应用2019上架谷歌Play商店适配指导

1. 谷歌API LEVEL要求，应用2019上架谷歌Play商店需要应用升级自己的targetSdkVersion到28，具体参考：
https://android-developers.googleblog.com/2019/02/expanding-target-api-level-requirements.html
2. 2018年要求升级targetSdkVersion到26；
3. P版本上通过targetSdkVersion控制生效的变更需要重点关注，其他版本变更的适配请关注：
https://developer.android.google.cn/distribute/best-practices/develop/target-sdk.html

Google P版本变化点		说明	第三方应用影响	适配指导
ART	非SDK接口管控	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。	所有使用非SDK接口的三方应用	• 重新做接口扫描 • 关注应用有没有使用深灰名单和黑名单非SDK接口，需要适配和整改 • 无法整改的接口说明详细使用场景和理由反馈给谷歌申请加灰名单
	内联方法不能跨dex限制	如果应用内内联方法出现跨dex场景，系统将会直接abort，导致应用出现崩溃	使用热修复+插件框架的应用（淘宝、爱奇艺等等）	• 尽量避免使用不同的classloader来加载相关的（有互相调用可能）类 • 如果一定要这样做的话，需要避免内联(比如在不期望被inline的函数里面加个try catch，这样compiler就不会将这个函数inline)
Framework	Apache HTTP 客户端弃用	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。	1.使用热修复或者插件框架的应用 2.TargetSdkVersion>=28的应用	• 如果要继续使用 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"/> • 使用 HttpURLConnection 类替代apache-http

Android Q版本变更列表

Google Q版本变化点		说明	第三方应用影响	Q版本处理策略
隐私和安全	应用存储空间限制	• 应用通过File接口只能读写应用自己沙箱目录的文件，无法直接读写沙箱外文件 • 应用需要使用安卓提供的接口读写沙箱外的文件	对应用需要读写应用沙箱外文件的场景都有影响	需要应用适配解决
	禁止应用读取deviceid和SN号	• 通过新增的权限控制：READ_PRIVILEGED_PHONE_STATE • 只有系统签名的应用才能申请，三方应用无法申请	影响应用依赖deviceid的功能，包括：数据上报、用户画像、推荐、广告等功能；金融类的应用需要重点验证对用户功能是否会产生影响	需要应用适配解决
	Wifi Mac地址随机化	• P版本已有该功能，未使能，可以在开发者选项打开 • Q版本默认使能该特性		
	三方应用无法获取手机的USB序列号	• targetSdkVersion>=Q的应用有影响 • 新增权限控制：android.permission.MANAGE_USB • 该权限需要系统签名才能申请	同上	应用升级TargetSdkVersion需要适配
	不允许后台弹页面	• 禁止应用后台启动activity • 目前版本未真的禁止，有toast告警提示，需要开发者整改	影响应用后台弹页面的场景：闹钟、音乐锁屏、语音和视频电话	需要应用按照通知提醒的方式适配解决
	不允许应用隐藏图标	• 所有三方应用的图标都会显示 • 如果应用没有启动页面，点击图标跳转到应用的详情页面	只有服务的应用	• 需要应用检查是否会有图标的问题 • 如果确实需要隐藏，可以通过担保应用担保来适配
	禁止应用后台访问剪切板	• 除非应用程序是默认输入法编辑器（IME）或当前具有焦点的应用程序，否则应用程序无法访问剪贴板数据。	后台读取剪切板场景	需要应用适配解决
	位置权限三态化	• Android Q 允许用户指定应用从不、仅在使用期间（运行时），或者任何时候 (退到后台) 都能获取位置信息。 • 如果用户选择仅在使用，那么前台服务需要在manifest定义的前台服务中增加android:foregroundServiceType= “location” 才能访问位置信息	影响后台gps定位场景：百度地图、高德地图后台导航	• 后TargetSdkVersion>=Q，应用需要后台定位需要申请后台定位权限ACCESS_BACKGROUND_LOCATION • 前台服务设置type：Service.startForeground(Notification notif, int serviceTypes);
AOSP	Hidden API	• 名单类型变化，新增：max-o和max-p名单类型 • 名单变化：黑名单增加	所有使用非SDK接口的三方应用	• 使用最新Q版本的工具重新做接口扫描 • 有问题的接口需要应用适配和整改 • 无法整改的接口说明详细使用场景和理由反馈给谷歌申请加灰名单
	折叠屏	• 提供统一的折叠屏方案和适配接口	• 状态切换，应用页面重新加载问题 • 屏幕比例适配问题	需要应用适配解决

Android Q版本变更列表

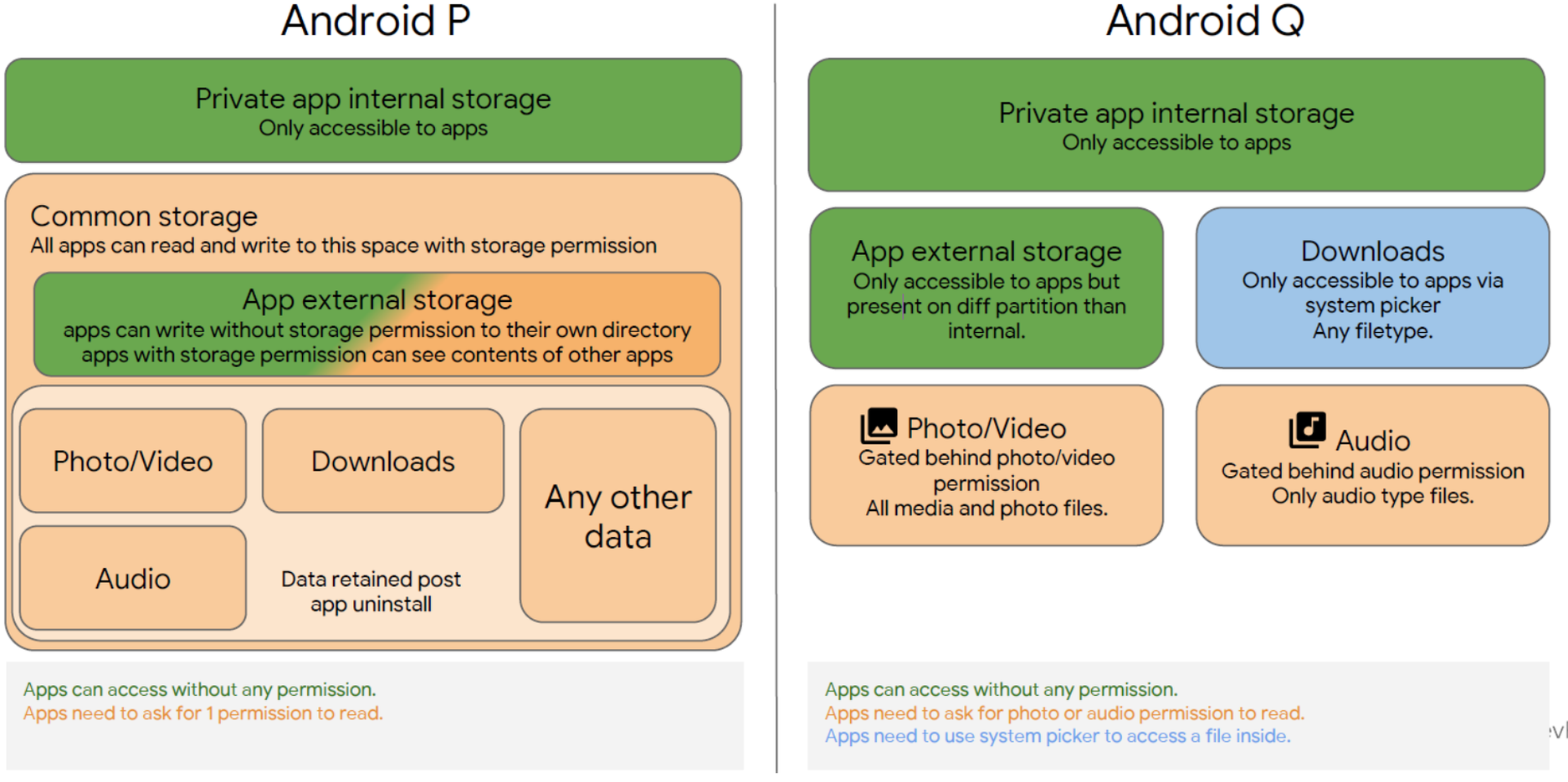
Google Q版本变化点		说明	第三方应用影响	Q版本处理策略
多媒体	并发录音	• P不支持并发录音，录音焦点不可抢占 • 支持并发录音，录音焦点可以抢占	应用后台录音场景，可能录到空数据	• 需要应用通过接口监听焦点状态变化，做出对应处理
	访问所有相机信息需要获得权限	• Android Q更改了默认情况下getCameraCharacteristics()方法返回的信息的广度。特别是，您的应用必须具有CAMERA权限才能访问此方法的返回值中包含的潜在设备特定元数据。	相机场景	需要应用适配解决
	MediaProvider	• “_data”值变化，返回的值不再是文件真实路径 • 查询sql语句管控，不合法的查询列会导致返回的查询数据为空，除了mediastore中定义的列以外的信息，其他的信息，无法查询成功 • 位置信息被删除，无法通过MediaProvider直接查询	影响应用的多媒体查询，可能导致查询结果为空	• 应用不能再通过获取的“_data”值作为文件真实路径判断文件是否存在或者直接进行读写 • 应用对MediaProvider的查询语句排查和整改 • 应用读取位置信息适配：需要应用动态申请ACCESS_MEDIA_LOCATION权限，并调用MediaStore的setRequireOriginal方法
系统应用	安装器安装应用接口废弃	• Q版本不允许使用intent（action为INSTALL_PACKAGE），在intent中通过 “file://” 附带文件路径拉起安装器安装应用 • 主要影响TargetSdkVersion<24的应用	应用内安装	使用FileProvider适配，使用content Uri替代file Uri解决
	Q版本通知规格变更	• 屏通知增加权限：对于TargetSdkVersion>=Q且通知设置了全屏属性(fullscreenIntent)，需要增加USE_FULL_SCREEN_INTENT权限通知 • 优先级(Importance)调整：增加通知助手NotificationAssistant去调整通知优先级 • 全屏通知在桌面状态显示成横幅：带有fullscreenIntent的通知在桌面显示成横幅，在锁屏界面显示成全屏；（前提：Importance>=4）	不适配可能出现： • 通知无法横幅显示 • 通知无法全屏显示 • 通知无法响铃、震动	需要应用适配解决

Android Q版本变更列表

Google Q版本变化点		说明	第三方应用影响	Q版本处理策略
OS	禁止三方应用调用dex2oat	• 通过selinux权限限制 • 只对TargetSdkVersion>=Q有影响	加固和需要调用dex2oat完成的功能	应用不要自己调用dex2oat
	OpenGL切换Vulkan	• 2D 绘制，hwui通路增加skiavulkan pipeline； • 3D 绘制，opengl接口增加Angle实现，可以选择通过vulkan实现opengl API • vulkan版本升级到vulkan1.1；	• 对外接口没有变化，不会导致兼容性问题，但是应用也需要排查有没有依赖OpenGL的私有接口 • 可能存在显示问题	应用排查是否依赖OpenGL的私有接口，如果有需要整改
短距	启用和禁用Wi-Fi的限制	• 在Android Q上运行的应用无法启用或停用Wi-Fi。WifiManager.setWifiEnabled()方法始终返回false。 • 如果需要，请使用设置面板提示用户启用和禁用Wi-Fi。	对应用直接停止和开启wifi的功能有影响	需要应用使用设置面板提示用户开启和禁用Wifi
	电话，Wi-Fi，蓝牙API所需的精确位置权限	• TargetSdkVersion>=Q:除非您的应用具有ACCESS_FINE_LOCATION权限，否则在Android Q上运行时，您的应用无法在Wi-Fi，Wi-Fi Aware或蓝牙API中使用多种方法。 • targetSdkVersion<Q:不受影响，只需要申请ACCESS_COARSE_LOCATION或者ACCESS_FINE_LOCATION即可	对通过wifi，网络和蓝牙接口定位有影响	应用升级TargetSdkVersion的话需要适配
	Wifi和蓝牙使用需要打开位置开关	• WiFi扫描/蓝牙扫描，需要开启定位服务	对应用的wifi和蓝牙扫描功能有影响	需要开发者开启wifi和蓝牙扫描之前判断位置开关是否开启，如果没有开启提示用户打开再发起扫描
兼容性	64位支持	• 新开发的应用，2019-8-1以后上架谷歌Play要求应用的native代码需要提供64位的版本 • 更新的应用，2019-11-1以后上架谷歌play要求应用的native代码需要提供64位的版本	需要上架谷歌Play的应用	• 应用自己的Native代码增加64位版本 • 使用的第三方的SDK的native代码增加64位版本
API LEVEL	2019谷歌Play上架API LEVEL政策	• 新开发的应用，2019-8-1以后上架谷歌Play要求应用的TargetSdkVersion>=28 • 更新的应用，2019-11-1以后上架谷歌play要求应用的TargetSdkVersion>=28	需要上架play商店的应用	重点关注TargetSdkVersion控制生效的一些特性，特别是P版本
	最小TargetSdkVersion要求至少23	• 应用的TargetSdkVersion<23，应用首次启动会弹框警告	所有三方应用	需要应用升级TargetSdkVersion到23

应用存储空间限制-特性介绍

1. Android Q为每个应用程序在外部存储设备提供了一个独立的存储沙箱，应用通过路径创建的文件都保存在应用的沙箱目录
2. 共享集合：如果应用的一些文件是用户选择下载保存的，应用卸载的时候用户不希望删除，这部分文件开发者可以通过MediaProvider接口保存在公共共享集合，包括：照片、视频、音乐和下载集合
3. 新的访问多媒体文件的权限：应用读写自己创建的文件不需要权限，但是如果需要读取其他应用创建的多媒体文件就需要申请对应的权限，通过MediaProvider接口读取
4. 读写其他应用的下载公共集合文件需要使用SAF的方式读写
5. 目前版本该特性没有默认开启，需要开发者通过命令开启：adb shell sm set-isolated-storage on
6. 参考谷歌提供的官方适配指导文档：https://developer.android.google.cn/preview/privacy/scoped-storage



应用存储空间限制-兼容性影响分析

问题1：文件分享，比如应用通过微信、qq、微博等方式分享图片，分享失败

问题2：应用使用其他应用打开文件，比如qq调用wps打开好友发的pdf文件

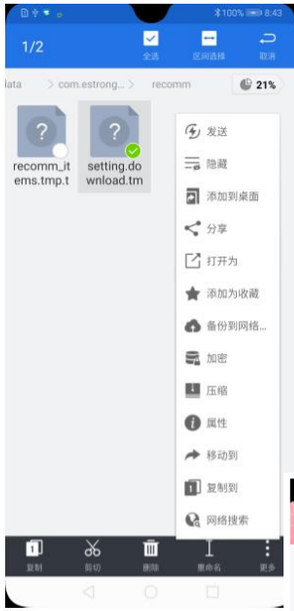
1.问题分析

- 分享方应用使用file:// URI分享应用沙箱文件，由于沙箱化变更，应用无法通过文件路径直接访问应用沙箱外的任何文件，会导致接收文件方应用微博、qq和微信应用端报“文件找不到”的问题
- 接收方应用不支持FileProvider的content uri方式，打开文件失败

2.适配指导：

- 分享方统一使用FileProvider的方式分享文件
- 接收方需要支持FileProvider的URI方式读取文件
- 参考适配指导文档：

https://developer.android.com/training/secure-file-sharing
https://developer.android.com/reference/android/support/v4/content/FileProvider#SpecifyFiles



应用存储空间限制-使用FileProvider共享文件适配指导

1.指定应用的FileProvider

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp">
    <application
        ...>
        <provider
            android:name="android.support.v4.content.FileProvider"
            android:authorities="com.example.myapp.fileprovider"
            android:grantUriPermissions="true"
            android:exported="false">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/filepaths" />
        </provider>
        ...
    </application>
</manifest>
```

2.指定应用分享的文件路径

```
<paths>
    <files-path path="images/" name="myimages" />
</paths>
```

3.获得分享文件的Content URI

```
   /*
    * Get a File for the selected file name.
    * Assume that the file names are in the
    * imageFilename array.
    */
   File requestFile = new File(imageFilename[position]);
   /*
    * Most file-related method calls need to be in
    * try-catch blocks.
    */
   // Use the FileProvider to get a content URI
   try {
       fileUri = FileProvider.getUriForFile(
               MainActivity.this,
               "com.example.myapp.fileprovider",
               requestFile);
   } catch (IllegalArgumentException e) {
       Log.e("File Selector",
             "The selected file can't be shared: " + requestFile.toString());
   }
```

4.临时授予文件接收方的文件读写权限

```
if (fileUri != null) {
    // Grant temporary read permission to the content URI
    resultIntent.addFlags(
        Intent.FLAG_GRANT_READ_URI_PERMISSION);
}
```

5.分享文件

```
if (fileUri != null) {
    ...
    // Put the Uri and MIME type in the result Intent
    resultIntent.setDataAndType(
            fileUri,
            getContentResolver().getType(fileUri));
    // Set the result
    MainActivity.this.setResult(Activity.RESULT_OK,
            resultIntent);
    } else {
        resultIntent.setDataAndType(null, "");
        MainActivity.this.setResult(RESULT_CANCELED,
                resultIntent);
    }
}
```

6.文件接收方读取文件

```
// Get the file's content URI from the incoming Intent
Uri returnUri = returnIntent.getData();
/*
 * Try to open the file for "read" access using the
 * returned URI. If the file isn't found, write to the
 * error log and return.
 */
try {
    /*
     * Get the content resolver instance for this context, and use it
     * to get a ParcelFileDescriptor for the file.
     */
    inputPFD = getContentResolver().openFileDescriptor(returnUri, "r");
} catch (FileNotFoundException e) {
    e.printStackTrace();
    Log.e("MainActivity", "File not found.");
    return;
}
// Get a regular file descriptor for the file
FileDescriptor fd = inputPFD.getFileDescriptor();
```

应用存储空间限制-通过share uid共享文件适配指导

补充：对于共享uid的多个应用，应用的sandbox目录也是共享的，/sdcard/Android/sandbox下创建的目录名称不是包名，而是 shared-$(uid name)

- 同系列的应用可以通过此机制共享私有数据的访问
- 多个应用共享sandbox目录时，只有在最后一个应用卸载时，才会删除sandbox下的目录

```
drwxrwx--x   3 system         sdcard_rw  3488 2019-02-19 18:29 shared-android.uid.system
drwxrwx--x   3 u0_a22         sdcard_rw  3488 2019-02-19 18:29 shared-android.uid.systemui
drwxrwx--x   3 u0_a29         sdcard_rw  3488 2019-02-19 18:29 shared-com.android.emergency.uid
drwxrwx--x   3 u0_a96         sdcard_rw  3488 2019-02-19 18:29 shared-com.google.ar.core
drwxrwx--x   3 u0_a38         sdcard_rw  3488 2019-02-19 18:29 shared-com.huawei.android.weather
drwxrwx--x   3 u0_a30         sdcard_rw  3488 2019-02-19 18:29 shared-com.huawei.hiaction.share_user
drwxrwx--x   4 u0_a71         sdcard_rw  3488 2019-02-19 18:30 shared-com.huawei.rcsserviceapplication
HWHMA:/sdcard/Android/sandbox #
```

应用存储空间限制-兼容性影响分析

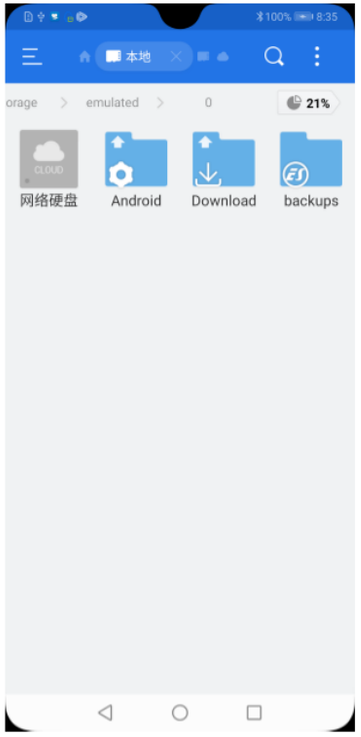
问题3：读写应用沙箱外文件场景，比如：文件管理器、手机管家空间清理、微信给好友发送本地文件
问题4：通过SAF的方式修改头像失败，比如头条、滴滴出行等等

1.问题分析

- 应用只能在沙箱内随意读写文件，无法通过路径读写应用沙箱外文件（包括公共文件），所以可以看到es文件管理器只能看到自己沙箱的文件和目录；
- 测试也发现一些应用通过SAF的方式修改用户头像会出现文件找不到的问题，该问题和开发者使用SAF的方式有关系，需要开发者排查，应用可以通过DocumentUI拿到文件的Content Uri，通过context.getContentResolver().openFileDescriptor(url, “r”)就可以读写文件

2.适配指导：

- 读取其他应用生成的多媒体文件可以通过MediaProvider接口读取，具体适配指导请参考PPT12和13页；
- 需要读写任意指定目录的文件需要通过SAF的方式读写
- SAF适配指导链接：https://developer.android.com/guide/topics/providers/document-provider



应用存储空间限制-SAF适配指导

1.搜索文档

```
private static final int READ_REQUEST_CODE = 42;
...
/**
 * Fires an intent to spin up the "file chooser" UI and select an image.
 */
public void performFileSearch() {

    // ACTION_OPEN_DOCUMENT is the intent to choose a file via the system's file
    // browser.
    Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);

    // Filter to only show results that can be "opened", such as a
    // file (as opposed to a list of contacts or timezones)
    intent.addCategory(Intent.CATEGORY_OPENABLE);

    // Filter to show only images, using the image MIME data type.
    // If one wanted to search for ogg vorbis files, the type would be "audio/ogg".
    // To search for all documents available via installed storage providers,
    // it would be "*/*".
    intent.setType("image/*");

    startActivityForResult(intent, READ_REQUEST_CODE);
}
```

2.通过传回文件的URI读写文件

```
@Override
public void onActivityResult(int requestCode, int resultCode,
        Intent resultData) {

    // The ACTION_OPEN_DOCUMENT intent was sent with the request code
    // READ_REQUEST_CODE. If the request code seen here doesn't match, it's the
    // response to some other intent, and the code below shouldn't run at all.

    if (requestCode == READ_REQUEST_CODE && resultCode == Activity.RESULT_OK) {
        // The document selected by the user won't be returned in the intent.
        // Instead, a URI to that document will be contained in the return intent
        // provided to this method as a parameter.
        // Pull that URI using resultData.getData().
        Uri uri = null;
        if (resultData != null) {
            uri = resultData.getData();
            Log.i(TAG, "Uri: " + uri.toString());
            showImage(uri);
        }
    }
}
```

3.读文件1

```
private Bitmap getBitmapFromUri(Uri uri) throws IOException {
    ParcelFileDescriptor parcelFileDescriptor =
            getContentResolver().openFileDescriptor(uri, "r");
    FileDescriptor fileDescriptor = parcelFileDescriptor.getFileDescriptor();
    Bitmap image = BitmapFactory.decodeFileDescriptor(fileDescriptor);
    parcelFileDescriptor.close();
    return image;
}
```

4.读文件2

```
private String readTextFromUri(Uri uri) throws IOException {
    InputStream inputStream = getContentResolver().openInputStream(uri);
    BufferedReader reader = new BufferedReader(new InputStreamReader(
            inputStream));
    StringBuilder stringBuilder = new StringBuilder();
    String line;
    while ((line = reader.readLine()) != null) {
        stringBuilder.append(line);
    }
    fileInputStream.close();
    parcelFileDescriptor.close();
    return stringBuilder.toString();
}
```

5.删除文件

```
DocumentsContract.deleteDocument(getContentResolver(), uri);
```

6.编辑文件

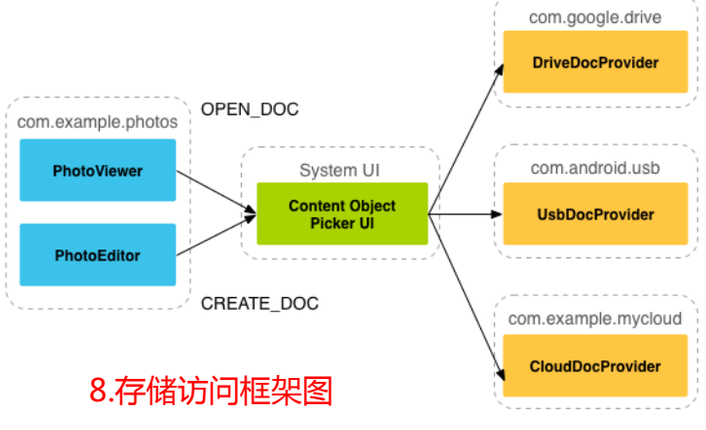
```
private void alterDocument(Uri uri) {
    try {
        ParcelFileDescriptor pfd = getActivity().getContentResolver().
                openFileDescriptor(uri, "w");
        FileOutputStream fileOutputStream =
                new FileOutputStream(pfd.getFileDescriptor());
        fileOutputStream.write(("Overwritten by MyCloud at " +
                System.currentTimeMillis() + "\n").getBytes());
        // Let the document provider know you're done by closing the stream.
        fileOutputStream.close();
        pfd.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

7.新建文件

```
private static final int WRITE_REQUEST_CODE = 43;
...
private void createFile(String mimeType, String fileName) {
    Intent intent = new Intent(Intent.ACTION_CREATE_DOCUMENT);
    // Filter to only show results that can be "opened", such as
    // a file (as opposed to a list of contacts or timezones).
    intent.addCategory(Intent.CATEGORY_OPENABLE);

    // Create a file with the requested MIME type.
    intent.setType(mimeType);
    intent.putExtra(Intent.EXTRA_TITLE, fileName);
    startActivityForResult(intent, WRITE_REQUEST_CODE);
}
```

8.存储访问框架图



应用存储空间限制-兼容性影响分析

问题5：读写本地多媒体文件，比如微信、qq给好友发送本地图片和视频；发图片微博等等

问题6：图片保存的路径问题，应用直接通过路径保存文件都是保存在应用的沙箱路径，无法保存到系统相机的拍照目录DCIM

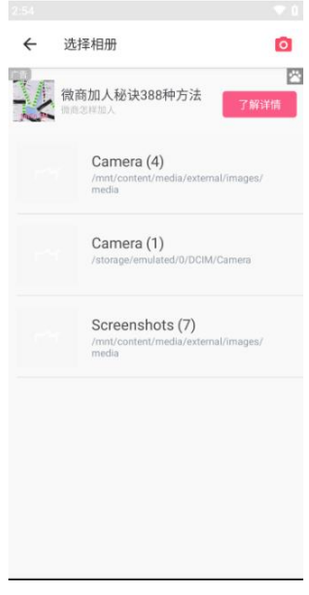
问题7：多媒体文件删除问题，比如三方图库照片无法删除

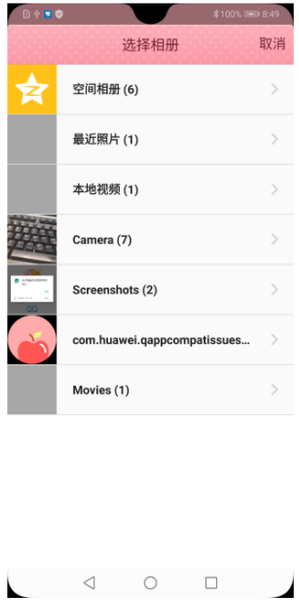
1.问题分析

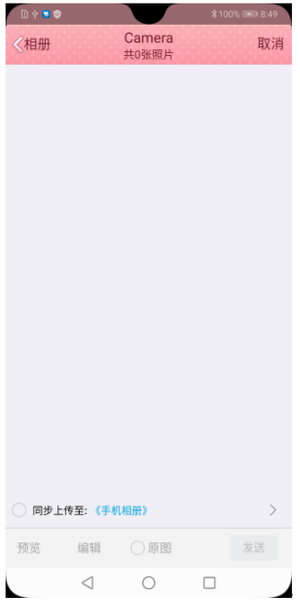
- P版本通过MediaProvider查询图片的“_data”字段值为图片的真实文件路径，应用也可以直接路径读取文件，或者通过路径判断图片文件是否存在；
- P版本通过路径直接访问多媒体文件，包括读和写其他应用生成的文件；
- Q版本通过MediaProvider查询图片的“_data”字段值只有图片是应用沙箱内的图片，返回的才是文件的真实路径，其他图片返回的是非文件真实路径，**通过该路径判断文件是否存在会出现问题**
- Q版本无法通过路径读写应用沙箱外的所有文件，需要读写沙箱外的多媒体文件，需要使用MediaProvider接口

2.适配指导：

- **不要通过路径进行文件操作**
- 如果需要读取其他应用生成的多媒体文件，需要申请对应的权限
- 查询图片建议使用MediaProvider查询，需要注意的是查询的“_data”值不能再认为是文件的真实路径
- 保存图片到沙箱外的公共文件，需要使用SAF或者MediaProvider的接口








```
kirin980:/sdcard/android/sandbox/com.mt.mtxx.mtxx/DCIM/camera # ls
mtxx_album_take_photo_temp_mh1551422698767.jpg
```

```
jiyun    : image path:/mnt/content/media/external/images/media/20, fi
jiyun    : image path:/mnt/content/media/external/images/media/21, fi
jiyun    : image path:/mnt/content/media/external/images/media/22, fi
jiyun    : image path:/mnt/content/media/external/images/media/23, fi
jiyun    : image path:/mnt/content/media/external/images/media/43, fi
jiyun    : image path:/mnt/content/media/external/images/media/44, fi
jiyun    : image path:/mnt/content/media/external/images/media/45, fi
jiyun    : image path:/mnt/content/media/external/images/media/46, fi
jiyun    : image path:/mnt/content/media/external/images/media/3208,
jiyun    : image path:/mnt/content/media/external/images/media/3209,
jiyun    : image path:/mnt/content/media/external/images/media/3210,
jiyun    : image path:/mnt/content/media/external/images/media/3231,
```

应用存储空间限制-其他应用的多媒体文件读写适配指导

多媒体文件读写适配指导：

1.应用读取自己创建的多媒体文件不需要权限，读取其他应用创建的多媒体文件需要申请对应的权限：

- 音乐文件：android.permission.READ_MEDIA_AUDIO
- 照片文件：android.permission.READ_MEDIA_IMAGES
- 视频文件：android.permission.READ_MEDIA_VIDEO

2.通过MediaProvider查询并读取文件参考实现：

```
public static List<Uri> loadPhotoFiles(Context context) {
    Log.e(TAG, "loadPhotoFiles");
    List<Uri> photoUris = new ArrayList<Uri>();
    Cursor cursor = context.getContentResolver().query(
            MediaStore.Images.Media.EXTERNAL_CONTENT_URI, new String[]{MediaStore.Im-
ages.Media._ID}, null, null, null);
    Log.e(TAG, "cursor size:" + cursor.getCount());
    while (cursor.moveToNext()) {
        int id = cursor.getInt(cursor
                .getColumnIndex(MediaStore.Images.Media._ID));
        Uri photoUri = Uri.parse(MediaStore.Images.Media.EXTERNAL_CONTENT_URI.toString() +
File.separator + id);
        Log.e(TAG, "photoUri:" + photoUri);
        photoUris.add(photoUri);
    }
    return photoUris;
}
```

```
public static Bitmap getBitmapFromUri(Context context, Uri uri) throws IOException {
    ParcelFileDescriptor parcelFileDescriptor =
            context.getContentResolver().openFileDescriptor(uri, "r");
    FileDescriptor fileDescriptor = parcelFileDescriptor.getFileDescriptor();
    Bitmap image = BitmapFactory.decodeFileDescriptor(fileDescriptor);
    parcelFileDescriptor.close();
    return image;
}
```

3.通过MediaProvider保存文件到公共集合目录参考实现：

```
    ContentValues values = new ContentValues();
    values.put(MediaStore.Images.Media.DISPLAY_NAME, displayName);
    values.put(MediaStore.Images.Media.DESCRIPTION, description);
    values.put(MediaStore.Images.Media.MIME_TYPE, mimeType);
    values.put(MediaStore.Images.Media.PRIMARY_DIRECTORY, savePrimaryDir);
    values.put(MediaStore.Images.Media.SECONDARY_DIRECTORY, saveSecondaryDir);
    Uri url = null;
    String stringUrl = null;       /* value to be returned */
    ContentResolver cr = context.getContentResolver();
    try {
        url = cr.insert(uri, values);
        if (url == null) {
            return null;
        }
        byte[] buffer = new byte[BUFFER_SIZE];
        ParcelFileDescriptor parcelFileDescriptor = cr.openFileDescriptor(url, "w");
        FileOutputStream fileOutputStream =
                new FileOutputStream(parcelFileDescriptor.getFileDescriptor());
        InputStream inputStream = context.getResources().getAssets().open(saveFileName);
        while (true) {
            int numRead = inputStream.read(buffer);
            if (numRead == -1) {
                break;
            }
            fileOutputStream.write(buffer, 0, numRead);
        }
        fileOutputStream.flush();
    } catch (Exception e) {
        Log.e(TAG, "Failed to insert media file", e);
        if (url != null) {
            cr.delete(url, null, null);
            url = null;
        }
    }
    if (url != null) {
        stringUrl = url.toString();
    }
    return stringUrl;
}
```

1.插入数据库一条记录，可以指定保存多媒体文件的一级目录和二级目录

2.通过openFileDescriptor写文件实现文件的最终保存

应用存储空间限制-其他应用的多媒体文件读写适配指导

多媒体文件读写适配指导：

4.通过MediaProvider保存文件的目录设置说明：

可以通过PRIMARY_DIRECTORY和SECONDARY_DIRECTORY字段来设置一级目录和二级目录，

- 一级目录必须是和MIME type的匹配的根目录下的Public目录，一级目录可以不设置，不设置时会放到默认的路径；
- 二级目录可以不设置，不设置时直接保存在一级目录下
- 应用生成的文档类文件，代码里面默认不设置时，一级是Downloads目录，也可以设置为Documents目录，建议推荐三方应用把文档类的文件一级目录设置为Documents目录。
- 一级目录MIME type，默认目录、允许的目录映射以及对应的读取权限如下表所示：

数据类型	媒体数据库 TABLE ID	MIME参数	可以设置的一级目录	默认一级目录	默认二级目录	对应媒体权限
音频	AUDIO_MEDIA/ AUDIO_MEDIA_ID	audio/mpeg	Environment.DIRECTORY_ALARMS Environment.DIRECTORY_MUSIC Environment.DIRECTORY_NOTIFICATIONS Environment.DIRECTORY_PODCASTS Environment.DIRECTORY_RINGTONES	Environment.DIRECTORY_MUSIC	无	READ_MEDIA_AUDIO
音乐专辑艺术家	AUDIO_ALBUMART/ AUDIO_ALBUMART_ID	image/jpeg	Environment.DIRECTORY_MUSIC	Environment.DIRECTORY_MUSIC	.thumbnails	READ_MEDIA_AUDIO
音乐播放列表	AUDIO_PLAYLISTS AUDIO_PLAYLISTS_ID	NA	Environment.DIRECTORY_MUSIC	Environment.DIRECTORY_MUSIC	无	READ_MEDIA_AUDIO
图片	IMAGES_MEDIA/ IMAGES_MEDIA_ID	image/jpeg	Environment.DIRECTORY_PICTURES Environment.DIRECTORY_DCIM	Environment.DIRECTORY_PICTURES	无	READ_MEDIA_IMAGES
图片缩略图	IMAGES_THUMBNAILS IMAGES_THUMBNAILS_ID	image/jpeg	Environment.DIRECTORY_PICTURES	Environment.DIRECTORY_PICTURES	.thumbnails	READ_MEDIA_IMAGES
视频	VIDEO_MEDIA/ VIDEO_MEDIA_ID	video/mp4	Environment.DIRECTORY_MOVIES Environment.DIRECTORY_DCIM	Environment.DIRECTORY_MOVIES	无	READ_MEDIA_VIDEO
视频缩略图	VIDEO_THUMBNAILS/ VIDEO_THUMBNAILS_ID	image/jpeg	Environment.DIRECTORY_MOVIES	Environment.DIRECTORY_MOVIES	.thumbnails	READ_MEDIA_VIDEO
下载数据	DOWNLOADS DOWNLOADS_ID	NA	Environment.DIRECTORY_DOWNLOADS	Environment.DIRECTORY_DOWNLOADS	无	NA
其他文件	FILES FILES_ID	NA	Environment.DIRECTORY_DOWNLOADS Environment.DIRECTORY_DOCUMENTS	Environment.DIRECTORY_DOWNLOADS	无	NA（根据文件类型，如果有媒体文件会按照媒体文件要求权限）

应用存储空间限制-其他应用的多媒体文件读写适配指导

3.写其他应用生成的多媒体文件：应用只有自己插入的多媒体文件的写权限，没有别的应用插入的多媒体文件的写权限，直接通过下面的方法删除其他应用的多媒体文件会删除失败

```
context.getContentResolver().delete(uri, null, null))
```

适配建议：
方式1：如果应用需要修改其他应用插入的多媒体文件，需要作为**系统默认应用**，比如作为系统默认图库，可以删除和修改其他应用的图片和视频文件；作为系统的默认音乐播放软件，可以删除和修改其他应用的音乐文件。

```
<manifest>
    <!-- Use the old permission for devices running Android 9 (API level 28) or
         lower. -->
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
        android:maxSdkVersion="28" />

    <!-- Use the new "read audio" permission for Android Q. -->
    <uses-permission
        android:name="android.permission.READ_MEDIA_AUDIO" />

    <application>
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
                   android:name="android.intent.category.LAUNCHER" />

                <!-- You must include these categories for your app
                     to be considered for the ROLE_MUSIC role. -->
                <category android:name="android.intent.category.APP_MUSIC" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

方式2：使用ContentResolver对象查找文件并进行修改或者删除。执行修改或删除操作时，捕获RecoverableSecurityException，以便您可以请求用户授予您对多媒体文件的写入权限。

4.图片的**地理位置信息**读取适配：一些照片在其Exif元数据中包含位置信息，允许用户查看拍摄照片的位置。由于此位置信息是敏感的，因此默认情况下Android Q会对位置信息从MediaProvider数据库中删除，应用如果需要获取位置信息，请完成以下步骤：

- 将新的ACCESS_MEDIA_LOCATION权限添加到您应用的AndroidManifest.xml中。(如果应用是默认的图库应用，不需要申请该权限)
- 从您的MediaStore对象，调用setRequireOriginal()，传入照片的URI。

```
RoleManager roleManager = getSystemService(RoleManager.class);
if (roleManager.isRoleAvailable(RoleManager.ROLE_MUSIC)) {
    if (roleManager.isRoleHeld(RoleManager.ROLE_MUSIC)) {
        // This app is the default music app.
     } else {
        // This app isn't the default music player, but the role is available,
        // so request it.
        Intent roleRequestIntent = roleManager.createRequestRoleIntent(
                RoleManager.ROLE_MUSIC)
        startActivityForResult(roleRequestIntent, ROLE_REQUEST_CODE)
    }
}

@Override
protected void onActivityResult(int requestCode, int resultCode,
                                @Nullable Intent data) {
    if (requestCode == ROLE_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            // Your app has the role that you requested.
        }
    }
}
```

多媒体类型	Intent过滤器	Role类型
图片和视频	android.intent.category.APP_GALLERY	RoleManager.ROLE_GALLERY
音乐	android.intent.category.APP_MUSIC	RoleManager.ROLE_MUSIC

应用存储空间限制-兼容性影响分析

问题8：新安装应用卸载，应用数据清空问题

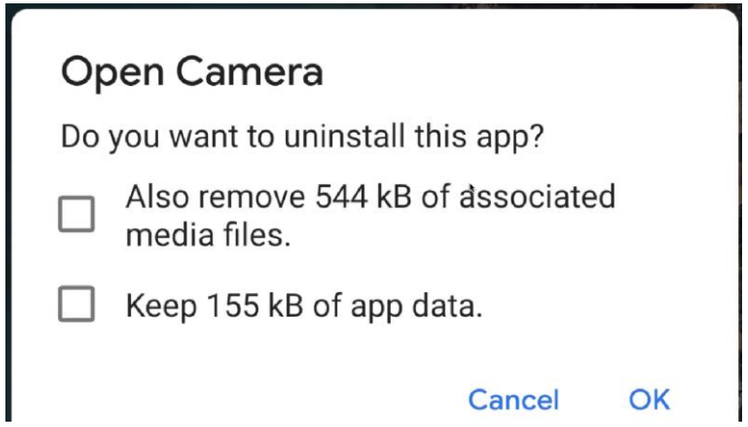
1.问题分析

- 通过测试发现应用在sd卡根目录创建应用自己的目录保存用户的文件的做法是很普遍的；
- P版本应用在sd卡根目录保存的应用数据卸载时不会被清楚，但是Q版本这部分数据实际保存的是应用的沙箱目录下，卸载的时候回全部被清除；

2.适配指导：

- 应用不想卸载删除的文件通过SAF或者是MediaProvider的接口保存在公共目录，不要放在应用的沙箱目录，公共目录文件卸载删除默认会弹框提示，对应右边弹框的第一个勾选，默认保留，勾选删除，**谷歌后续的版本计划把这个勾选去掉，意味着应用保存到公共集合目录的文件卸载的时候不会提示用户删除**；
- 对于应用的沙箱文件，应用如果不想卸载的时候被删除，需要应用在manifest文件增加：<application android:fragileUserData=”true” />，这样卸载应用的时候，系统弹出的对话框中才会有第二个勾选框出现（不增加该属性是不会有第二个勾选出现），默认删除，勾选保留。

应用分类	应用名称	应用版本	用户数据生成场景	用户数据内部存储保存路径（手机版本：ELLE-AL00-100SP1）	卸载应用后下载文件或保存数据是否被删除
通讯社交	微信	7.0.3	保存好友发送的图片、视频、文档	内部存储/Tencent/MicroMsg/WeiXin	否
	QQ	7.9.8		视频保存路径：内部存储/DCIM/Camera	否
拍照类	BeautyCam美颜相机	8.2.40	拍摄照片、录制视频	图片视频都保存在路径：内部存储/DCIM/Camera	否
	美图秀秀	8.4.2.0		二、拍照后点击存到美图相册：	/内部存储/DCIM/Camera路径下不会删除
	Faceu激萌	4.6.9		跟拍录制视频：	/内部存储/DCIM/Camera路径下不会删除
视频类	爱奇艺	10.1.5	缓存离线视频	内部存储/Android/data/com.qiyi.video/files/app/download/video	是
	优酷	7.6.4		内部存储/youku/offlinedata	否
	腾讯视频	6.7.0.18223		内部存储/Android/data/com.tencent.qqlive/files/videos_mdzUS	是
音乐类	网易云音乐	5.9.0	下载音乐	内部存储/netease/cloudmusic/Music	否
	QQ音乐	8.9.6.13		内部存储/qqmusic/song/	否
	酷狗音乐	9.1.5		内部存储/kgmusic/download/	否
导航类	高德地图	8.85.0.2275	下载离线地图	内部存储/autonavi/data/navi/comPile_v2/chn/	否（重装应用后，已下载的离线地图应用内还显示已下
	百度地图	10.13.3		内部存储/BaiduMap/offline/	否（重装应用后，已下载的离线地图应用内还显示已下
	腾讯地图	8.4.0		内部存储/SOSOMap/data/v3/	否（重装应用后，已下载的离线地图应用内还显示已下
购物类	手机淘宝	8.4.0	保存商品图片	内部存储/Pictures/taobao/	否
	苏宁易购	7.4.9		内部存储/Pictures/suning/	否
	蘑菇街	11.6.3.12953		内部存储/mogujie/transformer/mogujie/	否
听书类	喜马拉雅	6.5.57.3	下载书语音包	内部存储/Android/data/com.xumalaya.ting.android/files/download/	是
	蜻蜓FM	8.2.7		内部存储/QTDownloadRadio	否
	追书神器	4.36		内部存储/ZhuiShuShenQi/Chapter/	否
办公	WPSOffice	11.4.3	新建word、excel、PPT等	路径自定义	否
	网易邮箱	6.12.2	下载邮件中图片、视频、文件等	视频：路径保存时可自定义	否
	QQ邮箱	5.5.8	下载邮件中图片、视频、文件等	（路径可自定义）	否
漫画和小说	书旗小说	10.7.7.75	下载书	内部存储/shuqi/shuqi/chaptercache/1245479984	否
	网易漫画	4.6.4	下载漫画	内部存储/Android/data/com.netease.cartoonreader/book/	是
	微博动漫	7.5.0		内部存储/wbcomic/wbdownload/	否
直播类	抖音短视频	5.1.1	下载短视频	内部存储/DCIM/Camera	否
下载类	迅雷	5.71.2.5890	下载小视频	内部存储/Android/obb/com.xunlei.downloadprovider/ThunderDownload/	是
	QQ浏览器	9.0.2.4800	下载文件（图片、安装包等）	安装包：内部存储/QQBrowser/安装包	否
	UC浏览器	12.2.8.1008		内部存储/UCDownloads/	否
存储类	百度网盘	9.6.1	下载文件（图片、音乐）	默认下载位置：内部存储/BaiduNetdisk/	否
	微盘	3.4.15	下载文档	内部存储/微盘/	否
	UC浏览器	12.2.8.1008	UC浏览器功能内UC网盘功能中下载图片	内部存储/UCDownloads/CloudDrive/	否



应用存储空间限制-兼容性影响分析

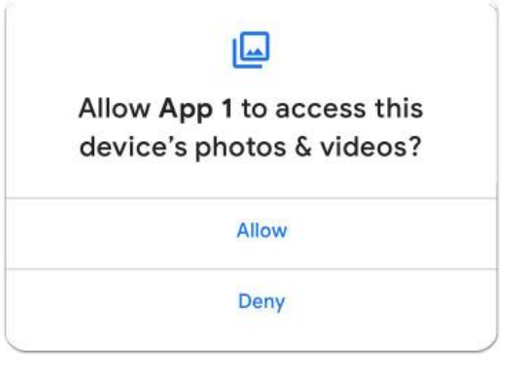
问题9：Hota升级不受沙箱影响的应用卸载残留数据，重新安装之后没有权限读写的问题。

1.问题分析

- hota升级到Q版本之前已经安装的应用，并且用户授权应用存储权限，升级到Q版本将豁免沙箱影响，沙箱特性不生效；
- 应用卸载重新安装之后，会受沙箱特性影响，所以卸载之后保留的用户的文件应用没有权限再通过路径读取，必须要通过SAF或者MediaProvider的方法读取；

2.适配指导：

- 应用需要考虑历史数据迁移，希望重新安装找回来的文件通过谷歌的SAF或者MediaProvider的接口迁移到公共目录；

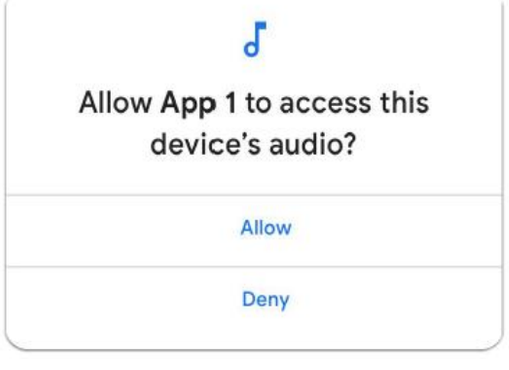


问题10：存储权限变更问题。

1.因为对于TargetSdkVersion<Q的应用，谷歌提供了兼容方案，所以不适配也不会有权限问题。

2.TargetSdkVersion>=Q的应用，需要适配增加新的存储权限申请，否则会报权限问题：

- <uses-permission android:name="android.permission.READ_MEDIA_AUDIO" />
- <uses-permission android:name="android.permission.READ_MEDIA_IMAGES" />
- <uses-permission android:name="android.permission.READ_MEDIA_VIDEO" />



禁止应用读取设备标识符信息

1.特性介绍

禁止获取device id：

- 新权限：READ_PRIVILEGED_PHONE_STATE（需要系统签名才能申请）
- 老权限：READ_PHONE_STATE

Mac地址随机化：

- P版本没有默认开启，需要通过开发者选项手动开启
- Q版本默认开启

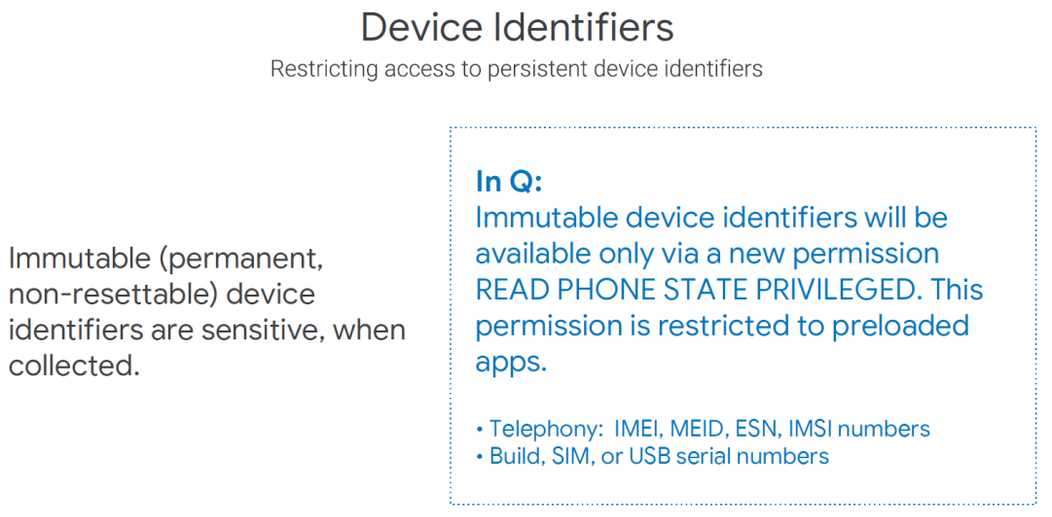
2.兼容性影响

- TargetSdkVersion<Q并且没有申请READ_PHONE_STATE，或者TargetSdkVersion>=Q，获取device id会抛异常SecurityException
- TargetSdkVersion<Q并且申请了READ_PHONE_STATE，通过getDeviceId接口读取的值为Null
- 通过Build和Build.getSerial()返回的值为“Unknown”
- 90%+的应用都有影响，影响很大，金融类的应用重点测试

3.适配建议

- 指导链接： https://developer.android.com/training/articles/user-data-ids
- 其他可替代方案：通过Android ID替代device id

```
String ANDROID_ID =
Settings.System.getString(getContentResolver(),Settings.Secure.ANDROID_
ID);
```



禁止应用后台弹页面

1.特性介绍

- 目前版本只是弹toast提示开发者需要整改的页面，未开启禁止功能，需要开发者自查和整改，后面的版本会真的禁止
- Beta1版本功能使能方法：系统设置->开发者选项->允许系统执行后台活动

2.影响场景

- 闹钟
- 音乐锁屏
- 语音电话、视频电话

3.适配指导

- 使用通知替代后台直接弹页面，具体可参考谷歌提供的适配指导：

https://developer.android.google.cn/preview/privacy/background-activity-starts

请使用高优先级通知，并提供一个全屏 intent：

```
Intent fullScreenIntent = new Intent( packageContext: this, DownloadProviderTestActivity.class);
PendingIntent fullScreenPendingIntent = PendingIntent.getActivity( context: this, requestCode: 0, fullScreenIntent, PendingIntent.FLAG_UPDATE_CURRENT);
Notification notification = new NotificationCompat.Builder( context: this, channelId: "upgrade")
        .setSmallIcon(R.drawable.if_apple_2003193)
        .setContentTitle("Incoming call")
        .setContentText("(919) 555-1234")
        .setPriority(NotificationCompat.PRIORITY_HIGH)
        .setCategory(NotificationCompat.CATEGORY_CALL)
        .setFullScreenIntent(fullScreenPendingIntent, highPriority: true).build();
NotificationManager mNotificationManager =
        (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
// mId allows you to update the notification later on.
mNotificationManager.notify( id: 1, notification);
```

后台地理权限

1.特性介绍

- Q之前只有ACCESS_FINE_LOCATION和ACCESS_COARSE_LOCATION
- Q版本ACCESS_FINE_LOCATION和ACCESS_COARSE_LOCATION只代表仅前台位置使用权限，另外新增了ACCESS_BACKGROUND_LOCATION权限，代表始终位置使用权限，如果需要始终使用位置信息，需要申请这个权限
- 应用的TargetSdkVersion<Q，会默认请求ACCESS_BACKGROUND_LOCATION权限，但是用户可以拒绝，可以选择仅前台使用
- 前台定义：应用有可见的Activity或者前台服务
- 如果用户选择仅前台使用允许，应用的页面退后台，通过启动前台服务让应用处于前台状态，必须把前台服务标为：foregroundServiceType=“location”，才能获取位置信息。

2.影响场景

- 地图类应用的后台导航功能需要测试和适配

3.适配指导

- 谷歌提供的适配指导：https://developer.android.google.cn/preview/privacy/device-location
- 如果应用需要后台导航：因为O版本的“后台位置限制”管控，建议地图类的应用还是通过前台服务让应用处于前台状态，不受“后台位置限制”管控，并且前台服务增加foregroundServiceType=“location”，不受Q版本前台服务位置限制；
- 如果应用需要在后台定期获取地理讯息：检测若没得到 “始终允许”，先向用户提示说明权限的出发点，让用户主动选择“始终允许”。另外前台服务不需加上foregroundServiceType的属性。但是可能的问题就是应用切后台会受到“后台位置限制”，只能30分钟访问一次位置信息，所以如果有这种定期检查位置信息的需求建议采用这个方式；
- 如果您的应用不需要后台定位，最佳做法是升级sdkversion到Q，并且不要申请后台定位权限

```
<service
    android:name="MyNavigationService"
    android:foregroundServiceType="location" ... >
    ...
</service>
```

后台地理权限-适配指导

1.地图后台导航场景适配指导

- 可以不用申请后台定位权限：ACCESS_BACKGROUND_LOCATION
- 需要启动前台服务，并且要新增设置前台服务的foregroundServiceType，有两种设置方法：

一种是通过Manifest文件静态设置；

```
<service
    android:name="MyNavigationService"
    android:foregroundServiceType="location" ... >
    ...
</service>
```

另外一种是代码动态设置：

Service.startForeground(Notification notif, int serviceTypes);

- 启动前台服务之前先判断是否有前台定位的权限：

```
boolean permissionAccessCoarseLocationApproved =
    ActivityCompat.checkSelfPermission(this,
        permission.ACCESS_COARSE_LOCATION) ==
        PackageManager.PERMISSION_GRANTED;

if (permissionAccessCoarseLocationApproved) {
    // App has permission to access location in the foreground. Start your
    // foreground service that has a foreground service type of "location".
} else {
   // Make a request for foreground-only location access.
   ActivityCompat.requestPermissions(this, new String[] {
        Manifest.permission.ACCESS_COARSE_LOCATION},
       your-permission-request-code);
}
```

2.定期后台定位场景适配指导

- 需要申请后台定位权限：ACCESS_BACKGROUND_LOCATION
- 做好权限是否授予校验：检查用户有没有主动授予应用后台定位的权限，如果没有授予，说明原因，让用户主动授予，具体代码可参考：

```
boolean permissionAccessCoarseLocationApproved =
    ActivityCompat.checkSelfPermission(this, permission.ACCESS_COARSE_LOCATION)
        == PackageManager.PERMISSION_GRANTED;

if (permissionAccessCoarseLocationApproved) {
   boolean backgroundLocationPermissionApproved =
           ActivityCompat.checkSelfPermission(this,
               permission.ACCESS_BACKGROUND_LOCATION)
               == PackageManager.PERMISSION_GRANTED;

   if (backgroundLocationPermissionApproved) {
       // App can access location both in the foreground and in the background.
       // Start your service that doesn't have a foreground service type
       // defined.
   } else {
       // App can only access location in the foreground. Display a dialog
       // warning the user that your app must have all-the-time access to
       // location in order to function properly. Then, request background
       // location.
       ActivityCompat.requestPermissions(this, new String[] {
           Manifest.permission.ACCESS_BACKGROUND_LOCATION},
           your-permission-request-code);
   }
} else {
   // App doesn't have access to the user's location at all. Make full request
   // for permission.
   ActivityCompat.requestPermissions(this, new String[] {
        Manifest.permission.ACCESS_COARSE_LOCATION,
        Manifest.permission.ACCESS_BACKGROUND_LOCATION
        },
        your-permission-request-code);
}
```

启用和禁用Wi-Fi的限制

1.特性介绍

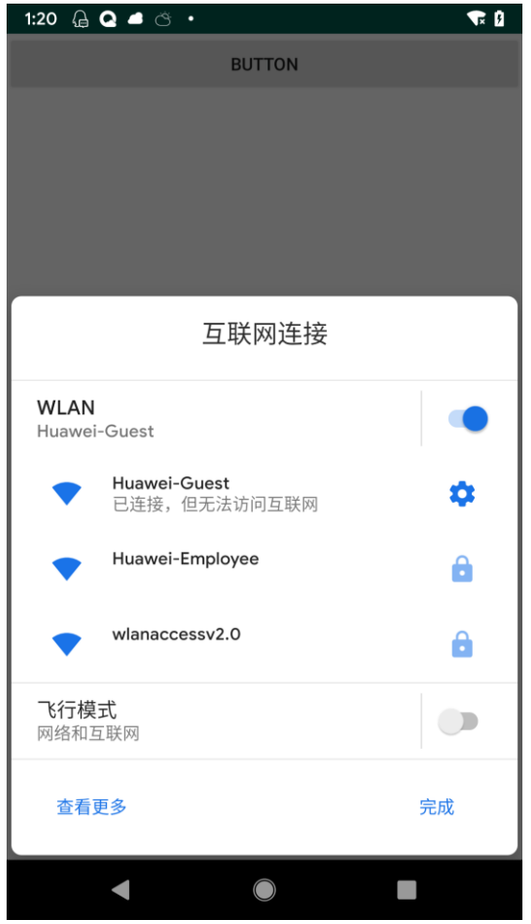
- 在Android Q上运行的应用无法启用或停用Wi-Fi。WifiManager.setWifiEnabled()方法始终返回false。
- 如果需要，请使用设置面板提示用户启用和禁用Wi-Fi。

2.影响场景

- 应用的Wifi自动启动和禁用功能

3.适配指导

```
Intent panelIntent = new Intent(Settings.Panel.ACTION_INTERNET_CONNECTIVITY);
startActivityForResult(panelIntent, 100);
```



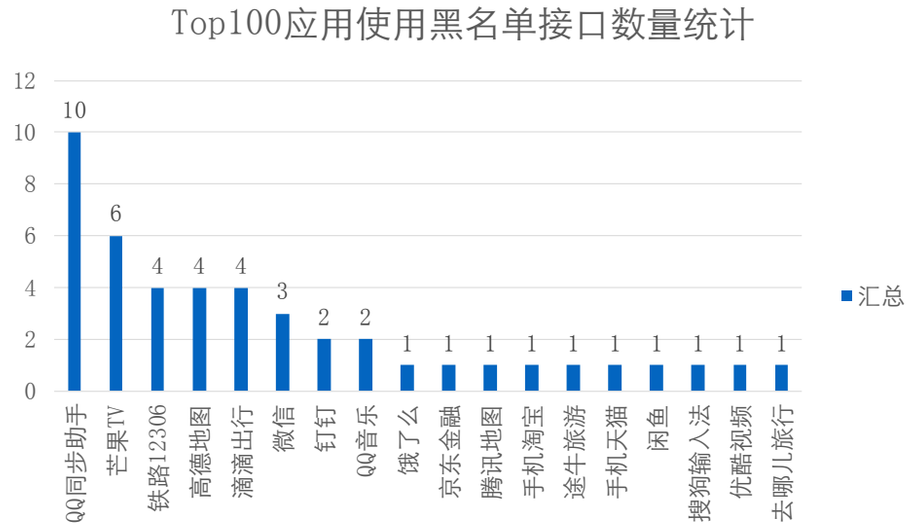
非SDK管控（Hidden API）

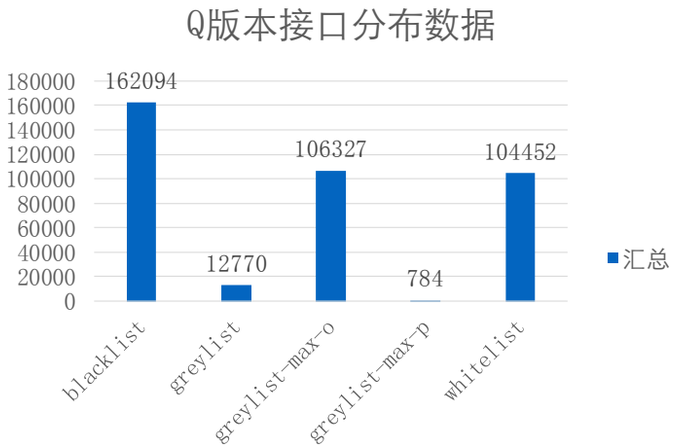
1.特性介绍

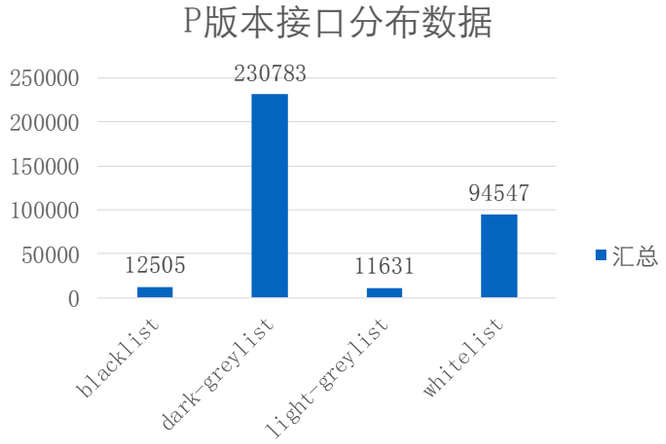
- Q版本新增max-o（应用的targetSdkVersion>O无法使用）和max-p（应用的targetSdkVersion>P无法使用）；
- 统计新增P版本的10w+个接口被加入了目前Q版本的黑名单；

2.适配建议：

- 谷歌提供的适配指导：https://developer.android.google.cn/distribute/best-practices/develop/restrictions-non-sdk-interfaces
- 使用谷歌提供的非SDK扫描工具查看应用使用的非SDK接口：https://android.googlesource.com/platform/prebuilts/runtime/+/master/appcompat/（备注：Q版本对应的的扫描工具已经更新，开发者可以下载扫描）
- 无法整改的接口说明详细使用场景和理由反馈给谷歌申请加灰名单：https://issuetracker.google.com/issues/new?component=328403&template=1027267







Top500应用扫描结果

名单类型	被使用的接口数
黑名单	236
max-o	149
max-p	45

名单类型	涉及的应用个数
黑名单	67
max-o	54
max-p	88

安装应用接口废弃

1.影响场景：

- 对TargetSdkVersion<24的应用有影响应用内安装应用，包括自更新和安装其他应用

TargetSdkVersion	Q之前	Q
<24	可以通过file:// URI安装	不可以通过file:// URI安装
>=24	不可以通过file:// URI安装，抛异常	不可以通过file:// URI安装

2.适配建议

- 需要申请android.permission.REQUEST_INSTALL_PACKAGES权限。
- 通过FileProvider使用content:// URI安装，代码如下：

```
Uri installUri = FileProvider.getUriForFile(getApplicationContext(),
    "com.huawei.qappcompatissues.fileprovider", apkFile);
Intent intent = new
    Intent(Intent.ACTION_INSTALL_PACKAGE).setData(installUri);
intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
startActivity(intent);
```

录音变更（并发录音、三方录音source管控）

1.特性介绍

- P版本录音焦点不可被抢占，Q版本录音焦点可以被抢占
- 敏感源取决于应用下发的source：AUDIO_SOURCE_CAMCORDER和AUDIO_SOURCE_VOICE_COMMUNICATION
- 抢占规则：只是限制不敏感音源无法抢占敏感音源，其他情况都可以随意抢占

2.影响场景

- 三方应用录音焦点被抢占录音数据为空。
- 应用的录音功能需要重点测试

录音变更适配指导

通过下面三个方式注册回调监听：

- AudioRecoder.registerAudioRecordingCallback(Executor executor, AudioManager.AudioRecordingCallback cb);
- MediaRecoder.registerAudioRecordingCallback(Executor executor, AudioManager.AudioRecordingCallback cb);
- AudioManager.registerAudioRecordingCallback(AudioRecordingCallback cb, Handler handler);

回调信息的使用：

```
new AudioManager.AudioRecordingCallback() {
    @Override
    public void onRecordingConfigChanged(List<AudioRecordingConfiguration> configs) {
        for (AudioRecordingConfiguration arc : configs){
            arc.getClientAudioSessionId(); \\ client的session ID,每个录音唯一
            arc.isClientSilenced(); \\ cilent是否静音
            arc.getAudioSource(); \\ client 使用的录音source
        }
    }
};
```

系统会回调当前所有的cilent信息， 只要任意一个cilent状态发生了增删改

1、使用 getClientAudioSessionId() 与 AudioRecord的 getAudioSessionId 进行对比，可以知道哪个cilent是应用自己的

2、使用 isClientSilenced() 可以知道是否被静音

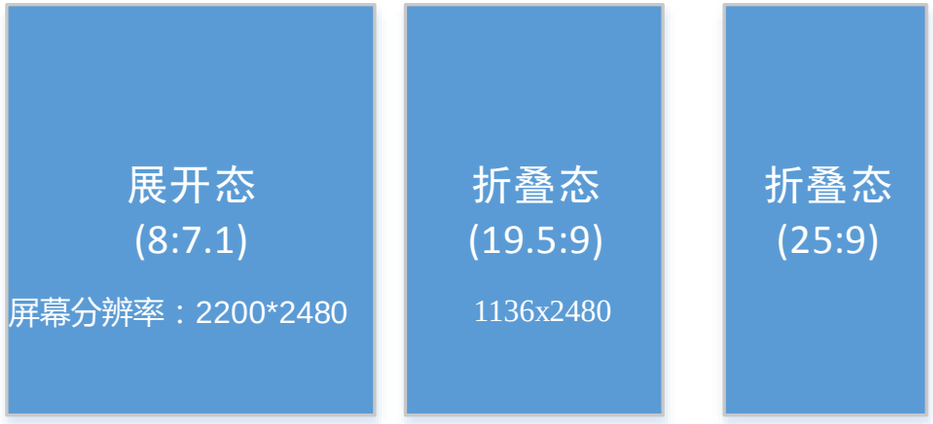
3、getAudioSource可以知道是否有voip、camera等录音场景，应用可以初步判断优先级

折叠屏

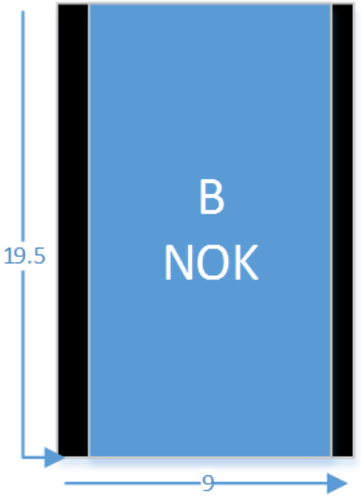
1.MATEX物理形态概述

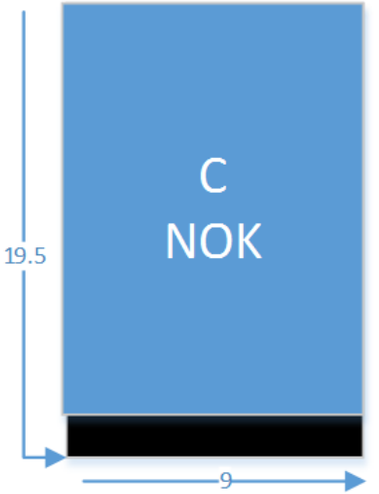
对于通用软件，可能存在如下形态，即：

- 展开态，全屏点亮工作
- 折叠主屏态：折叠态，主屏工作
- 折叠副屏态：折叠态：副屏工作



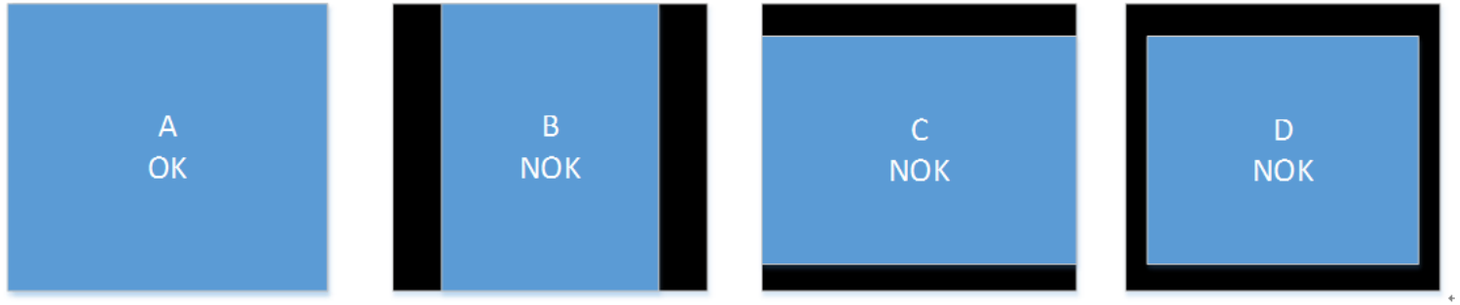
主屏规格定义：应用在主屏折叠下，可以撑满全屏显示，且在横竖屏切换形态下，布局和操作按键都正常，不出现任意方向的黑边。



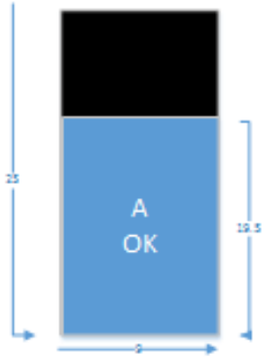


2.静态适配需求规格描述

大屏规格定义：应用在大屏展开下，可以撑满全屏显示，且在横竖屏切换形态下，布局和操作按键都正常，不出现任意方向的黑边。



副屏规格：25 ：9
副屏下，默认不以全屏显示，以主屏比例显示，即在副屏下也显示19.5:9居下显示。应用只需要做好19.5:9的适配即可，副屏以19.5:9 在屏幕下方显示由系统统一控制。

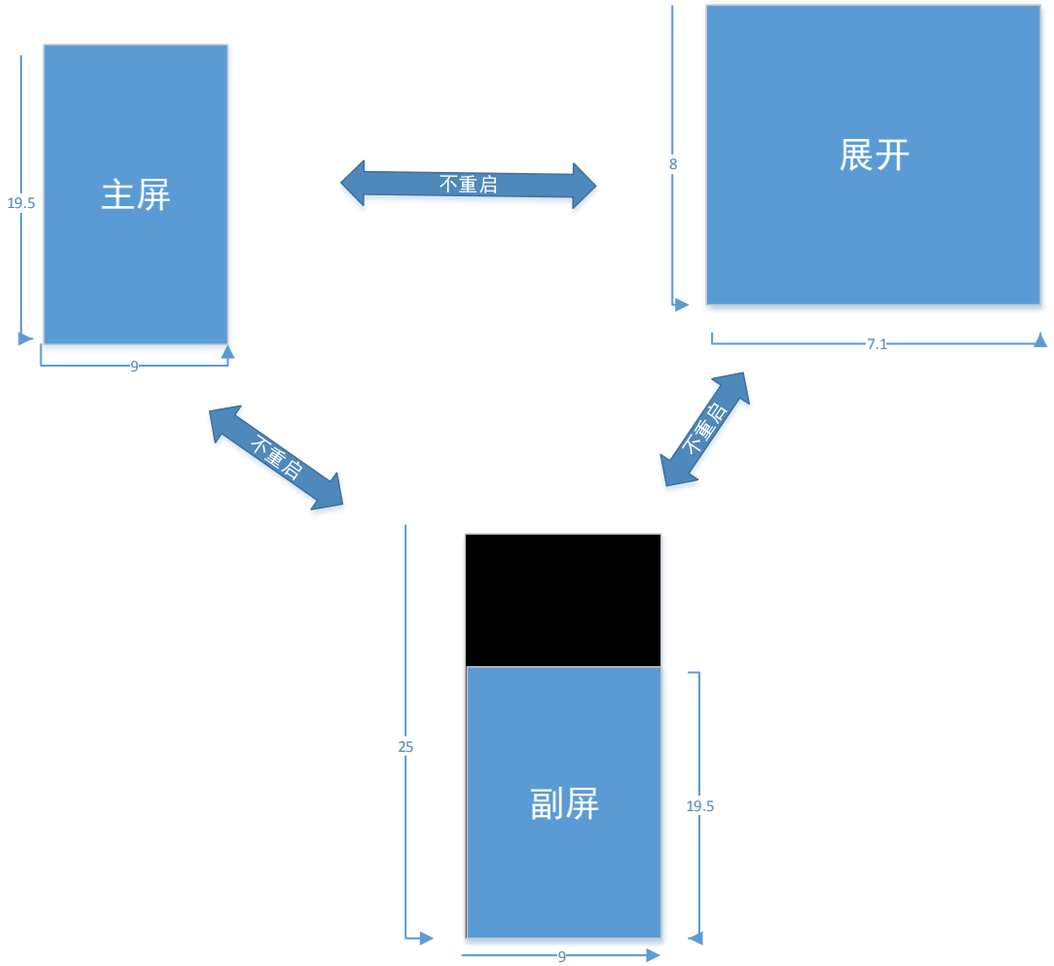


折叠屏

3.动态适配需求规格

- 任意形态下打开一个应用，满足以上静态显示规格
- 当用户物理形态切换时，应用需要做到不重启应用，并自动在新的设备形态下按照静态规格布局。

即：在用户进行展开，折叠等操作， 应用任务不中断，自动适应各种屏幕下的静态布局规格。



折叠屏-适配指导

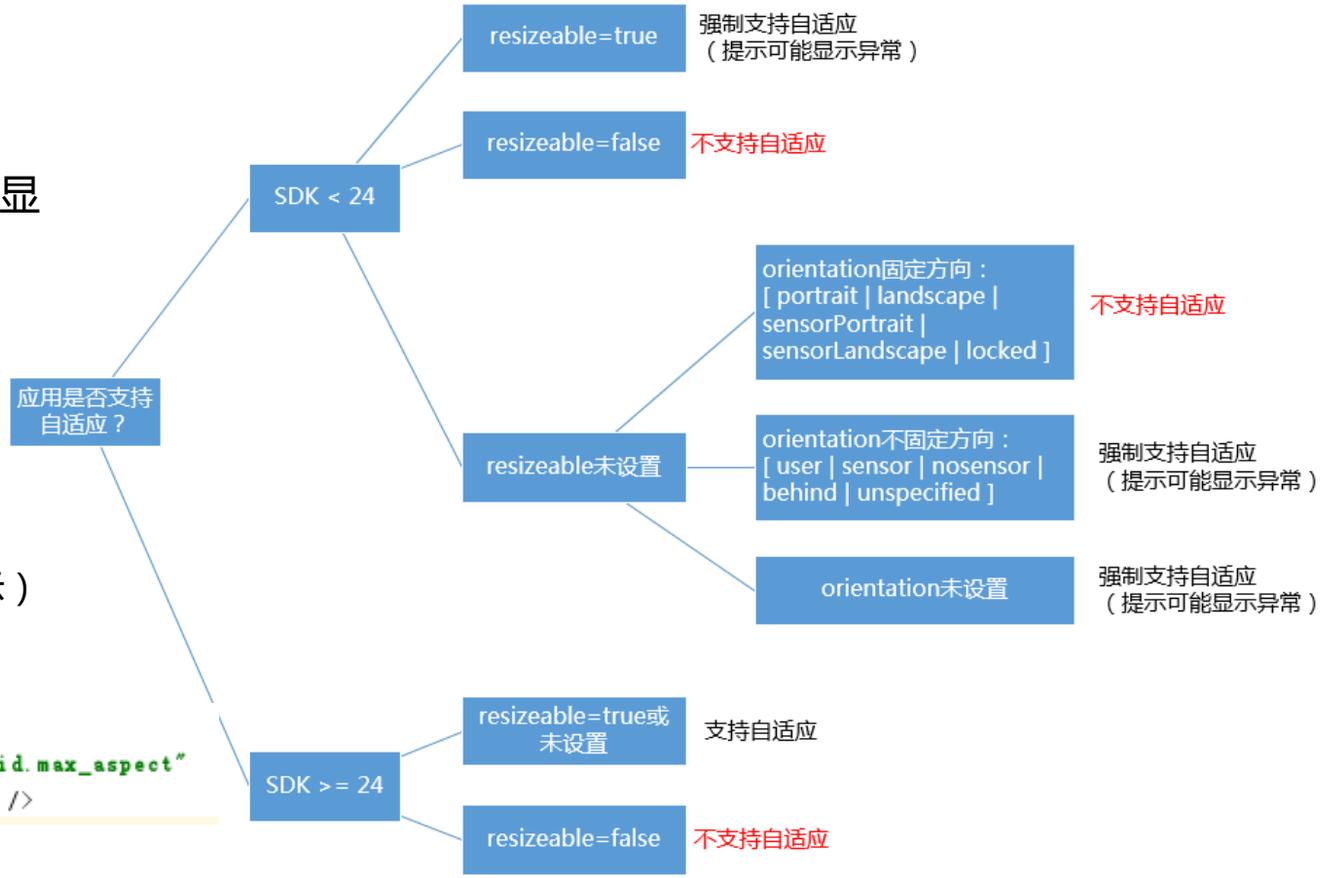
1.屏幕比例适配

推荐适配方式：应用支持自适应能力适配

在 manifest 文件的 <activity> 或 <application> 节点中设置 android:resizeableActivity 的值为true，可声明应用支持自适应显示，Activity 将能以分屏和 freeform 模式启动。

```
<activity android:name=".MainActivity" android:resizeableActivity="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>

        <category android:name="android.intent.category.LAUNCHER"/>
```



其他方式：设置应用支持的最大比例和最小比例适配

- 最大宽高比申明： maxaspecratio：2.4（2.4表明在主副屏下满屏显示）

```
<activity
    android:name=".MainActivity"
    android:maxAspectRatio="2.4">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>

        <category android:name="android.intent.category.LAUNCHER"/>
```

8.0及以上的版本才支持

```
<meta-data
    android:name="android.max_aspect"
    android:value="2.4" />
```

- 最小宽高比申明： minaspectratio：1.0 （1.0表示在展开态全屏显示）

Q以前的版本

```
<meta-data android:name="android.min_aspect" android:value="1.0" />
```

Q以后的版本

```
<activity android:minAspectRatio="1.0">
 ...
</activity>
```

折叠屏-适配指导

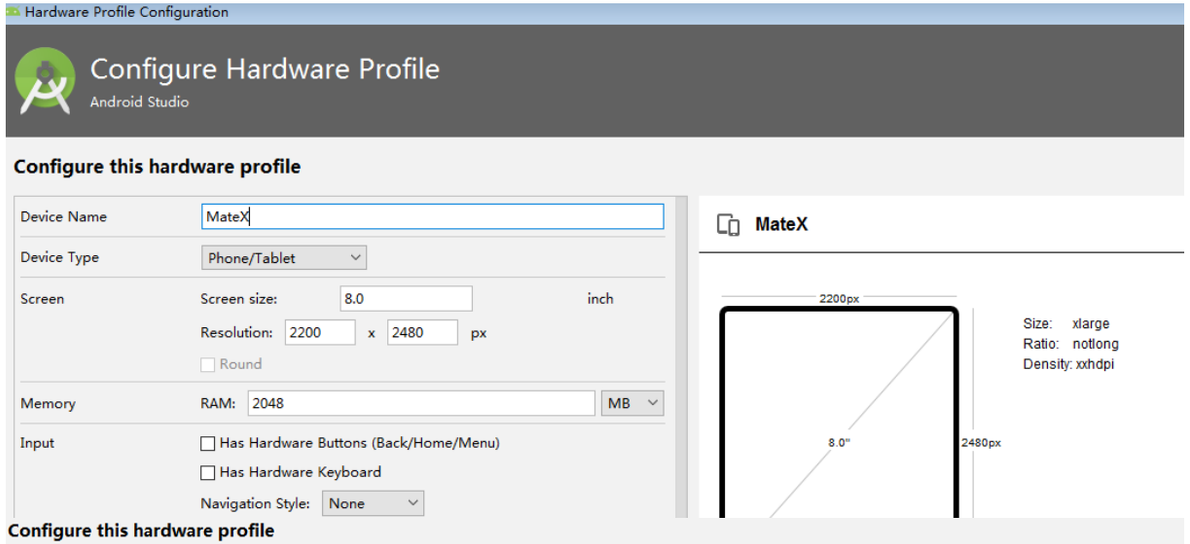
2.切换显示比例应用不重启适配

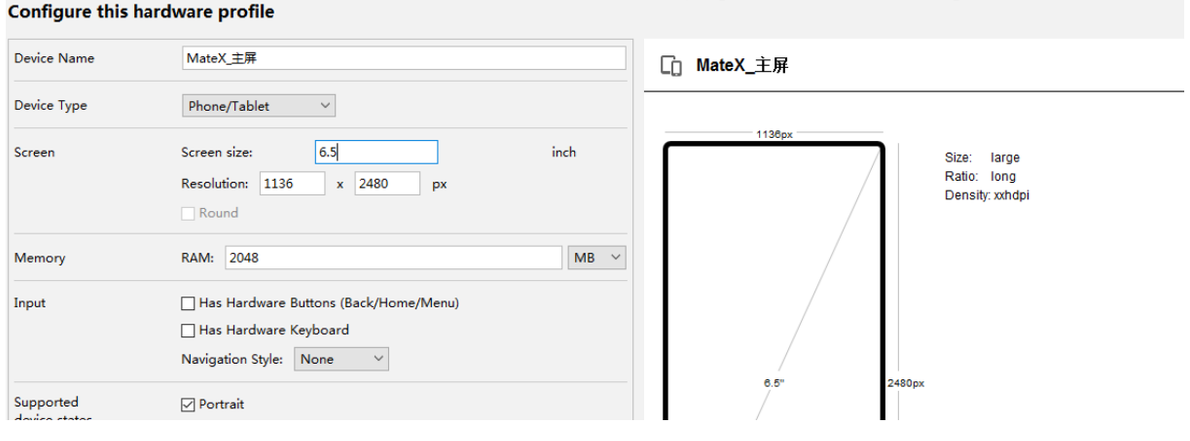
- 在 manifest 文件的 <activity> 节点中的 android:configChanges 属性增加 screenSize|smallestScreenSize|screenLayout字符串，当屏幕比例变化时，系统会回调 Activity 的 onConfigurationChanged() 方法，而避免 Activity 重新启动。

```
<activity
    android:name=".MainActivity"
    android:screenOrientation="portrait"
    android:configChanges="screenSize|smallestScreenSize|screenLayout">
```

- 通过接口newConfig.**screenHeightDp**和newConfig.**screenWidthDp**获取屏幕尺寸信息调整页面布局

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    Log.e( tag: "jiyun",  msg: "newConfig.screenHeightDp:" + newConfig.screenHeightDp
            + ", newConfig.screenWidthDp" + newConfig.screenWidthDp);
}
```





3.调试方法

- 模拟器调试，谷歌计划在beta2版本模拟器支持折叠屏模拟调试，目前只能通过设置不同的分辨率模拟器验证显示比例适配效果
- 非折叠屏真机模拟调试

切全屏：adb shell wm size 2200x2480

切主屏（副屏）：adb shell wm size 1136x2480

- 折叠屏真机调试

谢谢大家！

MAKE it POSSIBLE