锐动 Android 视频编辑 SDK

锐动天地
www.17rd.com

作者：RD Android Team

版本：Beta 版

日期：2017-4-26

1 编写目的

预期读者：

有视频开发经验或者无经验的，打算或者正在使用“锐动 Android 视频编辑 SDK”的相关工程师。

- Android 软件工程师。
- 产品经理。
- QA

2 名词解释

- 分辨率：用于计算机视频处理的图像，以水平和垂直方向上所能显示的像素数来表示分辨率。常见视频分辨率的有 1080P 即 1920x1080，720P 即 1080x720，640x480 等。
- 宽高比：视频分辨率的宽高比，常见的有 16:9，4:3，1:1。锐动视频编辑 SDK 对各宽高比的视频都支持编辑，导出的默认分辨率是 640x360，宽高比是 16:9。
- 帧率：每秒的帧数(fps)或者说帧率表示图形处理器处理场时每秒钟能够更新的次数。
- 码率： 数据传输时单位时间传送的数据位数,一般我们用的单位是 kbps 即千位每秒。
- 素材：来自系统相机，其他 app，电脑，网络的照片，音乐，视频等。
- 视频分割： 把视频分割为若干个片段，可用于从视频中截取一个或者多个精彩瞬间，或者删除不喜欢的片段，分割后的多个片段也可以用于重新调整顺序，视频合并等。
- 视频裁切： 从视频画面中裁切出来仅需要的部分，锐动视频编辑 SDK 针对手机竖

屏拍摄的画面，支持动态调整裁切区域，帮助用户不同时间点选择自己需要的画面。

- 视频旋转：对视频进行 90°，180°，270°旋转，一般用于矫正用户手机上录制的头像横置的视频。
- 视频镜像：对视频左右镜像。
- 视频变速：对视频播放速度调整，实现慢镜头或者快镜头效果。
- 视频配乐：选取本地或者网络音乐作为视频的背景音乐。
- 视频配音：通过 mic phone 对视频配音。
- 视频字幕：使用文字标注视频。
- 视频滤镜：调整视频的画面颜色效果。
- 视频特效：在视频里面增加特效动画或声音
- 合并转场：两个视频片段之间的衔接效果。
- 主题（模版）： 应用视频的整体效果方案。

3 集成步骤

3.1 运行环境

- Android 4.1（api 16）以上；
- 处理器：双核 1GHz 以上 CPU(目前只支持 ARM CPU, X86、MIPS 暂不支持)；

  推荐四核 1.2GHz 以上 CPU
- 内存：1 GB 以上；

3.2 注册申请 AppKey 和 APPSECRET

1、 登录 http://www.rdsdk.com 注册用户

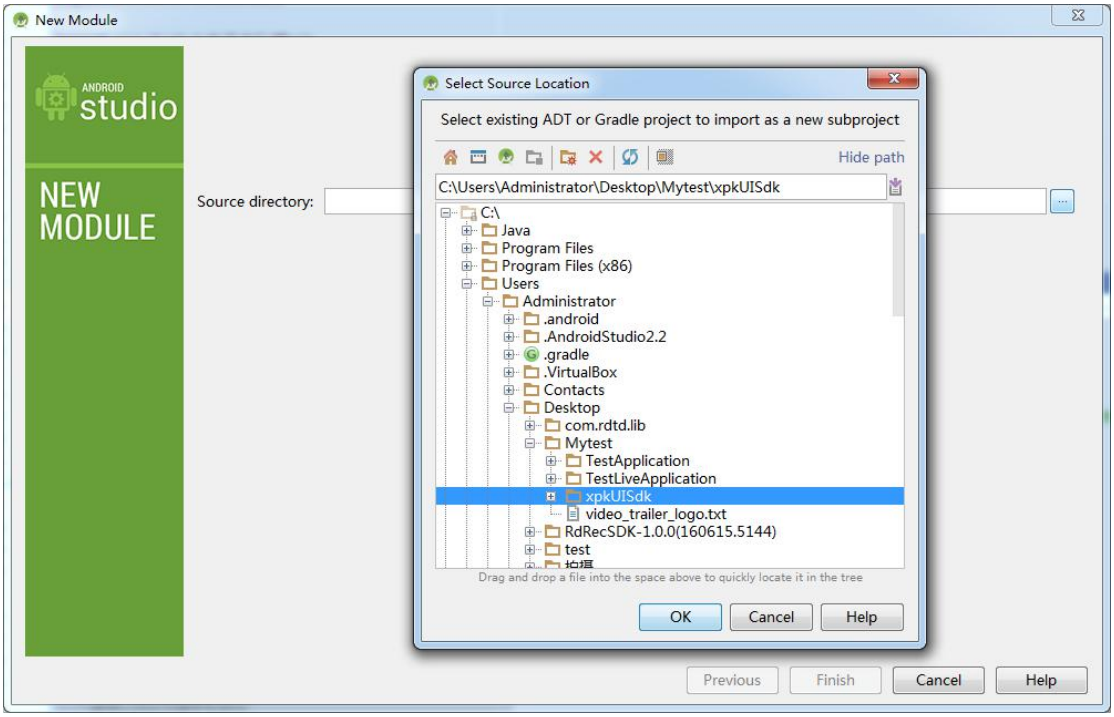
2、 登录注册好的用户

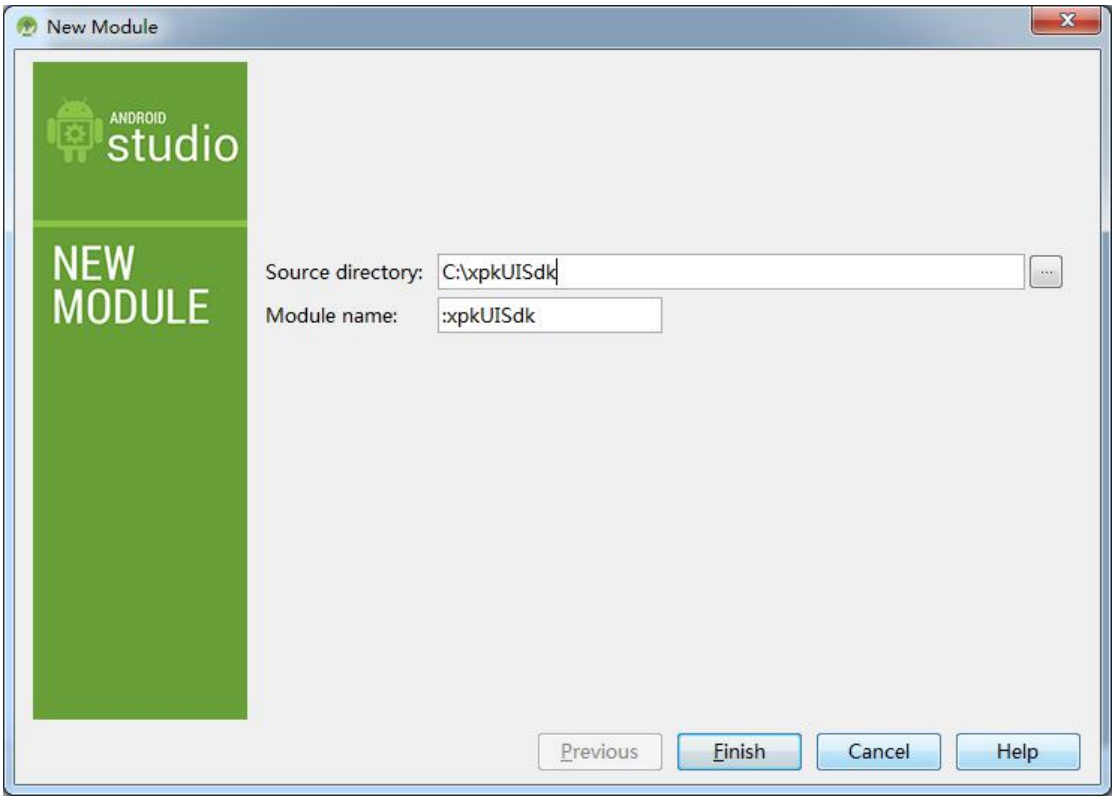
3、 进入视频云管理 点击（新增）获取应用的 appkey 、appsecret

3.3 下载并导入 SDK

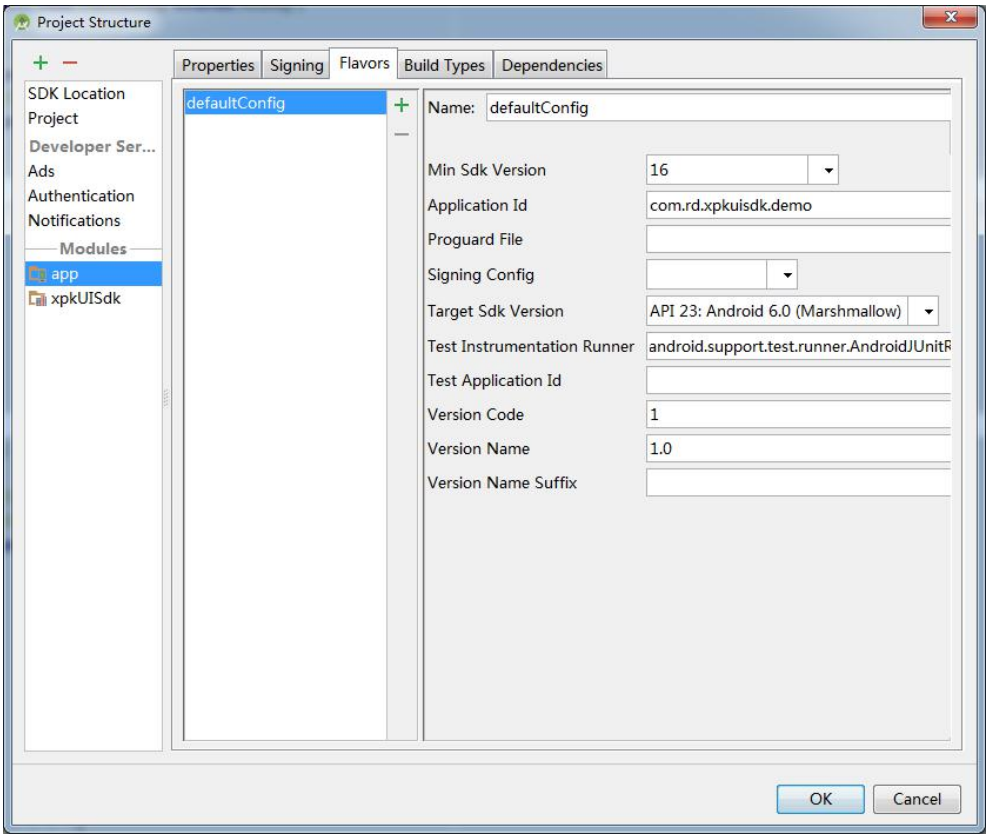
3.3.1 Android Studio 导入 xpkUISdk

点击 File--->Import Module，选择路径，填写“Module name”，默认为 xpkUISdk，点击“finish”。

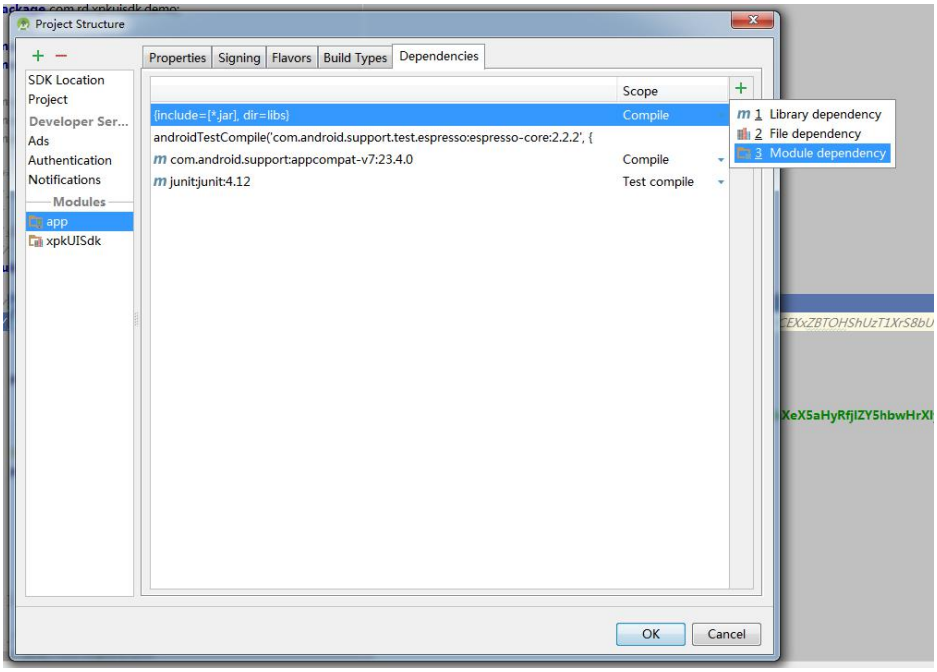




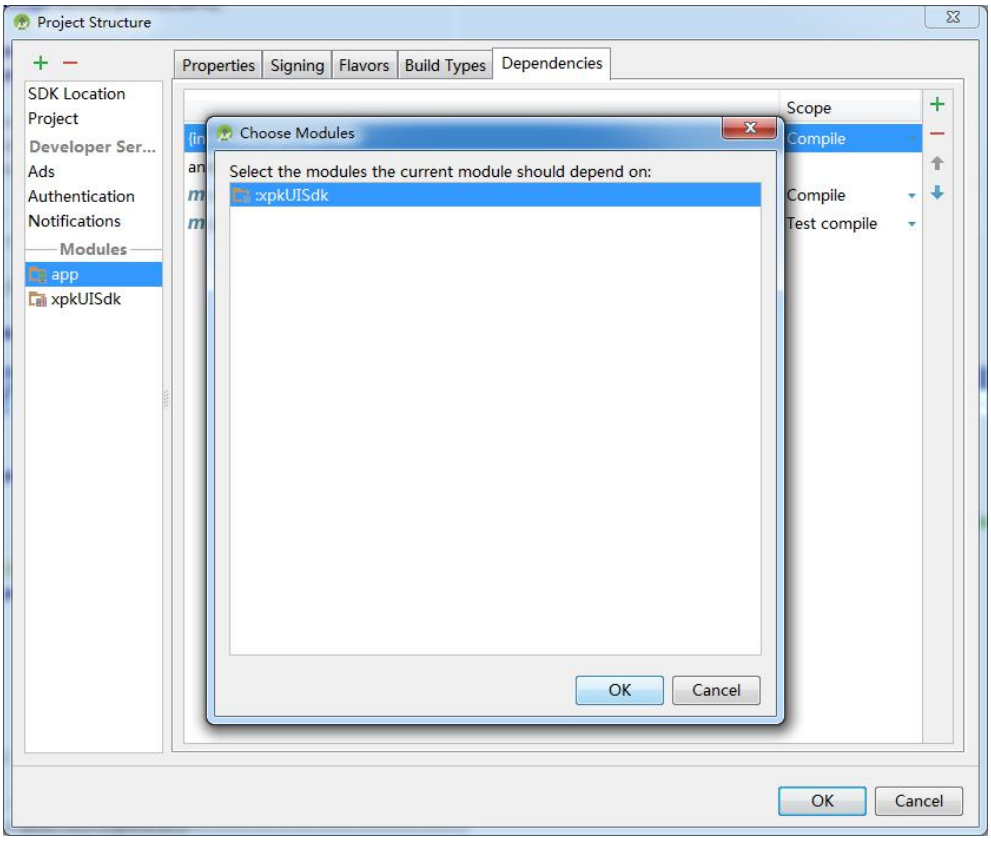
之后点击 File--->Project Structure，选择你自己工程的 Module，在 Flavors 选项卡中设置 Min Sdk Version 不低于 16



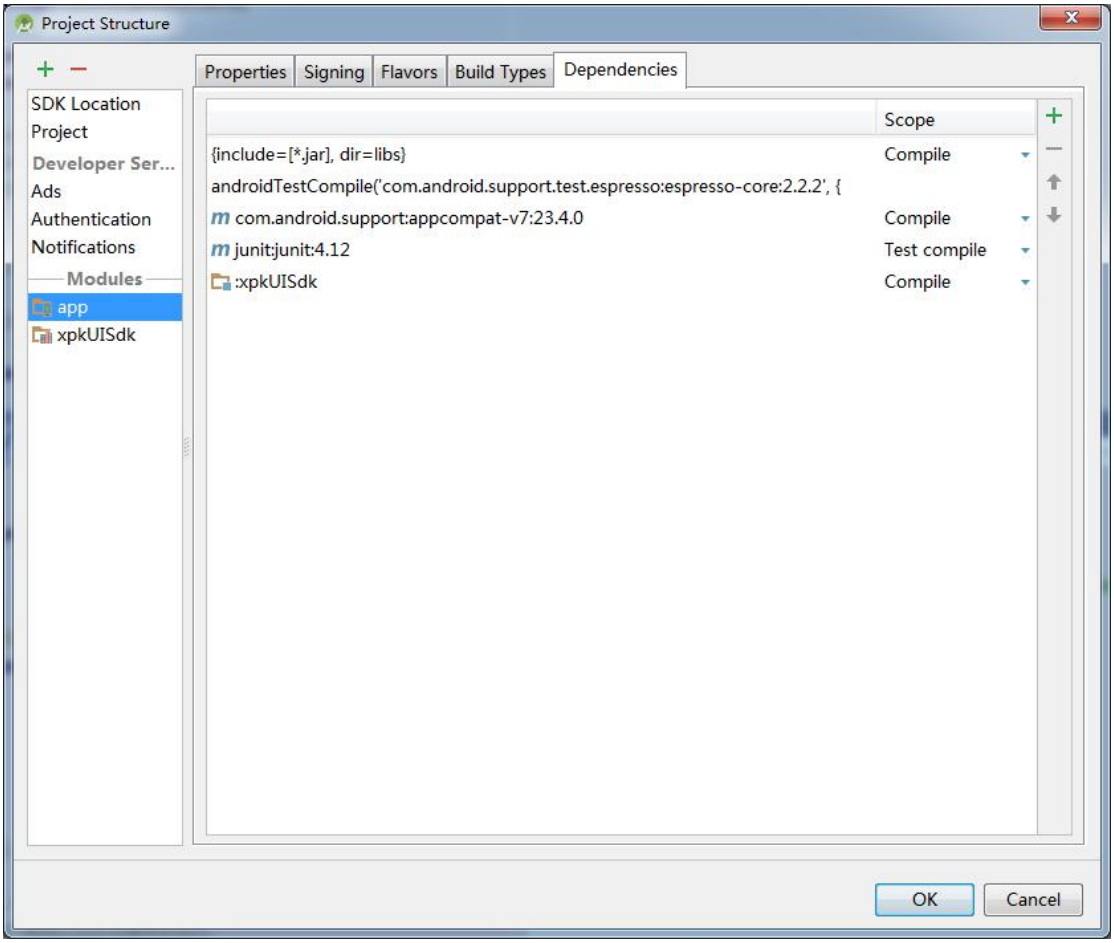
然后切换到 Dependencies 选项卡并点击右侧“+”号在弹出的下拉菜单选择 Module Dependency 点击。



弹出的小窗口中选择 SDK 相关的 Modules,“OK”



添加成功后点击 Project Structure 设置界面的“OK”完成导入工作。



关于集成 NDK SO 库引发冲突解决办法如下:

步骤一：sdk library 模块过滤

```
android {
  compileSdkVersion 19
  buildToolsVersion "23.0.2"
   packagingOptions {
     //过滤掉本地libs/armeabi/和libs/armeabi-v7a/中的部分so，以jcenter
     exclude 'lib/armeabi/libRdBase.so'
     exclude 'lib/armeabi/libLiveRecorder.so'
     exclude 'lib/armeabi/libRecorderKernel.so'
     exclude 'lib/armeabi-v7a/libRdBase.so'
     exclude 'lib/armeabi-v7a/libLiveRecorder.so'
     exclude 'lib/armeabi-v7a/libRecorderKernel.so'
  }
```

步骤二:application 模块中过滤

编辑 SDK NDK 包含以下架构的 SO 库：

- armeabi-v7a

建议在 Module 的 build.gradle 文件中使用 NDK 的“abiFilter”配置，设置支持的 SO 库架构。

如果在添加“abiFilter”之后 Android Studio 出现以下提示：

```
NDK integration is deprecated in the current plugin. Consider trying the new experimental plugin.
```

则在项目根目录的 gradle.properties 文件中添加：

```
android.useDeprecatedNdk=true
```

在集成 NDK SO 库时，请注意只保留支持的架构 SO 库，参考截图配置

```
defaultConfig {
    minSdkVersion 16
    targetSdkVersion 23
    ndk {
        abiFilters "armeabi", "armeabi-v7a"
    }
}
```

Android studio 上绑定 SDK javadoc 文档

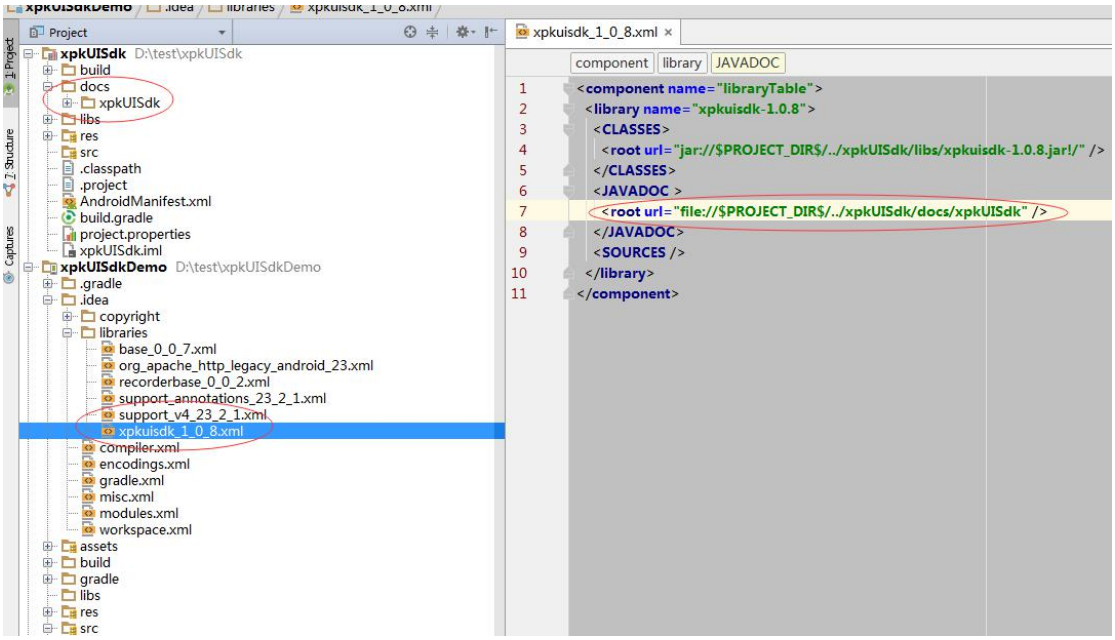
Android Studio 中使用 SDK jar 包时，会在$PROJECT_DIR$/.idea/libraries 目录中生成 xpkuisdk_$VERSTION$.xml 文件,添加以下配置：

```
<JAVADOC >
  <root url="file://$PROJECT_DIR$/../xpkUISdk/docs/xpkUISdk" />
</JAVADOC>
```

其中$PROJECT_DIR$为项目根目录；$VERSTION$代表 SDK 版本，如版本是 1.0.9，则该 XML 文件则为 xpkuisdk_1_0_9.xml；

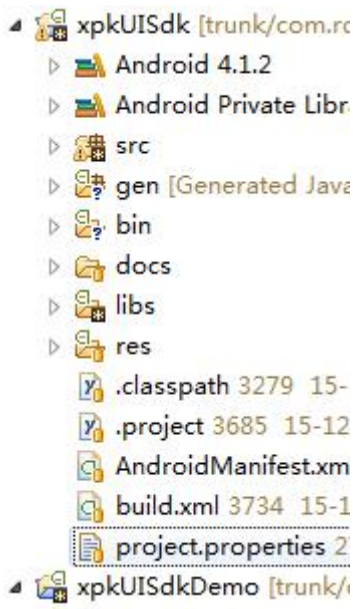
当前配置演示是以 SDK demo 项目进行的，需要根据实际集成环境进行修改。

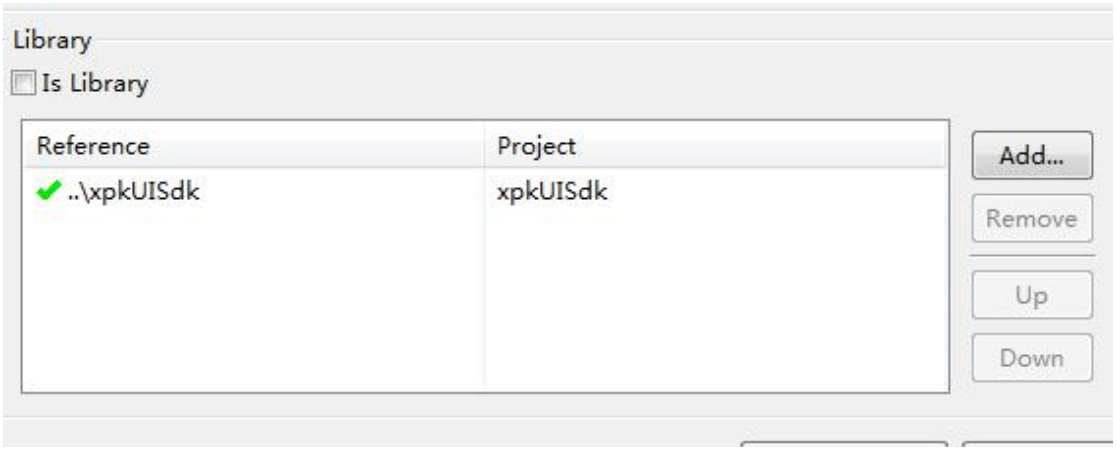
参考如下截图:



添加完 JAVADOC 后，将是如下结果：

3.3.2 Eclipse 导入 xpkUISdk





3.3.3 准备 AndroidManifest.xml (权限，注册 Activity)

a.添加权限：

```
<!-- 必要权限开始 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<!-- 需要支持后台或休眠后保存输出时，需添加的权限 -->
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

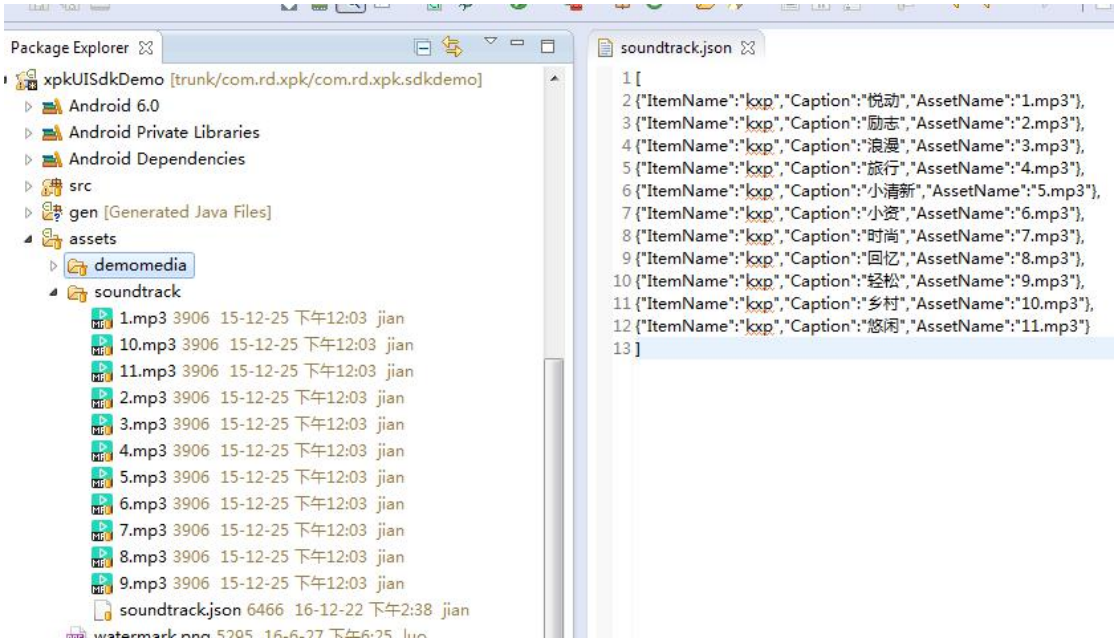
b.注册必要的 Activity：

```
<activity
    android:name="com.rd.xpkuisdk.SelectMediaActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait" />
<activity
    android:name="com.rd.xpkuisdk.VideoPreviewActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait" />
<activity
    android:name="com.rd.xpkuisdk.EditPreviewActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait" />
<activity
    android:name="com.rd.xpkuisdk.CropVideoActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait" />
<activity
    android:name="com.rd.xpkuisdk.VideoEditActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait"
    android:windowSoftInputMode="stateAlwaysHidden|adjustResize" />
<activity
    android:name="com.rd.xpkuisdk.CropVideoVerActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait" />
<activity
    android:name="com.rd.xpkuisdk.SpeedPreviewActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait" />
<activity
    android:name="com.rd.xpkuisdk.ImageDurationActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait" />
<activity
    android:name="com.rd.xpkuisdk.ExtPhotoActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait"
    android:windowSoftInputMode="stateAlwaysHidden|adjustResize" />
```

如果不想影响当前项目的 AndroidManifest.xml，Eclipse 下可以合并 xpkUISdk 中的配置，修改 project.properties 文件如下：

```
android.library.reference.1=../xpkUISdk
manifestmerger.enabled=true
```

3.3.4 替换内置配音



替换对应音频文件和 soundtrack. json

3.3.5 调用 sdk 初始化

```
public class AppImpl extends Application {
    /**
     * 已获取的AppKey
     */
    public static final String APP_KEY = "27eb1f31f0f52501";
    /**
     * 已获取的AppSecret
     */
    public static final String APP_SECRET = "3ac8a8b40ce5bd5c37642978ef097731TI

    @Override
    public void onCreate() {
        super.onCreate();
        initXpkSdk();
    }

    /**
     * 初始化SDK
     */
    public void initXpkSdk() {
        // 这里确定是否启用日志，在调试阶段可开启该选项，方便定位问题。打包为发行版时，此启用无效
        XpkSdk.enableDebugLog(true);
        // 自定义根目录，如果为空则默认为/sdcard/xpk
        String strCustomPath = Environment.getExternalStorageDirectory()
                + "/xpk";
        // sdk初始
        XpkSdk.init(AppImpl.this,strCustomPath, APP_KEY, APP_SECRET,
                new SdkHandler().getCallBack());
        // 自定义Crash handler,实际项目可不加入
        MyCrashHandler.getInstance().init(this);
    }
}
```

其中的初始化 SDK 函数

boolean com.rd.xpkuisdk.XpkSdk.init(Context context, String strCustomRootPath, String appkey, String appScrect, ISdkCallBack iCallback)

Parameters:

context 应用上下文

strCustomRootPath 自定义的工作目录，不设置则为默认 xpk

appkey 在平台申请的 Appkey

appScrect 在平台申请的 appScrect

iCallback 回调接口

Returns:

返回 true 代表正常初始化 SDK

xpkUISDK 回调接口为：

```
public interface ISdkCallBack {
/**
 * 目标视频的路径
 *
 * @param context
 *            应用上下文
 * @param exportType
 *            回调类型 来自简单录制 {@link XpkSdk#XPK_CAMERA_EXPORT}<br>
 *            来自录制编辑{@link XpkSdk#XPK_CAMERA_EDIT_EXPORT}<br>
 *            来自编辑导出{@link XpkSdk#XPK_EDIT_EXPORT}<br>
 *            来自普通截取视频导出{@link XpkSdk#XPK_TRIMVIDEO_EXPORT}<br>
 *            来自定长截取视频导出{@link XpkSdk#XPK_TRIMVIDEO_DURATION_EXPORT}<br>
 * @param videoPath
 */
public void getVideoPath(Context context, int exportType, String videoPath);
/**
 * 响应截取视频时间
 *
 * @param context
 *            应用上下文
 * @param exportType
 *            回调类型 来自普通截取视频的时间导出{@link XpkSdk#XPK_TRIMVIDEO_EXPORT}<br>
 *            来自定长截取视频的时间导出{@link XpkSdk#XPK_TRIMVIDEO_DURATION_EXPORT}<br>
 * @param startTime
```

```
     *            开始时间
     * @param endTime
     *            结束时间
     */
    public void getVideoTrimTime(Context context, int exportType,
                                 int startTime, int endTime);
    /**
     * 响应确认截取按钮
     *
     * @param context
     *            应用上下文
     * @param exportType
     *            来自普通截取的确认 {@link XpkSdk#XPK_TRIMVIDEO_EXPORT}
     *            来自定长截取的确认 {@link XpkSdk#XPK_TRIMVIDEO_DURATION_EXPORT}<br>
     */
    public void getVideoTrim(Context context, int exportType);
    /**
     * 使用自定义相册时响应进入相册（只显示照片、图片）
     *
     * @param context
     *            应用上下文
     */
    public void getPhoto(Context context);
    /**
     * 使用自定义相册时响应进入相册（只显示视频）
     *
     * @param context
     *            应用上下文
     */
    public void getVideo(Context context); }
```

初始化成功后才能正常使用其他功能接口,还要注意系统权限的申请，否则 SDK 在系统 api 级别 >= 23（android 6.0+）下不能正常使用

```
// 请求code:读取外置存储
private static final int REQUEST_CODE_READ_EXTERNAL_STORAGE_PERMISSIONS = 1;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    //setContentView(R.layout.activity_main);

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M
            && !XpkSdk.isInitialized()) {
        if (checkSelfPermission(Manifest.permission.READ_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED) {
            requestPermissions(
                    new String[] { Manifest.permission.READ_EXTERNAL_STORAGE },
                    REQUEST_CODE_READ_EXTERNAL_STORAGE_PERMISSIONS);
        }
    }
```

```
@SuppressLint("NewApi")
@Override
public void onRequestPermissionsResult(int requestCode,
                                       String[] permissions, int[] grantResults) {
    switch (requestCode) {
        case REQUEST_CODE_READ_EXTERNAL_STORAGE_PERMISSIONS: {
            for (int i = 0; i < permissions.length; i++) {
                if (grantResults[i] == PackageManager.PERMISSION_GRANTED) {
                    if (!XpkSdk.isInitialized()) {
                        ((AppImpl) getApplication()).init();
                    }
                } else {
                    Toast.makeText(this, "未允许读写存储！", Toast.LENGTH_SHORT).show();
                    finish();
                }
            }
        }
        break;
        default: {
            super.onRequestPermissionsResult(requestCode, permissions,
                    grantResults);
        }
    }
}
```

3.4 指定配置参数

SDK 初始化成功之后，需要部分自定义功能时需指定配置参数，不指定时也不会影响使用，只是按照默认配置进行调用。

a.编辑界面配置：

```
UIConfiguration uiConfig = new UIConfiguration.Builder()
        //设置横竖屏(支持横屏，竖屏，自动)
        .setOrientation(orientation)
        //设置是否使用自定义相册（传 false 将调用秀拍客相册）
        .useCustomAlbum(useCustomAlbum)
        //设置是否开启向导化
        .enalbeWizard(enable)
        // 设置秀拍客相册支持格式
        .setAlbumSupportFormat(albumSupportFormatType)
        // 设置配乐界面风格
        .setSoundTrackType(soundTrakLayoutType)
        // 设置滤镜界面风格
        .setFilterType(filterLayoutType)
        //设置视频默认比例
        .setVideoProportion(proportion)
        //设置编辑导出功能模块显示与隐藏
        .setEditAndExportModuleVisibility(editAndExportModules,visibility)
        //设置片段编辑功能模块显示与隐藏
        .setClipEditingModuleVisibility(clipEditingModules,visibility)
        //设置 MV 及 MV 资源 url，详见自定义 MV 资源
        .enableMV(boolean enable, String url)
        //配乐方式 2 (setSoundTrackType(UIConfiguration.SOUND_TRACK_LAYOUT_2))有效,
    设置自定义的网络音乐
        .setMusicUrl(musicUrl)
        .get();
```

- 其中 EditAndExportModules 是编辑导出功能模块枚举

```
public enum EditAndExportModules {
        /** 配乐 */
        SOUNDTRACK,
        /** 配音 */
        DUBBING,
        /** 滤镜 */
        FILTER,
        /** 字幕 */
```

```
        TITLING,
        /** 特效 */
        SPECIAL_EFFECTS,
        /** 片段编辑 */
        CLIP_EDITING,
    }
```

- ClipEditingModules 是片段编辑模块枚举

```
public enum ClipEditingModules {
        /** 视频调速 */
        VIDEO_SPEED_CONTROL,
        /** 设置图片时长 */
        IMAGE_DURATION_CONTROL,
        /** 复制 */
        COPY,
        /** 图片视频编辑 */
        EDIT,
        /** 视频比例 */
        PROPORTION,
        /** 调序 */
        SORT,
        /** 截取 */
        TRIM,
        /** 分割 */
        SPLIT,
        /** 文字板 */
        TEXT,
    }
```

- 若选择自定义相册,将会回调以下接口：

```
com.rd.xpkuisdk.ISdkCallBack.getVideo(context) //添加视频
com.rd.xpkuisdk.ISdkCallBack.getPhoto(context)//添加图片
```

在上述方法中调用自定义相册，选择视频或图片完成后，将它们的路径通过调用以下方法完成选择：

```
com.rd.xpkuisdk.XpkSdk.onCustomizeAlbum(
                                    Contextcontext, //应用上下文
                                    ArrayList<String> medialist //视频或图片路径集合)
```

b.导出视频配置：

```
ExportConfiguration exportConfig = new ExportConfiguration.Builder()
        //设置保存导出视频时的码流 单位 M
        .setVideoBitRate(bitRate)
        //设置导出视频时长 单位 ms（设置为 0 或默认不设置将导出完整视频）
        .setVideoDuration(exportVideoDuration)
        //设置导出视频保存路径（设 null 将保存到默认路径）
        .setSavePath(savePath)
```

```
//设置视频片尾图片路径（设 null 将没有片尾）
.setTrailerPath(trailerPath)
//设置视频片尾时长 单位 ms（默认为 2000ms）
.setTrailerDuration(trailerDuration)
// 设置是否显示水印
.enableWatermark(enable)
// 设置水印位置（默认为左下角）
.setWatermarkPosition(watermarkPosition)
.get();
```

c.录制拍摄配置：

```
CameraConfiguration cameraConfig = new CameraConfiguration.Builder()
        //设置限制录制的最小视频时长 单位为秒 0 代表没有限制
        .setVideoMinTime(maxTime)
        //设置限制录制的最大视频时长 单位为秒 0 代表没有限制
        .setVideoMaxTime(maxTime)
        //确定是否支持拍照模式下点击拍照按钮立即返回
        .setTakePhotoReturn(taskPhotoReturn)
        //设置录制时启动默认页面方式
        .setCameraUIType(type)
        //录制时静音
        .setAudioMute(mute)
        // 设置是否默认为后置摄像头
        .setDefaultRearCamera(setDefaultRearCamera)
        //设置是否启用人脸识别及贴纸
        .setEnableFace(boolean enable)
        //设置人脸贴纸/挂件鉴权证书
        .setPack(byte[] _pack)
        .get();
```

其中 CameraUIType 可选常量定义如下：

```
//代表默认启动 16：9 宽屏录制界面并可切换到 1：1 界面
CameraConfiguration.WIDE_SCREEN_CAN_CHANGE
//代表默认启动 1：1 界面并可切换到 16：9 宽屏录制界面
CameraConfiguration.SQUARE_SCREEN_CAN_CHANGE
//代表默认启动 1：1 界面并不可切换到 16：9 宽屏录制界面
CameraConfiguration.ONLY_SQUARE_SCREEN
```

d.*人脸识别及贴纸配置：详见使用人脸贴纸

指定配置参数例子：

```
XpkSdk.getXpksdkService().initConfiguration(
        exportConfig, //导出配置
        uiConfig, //界面配置
```

cameraConfig //录制拍摄配置);

3.5 录制视频

3.5.1 录制视频或拍照

进入该界面调用以下接口：

```
public static void com.rd.xpkuisdk.onXpkCamera(
        Context context,   //应用上下文
        int requestCode   //用于 ActivityResult 返回)
```

录制视频或录制视频并编辑导出完成后将会通过 ActivityResult 返回视频地址：

```
data.getStringExtra(XpkSdk.INTENT_KEY_VIDEO_PATH);  //返回的视频地址
data.getStringExtra(XpkSdk.INTENT_KEY_PICTURE_PATH);  //返回的图片地址
```

其中回调类型 exportType 参数值定义如下：

3.6 编辑视频

3.6.1 进入 sdk 的功能界面 (选择媒体资源)

```
void com.rd.xpkuisdk.XpkSdk.onXpkVideo(Context context//应用上下文)
```

3.6.2 进入 sdk 的编辑界面 (直接进入编辑界面)

```
void com.rd.xpkuisdk.XpkSdk.onXpkEdit(
    Context context,//应用上下文
    ArrayList<String> list //媒体路径集合(可用的图片或视频的路径)
    int requestCode     //ActivityResult 的返回值
    )
```

3.6.3 进入 sdk 的截取界面 (直接进入截取界面)

通过以下接口可以直接进入截取界面：

```
void com.rd.xpkuisdk.XpkSdk.onXpkTrimVideo(
    Context context, //应用上下文
    String videoPath, //媒体路径
```

String savePath,//视频截取后的保存路径 设置 null 将会保存到默认路径.../xpk/xpkVideos/

String title, //标题栏文字内容 设置 null 标题将会设为“截取”

int titleBarColor, //标题栏背景色

String buttonCancelText, //取消按钮文字 设置 null 将会显示图标 ×

String buttonConfirmText, //确认按钮文字 设置 null 将会显示图标 √

int buttonColor //按钮背景色

);

完成截取需分为两个步骤，第一步是先确认截取返回方式，第二步通过返回方式将最终的结果通过对应的回调接口进行返回。

第一步：确认截取返回方式

确认截取返回方式时将会回调以下接口：

```
com.rd.xpkuisdk.ISdkCallBack.getVideoTrim(
    context,//应用上下文
    exportType //回调类型
    )
```

其中回调类型 exportType 参数值定义如下：

```
XpkSdk.XPK_TRIMVIDEO_EXPORT //普通截取

XpkSdk.XPK_TRIMVIDEO_DURATION_EXPORT //定长截取
```

第二步：设置截取返回方式

通过以下接口完成截取返回方式设置:

```
void com.rd.xpkuisdk.XpkSdk.onVideoTrim(
    Context context, //应用上下文
    int returnType //需要给定的截取返回方式

    )
```

其中截取返回方式 returnType 参数值定义如下：

```
RETURN_TRIM_VIDEO = 0//返回截取后视频
```

完成截取并导出视频后将会回调以下接口：

```
com.rd.xpkuisdk.ISdkCallBack.getVideoPath(
    context, //应用上下文
    exportType,//截取并导出视频回调类型
    videoPath//截取并导出后的视频路径
    )
```

其中回调类型 exportType 参数值定义如下：

XpkSdk.XPK_TRIMVIDEO_EXPORT //普通截取

XpkSdk.XPK_TRIMVIDEO_DURATION_EXPORT //定长截取

RETURN_TRIM_TIME = 1//返回截取时间

截取界面将会返回截取视频的开始和结束时间 并执行以下回调：

```
com.rd.xpkuisdk.ISdkCallBack.getVideoTrimTime(
    context, //应用上下文
    exportType,//返回截取时间回调类型
    startTime,//开始时间(单位 ms)
    endTime //结束时间(单位 ms)
    )
```

其中回调类型 exportType 参数值定义如下：

XpkSdk.XPK_TRIMVIDEO_EXPORT //普通截取

XpkSdk.XPK_TRIMVIDEO_DURATION_EXPORT //定长截取

3.6.4 导出视频

编辑完成后，根据导出配置生成视频，导出完成后将会回调以下接口：

```
com.rd.xpkuisdk.ISdkCallBack.getVideoPath(
    context, //应用上下文
    exportType,//导出视频回调类型
    videoPath//视频路径
    )
```

其中回调类型 exportType 参数值定义如下：

XpkSdk.XPK_EDIT_EXPORT //普通编辑导出完成后的返回

XpkSdk.XPK_CAMERA_EDIT_EXPORT //代表录制完成并编辑导出后的返回

视频导出完成后还可以通过 ActivityResult 返回视频的地址：

data.getStringExtra(XpkSdk.XPK_EDIT_RESULT)

3.7 调用相册

调用以下接口可直接进入秀拍客相册：

```
XpkSdk.onXpkAlbum(
            context, //应用上下文
            formatType, //相册支持的类型
            requestCode //用于 ActivityResult 返回)
```

其中回调类型 exportType 参数值定义如下：

```
UIConfiguration.ALBUM_SUPPORT_DEFAULT //默认（同时支持图片与视频）
UIConfiguration.ALBUM_SUPPORT_VIDEO_ONLY //仅支持视频
UIConfiguration.ALBUM_SUPPORT_IMAGE_ONLY //仅支持图片
```

确认选择图片或视频后将会通过 ActivityResult 返回它们的本地路径

```
data .getStringArrayListExtra(XpkSdk.XPK_ALBUM_RESULT)
```

3.8 压缩视频

压缩视频配置类示例：

```
CompressConfiguration compressConfig=new CompressConfiguration.Builder()
        // 设置压缩视频码率
        .setBitRate(compressBitRate)
       // 是否显示水印
        .enableWatermark(enableCompressWatermark)
        // 显示水印的位置
        .setWatermarkPosition(ompressWatermarkPosition)
        // 设置视频分辨率
        .setVideoSize(compressVideoWidth, compressVideoHeight)
        .get()


XpkSdk.getXpksdkService().initCompressConfiguration(compressConfig)
```

调用接口

```
XpkSdk.onCompressVideo(this, videoPath, iCompressVideoCallback);
```

其中 iCompressVideoCallback 是回调函数

```
 public interface ICompressVideoCallback {

      void onCompressStart();    //开始压缩
```

```
    void onProgress(int progress, int max);   //压缩进度
    void onCompressComplete(String mediaPath);   //压缩完成
    void onCompressError(String errorLog);    //压缩失败
}
```

3.9 使用自定义播放器

com.rd.xpkuisdk.ui.XpkVideoView

3.9.1 在布局文件中添加播放器

```
<com.rd.xpkuisdk.XpkVideoView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

3.9.2 播放器设置 listener

void com.rd.xpkuisdk.XpkVideoView.setPlayerListener(XpkPlayerListener listener)

3.9.3 播放器接口

```
setVideoPath(String videoPath) //设置视频路径
boolean isPlaying() //判断播放器是否正在播放
void start() //开始播放
void pause() //暂停播放
int getCurrentPosition() //获取当前播放进度
void seekTo(int position) //跳到指定播放位置(单位 ms)
int getDuration() //获取媒体时长,单位 ms
```

3.10 响应退出

应用结束时，需调用一下 SDK 的退出接口 XpkSdk.exitApp,一般是在最后退出 Activity 中的 onDestroy 方法内，具体实现参见 demo。

3.11 分享视频

暂无

3.12 打包混淆

需要在 proguard.cfg 文件中添加如下配置项：

-dontwarn com.rd.**

-keep class com.rd.** { *; }

-keepclassmembers class com.faceunity.wrapper {*;}

-keepclassmembers class com.spap.** {*;}

3.13 自定义 MV 资源

通过以下配置可以启用 MV 并设置获取 MV 资源的 url

UIConfiguration .enableMV(boolean enable, String url)

获取 MV 资源的 url 需满足以下定义：

请求格式：

URL	http://**
支持格式	JSON
HTTP 请求验证	POST/GET
请求次数限制	否

请求参数：

参数名	必选	类型及范围	说明
type	是	string	android ios

正常返回结果：

返回键	类型	返回值	说明
state	Boolean	请求状态	true
code	int	请求状态码	200 正常
message	String	请求状态文本	
result	JsonObject	请求返回参数	
data	JsonArray	mv 列表	
name	String	mv 名称	不能为 null 且唯一
img	String	mv 图片	不能为 null 且唯一
url	String	mv 链接	不能为 null 且唯一
enname	String	英文名称	不能为 null 且唯一

参考 Demo

请求参数:{"type":"android"}

返回数据:

3.14 使用人脸贴纸

3.14.1 指定配置参数：

a.是否开启人脸识别及贴纸:

```
CameraConfiguration cameraConfig = new CameraConfiguration.Builder()
        //设置是否启用人脸识别及贴纸
        .setEnableFace(boolean enable)
        //设置人脸识别鉴权证书
        .setPack(byte[] _pack)
        .get();
```

b.配置 FaceuConfig:

```
FaceuConfig fc= new FaceuConfig();
//设置基础库本地路径（必须,否则无法开启人脸识别及贴纸）
fc.setV3Path(String v3path)
//设置美颜基础库本地路径（必须,否则无法启用美颜）
fc.setBeautyPath(String BeautyPath)
//设置开启美颜时美白的等级（参数值 0.0f-1.0f,开启人脸识别且开启美颜之后生效）
fc.setColor_level(0.48f);
//设置开启美颜时磨皮的等级(参数值 0.0f-4.0f,开启人脸识别且开启美颜之后生效)
fc.setBlur_level(4.0f);
//设置开启美颜时瘦脸的等级（有效参数值 0.0f-2.0f,开启人脸识别且开启美颜之后生效）
fc.setCheek_thinning(0.68f);
//设置美颜时大眼的等级(有效参数值 0.0f-4.0f,开启人脸识别且开启美颜之后生效)
fc.setEye_enlarging(1.53f);
//新增单个人脸贴纸本地资源
fc.addFaceu(path, //.mp3 人脸贴纸本地路径
            icon, //人脸贴纸本地图标
            title //说明);
//是否启用加载网络化的人脸贴纸
fc.enableNetFaceu(true);
//设置网络人脸贴纸的请求接口 url
fc.setUrl(url);
```

完成初始配置：

```
XpkSdk.getXpksdkService().initFaceuConfig(
```

faceuConfig //人脸贴纸配置);

3.14.2 自定义资源：

获取人脸贴纸资源的 url 需满足以下定义：

请求格式：

URL	http://**
支持格式	JSON
HTTP 请求验证	POST/GET
请求次数限制	否

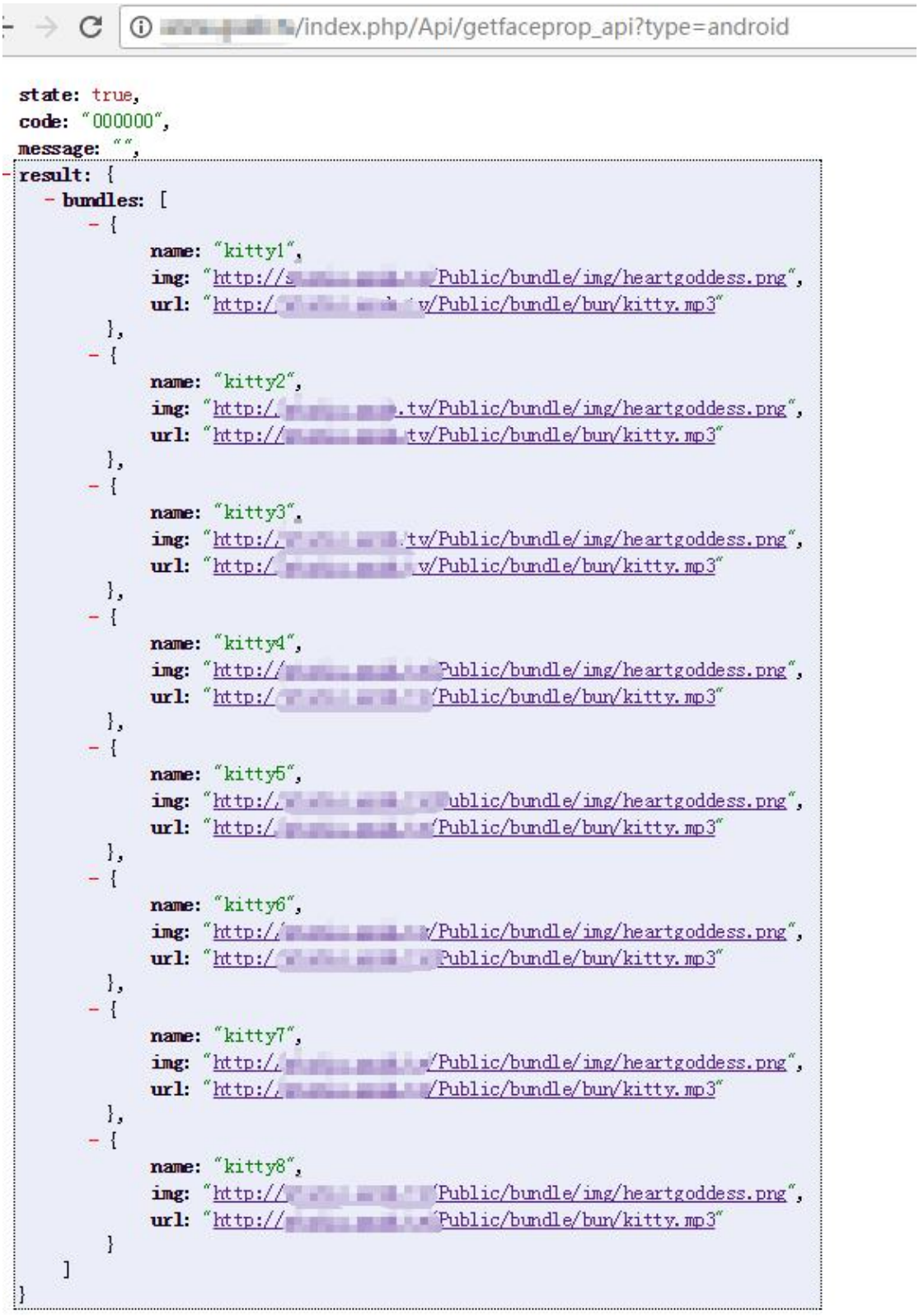
请求参数：

参数名	必选	类型及范围	说明
type	是	string	android ios

正常返回结果：

返回键	类型	返回值	说明
state	Boolean	请求状态	
code	int	请求状态码	
message	String	请求状态文本	
result	JsonObject	请求返回参数	
bundles	JsonArray	贴纸列表	
name	String	贴纸名称	
img	String	贴纸图片	
url	String	贴纸链接	

返回资源数据演示：


/index.php/Api/getfaceprop_api?type=android
state: true,
code: "000000",
message: "",
result: {
bundles: [
{
name: "kitty1",
img: "http://s /Public/bundle/img/heartgoddess.png",
url: "http:/ v/Public/bundle/bun/kitty.mp3"
},
{
name: "kitty2",
img: "http:/ .tw/Public/bundle/img/heartgoddess.png",
url: "http:// tw/Public/bundle/bun/kitty.mp3"
},
{
name: "kitty3",
img: "http:/ tw/Public/bundle/img/heartgoddess.png",
url: "http:/ v/Public/bundle/bun/kitty.mp3"
},
{
name: "kitty4",
img: "http:// Public/bundle/img/heartgoddess.png",
url: "http:/ /Public/bundle/bun/kitty.mp3"
},
{
name: "kitty5",
img: "http:/ ublic/bundle/img/heartgoddess.png",
url: "http:/ /Public/bundle/bun/kitty.mp3"
},
{
name: "kitty6",
img: "http:/ /Public/bundle/img/heartgoddess.png",
url: "http:/ Public/bundle/bun/kitty.mp3"
},
{
name: "kitty7",
img: "http:/ /Public/bundle/img/heartgoddess.png",
url: "http:/ /Public/bundle/bun/kitty.mp3"
},
{
name: "kitty8",
img: "http:// Public/bundle/img/heartgoddess.png",
url: "http:// Public/bundle/bun/kitty.mp3"
}
]
}

3.15 自定义 Music 资源

通过以下配置可以启用自定义 music 并设置获取 music 资源的 url

UIConfiguration .setSoundTrackType(UIConfiguration.SOUND_TRACK_LAYOUT_2))

UIConfiguration.setMusicUrl(musicUrl)//若不设置 url 就是使用默认的网络配乐

获取 Music 资源的 url 需满足以下定义：

请求格式：

URL	http://**
支持格式	JSON
HTTP 请求验证	POST/GET
请求次数限制	否

请求参数：

参数名	必选	类型及范围	说明
type	是	string	android 或 ios

正常返回结果：

返回键	类型	返回值	说明
state	Boolean	请求状态	true
code	int	请求状态码	200 正常
message	String	请求状态文本	
result	JsonObject	请求返回参数	
bgmusic	JsonArray	music 列表	

name	String	名称	不能为 null 且唯一
icon	String	图片	不能为 null 且唯一
url	String	music 链接	不能为 null 且唯一

参考 demo:

请求参数:{"type":"android"}

返回数据: