Android中直播视频技术探究之---桌面屏幕视频数据源采集功能分析

# ****一、前言****

之前介绍了[Android](https://www.2cto.com/kf/yidong/Android/" \t "https://www.2cto.com/kf/201608/_blank)直播视频中一种视频源数据采集：摄像头Camera视频数据采集分析中介绍了利用Camera的回调机制，获取摄像头的每一帧数据，然后进行二次处理进行推流。现在我们在介绍另外一种视频源类型，屏幕桌面视频数据源，这个就是录屏功能，这个在Android中的利用场景现阶段也是很多的，比如像斗鱼这些直播app，在录制游戏的时候用的还是比较多的，因为现在的移动游戏也那么火，游戏视频直播也是一个很不错的产业！

# 二、技术方案

下面先来分析一下Android中录制屏幕采用的方案有哪些？现阶段Android中录制屏幕的话大致就两种方案：

**1、Android5.0以下，获取root权限使用adb shell screenrecord命令功能进行录制**

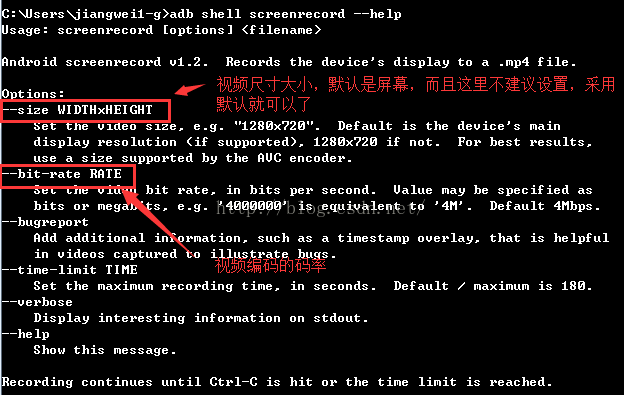
**2、Android5.0以上，使用**[系统](https://www.2cto.com/os/" \t "https://www.2cto.com/kf/201608/_blank)**Api功能：MediaProjection和VirtualDisplay**

这两种方式其实在底层的实现机制是一样的，因为在5.0之前，Google可能介于录制屏幕功能具有一定的危险性，所以就没有开放此功能，但是不知为何5.0开放了，当然开放了就立马产生了一个[漏洞](https://www.2cto.com" \t "https://www.2cto.com/kf/201608/_blank)，这个下一篇文章中介绍。从安全角度考虑，在屏幕截屏和录制功能不应该暴露出来，因为这个功能太危险了，之前有报道某某APP利用这个功能，在后台录制屏幕，然后传递到服务端，那么当一个用户在输入账号信息的时候，全部被录制了，那么就存在非常大的风险。

# ****三、使用命令功能录制视频****

下面先来介绍一下Android5.0一下录制屏幕功能

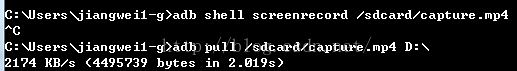
Android中在截屏时可以使用**adb shell screencap**命令，录制视频时可以使用adb shell screenrecord，先来看看这个命令的用法：



(1)、基本使用  
录制默认分辨率，默认4Mbps，默认180s的视频，保存到sdcard上名为FILENAME.mp4  
**$adb shell screenrecord /sdcard/FILENAME.mp4**  
  
(2)、指定分辨率(参数：--size)  
//分辨率为112x112,建议不要指定分辨率，使用默认分辨率效果最佳  
**$adb shell screenrecord --size 112x112 /sdcard/FILENAME.mp4  
注意：**分辨率不是完全可以随意定制的，比如在我手机上录制100x100的会提示错误：  
**$adb shell screenrecord --size 100x100 /sdcard/FILENAME.mp4**  
  
(3)、指定比特率(参数：--bit-rate)  
设置比特率为8Mbps，比特率越大，文件越大，画面越清晰  
**$adb shell screenrecord --bit-rate 8000000 /sdcard/FILENAME.mp4**  
  
(4)、旋转(参数：--rotate)  
旋转90度  
**$adb shell screenrecord --rotate /sdcard/FILENAME.mp4  
  
注意：**不支持声音，如有需要，另外独立录制。此命令需要root权限才可以使用，这个也是已非常大的局限性！

下面来看一下案例：

使用**adb shell screenrecord /sdcard/capture.mp4** 命令录制视频存到SD卡中，然后在使用adb pull导出到本地即可



看看效果：



功能我们可以看到了，可以发现其实这个功能测试用的比较多，他们在测试的时候，有时候复杂的测试场景难以重现，就可以使用录制屏幕功能，来跟踪问题。

那么现在如果一个应用获取到了root功能，就开始偷偷的在后台录制视频，然后把录制之后的视频文件发到服务端，然后进行分析，那么是非常危险的。

虽然这个功能可以了，但是我们发现，在直播的时候，如何进行数据流采取然后推流到服务端呢？因为在视频采集推流的时候，都是获取视频流数据，在底层使用rtmp进行数据打包发送服务端，那么这里有人会想，能不能一边录制视频到本地文件，然后一边在去读取这个文件流字节，进行推流？其实这种想法是好的，但是没有那么做的，因为这效率问题，其次是文件读写不同步问题，这里有一个思路就是，看看adb screenrecord命令的[源码](https://www.2cto.com/ym/" \t "https://www.2cto.com/kf/201608/_blank)了，因为他最终是采样数据写到本地文件的，那么在这个之前我们可以进行获取视频流即可，但是这个不是本文的重点，因为这个要是介绍的话，那就太多了，而且随着Android系统提升，未来用的多的是Android系统更高版本，所以这里要注重介绍的是5.0加的MediaProjection类来进行屏幕录制。

# ****四、使用开放Api进行录制视频****

Android5.0中添加了一套Api来进行屏幕录制功能，就是MediaProjection+VirtualDiaplay，用法也是非常简单的，在使用之前需要申请一下权限，然后获取权限之后，就开始使用即可！

## ****第一、使用流程介绍****

下面来看看使用流程：

1》使用getSystemService(Context.MEDIA\_PROJECTION\_SERVICE)方法获取MediaProjectionManager对象

2》开启授权Intent：startActivityForResult(mProjectionManager.createScreenCaptureIntent(),PERMISSION\_CODE);

3》授权之后在onActivityResult方法中进行处理：



这里看到了，利用返回的resultCode和data数据，得到MediaProjection对象，在注册一个回调接口，这个接口就是监听录制的状态信息，最后在创建一个VirtualDisplay

4》创建VirtualDisplay虚拟画面



参数说明：

\* 第一个参数：虚拟画面名称  
\* 第二个参数：虚拟画面的宽度  
\* 第三个参数：虚拟画面的高度  
\* 第四个参数：虚拟画面的标志  
\* 第五个参数：虚拟画面输出的Surface  
\* 第六个参数：虚拟画面回调接口

其中最重要的就是第五个参数，录制画面输出的地方，他这里介绍的是一个Surface类型，那么我们如果想介绍录制之后的视频数据，就需要创建一个Surface类型即可。在这里有三个方式创建Surface类型：

1》如果想截屏，那么可以使用ImageReader类的getSurface方法获取

2》如果想录制视频进行编码，可以使用MediaRecorder类的getSurface方法获取

3》最后如果想录制视频进行编码，并且获取视频流的话，使用MediaCodec的createInputSurface方法获取

这里我们先介绍前面两个功能，后续的那个功能再详细介绍！

# ****第二、截图功能介绍****

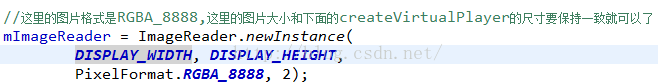
先来看看截图功能，其实在Android中截图大体上就三种方式：

**第一种使用adb shell screencap命令，但是需要root权限**

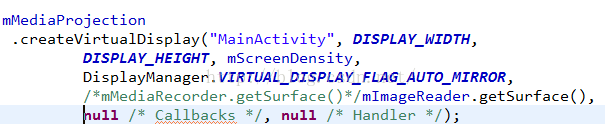
**第二种使用getWindow().getDecorView()获取当前Activity页面的View数据，然后进行保存Bitmap，但是这种方式有限制，一个是他只能截取当前应用的图，而且截取的图片不包括状态栏。**

**第三种使用MediaProjection和ImageReader来进行截图**

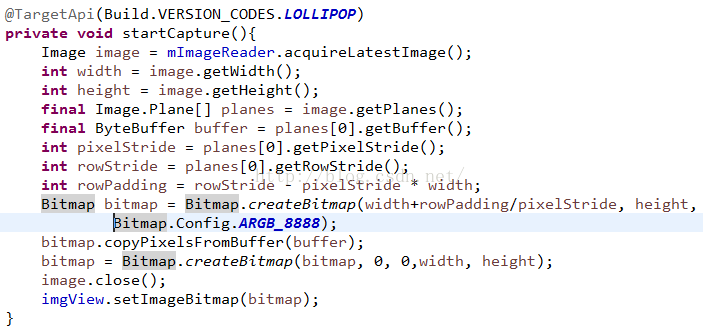
下面就来详细介绍如何使用MediaProjection来进行截图



先声明一个ImageReader类，参数比较简单，不过要注意的是这里的尺寸高宽要和下面创建VirtualDisplay一直：



然后把ImageReader的getSurface设置成画面的输出即可。接下来就开始截图：



从ImageReader中获取Image对象，然后得到图片字节数据，生成一个Bitmap即可。

下面来看看效果：



这里把状态栏也成功截取了！到这里就实现了MediaProjection的第一个功能实现屏幕截图功能。

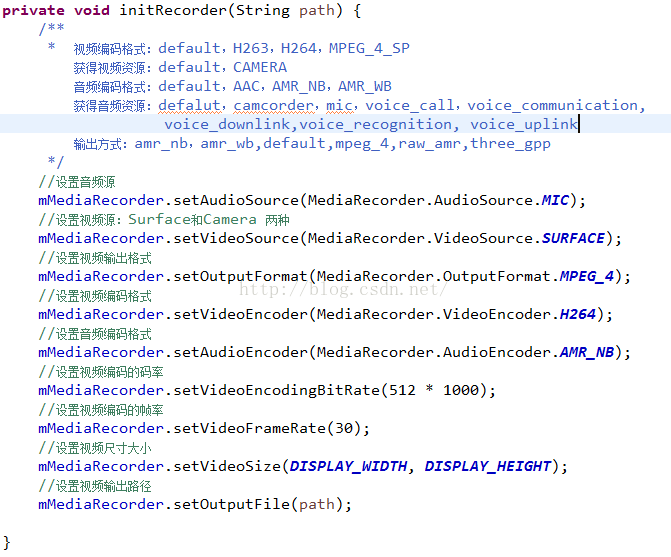
## ****第三、屏幕录制功能介绍****

这里先使用MediaRecorder类进行数据的编码工作，而我们真正要进行推流的话需要使用MediaCodec，因为我们需要获取到编码之后的视频流数据，而MediaRecorder是做不到这点的，但是本文只是先介绍录制功能，下一篇文章在详细介绍MediaCodec类功能！

使用MediaRecorder类很简单，以及后面介绍MediaCodec都一样，首先是初始化设置编码器的参数信息，而这些参数信息就那些：

视频尺寸高宽、视频帧率，视频采样率(码率)，视频编码格式等

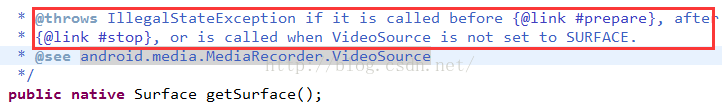
但是MediaRecorder还需要一些其他设置，比如设置视频来源，这里一般就两个来源，一个是摄像头Camera，一个是Surface类型，还有就是视频的输出格式，以及视频的输出文件路径等，当然MediaRecorder还支持音频录制，参数设置和视频差不多：



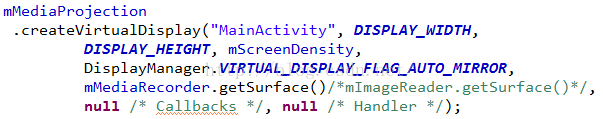
这里有一些默认格式参数选择：

**视频编码格式：default，H263，H264，MPEG\_4\_SP  
获得视频资源：default，CAMERA  
音频编码格式：default，AAC，AMR\_NB，AMR\_WB  
获得音频资源：defalut，camcorder，mic，voice\_call，voice\_communication,voice\_downlink,voice\_recognition, voice\_uplink  
输出方式：amr\_nb，amr\_wb,default,mpeg\_4,raw\_amr,three\_gpp**

初始化完了MediaRecorder之后，需要在调用prepare方法做准备，不然后续再调用getSurface会报错的：



这个方法的时候总是遇到这种错误，大致原因就是这个方法必须在prepare方法和stop方法中间调用，或者是视频源必须是Surface类型的。下面再来看看创建VirtualDisplay类：



调用getSurface方法获取到VirtualDisplay的输出画面。然后就需要调用开始编码方法start即可。

下面来看一下效果：



首先先需要系统进行授权，点击立即开始，就开始录制编码了，这里把编码之后的文件保存到sd中，我们在使用adb pull命令导出来，效果这里就不演示了，和上面的adb shell screenrecord命令产生的效果一样！

# ****五、技术概要****

1、介绍adb shell screenrecord命令录制视频，但是有局限性就是需要root权限

2、介绍了MediaProjection+ImageReader类进行截图功能

3、介绍了MediaProjection+MediaRecorder类进行视频录制功能

# ****六、遗留问题****

采用MediaProjection和MediaRecorder录制视频，或者是命令录制都是保存到本地文件了，但是在真正推流的过程中需要获取到视频流数据，所以这种方式不适合后续的推流工作，下一篇文章就要介绍如何MediaCodec进行视频编码，这个编码类是可以获取到编码之后的视频流数据的

# 七、总结

Android中在5.0之后放出了录制屏幕功能的api,但是其安全性还是值得考虑的，后续一篇文章就详细介绍一个关于MediaProjection的漏洞，导致用户的屏幕操作信息可能被泄露，而这些信息被泄露那么就代表这用户的账号信息存在很大的风险，但是既然有着api了，那么今年又是直播年，在录制游戏这个产业中，这个功能也是无可厚非的重要！