

<uses-sdk>

Google Play 会利用在应用清单中声明的 **<uses-sdk>** 属性，从不符合其平台版本要求的设备上**滤除您的应用**。在设置这些属性前，请确保您了解 [Google Play 过滤器](https://developer.android.google.cn/google/play/filters?hl=zh-cn) (https://developer.android.google.cn/google/play/filters?hl=zh-cn)。

语法：

```
<uses-sdk android:minSdkVersion (#min)="integer"
          android:targetSdkVersion (#target)="integer"
          android:maxSdkVersion (#max)="integer" />
```

包含它的文件：

<manifest> (https://developer.android.google.cn/guide/topics/manifest/manifest-element?hl=zh-cn)

说明：

您可以通过整数形式的 API 级别表示应用与一个或多个版本的 Android 平台的兼容性。**应用表示的 API 级别将与给定 Android 系统的 API 级别进行比较**，其结果**在不同 Android 设备上可能有所差异**。

尽管名称如此，但该元素实际用于指定 API 级别，**而非 SDK**（软件开发工具包）或 Android 平台的版本号。API 级别始终是单个整数。您无法通过 API 级别关联的 Android 版本号来推知 API 级别（例如，它不同于主版本号，也不同于主版本号与次版本号之和）。


另请阅读有关**对应用进行版本控制** (https://developer.android.google.cn/tools/publishing/versioning?hl=zh-cn)的文档。

属性：

android:minSdkVersion

一个用于指定应用运行所需最低 API 级别的整数。如果系统的 API 级别低于该属性中指定的值，Android 系统将阻止用户安装应用。您应始终声明该属性。

！ 注意：如果您不声明该属性，系统将假定默认值为“1”，这表示您的应用兼容所有 Android 版本。如果您的应用**并不兼容所有版本**（例如，它使用 API 级别 3 中引入的 API），并且您尚未声明正确的 minSdkVersion，则当应用安装在 API 级别小于 3 的系统上时，应用将在运行时**尝试访问不可用的 API 时发生崩溃**。因此，请务必在 minSdkVersion 属性中声明合适的 API 级别。

android:targetSdkVersion 

一个用于指定应用的目标 API 级别的整数。如果未设置，其默认值与为 minSdkVersion 指定的值相等。

该属性用于通知系统，您**已针对目标版本进行测试**，并且**系统不应通过启用任何兼容性行为**，以保持您的应用与目标版本的向前兼容性。应用仍可在较低版本上运行（最低版本为 minSdkVersion）。

Android 会随新版本的推出而逐渐发展，在此过程中，某些行为乃至外观可能会发生变化。不过，**如果平台的 API 级别高于应用 targetSdkVersion 所声明的版本**，**系统便可通过启用兼容性行为**，确保应用继续以您所期望的方式工作。您可以将 **targetSdkVersion 指定为符合应用所运行平台的 API 级别**，**从而停用此类兼容性行为**。例如，如果将该值设置为“11”或更高，系统便可在应用运行在 Android 3.0 或更高版本的平台上时对其应用新的默认主题 (Holo)，还可在应用运行在更大屏幕上时停用**屏幕兼容性模式** (https://developer.android.google.cn/guide/practices/screen-compat-mode?hl=zh-cn)（因为对 API 级别 11 的支持隐含了对更大屏幕的支持）。

系统可根据您为该属性设置的值启用许多兼容性行为。 [Build.VERSION_CODES](https://developer.android.google.cn/reference/android/os/Build.VERSION_CODES?hl=zh-cn)

(https://developer.android.google.cn/reference/android/os/Build.VERSION_CODES?hl=zh-cn) 参考资料中的相应平台版本介绍了其中几种行为。

如要让应用与各 Android 版本保持同步，您应增加该属性的值，使其与最新 API 级别一致，然后在相应平台版本上对应用进行全面测试。

引入的版本：API 级别 4

android:maxSdkVersion

一个指定作为应用设计运行目标的最高 API 级别的整数。

在 Android 1.5、1.6、2.0 及 2.0.1 中，系统会在安装应用时，以及系统更新后重新验证应用时检查该属性的值。任一情况下，如果应用的 maxSdkVersion 属性低于系统本身使用的 API 级别，则系统将不允许安装应用。在系统更新后重新验证的情况下，这实际相当于将您的应用从设备中移除。

为说明该属性在系统更新后对应用的影响，请查看以下示例：

Google Play 上发布了一个在其清单中声明 maxSdkVersion="5" 的应用。一位设备运行 Android 1.6（API 级别 4）的用户下载并安装了该应用。几周后，该用户收到 Android 2.0（API 级别 5）OTA 系统更新。更新安装后，系统检查该应用的 maxSdkVersion 并顺利对其完成重新验证。应用仍可照常工作。不过，一段时间后，设备再次收到系统更新，这次是更新到 Android 2.0.1（API 级别 6）。更新完成后，系统无法再重新验证应用，因为此时系统本身的 API 级别 (6) 已超过该应用支持的最高级别 (5)。系统会使该应用对用户不可见，实际相当于将应用从设备上删除。

⚠

警告：不建议声明该属性。首先，您没有必要设置该属性，并将其作为阻止您的应用部署至新版本 Android 平台的一种手段。从设计上讲，新版本平台完全向后兼容。只要您的应用仅使用标准 API 并遵循部署最佳实践，其应能够在新版本平台上正常工作。其次，请注意在某些情况下，声明该属性可能会导致您的应用在系统更新至更高 API 级别后从用户设备中移除。大多数可能安装您应用的设备都会定期收到 OTA 系统更新，因此您应在设置该属性前考虑这些更新对应用的影响。

引入的版本：API 级别 4

★

在安装或重新验证时，未来的 Android 版本（Android 2.0.1 之后的版本）将不再检查或强制执行 maxSdkVersion 属性。不过，在向用户呈现可供下载的应用时，Google Play 会继续使用该属性作为过滤器。

引入的版本：

API 级别 1

什么是 API 级别？

API 级别是对 Android 平台版本提供的框架 API 修订版进行唯一标识的整数值。

Android 平台提供一种框架 API，应用可利用它与底层 Android 系统进行交互。该框架 API 由以下部分组成：

- 一组核心软件包和类
- 一组用于声明清单文件的 XML 元素和属性
- 一组用于声明和访问资源的 XML 元素和属性
- 一组 Intent
- 一组应用可请求的权限，以及系统中包括的权限强制执行

Android 平台的每个后续版本均可包括其提供的 Android 应用框架 API 的更新。

框架 API 更新的设计用途是使新 API 与早期版本的 API 保持兼容。换言之，大多数 API 更改都是新增更改，并且会引入新功能或替代功能。在 API 的某些部分得到升级时，系统会弃用经替换的旧版部分，但不会将其移除，以便其仍可供现有应用使用。**在极少数情况下**，系统可能会修改或移除 API 的某些部分，但**通常只有在为确保 API 稳健性以及应用或系统安全性时**，才需要进行此类更改。所有其他来自早期修订版的 API 部分都将继续保留，不做任何修改。

Android 平台提供的框架 API 使用称为“API 级别”的整数标识符指定。每个 Android 平台版本恰好支持一个 API 级别，但隐含对所有早期 API 级别（低至 API 级别 1）的支持。Android 平台初始版本提供的是 API 级别 1，后续版本的 API 级别则依次增加。

下表列出了各 Android 平台版本所支持的 API 级别。如需了解有关运行各版本的设备相对数量的信息，请参阅“[平台版本](https://developer.android.google.cn/about/dashboards?hl=zh-cn)”信息中心页面 (https://developer.android.google.cn/about/dashboards?hl=zh-cn)。

平台版本	API 级别	VERSION_COD
Android 10.0 (https://developer.android.google.cn/preview?hl=zh-cn)	29 (https://developer.android.google.cn/sdk/api_diff/29/changes?hl=zh-cn)	Q (https://developer.android.google.cn/about/dashboards?hl=zh-cn#Q)

Android 9 (https://developer.android.google.cn/about/versions/pie?hl=zh-cn)	28 (https://developer.android.google.cn/sdk/api_diff/28/changes?hl=zh-cn)	P (https://developer.android.google.cn/sdk/api_diff/28/changes?hl=zh-cn#P)
Android 8.1 (https://developer.android.google.cn/about/versions/oreo/android-8.1?hl=zh-cn)	27 (https://developer.android.google.cn/sdk/api_diff/27/changes?hl=zh-cn)	O_MR1 (https://developer.android.google.cn/sdk/api_diff/27/changes?hl=zh-cn#O_MR1)
Android 8.0 (https://developer.android.google.cn/about/versions/oreo?hl=zh-cn)	26 (https://developer.android.google.cn/sdk/api_diff/26/changes?hl=zh-cn)	O (https://developer.android.google.cn/sdk/api_diff/26/changes?hl=zh-cn#O)
Android 7.1.1 Android 7.1 (https://developer.android.google.cn/about/versions/nougat/android-7.1?hl=zh-cn)	25 (https://developer.android.google.cn/sdk/api_diff/25/changes?hl=zh-cn)	N_MR1 (https://developer.android.google.cn/sdk/api_diff/25/changes?hl=zh-cn#N_MR1)
Android 7.0 (https://developer.android.google.cn/about/versions/nougat/android-7.0?hl=zh-cn)	24 (https://developer.android.google.cn/sdk/api_diff/24/changes?hl=zh-cn)	N (https://developer.android.google.cn/sdk/api_diff/24/changes?hl=zh-cn#N)
Android 6.0 (https://developer.android.google.cn/about/versions/marshmallow/android-6.0?hl=zh-cn)	23 (https://developer.android.google.cn/sdk/api_diff/23/changes?hl=zh-cn)	M (https://developer.android.google.cn/sdk/api_diff/23/changes?hl=zh-cn#M)
Android 5.1 (https://developer.android.google.cn/about/versions/android-5.1?hl=zh-cn)	22 (https://developer.android.google.cn/sdk/api_diff/22/changes?hl=zh-cn)	LOLLIPOP_MR1 (https://developer.android.google.cn/sdk/api_diff/22/changes?hl=zh-cn#LOLLIPOP_MR1)
Android 5.0 (https://developer.android.google.cn/about/versions/android-5.0?hl=zh-cn)	21 (https://developer.android.google.cn/sdk/api_diff/21/changes?hl=zh-cn)	LOLLIPOP (https://developer.android.google.cn/sdk/api_diff/21/changes?hl=zh-cn#LOLLIPOP)
Android 4.4W	20 (https://developer.android.google.cn/sdk/api_diff/20/changes?hl=zh-cn)	KITKAT_WATCH (https://developer.android.google.cn/sdk/api_diff/20/changes?hl=zh-cn#KITKAT_WATCH)
Android 4.4 (https://developer.android.google.cn/about/versions/android-4.4?hl=zh-cn)	19 (https://developer.android.google.cn/sdk/api_diff/19/changes?hl=zh-cn)	KITKAT (https://developer.android.google.cn/sdk/api_diff/19/changes?hl=zh-cn#KITKAT)
Android 4.3 (https://developer.android.google.cn/about/versions/android-4.3?hl=zh-cn)	18 (https://developer.android.google.cn/sdk/api_diff/18/changes?hl=zh-cn)	JELLY_BEAN_MR1 (https://developer.android.google.cn/sdk/api_diff/18/changes?hl=zh-cn#JELLY_BEAN_MR1)
Android 4.2, 4.2.2 (https://developer.android.google.cn/about/versions/android-4.2?hl=zh-cn)	17 (https://developer.android.google.cn/sdk/api_diff/17/changes?hl=zh-cn)	JELLY_BEAN_MR1 (https://developer.android.google.cn/sdk/api_diff/17/changes?hl=zh-cn#JELLY_BEAN_MR1)
Android 4.1, 4.1.1 (https://developer.android.google.cn/about/versions/android-4.1?hl=zh-cn)	16 (https://developer.android.google.cn/sdk/api_diff/16/changes?hl=zh-cn)	JELLY_BEAN (https://developer.android.google.cn/sdk/api_diff/16/changes?hl=zh-cn#JELLY_BEAN)
Android 4.0.3, 4.0.4 (https://developer.android.google.cn/about/versions/android-4.0.3?hl=zh-cn)	15 (https://developer.android.google.cn/sdk/api_diff/15/changes?hl=zh-cn)	ICE_CREAM_SANDWICH (https://developer.android.google.cn/sdk/api_diff/15/changes?hl=zh-cn#ICE_CREAM_SANDWICH)
Android 4.0, 4.0.1, 4.0.2 (https://developer.android.google.cn/about/versions/android-4.0?hl=zh-cn)	14 (https://developer.android.google.cn/sdk/api_diff/14/changes?hl=zh-cn)	ICE_CREAM_SANDWICH (https://developer.android.google.cn/sdk/api_diff/14/changes?hl=zh-cn#ICE_CREAM_SANDWICH)
Android 3.2 (https://developer.android.google.cn/about/versions/android-3.2?hl=zh-cn)	13 (https://developer.android.google.cn/sdk/api_diff/13/changes?hl=zh-cn)	HONEYCOMB_MR2 (https://developer.android.google.cn/sdk/api_diff/13/changes?hl=zh-cn#HONEYCOMB_MR2)
Android 3.1.x (https://developer.android.google.cn/about/versions/android-3.1?hl=zh-cn)	12 (https://developer.android.google.cn/sdk/api_diff/12/changes?hl=zh-cn)	HONEYCOMB_MR1 (https://developer.android.google.cn/sdk/api_diff/12/changes?hl=zh-cn#HONEYCOMB_MR1)
Android 3.0.x (https://developer.android.google.cn/about/versions/android-3.0?hl=zh-cn)	11 (https://developer.android.google.cn/sdk/api_diff/11/changes?hl=zh-cn)	HONEYCOMB (https://developer.android.google.cn/sdk/api_diff/11/changes?hl=zh-cn#HONEYCOMB)
Android 2.3.4 Android 2.3.3	10 (https://developer.android.google.cn/sdk/api_diff/10/changes?hl=zh-cn)	GINGERBREAD_MR1 (https://developer.android.google.cn/sdk/api_diff/10/changes?hl=zh-cn#GINGERBREAD_MR1)

https://developer.android.google.cn/about/versions/android-2.3.3?hl=zh-cn	https://developer.android.google.cn/sdk/api_diff/10/changes?hl=zh-cn	https://developer.android.google.cn/sdk/api_diff/10/changes?hl=zh-cn#GINGERBREAD
Android 2.3.2 Android 2.3.1 Android 2.3 (https://developer.android.google.cn/about/versions/android-2.3?hl=zh-cn)	9 (https://developer.android.google.cn/sdk/api_diff/9/changes?hl=zh-cn)	GINGERBREAD (https://developer.android.google.cn/sdk/api_diff/9/changes?hl=zh-cn#GINGERBREAD)
Android 2.2.x (https://developer.android.google.cn/about/versions/android-2.2?hl=zh-cn)	8 (https://developer.android.google.cn/sdk/api_diff/8/changes?hl=zh-cn)	FROYO (https://developer.android.google.cn/sdk/api_diff/8/changes?hl=zh-cn#FROYO)
Android 2.1.x (https://developer.android.google.cn/about/versions/android-2.1?hl=zh-cn)	7 (https://developer.android.google.cn/sdk/api_diff/7/changes?hl=zh-cn)	ECLAIR_MR1 (https://developer.android.google.cn/sdk/api_diff/7/changes?hl=zh-cn#ECLAIR_MR1)
Android 2.0.1 (https://developer.android.google.cn/about/versions/android-2.0.1?hl=zh-cn)	6 (https://developer.android.google.cn/sdk/api_diff/6/changes?hl=zh-cn)	ECLAIR_0_1 (https://developer.android.google.cn/sdk/api_diff/6/changes?hl=zh-cn#ECLAIR_0_1)
Android 2.0 (https://developer.android.google.cn/about/versions/android-2.0?hl=zh-cn)	5 (https://developer.android.google.cn/sdk/api_diff/5/changes?hl=zh-cn)	ECLAIR (https://developer.android.google.cn/sdk/api_diff/5/changes?hl=zh-cn#ECLAIR)
Android 1.6 (https://developer.android.google.cn/about/versions/android-1.6?hl=zh-cn)	4 (https://developer.android.google.cn/sdk/api_diff/4/changes?hl=zh-cn)	DONUT (https://developer.android.google.cn/sdk/api_diff/4/changes?hl=zh-cn#DONUT)
Android 1.5 (https://developer.android.google.cn/about/versions/android-1.5?hl=zh-cn)	3 (https://developer.android.google.cn/sdk/api_diff/3/changes?hl=zh-cn)	CUPCAKE (https://developer.android.google.cn/sdk/api_diff/3/changes?hl=zh-cn#CUPCAKE)
Android 1.1 (https://developer.android.google.cn/about/versions/android-1.1?hl=zh-cn)	2	BASE_1_1 (https://developer.android.google.cn/sdk/api_diff/2/changes?hl=zh-cn#BASE_1_1)
Android 1.0	1	BASE (https://developer.android.google.cn/sdk/api_diff/1/changes?hl=zh-cn#BASE)

API 级别在 Android 中的使用

在确保尽可能为用户和应用开发者提供最佳体验方面，API 级别标识符发挥着重要作用：

- 它允许 Android 平台描述其支持的最高框架 API 修订版
- 它允许应用描述其需要的框架 API 修订版
- 它允许系统协商在用户设备上安装应用，从而不安装非兼容版本的应用。

每个 Android 平台版本都将其 API 级别标识符存储在 Android 系统自身内部。

应用可利用框架 API 提供的清单元素 (`<uses-sdk>`) 来说明其可运行的最低和最高 API 级别，以及其支持的首选 API 级别。该元素具有以下三个重要属性：

- `android:minSdkVersion` — 指定能够运行应用的最低 API 级别。默认值为“1”。
- `android:targetSdkVersion` — 指定运行应用的目标 API 级别。在某些情况下，此属性允许应用使用在目标 API 级别中定义的清单元素或行为，而非仅限于使用针对最低 API 级别定义的元素或行为。
- `android:maxSdkVersion` — 指定能够运行应用的最高 API 级别。**重要说明：**在使用该属性之前，请先阅读 `<uses-sdk>` (<https://developer.android.google.cn/guide/topics/manifest/uses-sdk-element?hl=zh-cn>) 文档。

例如，如要指定应用运行所需的最低系统 API 级别，应用需在其清单中加入带 `android:minSdkVersion` 属性的 `<uses-sdk>` 元素。`android:minSdkVersion` 是一个整数值，对应能够运行应用的最低版本 Android 平台的 API 级别。

当用户试图安装应用，或在系统更新后重新验证应用时，Android 系统会先检查应用清单中的 `<uses-sdk>` 属性，然后将这些属性的值与其自己的内部 API 级别进行对比。只有在符合以下条件时，系统才允许安装开始：

- 如果声明 `android:minSdkVersion` 属性，其值必须小于或等于系统的 API 级别整数。如果未声明，则系统假定应用需要 API 级别 1。
- 如果声明 `android:maxSdkVersion` 属性，其值必须大于或等于系统的 API 级别整数。如果未声明，则系统假定应用没有最高 API 级别。如需了解有关系统如何处理该属性的详细信息，请阅读 [<uses-sdk>](https://developer.android.google.cn/guide/topics/manifest/uses-sdk-element?hl=zh-cn) (<https://developer.android.google.cn/guide/topics/manifest/uses-sdk-element?hl=zh-cn>) 文档。

如果在应用清单中进行声明，则 `<uses-sdk>` 元素的内容可能如下所示：

```
<manifest>
  <uses-sdk android:minSdkVersion="5" />
  ...
</manifest>
```

应用在 `android:minSdkVersion` 中声明 API 级别的主要原因是，告知 Android 系统，其正使用在指定 API 级别引入的 API。如果由于某种原因将应用安装在 API 级别较低的平台上，则它会在运行时试图访问不存在的 API 时发生崩溃。如果应用所需的最低 API 级别高于目标设备上平台版本的 API 级别，则系统不允许安装该应用，以防出现这种结果。

例如，[android.appwidget](https://developer.android.google.cn/reference/android/appwidget/package-summary?hl=zh-cn) (<https://developer.android.google.cn/reference/android/appwidget/package-summary?hl=zh-cn>) 软件包是随 API 级别 3 引入的。如果应用使用该 API，则必须使用“3”值声明 `android:minSdkVersion` 属性。随后，应用便可安装在 Android 1.5（API 级别 3）和 Android 1.6（API 级别 4）等平台上，但不能安装在 Android 1.1（API 级别 2）和 Android 1.0（API 级别 1）平台上。

如需了解有关如何指定应用 API 级别要求的详细信息，请参阅清单文件文档的 [<uses-sdk>](https://developer.android.google.cn/guide/topics/manifest/uses-sdk-element?hl=zh-cn) (<https://developer.android.google.cn/guide/topics/manifest/uses-sdk-element?hl=zh-cn>) 部分。

开发注意事项

下文提供您在开发应用时应考虑的 API 级别的相关信息。

应用向前兼容性

Android 应用一般向前兼容新版本的 Android 平台。

由于几乎所有对框架 API 的更改都是新增更改，所以使用 API 任何给定版本（其 API 级别所指定版本）开发的 Android 应用均向前兼容更新版本的 Android 平台以及更高 API 级别。应用应能在所有后期版本的 Android 平台上运行，除非在个别情况下，系统后来因某种原因将应用使用的某个 API 部分移除。

向前兼容性非常重要，因为许多 Android 设备都会接收 OTA 系统更新。用户可以安装您的应用并顺利使用，然后通过接收 OTA 更新升级到新版本的 Android 平台。安装更新后，您的应用将在新运行时版本的环境中运行，但此版本包含您应用所依赖的 API 和系统功能。

在某些情况下，在该 API 之下所做的更改（例如对底层系统本身的更改）可能会影响在新环境中运行的应用。因此，作为应用开发者，您必须了解应用在各系统环境中的外观和行为。为帮助您在各种版本的 Android 平台上测试应用，Android SDK 提供多个可供您下载的平台。每个平台均包含兼容的系统映像，**您可以通过在 AVD 中运行该映像来测试应用。**

应用向后兼容性

Android 应用未必向后兼容比其编译时所用目标版本更旧的 Android 平台版本。

每个新版本的 Android 平台都可能包含新的框架 API，例如能够让应用使用新的平台功能或替换现有 API 部分的 API。在新平台上运行时，应用可以使用这些新 API；且如上所述，在更新版本的平台（API 级别所指定的平台）上运行时，应用也可使用这些新 API。反之，由于早期版本的平台未包含新 API，因此使用新 API 的应用无法在这些平台上运行。

尽管 Android 设备降级至先前版本平台的情况不太可能发生，但您必须认识到，可能有许多设备运行的是早期版本的平台。即便是在接收 OTA 更新的设备中，有些设备的更新也可能会滞后，而且可能在相当长的时间内无法接收更新。

选择平台版本和 API 级别

在开发应用时，您需要选择编译应用所用的平台版本。一般而言，您应根据应用所支持平台的最低版本编译应用。

您可以在编译应用时依次降低其使用的目标版本，从而确定可能的最低平台版本。确定最低版本后，您应使用相应平台版本（和 API 级别）创建 AVD，然后对您的应用进行全面测试。请务必在应用的清单中声明 `android:minSdkVersion` 属性，并将其值设置为平台版本的 API 级别。

声明最低 API 级别

构建应用时，如果您使用最新平台版本中引入的 API 或系统功能，则应将 `android:minSdkVersion` 属性设置为最新平台版本的 API 级别。如此一来，便可确保用户只能在运行兼容版 Android 平台的设备上安装您的应用，并进而确保您的应用可以在用户设备上正常工作。

如果您的应用使用最新平台版本中引入的 API，但未声明 `android:minSdkVersion` 属性，则该应用可在运行最新版本平台的设备上正常工作，但无法在运行早期版本平台的设备上正常工作。在后一种情况下，当在运行时试图使用早期版本上不存在的 API 时，应用将发生崩溃。

针对更高 API 级别进行测试

编译应用之后，您应确保在应用的 `android:minSdkVersion` 属性所指定的平台上对其进行测试。为此，请使用您应用所需的平台版本创建 AVD。此外，为确保向前兼容性，您还应在所有 API 级别高于您应用 API 级别的平台上运行并测试您的应用。

Android SDK 提供多个可供您使用的平台版本（包括最新版本），以及可供您在必要时下载其他平台版本的更新器工具。

如要访问该更新器，请使用位于 `<sdk>/tools` 目录的 `android` 命令行工具。您可以通过执行 `android sdk` 来启动 SDK 更新器。您还可直接双击 `android.bat` (Windows) 或 `Android (OS X/Linux)` 文件。

如要在模拟器中的不同平台版本上运行应用，请为您想测试的每个平台版本创建 AVD。如需了解有关 AVD 的详细信息，请参阅[创建和管理虚拟设备](https://developer.android.google.cn/tools/devices?hl=zh-cn) (<https://developer.android.google.cn/tools/devices?hl=zh-cn>)。如要使用物理设备进行测试，请确保您知晓其所运行 Android 平台的 API 级别。请参阅本文顶部表格中所列的平台版本及其 API 级别。

按 API 级别过滤参考文档

Android Developers 网站在[每个参考文档页面的右上角](#)提供一个“Filter by API Level”控件。利用该控件，您可根据应用在其清单文件的 `android:minSdkVersion` 属性中所指定的 API 级别，只显示该应用实际可访问 API 部分的对应文档。

如要使用过滤功能，请选择页面搜索框下方用于启用过滤功能的复选框。然后将“Filter by API Level”控件设置为应用所指定的同一 API 级别。请注意，后期 API 级别中引入的 API 随即会灰显，并且系统会屏蔽其内容，因为您的应用将无法访问它们。

如果在文档中按 API 级别进行过滤，则您并不能查看各 API 级别新增或引入的 API — 它仅仅是提供一种途径，让您能够查看与给定 API 级别关联的整个 API，同时将后期 API 级别中引入的 API 元素排除在外。

如果您决定不想过滤 API 文档，只需使用复选框停用该功能。默认情况下，系统会停用 API 级别过滤，这样无论 API 级别如何，您都能查看完整的框架 API。

另请注意，各 API 元素的参考文档指定引入各元素的 API 级别。各文档页面内容区域的右上角会以“Since <api 级别>”的形式指定软件包和类的 API 级别。详细说明标头（位于右边距处）中会指定类成员的 API 级别。

Content and code samples on this page are subject to the licenses described in the [Content License](https://developer.android.google.cn/license?hl=zh-cn) (<https://developer.android.google.cn/license?hl=zh-cn>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2019-12-27 UTC.