




## 隐私策略更新 | Android 11 应用兼容性适配



谷歌开发者   
已认证的官方帐号

已关注

9 人赞同了该文章

作者 / Fred Chung

**Android 11 的最终版本已正式发布！**该版本延续了之前发行版本里不断改进的隐私策略，为用户提供更加完善的控制机制和透明度，并帮助应用更好地处理自身的数据。

其中很多优化将当前安全策略的最佳实践应用于最近的 Android 发行版本中（它们并不仅仅针对 Android 11）。在本文中，我们将以下面四个最佳实践作为切入点，助力您的应用设计与与时俱进，并计划开始进行兼容性测试。

1. 处理内容 URI 分享
2. 递增式权限申请
3. 在前台访问敏感数据
4. 使用可重置的标识符

### 为其它应用提供合适的 URI 权限

随着 Android 11 中 **软件包可见性** 的策略更新，目标 API 级别为 30 的应用对设备上已安装的其它软件包默认仅拥有受限的可见性。这样的设计旨在为应用“查看”设备上的其它已安装软件包时，提供更好的“问责”制度。

为了简化迁移，对于常见的应用场景，我们提供了 **实现指南**。通常，应用需要具备对其它已安装软件包的可见性（通过 **PackageManager** API 验证）才可以和其它软件包进行交互。该特性通常应用于诸如：启动服务，或者访问属于其它应用的 Content Provider。

您访问 Content Provider 的模式可能不是通过发送显式 intent 到某个特定的应用，而是通过发送隐式 intent。这样的话，您无法预判接收端应用（最终处理这个 intent 的应用）的目标 API 等级，而这个等级决定了接收端应用是否会受到 Android 11 中引入的应用包可见性限制的影响。

为了保证接收端应用能够“查看”您的软件包，从而能够访问任何共享的 URI，您需要在 intent 中添加 `FLAG_GRANT_READ_URI_PERMISSION` 和/或者 `FLAG_GRANT_WRITE_URI_PERMISSION`。请注意，写入权限并不包含读取访问权限。当被 intent 触发以后，接收端应用会被授予对相关 URI 的临时访问权限。

```
val shareIntent = Intent(Intent.ACTION_VIEW).apply {  
    flags = Intent.FLAG_GRANT_READ_URI_PERMISSION
```

▲ 赞同 9 ▼

添加评论

🔗 分享

❤️ 喜欢

★ 收藏

📄 申请转载

...



随着应用的目标 SDK 版本的更新（即使更新到 Android 11 之前的版本），请您特别关注涉及到与其它应用分享 Content Provider 访问权限的用例，并确保授予适当的 URI 权限。无论哪个应用是这 content provider 的拥有者，这个策略都管用。

通常，我们要将数据访问程度限制为当前任务所需的水平，如果您遵循了这个最佳实践，您的 Content Provider 访问权限应该是按照个别 **URI 模式** 设定的。只要做到这点，您的 Content Provider 就已经可以兼容 Android 11 了！

### 递增式申请权限

**Android 用户研究报告** 显示，在请求获取用户的授权时，那些符合用户期望值的请求更有可能被获准。因此，当您应用中的某个功能需要这些权限时，最佳实践是在上下文中 **请求权限**。

用户授予权限的原因排行。来源：**Android 用户研究报告**

△ 大多数用户会为了使用某个特定的功能而选择同意授权

这项策略对于敏感权限尤其适用，如位置访问权限。从 Android 10 开始，平台引入了细粒度的位置模型，区分了前台和后台位置访问。大多数位置场景仅需要前台访问，比如当用户在操作 Activity 的时候。

事实上，Google Play 已经出台了**相关政策**限制不必要的后台位置访问。要检查您的应用可能在哪些地方从后台访问位置，请参阅：**后台访问位置信息**文档。如果您的应用需要后台位置权限，比如地理围栏应用，请确保后台位置对您的功能设计是不可或缺的。

对于适用的应用，需要先申请前台位置权限，然后在稍晚些再申请后台位置权限。这种方法为用户提供了控制权限授予级别的选择。此外，您还可以有策略地显示一个权限申请的说明，或者设计一个合理的交互界面，为用户提供更多信息，以说明用户授予位置权限之后所获得的的功能提升。

Android 11 要求面向 API 级别为 30 的应用使用递增式位置权限请求。任何同时申请前台位置权限（无论是粗略位置还是精确位置）和后台位置权限的请求都会被忽略并且返回如下错误信息。



```
E/GrantPermissionsActivity: Apps targeting 30 must have foreground permission l
```

请注意在 `requestPermissions()` API 请求的任何其它非位置权限也会同时被忽略。

因为 `requestPermissions` API 接受一个由所需权限组成的数组作为参数，您现有的代码可能已有了同时申请多个权限的情况（如下所示），因此这里我们鼓励大家检查和审核一下自己的代码，如果代码修改影响到用户交互，则需要按需进行相应设计。

如果启用了 `ActivityCompat` 或者框架 API：

```
requestPermissions(  
    arrayOf(  
        // 不要同时申请前台位置权限和后台位置权限  
        // 因为所有同时申请的权限都会被忽略  
        // 而是通过增量式请求位置权限  
        android.Manifest.permission.ACCESS_COARSE_LOCATION,  
        android.Manifest.permission.ACCESS_BACKGROUND_LOCATION,  
  
        ...)  
    )  
)
```

同样地，如果启用了 **Jetpack Activity 库**：

```
// 使用 Activity 库  
val requestPermissionsLauncher =  
    registerForActivityResult(ActivityResultContracts.RequestMultiPlePermissions(  
        map: MutableMap<String, Boolean> ->  
            ...  
    )  
...  
  
requestPermissionsLauncher.launch(  
    arrayOf(  
        // 不要同时申请前台位置权限和后台位置权限  
        // 因为所有同时申请的权限都会被忽略  
        // 而是通过增量式请求位置权限  
        android.Manifest.permission.ACCESS_COARSE_LOCATION,  
        android.Manifest.permission.ACCESS_BACKGROUND_LOCATION,  
  
        ...)  
    )  
)
```

## 合理访问位置、麦克风和相机

Android 的系统设计支持公开透明地访问敏感数据，比如麦克风、相机和位置。例如，应用在前台的时候，也就是用户能看到应用界面的时候，才可以使用麦克风和相机。这样可以提高公开透明性，所以用户可以在知情的情况下启用相关特性。

如果您的应用包含访问敏感数据的前台服务，请确认应用场景中包含直接的用户交互，使用户可以控制所执行的任务。例如，在一个视频会议应用中，您可以使用一个前台服务来支持活跃的会议进程，其中会涉及到访问麦克风和相机。其中应该包含一个对于用户可见的用于启动和停止会议进程的操作，也就是该前台服务。



此外，您的应用必须正确设置 `foregroundServiceType` 属性来表明位置、麦克风或者相机的用途。这样可以为应用增加系统可见性，同时在 Android 11 中也是必须配置的属性。更多信息请访问：[Android 11 中的前台服务](#)。

您可能需要在 `AndroidManifest` 中声明多种数据类型的用途。

```
android:foregroundServiceType = "microphone location camera"
```

如果您的功能是基于 `WorkManager` 的 [长时间运行的 worker](#) 来实现的，那么它实际上是运行在叫做 `SystemForegroundService` 的前台服务上。您应该在应用的 `AndroidManifest` 中包含适合的前台服务类型，它会同 Jetpack 库的 AAR [AndroidManifest 文件](#) 合并。

在应用的 `AndroidManifest` 中添加下面的声明，并且在其中定义所需的前台服务类型。

```
<service
    android:name="androidx.work.impl.foreground.SystemForegroundService"
    android:foregroundServiceType="location|microphone"//这里选择和您的应用相关
    tools:node="merge" />
```

当您需要将 worker 以前台服务运行时，您需要将合适的前台服务类型传入 `ForegroundInfo` 对象。传入的服务类型必须和上面在 `AndroidManifest` 中添加的声明一致或者是其子集。

```
setForeground(
    ForegroundInfo(NOTIF_ID,
        notification.build(),
        // 前台服务类型
        ServiceInfo.FOREGROUND_SERVICE_TYPE_LOCATION or ...)
)
```

## 弃用不可重置标识符

Android 系统使用了一些不可重置的硬件标识符，比如 IMEI 以支持各种操作系统功能。出于隐私方面的考虑，这些相对“强大”的持久性和唯一性的标识符不适合用于大部分应用场景。

从 Android 10 开始，系统对不可重置的设备标识符 [实施了限制](#)。比如，只有带有 `READ_PRIVILEGED_PHONE_STATE` 权限的系统应用才可以通过 [getSimSerialNumber\(\)](#) 方法访问 SIM 卡的硬件标识符。在 Android 11 中，操作系统对 [getIccId\(\)](#) 方法也增加了类似的限制来进一步 [限制访问权限](#)，现在该方法仅返回空字符串。

对于需要使用 SIM 卡信息作为唯一性标识的应用，需要在 Android 11 里进行“空字符串”的兼容性检查。一个替代方案是使用 [getSubscriptionId\(\)](#) 方法，它会针对设备上指定的 SIM 卡信息返回一个以数字 1 开头的唯一索引值，也就是说，如果同一张 SIM 卡被重新安装到设备上的话，它会保持之前的订阅标识符，除非设备恢复出厂设置。更多请参阅：[唯一标识符最佳做法](#)。

平台和 Google Play 服务为应用提供了一些其它的 [标识符](#)，提供各种唯一性、可重置性和有作用域限制的标识符，适用于各种不同的应用场景。更多请参阅：[唯一标识符最佳做法](#)。

以上内容能够帮助大家更快更新适配最新的 API，并设计出对隐私更友好的应用。更多资源请参阅：

- [Android 11 中的改进](#)
- [隐私设置最佳实践](#)

发布于 09-18



推荐阅读



Android开发不能不会——教你如何开发扫二维码功能

编程花无缺



Android安全——客户端安全要点

brucevanfdm

微信抢红包插件与Android辅助功能

逢年过节大家都少不了发微信红包，通过微信红包来表达祝福。同时，微信还有拼手气群红包。各种群好友群，亲戚群，工作群逢年过节常常会有红包可抢。抢红包的秘诀是：“网速要好，手速要快”...

古城



基于Android平台设计与实现

Jomes...

还没有评论

写下你的评论... 