

Android 9 Pie

开发者手册

本手册包含哪些内容？

本手册旨在帮助您快速了解 Android 9。在本手册中，您将深入了解此系统的新特性和 API、以及如何迁移您的应用到 Android 9 上。我们期待这些分享会对您在新系统的使用方面有所帮助。

此外，手册还有一些新系统开发的常见问题以及您可以参考的一些资源。我们希望这些内容可以帮助您率先感受 Android 9 给您带来的各种创新和改善，以确保您的应用脱颖而出。

我们一直在不断寻找各种方法来改善我们的产品和服务，为此，我们也会定期更新本手册，以便为您提供最新信息。这意味着，当您使用本手册所描述的某些功能或特点时，他们可能会发生变化，在这种情况下，请遵循相应产品中的说明，或访问相应产品的帮助中心。

目录：

如何阅读本手册	3
要了解本手册的内容，请查看此示例页面	
<hr/>	
第 1 章	
Android 9 介绍	4
快速了解 Android 9 的新功能和 API，方便开发者进行高效的开发和迁移	
<hr/>	
第 2 章	
迁移到 Android 9	54
帮助您快速了解迁移到 Android 9 的方法、步骤和手段	
<hr/>	
第 3 章	
常见问题	74
Android 9 开发中的常见问答和建议	
<hr/>	
第 4 章	
相关资源	80
关注 Android 开发者网站、微信公众号，并在帮助中心查找相关问题的答案	

如何阅读本手册

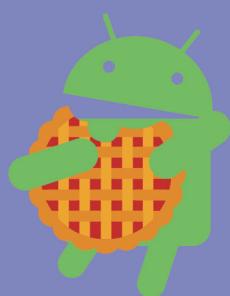
开发新手，可以从第一章读起，内有 Android 9 的系统介绍与功能更新；

经验高手，直接从第二章读起，以便快速迁移现有应用；

为了使开发者深入了解新系统的应用和功能，您可以在更多资源章节中扫描二维码进入视频主页，观看更加生动的功能解析视频。

第1章

Android 9 介绍



目录：

1.1 Android 9 操作系统概述

1.2 Android 9 功能和 API

1.2.1 利用 Wi-Fi RTT 进行室内定位

1.2.2 显示屏缺口支持

1.2.3 通知

1.2.4 多摄像头支持和摄像头更新

1.2.5 适用于可绘制对象和位图的 ImageDecoder

1.2.6 动画

1.2.7 HDR VP9 视频、HEIF 图像压缩和 Media API

1.2.8 JobScheduler 中的流量费用敏感度

1.2.9 Neural Networks API 1.1

1.2.10 自动填充框架



1.2.11 安全增强功能

1.2.12 Android 备份

1.2.13 无障碍功能

1.2.14 旋转

1.2.15 文本

1.2.16 设备端系统跟踪

1.3 Android 9 行为变更指南

1.3.1 针对 Android 9 的所有应用

1.3.2 针对 API 28+ 级别的所有应用

1.1 Android 9 操作系统概述



Android 9 利用人工智能技术，让您的手机更加智能、简洁与人性化。请通过本手册了解 Android 9 的所有新特性，同时也希望各位开发者能够借助 Android 9，全面提升应用性能，打造出出色体验，让您的应用与用户走得更近！

以机器学习为核心，打造更为智能的手机：Android 9 赋予手机强大的学习能力，系统能够根据用户在使用过程中展露的习惯与偏好，进行自我学习与适应——从强劲续航到人性化应用推荐，Android 9 都能想您所想，保障持久流畅的用户体验。

人机交互，就是这么容易：“让智能手机更加智能”是我们向前迈进的重要一步。但是，如何把握好用户与科技的关系——让科技以人为本，这一点也同样关键。在 Android 9 中，我们大幅度改进了用户界面，让它更简洁，也更易于操作；对于开发者而言，这些变更能够让用户更加容易搜索，使用和管理您的应用。

我们会继续努力将 Android 打造成一流的开放平台，助力全球开发者取得商业成功。期望各位能够利用 Android 9 这个新平台并借助 Google Play 中新增的功能，开发出优质的应用和游戏，通过最有效和安全的方式，为全球用户带去精彩体验！

1.2 Android 9 功能和 API



Android 9（API 级别 28）为用户和开发者引入了众多新特性和新功能。本章节重点介绍面向开发者的新功能。

请务必查阅 Android 9 行为变更以了解平台变更可能对应用产生影响的各个方面。

1.2.1

利用 Wi-Fi RTT 进行室内定位



Android 9 添加了对 IEEE 802.11mc Wi-Fi 协议（也称为 Wi-Fi Round-Trip-Time (RTT)）的平台支持，从而让您的应用可以利用室内定位功能。

在运行 Android 9 且具有硬件支持的设备上，应用可以使用 RTT API 来测量与附近支持 RTT 的 Wi-Fi 接入点 (AP) 的距离。设备必须已启用位置服务并开启 Wi-Fi 扫描（在 Settings > Location 下），同时您的应用必须具有 `ACCESS_FINE_LOCATION` 权限。

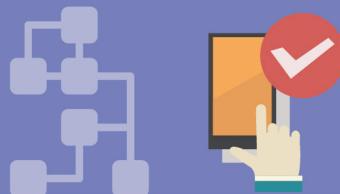
设备无需连接到接入点即可使用 RTT。为了保护隐私，只有手机可以确定与接入点的距离；接入点无此信息。

如果您的设备测量与 3 个或更多接入点的距离，您可以使用一个多点定位算法来预估与这些测量值最相符的设备位置。结果通常精准至 1 至 2 米。

通过这种精确性，您可以打造新的体验，例如楼内导航、基于精细位置的服务，如无歧义语音控制（例如，“打开这盏灯”），以及基于位置的信息（如“此产品是否有特别优惠？”）。



1.2.2 显示屏缺口支持



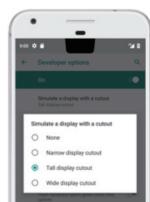
Android 9 支持最新的全面屏，其中包含为摄像头和扬声器预留空间的屏幕缺口。通过 `DisplayCutout` 类可确定非功能区域的位置和形状，这些区域不应显示内容。要确定这些屏幕缺口区域是否存在及其位置，请使用 `getDisplayCutout()` 函数。

全新的窗口布局属性 `layoutInDisplayCutoutMode` 让您的应用可以为设备屏幕缺口周围的内容进行布局。您可以将此属性设为下列值之一：

- `LAYOUT_IN_DISPLAY_CUTOUT_MODE_DEFAULT`
- `LAYOUT_IN_DISPLAY_CUTOUT_MODE_SHORT_EDGES`
- `LAYOUT_IN_DISPLAY_CUTOUT_MODE_NEVER`

可以按以下方法在任何运行 Android 9 的设备或模拟器上模拟屏幕缺口：

1. 启用开发者选项。
2. 在 Developer options 屏幕中，向下滚动至 Drawing 部分并选择 Simulate a display with a cutout。
3. 选择屏幕缺口的大小。



通过使用模拟器测试屏幕缺口

注：我们建议您通过使用运行 Android 9 的设备或模拟器测试屏幕缺口周围的内容显示。

1.2.3 通知



Android 9 引入了多个通知增强功能，可供以（API 级别 28）及以上版本作为目标平台的开发者使用。

提升短信体验

从 Android 7.0（API 级别 24）开始，您可以添加一个操作以回复短信或直接从通知中输入其他文本。Android 9 通过下列增强提升了该**功能**：

- 简化了针对对话参与者的支持：Person 类可用于识别参与对话的人员，包括他们的头像和 URI。现在，许多其他 API（如[addMessage\(\)](#)）均可利用 [Person] 类而不是 CharSequence。Person 类也支持构建器设计模式。
- 支持图像：现在，Android 9 可在手机的“短信通知”中显示图像。您可以使用对短信使用 [setData\(\)](#) 来显示图像。
- 将回复另存为草稿：当用户无意中关闭一个短信通知时，您的应用可以检索系统发送的 [EXTRA_REMOTE_INPUT_DRAFT](#)。您可以使用此 extra 预填充应用中的文本字段，以便用户可以完成他们的回复。
- 确定对话是否为群组对话。您可以使用 [setGroupConversation\(\)](#) 以明确确定对话是否为群组对话。

- 为 Intent 设置语义操作：`setSemanticAction()` 函数允许您为操作提供语义含义，如“标记为已读”、“删除”和“回复”等。
- SmartReply：Android 9 支持在您的短信应用中提供相同的建议回复。使用 `RemoteInput.setChoices()` 为用户提供一组标准回复。

渠道设置、广播和请勿打扰

Android 8 引入了通知渠道，允许您为要显示的每种通知类型创建可由用户自定义的渠道。Android 9 通过下列变更简化通知渠道设置：

- 屏蔽渠道组：现在，用户可以针对某个应用在通知设置中屏蔽整个渠道组。您可以使用 `isBlocked()` 函数确定何时屏蔽一个渠道组，从而不会向该组中的渠道发送任何通知。
此外，您的应用可以使用全新的 `getNotificationChannelGroup()` 函数查询当前渠道组设置。
- 全新的广播 Intent 类型：现在，当通知渠道和渠道组的屏蔽状态发生变更时，Android 系统将发送广播 Intent。拥有已屏蔽的渠道或渠道组的应用可以侦听这些 Intent 并做出相应的回应。有关这些 Intent 操作和 extra 的更多信息，请参阅 `NotificationManager` 参考中更新的常量列表。
- `NotificationManager.Policy` 有 3 种新的“请勿打扰”优先级类别：
 - `PRIORITY_CATEGORY_ALARMS` 优先处理警报。
 - `PRIORITY_CATEGORY_MEDIA` 优先处理媒体源的声音，如媒体和语音导航。
 - `PRIORITY_CATEGORY_SYSTEM` 优先处理系统声音。

- NotificationManager.Policy 还有 7 种新的“请勿打扰”常量，可以用来抑制视觉中断：
 - `SUPPRESSED_EFFECT_FULL_SCREEN_INTENT` 防止通知启动全屏 Activity。
 - `SUPPRESSED_EFFECT_LIGHTS` 屏蔽通知灯。
 - `SUPPRESSED_EFFECT_PEEK` 防止通知短暂进入视图（“滑出”）。
 - `SUPPRESSED_EFFECT_STATUS_BAR` 防止通知显示在支持状态栏的设备的状态栏中。
 - `SUPPRESSED_EFFECT_BADGE` 在支持标志的设备上屏蔽标志。
 - `SUPPRESSED_EFFECT_AMBIENT` 在支持微光显示的设备上屏蔽通知。
 - `SUPPRESSED_EFFECT_NOTIFICATION_LIST` 防止通知显示在支持列表视图（如通知栏或锁屏）的设备的列表视图中。

1.2.4

多摄像头支持和摄像头更新



在运行 Android 9 的设备上，您可以通过两个或更多物理摄像头来同时访问多个视频流。在配备双前置摄像头或双后置摄像头的设备上，您可以创建只配备单摄像头的设备所不可能实现的创新功能，例如无缝缩放、背景虚化和立体成像。通过该 API，您还可以调用逻辑或融合的摄像头视频流，该视频流可在两个或更多摄像头之间自动切换。

摄像头方面的其他改进还包括附加会话参数和 Surface 共享，前者有助于降低首次拍照期间的延迟，而后者则让摄像头客户端能够处理各种用例，而无需停止并启动摄像头视频流。我们还针对基于显示屏的 flash 支持和 OIS 时间戳访问新增了一些 API，用以实现应用级的图像稳定化和特效。

在 Android 9 中，多摄像头 API 支持单色摄像头，适用于具有 FULL 或 LIMITED 功能的设备。单色输出通过 `YUV_420_888` 格式实现，Y 为灰度，U (Cb) 为 128，V (Cr) 为 128。

在受支持的设备上，Android 9 还支持外置 USB/UVC 摄像头。

1.2.5 适用于可绘制对象和位图的 ImageDecoder



Android 9 引入了 `ImageDecoder` 类，可提供现代化的图像解码方法。使用该类取代 `BitmapFactory` 和 `BitmapFactory.Options` API。

`ImageDecoder` 让您可通过字节缓冲区、文件或 URI 来创建 `Drawable` 或 `Bitmap`。要解码图像，请首先以编码图像的来源为参数，调用 `createSource()`。然后，通过传递 `ImageDecoder.Source` 对象来调用 `decodeDrawable()` 或 `decodeBitmap()`，从而创建 `Drawable` 或 `Bitmap`。要更改默认设置，请将 `OnHeaderDecodedListener` 传递给 `decodeDrawable()` 或 `decodeBitmap()`。`ImageDecoder` 调用 `onHeaderDecoded()`，以图像的默认宽度和高度（若已知）为参数。如果编码图像是动画 GIF 或 WebP，`decodeDrawable()` 将返回 `Drawable`，它是 `AnimatedImageDrawable` 类的一个实例。

您可以使用不同的方法来设置图像**属性**:

- 要将解码的图像缩放到精确尺寸，请将目标尺寸传递给 `setTargetSize()`。您也可以使用样图尺寸来缩放图像。将样图尺寸直接传递给 `setTargetSampleSize()`。
- 要在缩放图像的范围内裁剪图像，请调用 `setCrop()`。
- 要创建可变位图，请将 `true` 传递给 `setMutableRequired()`。

通过 `ImageDecoder` 还可以为圆角或圆形遮罩之类的图像添加复杂的定制效果。以 `PostProcessor` 类的一个实例作为参数使用 `setPostProcessor()`，执行您所需的任何绘图命令。

注: 对 `AnimatedImageDrawable` 进行后处理时，效果会出现在动画的所有帧中。

1.2.6 动画



Android 9 引入了 `AnimatedImageDrawable` 类，用于绘制和显示 GIF 和 WebP 动画图像。

`AnimatedImageDrawable` 的工作方式与 `AnimatedVectorDrawable` 的相似之处在于，都是渲染线程驱动 `AnimatedImageDrawable` 的动画。渲染线程还使用工作线程进行解码，因此，解码不会干扰渲染线程的其他操作。这种实现机制允许您的应用在显示动画图像时，无需管理其更新，也不会干扰应用界面线程上的其他事件。

可使用 `ImageDecoder` 的实例对 `AnimatedImageDrawable` 进行解码。以下代码段演示如何使用 `ImageDecoder` 来解码 `AnimatedImageDrawable`：

KOTLIN

```
throws IOException {
    private fun decodeImage() {
        val decodedAnimation = ImageDecoder.decodeDrawable(
            ImageDecoder.createSource(resources,
            drawable.my_drawable))

        // Prior to start(), the first frame is displayed.
        (decodedAnimation as? AnimatedImageDrawable)?.start()
    }
}
```

JAVA

```
private void decodeImage() throws IOException {
    Drawable decodedAnimation = ImageDecoder.decodeDrawable(
        ImageDecoder.createSource(getResources(),
        drawable.my_drawable));

    if (decodedAnimation instanceof AnimatedImageDrawable) {
        // Prior to start(), the first frame is displayed.
        ((AnimatedImageDrawable) decodedAnimation).start();
    }
}
```

`ImageDecoder` 有几个允许您进一步修改图像的函数。例如，可使用 `setPostProcessor()` 函数来修改图像的外观，如应用圆形遮罩或圆角。

1.2.7

HDR VP9 视频、HEIF 图像压缩和 Media API



Android 9 新增了对 High Dynamic Range (HDR) VP9 Profile 2 的支持，因此，现在您可以在支持 HDR 的设备上为用户提供来自 YouTube、Play Movies 和其他来源的采用 HDR 的影片。

Android 9 为平台增加了对 HEIF (heic) 图像编码的支持。MediaMuxer 和 MediaExtractor 类中可支持 HEIF 静态图像示例 HEIF 改进了压缩，可节省存储空间和网络数据流量。借助 Android 9 设备上的平台支持，从后端服务器发送和使用 HEIF 图像轻而易举。确保应用兼容这种便于共享和显示的数据格式后，尝试在应用中使用 HEIF 作为图像存储格式。您可以使用 ImageDecoder 或 BitmapFactory 进行 jpeg 到 heicto 的转换，以通过 jpeg 获取位图，并且可以使用 HeifWriter 写入来自 YUV 字节缓冲区、Surface 或 Bitmap 的 HEIF 静态图像。

还可通过 AudioTrack、AudioRecord 和 MediaDrm 类获取媒体指标。

Android 9 向 MediaDRM 类添加了函数以获取指标、高带宽数字内容保护 (HDCP) 级别、安全级别和会话数，并对安全性级别和安全停止进行更多控制。

在 Android 9 中，AAudio API 包含 AAudioStream 属性，用于 usage、content type 和 input preset。使用这些属性可以创建针对 VoIP 或摄像机应用调整的流。您还可以设置 SessionID 将 AAudio 流与可包含音效的子混音相关联。使用 AudioEffect API 来控制音效。

Android 9 包含一个用于 DynamicsProcessing 的 AudioEffect API。借助该类，可以构建基于通道的音效，由各种类型（包括均衡、多频带压缩和限幅器）的多个阶段组成。频带和活动阶段的数量可配置，而且大多数参数可实时控制。

1.2.8 JobScheduler 中的流量费用敏感度



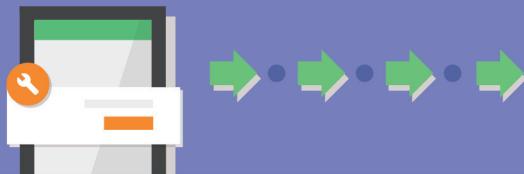
从 Android 9 开始，JobScheduler 可以使用运营商提供的网络状态信号来改善与网络有关的作业处理。

作业可以声明其预估的数据大小、信号预提取，并指定具体的网络要求。JobScheduler 然后根据网络状态管理工作。例如，当网络显示拥塞时，JobScheduler 可能会延迟较大的网络请求。如果使用的是不按流量计费的网络，则 JobScheduler 可运行预提取作业以提升用户体验（例如预提取标题）。

添加作业时，确保使用 `setEstimatedNetworkBytes()`、`setPrefetch()` 和 `setRequiredNetwork()`（如果适用），以帮助 JobScheduler 正确处理工作。在执行作业时，请确保使用 `JobParameters.getNetwork()` 返回的 `Network` 对象。否则，您将隐式使用设备的默认网络，其可能不符合您的要求，从而导致意外的流量消耗。

1.2.9

Neural Networks API 1.1



Android 8.1（API 级别 27）中引入了 Neural Networks API 以加快 Android 设备上机器学习的速度。Android 9 扩展和改进了该 API，增加了对九种新运算的支持：

- 元素级数学运算：
 - ANEURALNETWORKS_DIV
 - ANEURALNETWORKS_SUB
- 数组运算：
 - ANEURALNETWORKS_BATCH_TO_SPACE_ND
 - ANEURALNETWORKS_SPACE_TO_BATCH_ND
 - ANEURALNETWORKS_SQUEEZE
 - ANEURALNETWORKS_STRIDED_SLICE
 - ANEURALNETWORKS_TRANSPOSE
 - ANEURALNETWORKS_PAD
 - ANEURALNETWORKS_MEAN

已知问题：将 `ANEURALNETWORKS_TENSOR_QUANT8_ASYMM` 张量传递到 `ANEURALNETWORKS_PAD` 运算（在 Android 9 及更高版本中提供）时，NNAPI 的输出可能与较高级别机器学习框架（如 TensorFlow Lite）的输出不匹配。应只传递直到问题得到解决。`ANEURALNETWORKS_TENSOR_FLOAT32`

此外，API 还引入了一个新函数，即

`ANeuralNetworksModel_relaxComputationFloat32toFloat16()`，

允许您指定是否计算范围和精度低至 IEEE 754 16 位浮点格式的
`ANEURALNETWORKS_TENSOR_FLOAT32`。

1.2.10 自动填充框架



Android 9 引入了多项改进，自动填充服务可以利用这些改进进一步增强用户填写表单时的体验。

1.2.11 安全增强功能



Android 9 引入了若干安全**功能**，详见以下各节摘要说明：

Android Protected Confirmation

运行 Android 9 或更高版本的受支持设备赋予您使用 Android Protected Confirmation 的能力。使用该工作流时，您的应用会向用户显示提示，请他们批准一个简短的声明。应用可以通过这个声明再次确认，用户确实想完成一项敏感事务，例如付款。

如果用户接受该声明，Android 密钥库存收到并存储由密钥哈希消息身份验证代码（HMAC）保护的加密签名。Android 密钥库确认消息的有效性之后，您的应用可以使用在可信执行环境（TEE）下通过 trustedConfirmationRequired 生成的密钥。该签名具有很高的可信度，它表示用户已看过声明并同意其内容。

注意：Android 受保护的确认不会为用户提供安全信息通道。应用无法承担 Android 平台所提供之机密性保证之外的任何其他保证。尤其是，请勿使用该工作流显示您通常不会显示在用户设备上的敏感信息。

统一生物识别身份验证对话框

在 Android 9 中，系统代表您的应用提供生物识别身份验证对话框。该功能可创建标准化的对话框外观，风格和位置，让用户更加确认，他们在使用可信的生物识别凭据检查程序进行身份验证。

如果您的应用使用 FingerprintManager 向用户显示指纹身份验证对话框，请切换到改用 BiometricPrompt。BiometricPrompt 依赖系统来显示身份验证对话框。它还会改变其行为，以适应用户所选择的生物识别身份验证类型。

注：应用在使用中 BiometricPrompt 之前，先应该使用 `hasSystemFeature()` 函数以确保设备请立即获取 iTunes FEATURE_FINGERPRINT，FEATURE_IRIS 或 FEATURE_FACE。

如果设备不支持生物识别身份验证，可以回退为使用 `createConfirmDeviceCredentialIntent()` 函数验证用户的 PIN 码，图案或密码。

硬件安全性模块

运行 Android 9 或更高版本的受支持设备可拥有 StrongBox Keystarter，它是位于硬件安全性模块中的 Keystarter HAL 的一种实现。该模块包含以下组成部分：

- 自己的 CPU。
- 安全存储空间。
- 真实随机数生成器。
- 可抵御软件包篡改和未经授权线刷应用的附加机制。

检查存储在 StrongBox Keystarter 中的密钥时，系统会通过可信执行环境 (TEE) 证实密钥的完整性。

保护对密钥库进行的密钥导入

Android 9 通过利用 ASN.1 编码密钥格式将加密密钥安全导入密钥库库的功能，提高了密钥解密的安全性 Keystarter 随后会在密钥库中将密钥解密，因此密钥的内容永远不会以明文形式出现在设备的主机内存中。

具有密钥轮转的APK签名方案

Android 9 新增了对 APK Signature Scheme v3 的支持。该架构提供的选择可以在其签名块中为每个签名证书加入一条轮转证据记录。利用此功能，应用可以通过将 APK 文件过去的签名证书链接到现在签署应用时使用的证书，从而使用新签名证书来签署应用。

注：运行 Android 8.1（API级别27）或更低版本的设备不支持更改签名证书。如果应用的 minSdkVersion 为 27 或更低，除了新签名之外，可使用旧签名证书来签署应用。

只允许在未锁定设备上进行密钥解密的选项

Android 9 引入了 unlockedDeviceRequired 标志。此选项确定在允许使用指定密钥对任何正在传输或存储的数据进行解密之前，这些类型的密钥非常适合用于加密要存储在磁盘上的敏感数据，例如健康或企业数据。该标志为用户提供了更高的保证，即使手机丢失或被盗，在设备锁定的情况下，无法对数据进行解密。

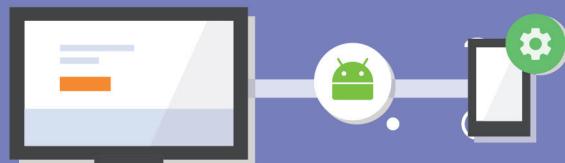
注：unlockedDeviceRequired 标志启用之后，仍然可以随时进行加密和签名验证。该标志可防止在设备解锁时“仅解密”数据。

在设备锁定时要确保密钥安全不被解密，可通过将true传递给 `setUnlockedDeviceRequired()` 函数启用该标志。完成该步骤之后，当用户的屏幕被锁定时，使用该密钥进行解密或签署数据的任何尝试都会失败。锁定设备在可以访问之前，需要 PIN 码，密码，指纹或者一些其他可信因素。

旧版加密支持

附带 Keymaster 4 的 Android 9 设备支持三重数据加密算法（简称三重 DES）。如果您的应用与需要三重 DES 的旧版系统进行互操作，请使用这种加密来加密敏感凭据。

1.2.12 Android 备份



Android 9 新增了与备份和还原有关的功能和开发者选项。这些更改的详细信息如以部分下所示。

客户端加密备份

Android 9 新增了对使用客户端密钥加密 Android 备份的支持。满足下列条件时会自动启用该支持**功能**：

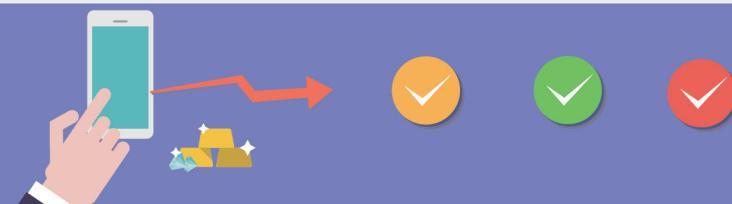
- 用户已使用 Android 9 或更高版本启用备份。
- 用户已为其设备设置屏幕锁定，需要 PIN 码，图案或密码才能解锁。

该隐私措施启用之后，从用户设备制作的备份还原数据时，会要求提供设备的 PIN 码，图案或密码。

定义备份所需的设备条件

如果你的应用数据包含敏感信息或偏好，Android 9 可让你定义设备条件（例如在客户端加密已启用或者正在进行本地设备到设备传输时），数据将依据该条件包括在用户的备份中。

1.2.13 无障碍功能



Android 9 引入了针对无障碍功能框架的增强**功能**，让您能够更轻松地为应用的用户提供更好的体验。

导航语义

Android 9 中的新增属性让您可以更轻松地定义无障碍服务（尤其是屏幕阅读器）如何从屏幕的某个部分导航到另一个部分。这些**属性**可帮助视力受损用户在应用界面的文本之间快速移动，并允许他们进行选择。

例如，在购物应用中，屏幕阅读器可以帮助用户从某个交易类别直接导航至下一个交易类别，在转到下一个类别之前，屏幕阅读器无需读取当前类别中的所有交易。

无障碍功能窗格标题

在 Android 8.1（API 级别 27）和更低版本中，无障碍服务有时无法确定屏幕的某个窗格是何时更新的，例如某个活动将一个片段替换为另一个片段的时候。窗格由按照逻辑关系分组，视觉上相关的界面元素组成，其中通常包含一个片段。

在 Android 9 中，可为这些窗格提供无障碍功能窗格标题，即可单独识别的标题。如果某个窗格具有无障碍功能窗格标题，当窗格改变时，无障碍

服务可接收更详细的信息。依靠这种功能，服务可以为用户提供有关界面变化的更精细信息。

要指定某个窗格的标题，请使用 `android:accessibilityPaneTitle` 属性。您也可以更新在运行时使用 `setAccessibilityPaneTitle()` 替换的某个界面窗格的标题。例如，您可以为某个 Fragment 对象的内容区域提供标题。

基于标题的导航

如果您的应用显示的文本内容包含逻辑标题，则对于表示这些标题的 View 实例，请将 `android:accessibilityHeading` 属性设置为 `true`。通过添加这些标题，无障碍服务可帮助用户直接从一个标题导航至下一个标题。任何无障碍服务都可以使用这种功能，以改善用户界面的导航体验。

群组导航和输出

传统上，屏幕阅读器一直使用 `android:focusable` 属性来确定何时应该将 `ViewGroup` 或一系列 `View` 对象作为一个整体进行读取。这样，用户就可以了解，这些视图在逻辑上彼此相关。

在 Android 8.1 和更低版本中，您需要将 `ViewGroup` 中的每个 `View` 对象标记为不可聚焦，并将 `ViewGroup` 本身标记为可聚焦。这种安排导致 `View` 的某些实例被标记为可聚焦，从而使得键盘导航变得更为繁琐。

从 Android 9 开始，如果将 `View` 对象标记为可聚焦会产生不良后果，则可以使用 `android:screenReaderFocusable` 属性 `android:focusable` 代替属性。屏幕阅读器聚焦在所有将 `android:screenReaderFocusable` 或 `android:focusable` 设置为 `true` 的元素上。

便捷操作

Android 9 新增了一些方便用户执行操作的支持**功能**：

访问提示：无障碍功能框架中的新增功能可让您在应用界面中访问提示。使用 `getTooltipText()` 读取提示文本，使用 `ACTION_SHOW_TOOLTIP` 和 `ACTION_HIDE_TOOLTIP` 来指示 View 的实例显示或隐藏提示。

新增全局操作：Android 9 在 `AccessibilityService` 类中引入了对两个额外设备操作的支持。您的服务可以帮助用户分别使用 `GLOBAL_ACTION_LOCK_SCREEN` 和 `GLOBAL_ACTION_TAKE_SCREENSHOT` 操作锁定其设备并进行屏幕截图。

窗口变更详情

Android 9 让您可以在应用同时重绘多个窗口时，更轻松地跟踪应用窗口的更新。当发生 `TYPE_WINDOWS_CHANGED` 事件时，可使用 `getWindowChanges()` API 来确定窗口发生的变更。在多窗口更新期间，每个窗口都会生成自己的一组事件。`getSource()` 函数返回与每个事件相关联的窗口的根视图。

如果应用已为其 View 对象定义无障碍功能窗格标题，您的服务将可以识别应用界面何时进行更新。`TYPE_WINDOW_STATE_CHANGED` 事件发生时，可使用 `getContentChangeTypes()` 所返回的类型来确定窗口发生的变更。例如，框架可以检测窗格何时有新标题或者窗格何时消失。

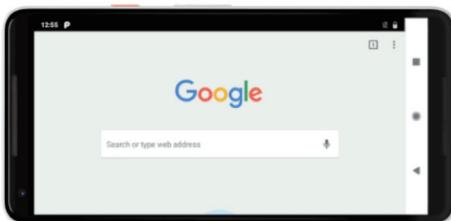
1.2.14 旋转



为避免无意的旋转，我们新增了一种模式，哪怕设备位置发生变化，也会固定在当前屏幕方向上。必要时用户可以通过按系统栏上的一个按钮手动触发旋转。

大多数情况下，对应用的兼容性影响微不足道。不过，如果您的应用有任何自定义旋转行为，或使用了任何非常规的屏幕方向设置，则可能会遇到以前用户旋转首选项始终设置为纵向时被忽视的问题。我们鼓励您审视一下您的应用所有关键活动中的旋转行为，并确保您的所有屏幕方向设置仍然提供最佳体验。

如需了解更多详情，请参阅 1.3 Android 9 行为变更指南。



一个新的旋转模式允许用户在必要时利用系统栏上的一个按钮手动触发旋转。

1.2.15 文本



Android 9 为平台提供了以下与文本相关的**功能**：

- 文本预先计算： PrecomputedText 类使您能提前计算和缓存所需信息，改善了文本渲染性能。它还使您的应用可以在主线程之外执行文本布局。
- 放大器： Magnifier 类是一种可提供放大器 API 的微件，可在所有应用中实现一致的放大器功能体验。
- Smart Linkify： Android 9 增强了 TextClassifier 类，该类可利用机器学习在选定文本中识别一些实体并建议采取相应的操作。例如，TextClassifier 可以让您的应用检测到用户选择了电话号码。然后，您的应用可以建议用户使用该号码拨打电话。TextClassifier 中的功能取代了 Linkify 类的功能。
- 文本布局：借助几种便捷函数和属性，可以更轻松地实现界面设计。

DEX 文件的 ART 提前转换

在运行 Android 9 或更高版本的设备上，Android 运行时 (ART) 提前编译器通过将应用软件包中的 DEX 文件转换为更紧凑的表示形式，进一步优化了压缩的 Dalvik Executable 格式 (DEX) 文件。此项变更可让您的应用启动更快并消耗更少的磁盘空间和内存。

这种改进特别有利于磁盘 I/O 速度较慢的低端设备。

1.2.16 设备端系统跟踪



Android 9 允许您通过设备记录系统跟踪记录，然后与您的开发团队分享这些记录的报告。该报告支持多种格式，包括 HTML。

通过收集这些跟踪记录，您可以获取与应用进程和线程相关的计时数据，并查看其他类型的具有全局意义的设备状态。

注：您无需设置您的代码来记录跟踪记录，但这样做可以帮助您查看应用代码的哪些部分可能会导致线程挂起或界面卡顿。

1.3

Android 9 行为变更指南



Android 9（API 级别 28）向 Android 系统引入了多项变更。当应用在 Android 9 平台上运行时，以下行为变更将影响所有应用，无论这些应用以哪个 API 级别为目标。所有开发者都应查看这些变更，并修改其应用以正确支持这些变更。

如需了解仅影响以 API 28 或更高级别为目标的应用的变更，请阅读[1.3.2 行为变更：以 API 级别 28+ 为目标的应用](#)。

1.3.1 针对 Android 9 的所有应用



用户行为隐私权变更

为了增强用户隐私，Android 9 引入了若干行为变更，如限制后台应用访问设备传感器、限制通过 Wi-Fi 扫描检索到的信息，以及与通话、手机状态和 Wi-Fi 扫描相关的新权限规则和权限组。

无论采用哪一种目标 SDK 版本，这些变更都会影响运行于 Android 9 上的所有应用。

- 后台对传感器的访问受限

Android 9 限制后台应用访问用户输入和传感器数据的能力。如果您的应用在运行 Android 9 设备的后台运行，系统将对您的应用采取以下**限制**：

- 您的应用不能访问麦克风或摄像头。
- 使用连续报告模式的传感器（例如加速度计和陀螺仪）不会接收事件。
- 使用变化或一次性报告模式的传感器不会接收事件。

如果您的应用需要在运行 Android 9 的设备上检测传感器事件，请使用前台服务。

- 限制访问通话记录

Android 9 引入 CALL_LOG 权限组并将 READ_CALL_LOG、WRITE_CALL_LOG 和 PROCESS_OUTGOING_CALLS 权限移入该组。在之前的 Android 版本中，这些权限位于 PHONE 权限组。

对于需要访问通话敏感信息（如读取通话记录和识别电话号码）的应用，该 CALL_LOG 权限组为用户提供了更好的控制和可见性。

如果您的应用需要访问通话记录或者需要处理去电，则您必须向 CALL_LOG 权限组明确请求这些权限。否则会发生 SecurityException。

注：因为这些权限已变更组并在运行时授权，用户可以拒绝您的应用访问通话记录。在这种情况下，您的应用应该能够妥善处理无法访问信息的状况。

如果您的应用已经遵循运行时权限最佳做法，则可以处理权限组的变更。

■ 限制访问电话号码

在未首先获得 READ_CALL_LOG 权限的情况下，除了应用的用例需要的其他权限之外，运行于 Android 9 上的应用无法读取电话号码或手机状态。

与来电和去电关联的电话号码可在手机状态广播（比如来电和去电的手机状态广播）中看到，并可通过 PhoneStateListener 类访问。但是，如果没有 READ_CALL_LOG 权限，则 PHONE_STATE_CHANGED 广播和 PhoneStateListener 提供的电话号码字段为空。

要从手机状态中读取电话号码，请根据您的用例更新应用以请求必要的权限：

- 要通过 PHONE_STATE Intent 操作读取电话号码，同时需要 READ_CALL_LOG 权限和 READ_PHONE_STATE 权限。

- 要从 `onCallStateChanged()` 中读取电话号码，只需要 `READ_CALL_LOG` 权限。不需要 `READ_PHONE_STATE` 权限。
- 限制访问 Wi-Fi 位置和连接信息

在 Android 9 中，应用进行 Wi-Fi 扫描的权限要求比之前的版本更严格。

类似的**限制**也适用于 `getConnectionInfo()` 函数，该函数返回描述当前 Wi-Fi 连接的 `WifiInfo` 对象。如果调用应用具有以下权限，则只能使用该对象的函数来检索 SSID 和 BSSID 值：

- `ACCESS_FINE_LOCATION` 或 `ACCESS_COARSE_LOCATION`
- `ACCESS_WIFI_STATE`

检索 SSID 或 BSSID 还需要在设备上启用位置服务（在 `Settings > Location` 下）。

- 从 Wi-Fi 服务函数中移除的信息

在 Android 9 中，下列事件和广播不接收用户位置或个人可识别数据方面的信息：

- `WifiManager` 中的 `getScanResults()` 和 `getConnectionInfo()` 函数。
- `WifiP2pManager` 中的 `discoverServices()` 和 `addServiceRequest()` 函数。
- `NETWORK_STATE_CHANGED_ACTION` 广播。

Wi-Fi 的 `NETWORK_STATE_CHANGED_ACTION` 系统广播不再包含 SSID（之前为 `EXTRA_SSID`）、BSSID（之前为 `EXTRA_BSSID`）或连接信息（之前为 `EXTRA_NETWORK_INFO`）。如果应用需要此信息，请改为调用 `getConnectionInfo()`。

- 电话信息现在依赖设备位置设置

如果用户在运行 Android 9 的设备上停用设备定位，则以下函数不提供结果：

- `getAllCellInfo()`
- `listen()`
- `getCellLocation()`
- `getNeighboringCellInfo()`

对使用非 SDK 接口的限制

为帮助确保应用稳定性和兼容性，此平台对某些非 SDK 函数和字段的使用进行了限制；无论您是直接访问这些函数和字段，还是通过反射或 JNI 访问，这些限制均适用。在 Android 9 中，您的应用可以继续访问这些受限的接口；该平台通过 toast 和日志条目提醒您注意这些接口。如果您的应用显示这样的 toast，则必须寻求受限接口之外的其他实现策略。如果您认为没有可行的替代策略，您可以提交错误以请求重新考虑此**限制**。

对非 SDK 接口的限制包含了更多重要信息。您应查阅该信息以确保您的应用继续正常工作。

安全行为变更

设备安全性变更

无论应用的目标平台版本如何，Android 9 添加的若干功能均可令应用的安全性得到改善。

■ 传输层安全协议 (TLS) 实现变更

系统的传输层安全协议 (TLS) 实现在 Android 9 中经历了若干次变更：

- 如果 `SSLocket` 的实例在创建时连接失败，系统会引发 `IOException` 而非 `NullPointerException`。
- `SSLEngine` 类可正常处理出现的任何 `close_notify` 提醒。

■ 更严格的 SECCOMP 过滤器

Android 9 对可供应用使用的系统调用做了进一步限制。此行为是 Android 8.0 (API 级别 26) 包含的 SECCOMP 过滤器的扩展。

注：此更改仅影响使用授权的系统调用的应用。

加密变更

Android 9 针对加密算法的实现和处理引入了几项变更。

- 参数和算法的 Conscrypt 实现

Android 9 在 Conscrypt 中实现了更多的算法参数。这些参数包括：AES、DESEDE、OAEP 和 EC。这些参数和许多算法的 Bouncy Castle 版本自 Android 9 起已被弃用。

注：EC 参数的 Conscrypt 实现仅支持已命名的曲线。

如果您的应用以 Android 8.1 (API 级别 27) 或更低版本为目标，则在请求这些已弃用算法之一的 Bouncy Castle 实现时，您将收到一条警告消息。然而，如果您以 Android 9 为目标平台，则这些请求会各自引发 NoSuchAlgorithmException。

- 其他变更

Android 9 引入了多项与加密有关的其他变更：

- 使用 PBE 密钥时，如果 Bouncy Castle 需要初始化矢量 (IV)，而您的应用未提供 IV，则会收到一条警告消息。
- ARC4 加密的 Conscrypt 实现允许您指定 ARC4/ECB/NoPadding 或 ARC4/NONE/NoPadding。

- Crypto Java 加密架构 (JCA) 提供程序现已被移除。因此，如果您的应用调用 SecureRandom.getInstance("SHA1PRNG", "Crypto")，将会发生 NoSuchProviderException。
- 如果您的应用从大于密钥结构的缓冲区中解析 RSA 密钥，将不会再发生异常。

不再支持 Android 安全加密文件

Android 9 完全取消了对 Android 安全加密文件 (ASEC) 的支持。

在 Android 2.2 (API 级别 8) 中，Android 引入了 ASEC 以支持 SD 卡加载应用功能。在 Android 6.0 (API 级别 23) 上，平台引入了一个可采用的存储设备技术，开发者可用它来代替 ASEC。

ICU 库更新

Android 9 使用 ICU 库版本 60。Android 8.0 (API 级别 26) 和 Android 8.1 (API 级别 27) 使用 ICU 58。

ICU 用于提供 android.icu package 下的公开 API，供 Android 平台内部用来提供国际化支持。例如，它用于实现 java.util、java.text 和 android.text.format 格式的 Android 类。

对 ICU 60 进行的更新包含许多细微但很有用的变更，这包括 Emoji 5.0 数据支持，改进了日期/时间格式，详见 ICU 59 和 ICU 60 版本说明中的介绍。

本次更新中的显著变更：

- 平台处理时区的方式已发生更改。
 - 平台能够更好地处理 GMT 和 UTC，不再将 UTC 与 GMT 混为一谈。
 - ICU 现在提供 GMT 和 UTC 的翻译版时区名称。此变更会影响“GMT”、“Etc/GMT”、“Etc/UTC”、“UTC”和“Zulu”之类时区的 android.icu 格式和解析行为。

- `java.text.SimpleDateFormat` 现在使用 ICU 提供 UTC /GMT 的显示名称，这意味着：
 1. 对于许多语言区域而言，设置 zzzz 的格式将会生成很长的本地化字符串。之前，对于 UTC 时区，它会生成“UTC”，而对于 GMT，则会生成“GMT+00:00”之类的字符串。
 2. 解析 zzzz 可识别“Universal Coordinated Time”和“Greenwich Mean Time”之类的字符串。
 3. 在所有语言里，如果应用接受“UTC”或“GMT+00:00”作为 zzzz 的输出，则可能会遇到兼容性问题。
- `java.text.DateFormatSymbols.getZoneStrings()` 的行为已变更：
 1. 与 `SimpleDateFormat` 类似，现在，UTC 和 GMT 也有长名称。对于 UTC 时区，DST 类型的时区名称（例如“UTC”、“Etc/UTC”和“Zulu”）变为 `GMT+00:00`（而不是硬编码字符串 `UTC`），这是在没有其他名称可用时的标准回退。
 2. 某些时区 ID 被正确地识别为其他地区的同义词，因此，Android 能够查找过时时区 ID（例如 `Eire`）对应的字符串，而之前无法解决此问题。
- 亚洲/河内不再是可识别的时区。因此，`java.util.TimeZones.getAvailableIds()` 不再返回此值，`java.util.TimeZone.getTimeZone()` 也不再识别它。此行为与现有的 `android.icu` 行为相符。
- `android.icu.text.NumberFormat.getInstance(ULocale, PLURALCURRENCYSTYLE).parse(String)` 函数甚至在解析合法相应币种文本时也会引发 `ParseException`。通过对 `PLURALCURRENCYSTYLE` 类型的相应币种文本使用自 Android 7.0（API 级别 24）以来所提供的 `NumberFormat.parseCurrency`，可避免此问题。

Android Test 变更

Android 9 引入了多项针对 Android Test 框架库和类结构的更改。这些变更可帮助开发者使用支持框架的公共 API，此外，在使用第三方库或自定义逻辑构建和运行测试时，这些变更还可提供更大的灵活性。

- 从框架移除的内容库

Android 9 将基于 JUnit 的类重新整理成三个内容库： android.test.base、 android.test.runner 和 android.test.mock。此变更允许您针对与您的项目依赖项搭配效果最好的 JUnit 版本运行测试。此版本的 JUnit 可能不同于 android.jar 提供的版本。

- 测试套件版本号变更

移除了 TestSuiteBuilder 类中的 `addRequirements()` 函数， TestSuiteBuilder 类本身也已弃用。`addRequirements()` 函数要求开发者提供类型为隐藏 API 的参数，结果令 API 失效。

Java UTF 解码器

UTF-8 是 Android 中的默认字符集。UTF-8 字节序列可由 `String(byte[] bytes)` 之类的 `String` 构造函数解码。

Android 9 中的 UTF-8 解码器遵循比以前版本中更严格的 Unicode 标准：这些变更包括：

- 非最短形式的 UTF-8（例如 <C0, AF>）被视为格式不正确。
- 替代形式的 UTF-8（例如 U+D800..U+DFFF）被视为格式不正确。
- 最大的子部分被单个 U+FFFD 取代。例如，在字节序列“41 C0 AF 41 F4 80 80 41”中，最大子部分为“C0”、“AF”和“F4 80 80”。其中“F4 80 80”可以是“F4 80 80 80”的初始子序列，但“C0”不能是任何形式正确的代码单位序列的初始子序列。因此，输出应为“A\uufffd\uufffdA\uufffdA”。

- 要在 Android 9 或更高版本中解码修改后的 UTF-8/CESU-8 序列，请使用 `DataInputStream.readUTF()` 函数或 `NewStringUTF()` JNI 函数。

使用证书的主机名验证

RFC 2818 中介绍了两种对照证书匹配域名的方法—使用 `subjectAltName` (SAN) 扩展程序中的可用名称，或者在没有 SAN 扩展程序的情况下，回退到 `commonName` (CN)。

然而，在 RFC 2818 中，回退到 CN 已被弃用。因此，Android 不再回退到使用 CN。要验证主机名，服务器必须出示具有匹配 SAN 的证书。不包含与主机名匹配的 SAN 的证书不再被信任。

网络地址查询可能会导致网络违规

要求名称解析的网络地址查询可能会涉及网络 I/O，因此会被视为阻塞性操作。对于主线程的阻塞性操作可能会导致停顿或卡顿。

`StrictMode` 类是一个有助于开发者检测代码问题的开发工具。

在 Android 9 及更高版本中，`StrictMode` 可以检测需要名称解析的网络地址查询所导致的网络违规。

您在交付应用时不应启用 `StrictMode`。否则，您的应用可能会遭遇异常，例如，在使用 `detectNetwork()` 或 `detectAll()` 函数获取用于检测网络违规的政策时，会出现 `NetworkOnMainThreadException`。

解析数字 IP 地址不被视为阻塞性操作。数字 IP 地址解析的工作方式与 Android 9 以前的版本中所采用的方式相同。

套接字标记

在低于 Android 9 的平台版本上，如果使用 `setThreadStatsTag()` 函数标记某个套接字，则当使用带 `ParcelFileDescriptor` 容器的 binder 进程间通信将其发送给其他进程时，套接字会被取消标记。

在 Android 9 及更高版本中，利用 binder 进程间通信将套接字发送至其他进程时，其标记将得到保留。此变更可能影响网络流量统计，例如，使用 `queryDetailsForUidTag()` 函数时。

如果您要保留以前版本的行为，即取消已发送至其他进程的套接字的标记，您可以在发送此套接字之前调用 `untagSocket()`。

更详尽的 VPN 网络功能报告

在 Android 8.1（API 级别 28）及更低版本中，`NetworkCapabilities` 类仅报告 VPN 的有限信息，例如 `TRANSPORT_VPN`，但会省略 `NET_CAPABILITY_NOT_VPN`。信息有限导致难以确定使用 VPN 是否会导致对应用的用户收费。例如，检查 `NET_CAPABILITY_NOT_METERED` 并不能确定底层网络是否按流量计费。

从 Android 9 及更高版本开始，当 VPN 调用 `setUnderlyingNetworks()` 函数时，Android 系统将会合并任何底层网络的传输和能力并返回 VPN 网络的有效网络能力作为结果。

在 Android 9 及更高版本中，已经检查 `NET_CAPABILITY_NOT_METERED` 的应用将收到关于 VPN 网络能力和底层网络的信息。

应用不再能访问 `/proc/net/xt_qtaguid` 文件夹中的文件

从 Android 9 开始，不再允许应用直接读取 `/proc/net/xt_qtaguid` 文件夹中的文件。这样做是为了确保与某些根本不提供这些文件的设备保持一致。

依赖这些文件的公开 API TrafficStats 和 NetworkStatsManager 继续按照预期方式运行。然而，不受支持的 cutils 函数（例如 `qtaguid_tagSocket()`）在不同设备上可能不会按照预期方式运行 — 甚至根本不运行。

现在强制执行 FLAG_ACTIVITY_NEW_TASK 要求

在 Android 9 中，您不能从非 Activity 环境中启动 Activity，除非您传递 Intent 标志 `FLAG_ACTIVITY_NEW_TASK`。如果您尝试在不传递此标志的情况下启动 Activity，则该 Activity 不会启动，系统会在日志中输出一则消息。

注：在 Android 7（API 级别 24）之前，标志要求一直是期望的行为并被强制执行。Android 7 中的一个错误会临时阻止实施标志要求。

屏幕旋转变更

从 Android 9 开始，对纵向旋转模式做出了重大变更。在 Android 8（API 级别 26）中，用户可以使用 Quicksettings 图块或 Display 设置在自动屏幕旋转和纵向旋转模式之间切换。纵向模式已重命名为旋转锁定，它会在自动屏幕旋转关闭时启用。自动屏幕旋转模式没有任何变更。

当设备处于旋转锁定模式时，用户可将其屏幕锁定到顶层可见 Activity 所支持的任何旋转。Activity 不应假定它将始终以纵向呈现。如果顶层 Activity 可在自动屏幕旋转模式下以多种旋转呈现，则应在旋转锁定模式下提供相同的选项，根据 Activity 的 `screenOrientation` 设置，允许存在一些例外情况（见下表）。

请求特定屏幕方向（例如，`screenOrientation=landscape`）的 Activity 会忽略用户锁定首选项，并且行为与 Android 8 中的行为相同。

可在 Android Manifest 中，或以编程方式通过 `setRequestedOrientation()` 在 Activity 级别设置屏幕方向首选项。

旋转锁定模式通过设置 WindowManager 在处理 Activity 旋转时使用的用户旋转首选项来发挥作用。用户旋转首选项可能在下列情况下发生变更。请注意，恢复设备的自然旋转存在偏差，对于外形与手机类似的设备通常设置为纵向：

- 当用户接受旋转建议时，旋转首选项变为建议方向。
- 当用户切换到强制纵向应用（包括锁定屏幕或启动器）时，旋转首选项变为纵向。

下表总结了常见屏幕方向的旋转行为：

屏幕方向	行为
未指定、user	在自动屏幕旋转锁定下，Activity 可以纵向或横向（以及颠倒纵向或横向）呈现。预期同时支持纵向和横向布局。
userLandscape	在自动屏幕旋转和旋转锁定下，Activity 可以横向或颠倒横向呈现。预期只支持横向布局。
userPortrait	在自动屏幕旋转和旋转锁定下，Activity 可以纵向或颠倒纵向呈现。预期只支持纵向布局。
fullUser	在自动屏幕旋转和旋转锁定下，Activity 可以纵向或横向（以及颠倒纵向或横向）呈现。预期同时支持纵向和横向布局。
sensor、fullSensor、 sensorPortrait、 sensorLandscape	旋转锁定用户将可选择锁定到颠倒纵向，通常为 180°。忽略旋转锁定模式首选项，视为自动屏幕旋转已启用。请仅在例外情况下并经过仔细的用户体验考量后再使用此项。

Apache HTTP 客户端弃用影响采用非标准 ClassLoader 的应用

在 Android 6 中，我们取消了对 Apache HTTP 客户端的支持。

此变更对大多数不以 Android 9 或更高版本为目标的应用没有任何影响。不过，此变更会影响使用非标准 ClassLoader 结构的某些应用，即使这些应用不以 Android 9 或更高版本为目标平台。

如果应用使用显式委托到系统 ClassLoader 的非标准 ClassLoader，则应用会受到影响。在 `org.apache.http.*` 中查找类时，这些应用需要委托给应用 ClassLoader。如果它们委托给系统 ClassLoader，则应用在 Android 9 或更高版本上将失败并显示 `NoClassDefFoundError`，因为系统 ClassLoader 不再识别这些类。为防止将来出现类似问题，一般情况下，应用应通过应用 ClassLoader 加载类，而不是直接访问系统 ClassLoader。

多个摄像头

在 Android 9 设备上运行的应用可以通过调用 `getCameraIdList()` 发现每个可用的摄像头。应用不应假定设备只有一个后置摄像头或只有一个前置摄像头。

例如，如果您的应用有一个用来切换前置和后置摄像头的按钮，则设备可能有多个前置或后置摄像头可供选择。您应浏览一下摄像头列表，检查每个摄像头的特征，然后决定向用户显示哪些摄像头。

1.3.2

针对 API 28+ 级别的所有应用



Android 9（API 级别 28）向 Android 系统引入了多项变更。以下行为变更仅影响以 API 28 或更高级别为目标的应用。将 targetSdkVersion 设为 API 28 或更高级别的应用必须进行修改，以便正确支持这些行为（如果适用）。

如需了解影响在 Android 9 上运行的所有应用的变更，则无论这些应用以哪个 API 级别为目标，都请参阅 1.3.1 针对 Android 9 的所有应用行为变更。

前台服务

针对 Android 9 或更高版本并使用前台服务的应用必须请求 [FOREGROUND_SERVICE](#) 权限。这是普通权限，因此，系统会自动为请求权限的应用授予此权限。

如果针对 Android 9 或更高版本的应用尝试创建一个前台服务且未请求 [FOREGROUND_SERVICE](#)，则系统会引发 `SecurityException`。

用户应用隐私权变更

如果您的应用以 Android 9 为目标平台，您应牢记以下行为变更。对设备序列信息和 DNS 信息进行的这些更新可增强用户隐私保护。

- 构建序列号弃用

在 Android 9 中，Build.SERIAL 始终设置为 "UNKNOWN" 以保护用户的隐私。如果您的应用需要访问设备的硬件序列号，您应改为请求 READ_PHONE_STATE 权限，然后调用 `getSerial()`。

- DNS 隐私

以 Android 9 为目标平台的应用应采用私有 DNS API。具体而言，当系统解析程序正在执行 DNS-over-TLS 时，应用应确保任何内置 DNS 客户端均使用加密的 DNS 查找与系统相同的主机名，或停用它而改用系统解析程序。

框架安全性变更

Android 9 包含可提升您的应用安全性的多个行为变更，但这些变更仅在您的应用以（API 级别 28）或更高级别为目标平台时才会生效：

- 默认情况下启用网络传输层安全协议 (TLS)

如果您的应用以 Android 9 或更高版本为目标平台，则默认情况下 `isCleartextTrafficPermitted()` 函数返回 false。如果您的应用需要为特定域名启用明文，您必须在应用的网络安全性配置中针对这些域名将 `cleartextTrafficPermitted` 显式设置为 true。

- 按进程分设基于网络的数据目录

为改善 Android 9 中的应用稳定性和数据完整性，应用无法再让多个进程共用同一 WebView 数据目录。此类数据目录一般存储 Cookie、HTTP 缓存以及其他与网络浏览有关的持久性和临时性存储。

在大多数情况下，您的应用只应在一个进程中使用 android.webkit 软件包中的类，例如 WebView 和 CookieManager。例如，您应该将所有使用 WebView 的 Activity 对象移入同一进程。您可以通过在应用的其他进程中调用 `disableWebView()`，更严格地执行“仅限一个进程”规则。该调用可防止 WebView 在这些其他进程中被错误地初始化，即使是从依赖内容库进行的调用也能防止。

如果您的应用必须在多个进程中使用 WebView 的实例，则必须先利用 `WebView.setDataDirectorySuffix()` 函数为每个进程指定唯一的数据目录后缀，然后再在该进程中使用 WebView 的给定实例。该函数会将每个进程的网络数据放入其在应用数据目录内自己的目录中。

注：即使您使用 `setDataDirectorySuffix()`，系统也不会跨应用的进程界限共享 Cookie 以及其他网络数据。如果应用中的多个进程需要访问同一网络数据，您需要自行在这些进程之间复制数据。例如，您可以调用 `getCookie()` 和 `setCookie()`，在不同进程之间手动传输 Cookie 数据。

- 以应用为单位的 SELinux 域名

以 Android 9 或更高版本为目标平台的应用无法利用可全球访问的 Unix 权限与其他应用共享数据。此变更可改善 Android 应用沙盒的完整性，具体地讲，就是要求应用的私有数据只能由该应用访问。

要与其他应用共享文件，请使用 content provider。

连接变更

- 连接数据计数和多路径

在以 Android 9 或更高版本为目标平台的应用中，系统计算并非当前默认网络的网络流量，例如，当设备连接 WLAN 时的蜂窝流量，并在 `NetworkStatsManager` 类中提供函数以查询该流量。

具体而言，`getMultipathPreference()` 现在将返回一个基于上述网络流量的值。从 Android 9 开始，此函数针对蜂窝数据返回 `true`，但当超过一天内累积的特定流量时，它将开始返回 `false`。在 Android 9 上运行的应用必须调用此函数并采用此提示。

ConnectivityManager.NetworkCallback 类现在将有关 VPN 的信息发送到应用。此变更让应用侦听连接事件变得更容易，而无需混用同步和异步调用，也无需使用有限的 API。此外，它还意味着将设备同时连接至多个 WLAN 网络或多个蜂窝网络时，信息传输可按预期工作。

- Apache HTTP 客户端弃用

在 Android 6 中，我们取消了对 Apache HTTP 客户端的支持。从 Android 9 开始，默认情况下该内容库已从 bootclasspath 中移除且不可用于应用。

要继续使用 Apache HTTP 客户端，以 Android 9 及更高版本为目标的应用可以向其 `AndroidManifest.xml` 添加以下内容：

```
<uses-library android:name=" org.apache.http.legacy" android:required=" false" />
```

注：拥有最低 SDK 版本 23 或更低版本的应用需要 `android:required="false"` 属性，因为在 API 级别低于 24 的设备上，`org.apache.http.legacy` 库不可用。（在这些设备上，Apache HTTP 类在 `bootclasspath` 中提供。）

作为使用运行时 Apache 库的替代，应用可以在其 APK 中绑定自己的 `org.apache.http` 库版本。如果进行此操作，您必须将该库重新打包（使用一个类似 Jar Jar 的实用程序）以避免运行时中提供的类存在类兼容性问题。

界面变更

- 视图焦点

0 面积的视图（即宽度或高度为 0）再也不能被聚焦。

此外，Activity 不再隐式分配触摸模式下的初始焦点。而是由您显式请求初始焦点（如若需要的话）。

- CSS RGBA 十六进制值处理

以 Android 9 或更高版本为目标的应用必须支持草案版 CSS 颜色模块级别 4 的行为，用于处理 4 和 8 个十六进制数字 CSS 颜色。

Chrome 自版本 52 以来便一直支持 CSS 颜色模块级别 4，但 WebView 目前停用此功能，因为现有 Android 应用被发现包含 Android ordering (ARGB) 中的 32 位十六进制颜色，这会导致渲染**错误**。

例如，对于以（API 级别 27）或更低版本为目标平台的应用，颜色 #80ff8080 目前在 WebView 中被渲染为不透明浅红色 (#ff8080)。先导部分（Android 会将其解读为 Alpha 部分）目前被忽略。如果某个应用以（API 级别 28）或更高版本为目标，则 #80ff8080 将被解读为 50% 透明浅绿 (#80ff80)。

- 文档滚动标签

Android 9 可正确处理文档的根标签是滚动标签的案例。在之前的版本中，滚动位置在 body 标签上设置，根标签的滚动值为零。Android 9 支持符合标准的行为，在这种行为中，滚动标签是根标签。

此外，直接访问 document.body.scrollTop、document.body.scrollLeft、document.documentElement.scrollTop 或 document.documentElement.scrollLeft 会因目标 SDK 的不同而具有不同的行为。要访问视口滚动值，请使用 document.scrollingElement（若有）。

- 自己暂停应用的通知

在 Android 9 之前，暂停的应用发出的通知会被取消。从 Android 9 开始，暂停的应用发出的通知将被隐藏，直至应用继续运行。

第 2 章

迁移到 Android 9



目录：

- 2.1** 运行 Android 9 的设备
- 2.2** Android 9 兼容性与测试
- 2.3** 更新目标版本并使用 Android 9
- 2.4** 电源管理功能更新
- 2.5** 非 SDK 接口的限制与测试



迁移到 Android 9

Android 9（API 级别 28）引入了一些您的应用可加以利用的新**功能**和 API，以及新行为变更。本章节概述了将应用迁移到 Android 9 的两个关键阶段的步骤：

- 确保与 Android 9 基本兼容

验证您的现有应用能够在新版本平台上全**功能**运行。在此阶段，您不需要使用新的 API，也不需要更改应用的 targetSdkVersion，但可能需要进行一些细微的更改。

- 以新平台为目标，使用 Android 9 SDK 编译，然后使用 Android 9 功能构建

当您准备好利用平台的新**功能**时，将 targetSdkVersion 更新至“28”，验证应用是否仍可按预期方式运行，然后开始使用新的 API。

2.1 运行 Android 9 的设备



如果您有一台兼容设备，则从制造商那里获取设备的 Android 9 系统镜像；
[点击此处](#)获取 Pixel 设备的出厂镜像。刷写系统镜像的一般说明位于此处。

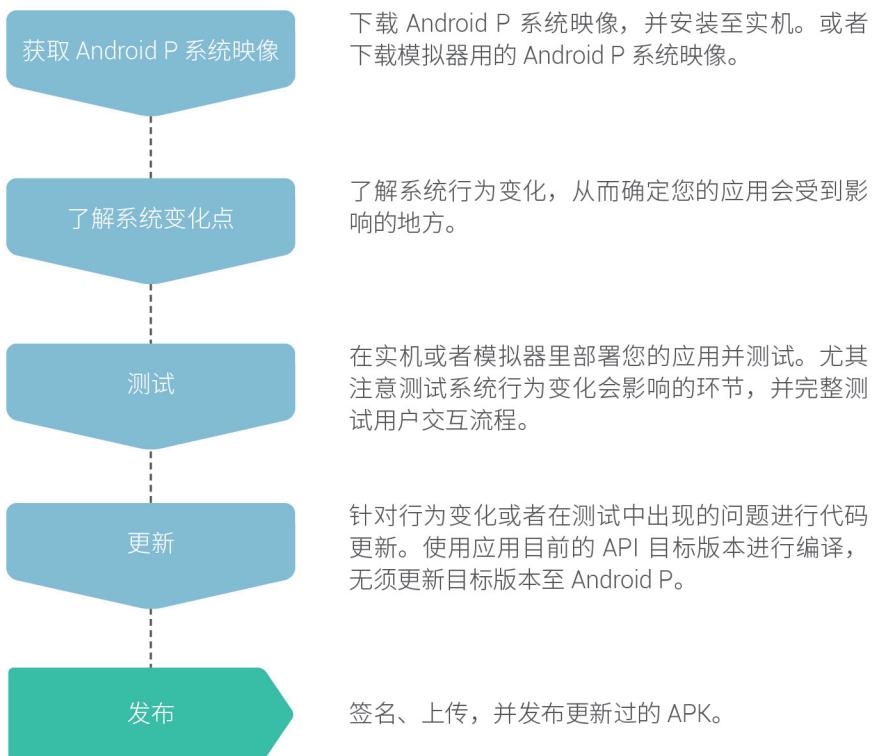
您还可以为 Android Emulator 下载 Android 9 系统镜像。它列于 SDK 管理器的 Android 9 下，显示为 Google APIs Intel x86 Atom System Image。

2.2 Android 9 兼容性与测试



确保与 Android 9 兼容

这一步的目标是确保您的现有应用在 Android 9 上可照常运行。由于一些平台变更可能影响应用的行为方式，因此可能需要进行一些调整，但您不需要使用新的 API 或更改 targetSdkVersion。



执行兼容性测试

与 Android 9 的兼容性测试多半与您准备发布应用时执行的测试属于同一类型。这时有必要回顾一下核心应用质量指南和测试最佳实践。

不过，测试还有另一个层面：Android 9 向 Android 平台引入了一些变化，即便不对 targetSdkVersion 做任何变动，仍可能影响应用的行为或令其根本无法运行。因此，您必须回顾表 1 中的关键变化，并对任何为适应这些变化而实现的修复进行测试。

表 1. 对在 Android 9 设备上运行的所有应用都有影响的关键变化。

变化	摘要
对非 SDK 接口的限制	现已禁止访问特定的非 SDK 接口，无论是直接访问，还是通过 JNI 或反射进行间接访问。尝试访问受限制的接口时，会生成 NoSuchFieldException 和 NoSuchMethodException 之类的错误。详情请参阅 对非 SDK 接口的限制 。
移除加密提供程序	从 Android 9 开始，Crypto JCA 提供程序已被移除。调用 SecureRandom.getInstance ("SHA1PRNG", "Crypto") 将会引发 NoSuchProviderException。
更严格的 UTF-8 解码器	在 Android 9 中，针对 Java 语言的 UTF-8 解码器比以往更严格，并且遵循 Unicode 标准。
禁止空闲应用访问相机、麦克风和传感器	在应用处于空闲状态时，不能再访问相机、麦克风或 SensorManager 传感器。

第三方兼容性测试资源

您也可以考虑借助第三方测试平台的兼容性测试资源进行测试。

目前，国内已经有四家测试平台可以提供 Android 9 的全面兼容性应用测试（具体免费时段由该测试平台决定）。这四家提供兼容性测试的平台分别是：



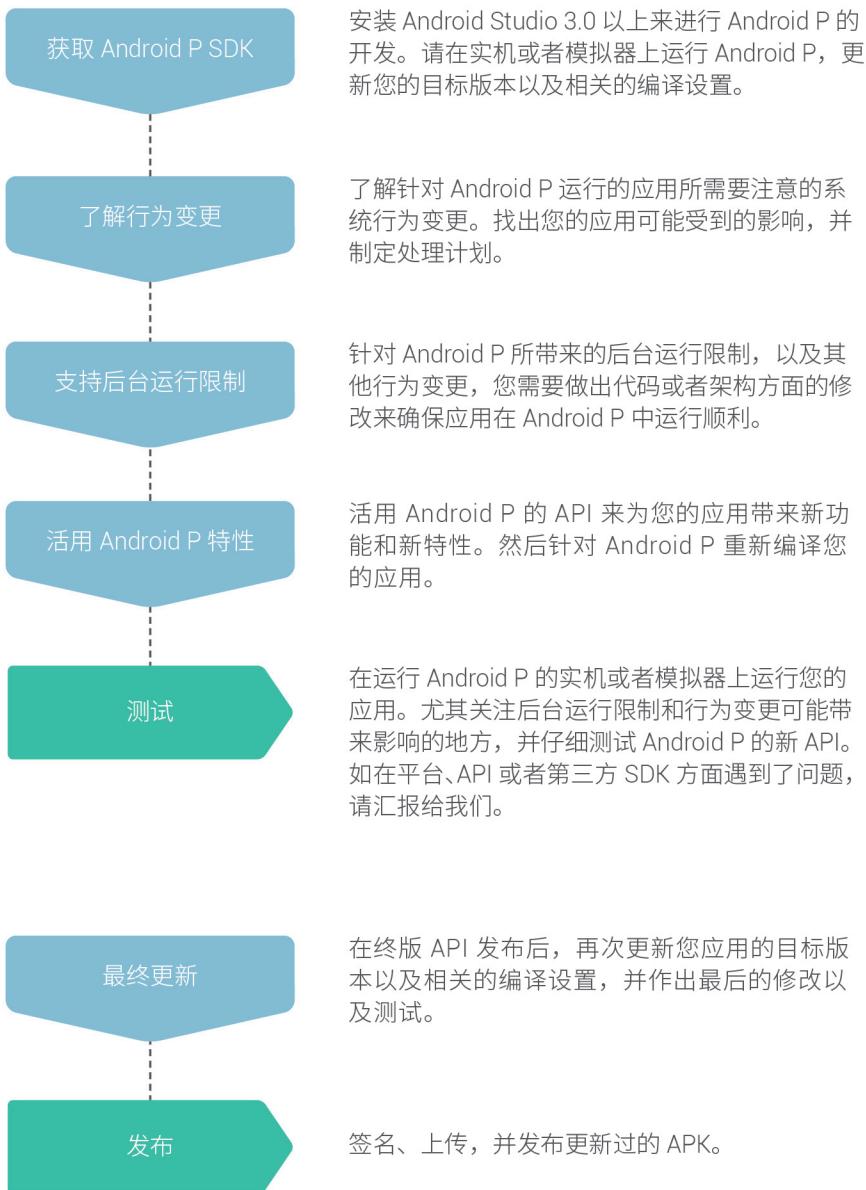
2.3 更新目标版本并使用 Android P



本章节解释如何通过将您的 `targetSdkVersion` 更新为 28 并增加 Android 9 中提供的新**功能**来实现对 Android 9 的全面支持。

除提供新 API 之外，在您将 `targetSdkVersion` 更新为 28 时，您会注意到 Android 9 还引入了一些行为变更。由于某些行为变更可能要求更改代码以避免冲突，因此，您应先查阅所有以 Android 9 为目标的应用的行为变更，了解在您更改 `targetSdkVersion` 后您的应用会受到哪些影响。

注：上述旨在确保平台兼容性的步骤是将应用以Android 9为目标的先决条件，因此请您务必先完成这些步骤。



获取 Android 9 SDK

您可以使用 Android Studio 3.1 或更高版本获取 SDK 软件包，以便利用 Android 9 构建应用。如果您暂时不需要 Android 9 中的新**功能**，只想针对该平台版本进行编译，您可以使用 Android Studio 3.1。Android Studio 3.2 提供了对 Android 9 功能的全面支持。

要设置任一版本的 Android Studio，请按照设置 Preview SDK 中介绍的步骤操作。

测试 Android 9 应用

完成以上准备工作后，您就可以构建应用，然后对其做进一步测试，确保以 Android 9（API 级别 28）为目标平台时它能正常工作。这时有必要再次回顾一下核心应用质量指南和测试最佳实践。

如果您构建应用时将 targetSdkVersion 设置为 P，应该注意特定的平台变化。即便您不实现 Android 9 中的新功能，其中的一些变化仍可能严重影响应用的行为或令其根本无法运行。

表 2 列出了这些变化以及可获得更多信息的链接。

表 2. targetSdkVersion 设置为 28 时影响应用的关键变化。

变化	摘要
前台服务权限	现在，想要使用前台服务的应用必须首先请求 FOREGROUND_SERVICE 权限。这是普通权限，因此，系统会自动为请求权限的应用授予此权限。在未获得此权限的情况下启动前台服务将会引发 SecurityException。
弃用 Bouncy Castle 加密	Android 9 弃用了几个来自 Bouncy Castle 提供程序中的加密技术，代之以由 Conscrypt 提供程序提供的加密技术。调用请求 Bouncy Castle 提供程序的 <code>getInstance()</code> 时，会生成 <code>NoSuchAlgorithmException</code> 错误。要解决这些错误，请不要在 <code>getInstance()</code> 中指定提供程序（也就是请求默认实现）。
移除对 <code>Build.serial</code> 的直接访问	现在，需要 <code>Build.serial</code> 标识符的应用必须请求 <code>READ_PHONE_STATE</code> 权限，然后使用 Android 9 中新增的新 <code>Build.getSerial()</code> 函数。
不允许共享	在 Android 9 中，针对 Java 语言的 UTF-8 解码器比以往更严格，并且遵循 Unicode 标准。
WebView 数据目录	现在，不允许应用在不同进程之间共享一个 WebView 数据目录。如果您的应用有多个进程使用 WebView、CookieManager 或 android.webkit 软件包中的任何其他 API，则在第二个进程调用 WebView 函数时，您的应用将会崩溃。
SELinux 禁止访问应用的数据目录	系统强制每个应用的 SELinux 沙盒对每个应用的私有数据目录强制执行逐个应用的 SELinux 限制。现在，不允许直接通过路径访问其他应用的数据目录。应用可以继续使用进程间通信 (IPC) 机制（包括通过传递 FD）共享数据。

2.4 电源管理功能更新



Android 9（API 级别 28）引入了一些新功能来改进设备电源管理。这些变化，连同先前版本中已经存在的功能，有助于确保将系统资源提供给最需要它们的应用。

电源管理功能可以分为两个类别：

应用待机群组：系统将根据用户的使用模式**限制**应用对 CPU 或电池等设备资源的访问。这是 Android 9 中新增的一项**功能**。

省电模式改进：开启省电模式后，系统会对所有应用施加**限制**。这是一项已有的**功能**，但在 Android 9 中得到了改进。

注：这些变化适用于所有应用，无论它们是否以 Android 9 为目标。

应用待机群组

Android 9 引入了一项新的电池管理**功能**，即应用待机群组。应用待机群组可以基于应用最近使用时间和使用频率，帮助系统排定应用请求资源的优先级。根据使用模式，每个应用都会归类到五个优先级群组之一中。系统将根据应用所属的群组**限制**每个应用可以访问的设备资源。

五个群组按照以下特性将应用分组：

- 活跃

如果用户当前正在使用应用，应用将被归到“活跃”群组中，例如：

- 应用已启动一个 Activity
- 应用正在运行前台服务
- 应用的同步适配器与某个前台应用使用的 content provider 关联
- 用户在应用中点击了某个通知

如果应用处于“活跃”群组，系统不会对应用的作业、报警或 FCM 消息施加任何**限制**。

- 工作集

如果应用经常运行，但当前未处于活跃状态，它将被归到“工作集”群组中。例如，用户在大部分时间都启动的某个社交媒体应用可能就属于“工作集”群组。如果应用被间接使用，它们也会被升级到“工作集”群组中。

如果应用处于“工作集”群组，系统会对它运行作业和触发报警的能力施加**轻度限制**。

- 常用

如果应用会定期使用，但不是每天都必须使用，它将被归到“常用”群组中。例如，用户在健身房运行的某个锻炼跟踪应用可能就属于“常用”群组。

如果应用处于“常用”群组，系统将对它运行作业和触发报警的能力施加**较强的限制**，也会对高优先级 FCM 消息的数量设定**限制**。

- 极少使用

如果应用不经常使用，那么它属于“极少使用”群组。例如，用户仅在入住酒店期间运行的酒店应用就可能属于“极少使用”群组。

如果应用处于“极少使用”群组，系统将对它运行作业、触发警报和接收高优先级 FCM 消息的能力施加严格限制。系统还会限制应用连接到网络的能力。

- 从未使用

安装但是从未运行过的应用会被归到“从未使用”群组中。系统会对这些应用施加极强的限制。

系统会动态地将每个应用归类到某个优先级群组，并根据需要重新归类。系统可能会依靠某个使用机器学习的预加载应用确定每个应用的使用可能性，并将应用归类到合适的群组。如果设备上不存在系统应用，系统默认将基于应用的最近使用时间对它们进行排序。更为活跃的应用将被归类到为应用提供更高优先级的群组，从而让应用可以使用更多系统资源。具体而言，群组决定应用运行作业的频率，应用可以触发报警的频率，以及应用可以接收高优先级 Firebase 云信息传递 (FCM) 消息的频率。这些限制仅在设备使用电池电量时适用，如果设备正在充电，系统不会对应用施加这些限制。

每个制造商都可以设定自己的标准来归类非活跃应用。您不应当尝试影响应用所属的群组。相反，您应当将精力放在确保应用在所属的群组内良好运行上。您的应用可以通过调用新函数

`UsageStatsManager.getAppStandbyBucket()` 查找当前属于哪个群组。

注：位于低电耗模式白名单中的应用不适用基于应用待机群组的限制。

最佳实践

如果您的应用已遵循低电耗模式和应用待机模式的最佳实践，那么处理新的电源管理功能不应是难事。不过，之前正常运行的一些应用行为现在可能会引起问题。

- 不要让系统处于一种不断变换应用所属群组的状态。系统的分组方式可以变化，每个设备制造商都可以选择使用自己的算法编写分组应用。相反，请确保您的应用无论处于哪一个分组时行为都很恰当。
- 如果应用没有启动器 Activity，那么它可能永远不会升级到“活跃”分组中。您需要重新设计应用，使之具有此类 Activity。
- 如果应用的通知不可操作，用户与通知交互将无法触发应用向“活跃”群组的升级。在这种情况下，您需要重新设计某些适当的通知，让它们允许用户响应。
- 类似地，如果应用在收到高优先级 FCM 消息时不显示通知，那么它不会向用户提供与应用交互的机会，也不会借此升级到“活跃”群组中。事实上，高优先级 FCM 消息的唯一预期目的是向用户推送通知，因此，这种情况永远都不应发生。如果您在某条 FCM 消息不触发用户交互时将其错误地标记为高优先级，它可能引起其他不良后果；例如，它可能导致您的应用耗尽配额，导致真正紧急的 FCM 消息被视为一般优先级。

注：如果用户重复忽略了某个通知，系统将向用户提供未来阻止该通知的选项。请不要为了让您的应用处于“活跃”群组而向用户滥发通知！

- 如果应用分为多个软件包，那么这些软件包可能处于不同的群组中，进而拥有不同的访问权限级别。请务必对软件包被归类到各个群组的此类应用进行测试，以便确保应用行为正常。

省电模式改进

Android 9 对省电模式进行了多处改进。设备制造商可以决定施加的确切限制。例如，在 AOSP 构建中，系统会应用以下**限制**：

- 系统会更积极地将应用置于应用待机模式，而不是等待应用空闲。
- 后台执行限制适用于所有应用，无论它们的目标 API 级别如何。
- 当屏幕关闭时，位置服务可能会被停用。
- 后台应用没有网络访问权限。

一如既往，一种比较好的做法是在省电模式激活时对您的应用进行测试。您可以通过设备的 Settings > Battery Saver 界面手动开启省电模式。

测试和问题排查

新的电源**管理**功能会影响在 Android 9 设备上运行的所有应用，无论应用是否以 Android 9 为目标。务必确保您的应用在这些设备上行为正常。

务必在各种条件下测试您的应用的主要用例，以便了解电源管理功能彼此之间的交互。您可以使用 Android 调试桥命令开关某些功能。

Android 调试桥命令

您可以使用 Android 调试桥 shell 命令测试多个电源管理功能。

应用待机群组

您可以使用 ADB 为您的应用手动指定应用待机群组。要更改应用的群组，请使用以下命令：

```
$ adb shell am set-standby-bucket packagename active|working_set|frequent|rare
```

您还可以使用该命令一次设置多个软件包：

```
$ adb shell am set-standby-bucket package1 bucket1 package2 bucket2...
```

要检查应用处于哪一个群组，请运行以下命令：

```
$ adb shell am get-standby-bucket [packagename]
```

如果您不传递 *packagename* 参数，命令将列出所有应用的群组。应用还可以调用新函数 [UsageStatsManager.getAppStandbyBucket\(\)](#)，在运行时查找所属的群组。

省电模式

可以使用多个命令测试您的应用在低电量条件下的行为。

注：您还可以使用设备的 Settings > Battery saver 界面将设备置于省电模式。

要模拟拔下设备电源时的情形，请使用以下命令：

```
$ adb shell dumpsys battery unplug
```

要测试设备在低电量条件下的行为，请使用以下命令：

```
$ adb shell settings put global low_power 1
```

完成测试后，您可以使用以下命令撤消设备的手动设置：

```
$ adb shell dumpsys battery reset
```

2.5 对非 SDK 接口的限制与测试



Android 9（API 级别 28）引入了针对非 SDK 接口的使用**限制**，无论是直接使用还是通过反射或 JNI 间接使用。无论应用是引用非 SDK 接口还是尝试使用反射或 JNI 获取其句柄，均适用这些**限制**。

一般来说，应用应当仅使用 SDK 中正式记录的类。这意味着，在您通过反射之类的语义来操作某个类时，不应打算访问 SDK 中未列出的函数或字段。

使用此类函数或字段很可能会破坏您的应用。

区分 SDK 接口和非 SDK 接口

一般来说，SDK 接口是指在 Android 框架软件包索引中记录的接口。对非 SDK 接口的处理是 API 抽象化的实现细节；其会随时更改，恕不另行通知。

Android 9 引入了针对非 SDK 接口的使用**限制**，无论是直接使用还是通过反射或 JNI 间接使用。无论应用是引用非 SDK 接口还是尝试使用反射或 JNI 获取其句柄，均适用这些**限制**。

测试非 SDK 接口

您可以通过下载 Android 9 对您的应用进行测试。系统将打印日志，如果您的应用访问某些“列入黑名单的”非 SDK 接口，系统还可能显示 toast。如果您的应用调用“列入黑名单的”非 SDK 接口，系统将引发错误。

注意 toast，它会提醒您注意被建议禁用的接口。此外，确保检查应用的日志消息，其中包含关于应用所访问的非 SDK 接口的更多详细信息，包括以 Android 运行时所使用的格式列出的声明类、名称和类型。日志消息还说明了访问方法：直接、通过反射或者通过 JNI。最后，日志消息显示调用的非 SDK 接口属于黑名单还是黑名单。

您可以使用 adb logcat 访问这些日志消息，消息将显示在正在运行的应用的 PID 下。例如，日志中的条目看上去可能类似下面这样：

```
Accessing hidden field Landroid/os/Message;->flags:l (light greylist, JNI)
```

列入黑名单的非 SDK 接口包含可以在 Android 9 中继续工作，但我们不能保证在未来版本的平台中能够继续访问的函数和字段。如果由于某种原因，您不能实现列入黑名单的 API 的替代策略，则可以提交错误，以便请求重新考虑此**限制**。

某些 API 位于“黑名单”中，比列入黑名单的函数更为严格。列入黑名单的函数会使应用引发如保留非 SDK 接口的结果部分的表中所列的异常。

Android 9 支持非 SDK 接口的 StrictMode（称为 detectNonSdkApiUsage），因此当您的应用调用列入黑名单的应用时，堆叠追踪将显示上下文。

调用非 SDK 接口的结果

下表详细说明了各种访问方式及其相应的结果。

访问方式	结果
Daivik 指令引用字段	引发 NoSuchFieldError
Daivik 指令引用函数	引发 NoSuchMethodError
通过 Class.getDeclaredField() 或 Class.getField() 反射	引发 NoSuchFieldException
通过 Class.getDeclaredMethod() 或 Class.getMethod() 反射	引发 NoSuchMethodException
通过 Class.getDeclaredFields() 或 Class.getFields() 反射	结果中未出现非 SDK 成员
通过 Class.getDeclaredMethods() 或 Class.getMethods() 反射	结果中未出现非 SDK 成员
通过 env->GetFieldID() 调用 JNI	返回 NULL, 引发 NoSuchFieldError
通过 env->GetMethodID() 调用 JNI	返回 NULL, 引发 NoSuchMethodError
前台服务权限	现在, 想要使用前台服务的应用必须首先请求 FOREGROUND_SERVICE 权限。这是普通权

查看更多关于非 SDK 接口的常见问题及建议, 请扫描下方二维码



第3章

常见问题



常见问题

从我们发布 Android 9 新版本以来，很多开发者都对当前常见应用在 Android P 上做了一些测试和应用，我们在里总结了一些常规的问题，以及它们发生的原因和建议的修改措施。

应用不兼容的常见原因：

- 使用了系统的 ClassLoader 加载 org.apache.http.* 的库

Android M 就已经开始移除对 Apache HTTP client 的支持。而 Android P 的系统 ClassLoader 已经不支持加载 org.apache.http.* 包（抛出 NoClassDefFoundError），应用必须用自定义的 ClassLoader 来加载，同时确保 org.apache.http.* 的路径包含在应用 classpath 上。

应用不应该再使用 org.apache.http.legacy 库，如果实在必须，可以将它打包进自己的 APK，同时改名以防止与运行时的版本冲突。

- 没有使用兼容 Android Pie 的加固服务

部分加固服务可能尚未兼容 Android Pie。开发者应该使用兼容 Android Pie 的加固服务。

- 直接调用 dex2oat

从一开始，dex2oat 就被设计为系统内部使用的编译部署工具，Android 从来都未支持过开发者直接调用 dex2oat 的场景。

如果您需要从内存中加载 dex 文件，而不愿在存储中留下痕迹，请使用 Android O 中新增的加载器 InMemoryDexClassLoader。

相关的 dex / so 文件亦不应直接操作或篡改，干扰或篡改系统内部加载 dex 的逻辑很可能会导致兼容性问题。

- 使用了非 SDK 接口

非 SDK 接口在每次版本更新中都有可能被改动，开发者应只使用 SDK 接口。

- 使用了不兼容的第三方的库

如果您使用的第三方库尚不支持 Android P 版本，请报告给其提供商，帮助推动它解决兼容性问题。

非 SDK 接口的限制名单

- 白名单

- Android SDK 本身
- 没有任何**限制**

- 黑名单

- 只能被 Android 系统及系统应用使用
- 无论 targetSdkVersion 都禁止使用
- 对应用开发者来说，相当于没有这些接口

- 深灰名单

- 没有发现应用在使用，但我们觉得有潜在的可能性
- 当 targetSdkVersion < P 时允许使用
- 当 targetSdkVersion >= P 时禁止使用（相当于黑名单）

- 浅灰名单

- 已有应用在使用的非 SDK 接口，仍然可以继续使用
- 将来会考虑提供相应的 SDK 接口
- 当 targetSdkVersion >= P 时系统提示**警告**

凹口屏幕 Display Cutout

- 不要硬编码状态栏的高度，请使用 `WindowInsetsCompat` 获取状态列的高度。
- 注意屏幕大小与显示范围的差异，请使用 `View.getLocationInWindow()`，而不是 `View.getLocationOnScreen()`。处理 `MotionEvent` 时，使用 `getX()` / `getY()`，而不是 `getRawX()` / `getRawY()`。
- 凹口可以置中或靠边，只会在屏幕短边出现，两条短边皆可有缺口。

屏幕旋转锁定

在 Android P 上，不论是自动旋转或旋转锁定（rotation lock），应用界面皆可以为纵向或横向，这取决于最上层可见 `Activity` 的 `screenOrientation` 设置。请不要再假设设备在旋转锁定时必定为纵向。

Inline 函数调用检查

在 Android P 中，如果调用某个 `inline` 方法的类与 `inline` 方法所在的类由不同的 `ClassLoader` 加载，就会主动发起 `abort` (`inline` 不允许跨 dex 文件)，导致应用 `crash`。请尽量避免用不同的 `ClassLoader` 来加载相关的（有互相调用可能）类，因为被调用类的方法可能已经被 `inline` 了。

空闲应用无法访问麦克风、摄像头和传感器

为了更好地保证隐私，Android P **限制**所有处于空闲状态的应用对麦克风、摄像头和所有 `SensorManager` 传感器的访问。当一个应用的 UID 空闲时，麦克风将会报告系统“无音频信号”，传感器将会停止报告事件。应用使用的摄像头也会断开连接，如果应用尝试使用它们，则会生成**错误**。在大多数情况下，这些**限制**不会为现有应用带来新的问题，但我们仍然建议您从应用中移除此类传感器请求。

前台服务权限

应用 target 到 P 版本后，在使用前台服务时必须申请 FOREGROUND_SERVICE 权限。这是一个一般性权限，应用只需在 manifest 中声明，系统会自动授予而无需询问用户。但若无此权限即运行前台服务，系统会抛出 SecurityException。

后台服务限制

自 Oreo 起，Target SDK >= 26 的应用若没有在前台显示，它的后台服务将受到系统**限制**。需长期运行的服务应迁移至前台服务，并让使用者注意到服务正在运行；或改用排程作业，例如 WorkManager 或 JobScheduler。

Google Play targetSdkVersion 政策

为了推动应用获得 Android 新版本提供的安全和性能提升，Google Play 应用市场要求其上的应用必须

- 从2018年8月起，新发布的应用必须将 targetSdkVersion 设置为26或更高
- 从2018年11月起，现有应用的升级必须将 targetSdkVersion 设置为26或更高
- 2019年之后，新发布或升级应用必须将 targetSdkVersion 设置为一年内发布的 Android 版本

第 4 章

相关资源



相关资源

[Android Developers 中文网站](#)



谷歌开发者微信公众号：[Google_Developers](#)



更多中文视频，请观看 [Google Developers 个人频道](#)



欢迎您提意见

我们始终致力于不断改善我们的产品和服务，以帮助您更好的了解、使用我们的产品，拓展您的应用业务。欢迎您关注谷歌开发者公众号，在留言板进行留言，与我们分享您对本手册的看法和意见，让我们了解您想在未来版本中看到哪些内容。

