VoIP，应用可以支持实时互联网电话操作，例如双向视频会议。

应用通过使用该功能暗示其还使用 `android.software.sip` 功能，除非这个父功能在声明时使用了 `android:required="false"`。

`android.software.webview`

应用显示来自互联网的内容。

自定义输入软件功能

`android.software.input_methods`

应用使用新的输入法，该输入法由开发者在 `InputMethodService` 中定义。

设备管理软件功能

`android.software.backup`

应用加入处理备份和恢复操作的逻辑。

`android.software.device_admin`

应用通过设备管理员来强制执行设备规范。

`android.software.managed_users`

应用支持二级用户和托管配置文件。

`android.software.securely_removes_users`

应用可**永久性**移除用户及其相关数据。

`android.software.verified_boot`

应用加入处理设备验证启动功能结果的逻辑，该逻辑可检测设备的配置在重新启动操作期间是否发生了变化。

媒体软件功能

`android.software.midi`

应用利用乐器数字化接口 (MIDI) 协议连接到乐器或输出声音。

`android.software.print`

应用加入打印设备上所显示文档的命令。

`android.software.leanback`

应用呈现专为在大屏幕（例如电视）上观看而设计的 UI。

`android.software.live_tv`

应用流式传输直播电视节目。

屏幕界面软件功能

`android.software.app_widgets`

应用使用或提供应用小部件，并且只应安装在带有可供用户嵌入应用小部件的主屏幕或类似位置的设备上。

`android.software.home_screen`

应用起到替代设备主屏幕的作用。

`android.software.live_wallpaper`

应用使用或提供包含动画的壁纸。

隐含功能要求的权限

一些硬件和软件功能常量是在相应 API 发布后提供给应用使用；例如，`android.hardware.bluetooth` 功能是在 Android 2.2（API 级别 8）中添加的，但它引用的 Bluetooth API 则是在 Android 2.0（API 级别 5）添加的。 因此，一些应用在能够利用 `<uses-feature>` 系统声明其需要某个 API 之前就已经具备了使用该 API 的能力。

为防止无意间提供这些应用，Google Play 会假定特定硬件相关权限规定，默认情况下需要底层硬件功能。 例如，使用蓝牙的应用必须在 `<uses-permission>` 元素中请求 `BLUETOOTH` 权限 — 对于旧版应用，Google Play 假定权限声明意味着，应用需要底层 `android.hardware.bluetooth` 功能，并根据该功能设置过滤。 表 2 所列权限隐含的功能要求等同于 `<uses-feature>` 元素中声明的功能。

请注意，`<uses-feature>` 声明（包括任何声明的 `android:required` 属性）始终优先于表 2 中权限所隐含的功能。对于上述任一权限，您都可以通过在 `<uses-feature>` 元素中使用 `android:required="false"` 属性显式声明隐含功能来停用根据隐含功能进行过滤。 例如，要想停用根据 `CAMERA` 权限进行任何过滤，您需要向清单文件添加下面这个 `<uses-feature>` 声明：

```
<uses-feature android:name="android.hardware.camera" android:required="false" />
```

表 2. 隐含设备硬件用途的设备权限。

类别	此权限...	...隐含此功能要求
蓝牙	BLUETOOTH	android.hardware.bluetooth （如需了解详情，请参阅针对蓝牙功能的特殊处理。）
	BLUETOOTH_ADMIN	android.hardware.bluetooth
相机	CAMERA	android.hardware.camera *和* android.hardware.camera.autofocus
定位	ACCESS_MOCK_LOCATION	android.hardware.location
	ACCESS_LOCATION_EXTRA_COMMANDS	android.hardware.location
	INSTALL_LOCATION_PROVIDER	android.hardware.location
	ACCESS_COARSE_LOCATION	android.hardware.location.network *和* android.hardware.location
	ACCESS_FINE_LOCATION	android.hardware.location.gps *和* android.hardware.location
麦克风	RECORD_AUDIO	android.hardware.microphone
电话	CALL_PHONE	android.hardware.telephony
	CALL_PRIVILEGED	android.hardware.telephony
	MODIFY_PHONE_STATE	android.hardware.telephony
	PROCESS_OUTGOING_CALLS	android.hardware.telephony
	READ_SMS	android.hardware.telephony
	RECEIVE_SMS	android.hardware.telephony
	RECEIVE_MMS	android.hardware.telephony
	RECEIVE_WAP_PUSH	android.hardware.telephony
	SEND_SMS	android.hardware.telephony
	WRITE_APN_SETTINGS	android.hardware.telephony
	WRITE_SMS	android.hardware.telephony
Wi-Fi	ACCESS_WIFI_STATE	android.hardware.wifi
	CHANGE_WIFI_STATE	android.hardware.wifi
	CHANGE_WIFI_MULTICAST_STATE	android.hardware.wifi