短视频

工程配置

产品文档

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2018 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

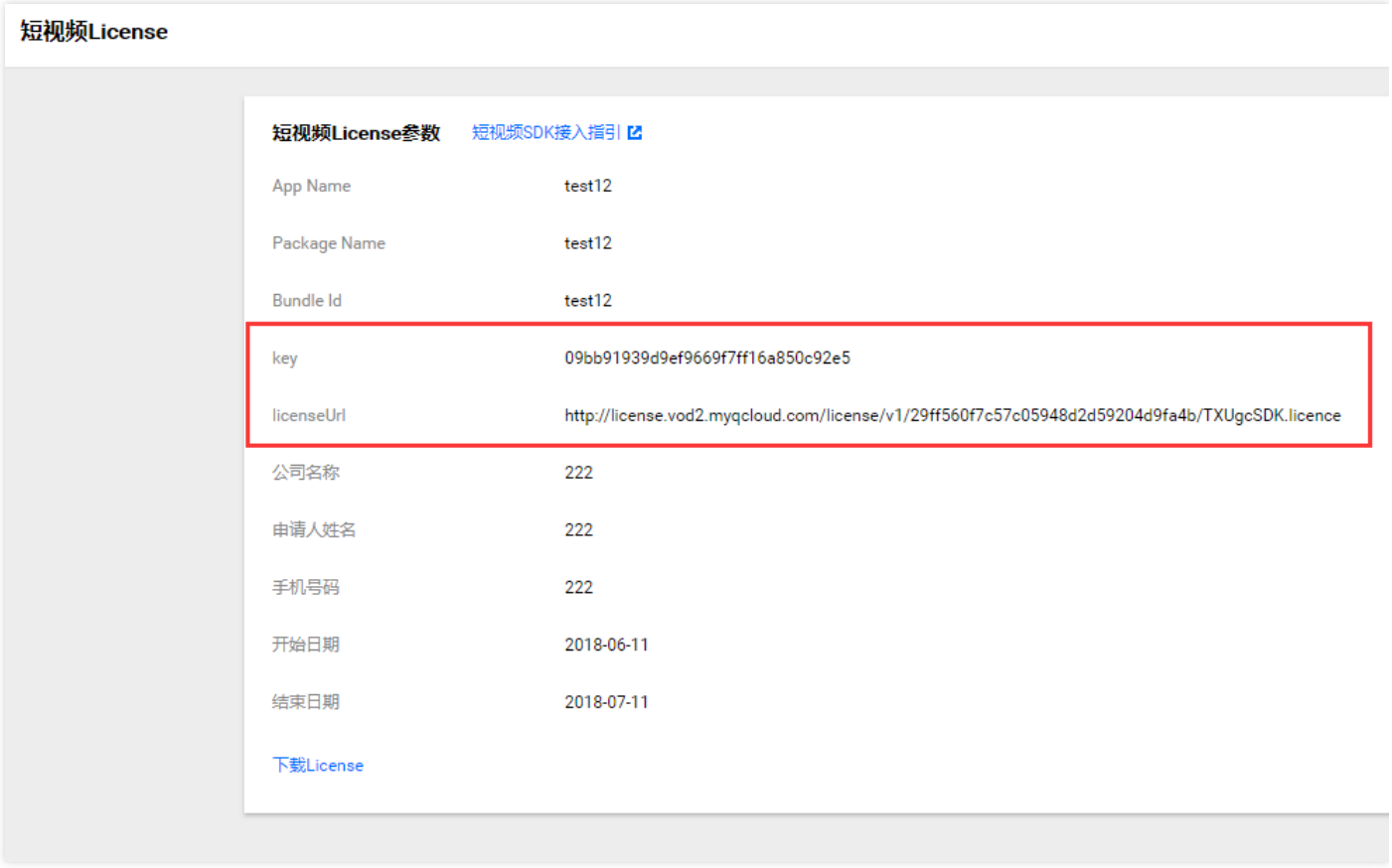
版权所有：腾讯云计算（北京）有限责任公司

工程配置
XCode

最近更新时间：2018-07-18 11:16:07

短视频 licence 集成

- 在 控制台 填写完信息后，会拿到 key 和 url，见下图。



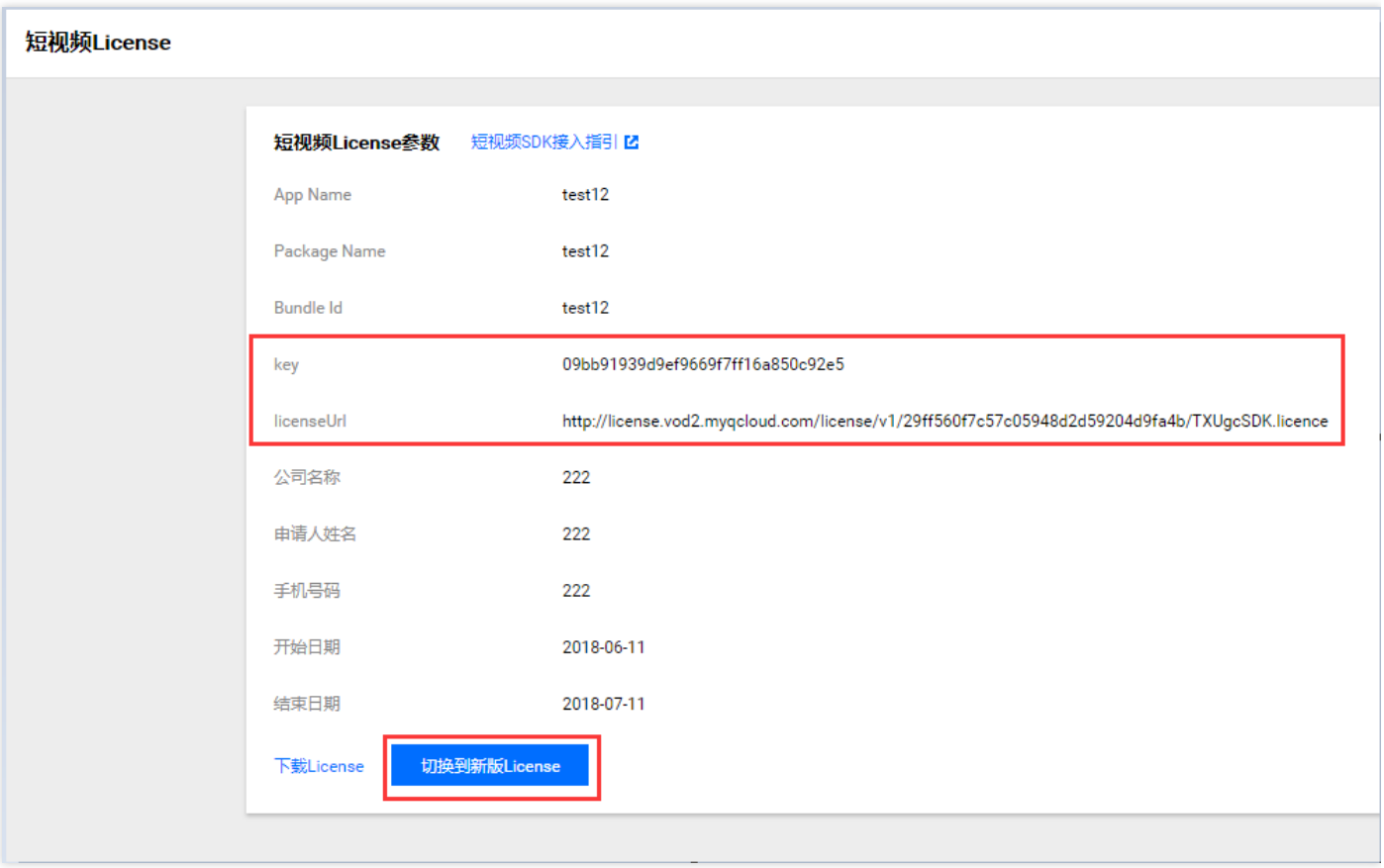
在您的应用中使用短视频功能之前（建议在 AppDelegate 中）把拿到的 key 和 url 设置到下面接口中

```
[TXUGCBase setLicenceURL:url key:key];
```

- 您可以选择是否打包 licence 到应用中：如果不选择打包，SDK 第一次使用需要访问网络；如果选择打包，把 TXUgcSDK.licence（名称要正确）拷贝到 App 中即可。
- 当您的 licence 过期了，可以登录腾讯云点播控制台进行续费，SDK 会自动续期，不需要您的应用做任何操作。
- 如果您的 licence 校验失败，您可以调用下面代码来查看 licence 信息是否填写错误。

```
NSLog(@"%@", [TXUGCBase getLicenceInfo]);
```

- 对于使用 4.7 版本 licence 的用户，如果您升级了 SDK 到 4.9 版本了，您可以登录控制台，单击下图的 **切换到新版License** 按钮生成对应的 key 和 url，按照上述操作集成即可。



Xcode 工程设置

支持平台

- SDK 支持 iOS 8.0 以上系统

开发环境

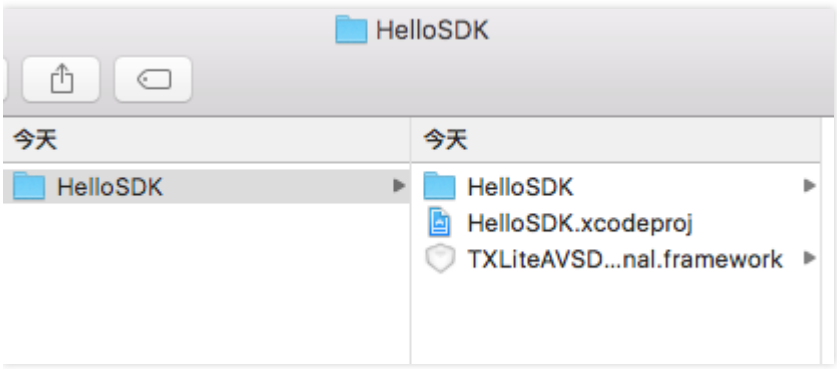
- Xcode 9 或更高版本
- OS X 10.10 或更高版本

Xcode工程设置

下面通过一个简单的 iOS Application 工程，说明如和在 Xcode 工程中配置 SDK。

拷贝 SDK 文件

在本例中，新建一个名字叫做 HelloSDK 的 iOS 工程，将下载下来的 TXLiteAVSDK_UGC.framework 拷贝至工程目录。目录结构如下图所示：

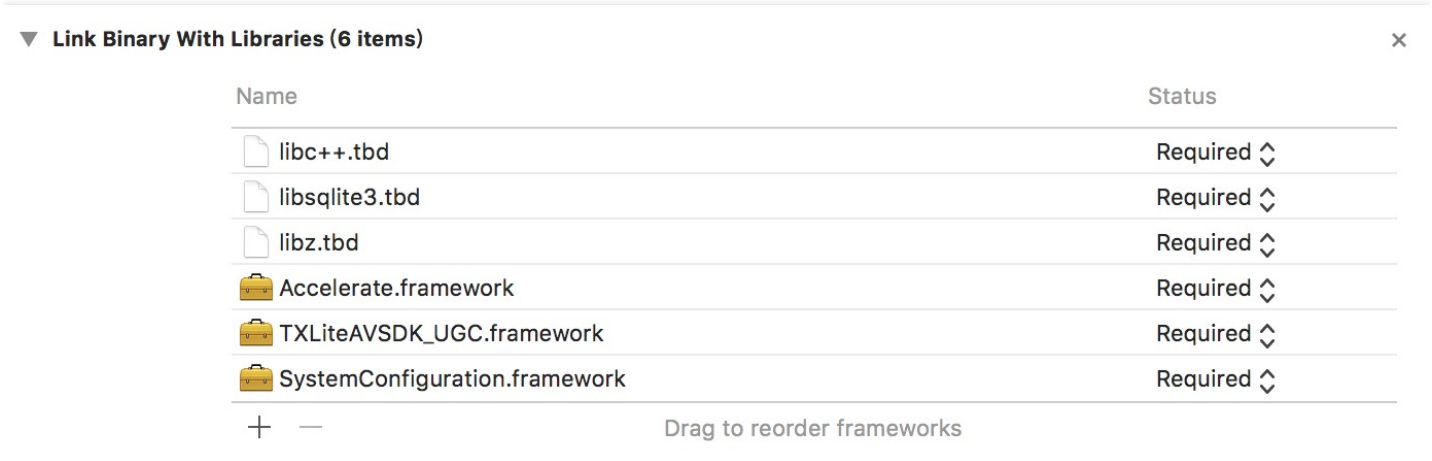


添加 Framework

在工程中添加 TXLiteAVSDK_UGC.framework ，同时还要添加以下系统依赖库：

1. Accelerate.framework
2. SystemConfiguration.farmework
3. libstdc++.tbd
4. libsqlite3.tbd
5. libz.tbd

所有系统依赖库添加完毕，工程依赖如下图所示：



添加 -ObjC

SDK 用到了一些类别的方法，加载类别方法需要在工程配置：Build Settings -> Linking -> Other Linker Flags 添加 -ObjC ，否则在程序运行的过程中可能因为找不到类别方法而报错。

版权所有：腾讯云计算（北京）有限责任公司

引用头文件

在需要使用SDK的文件中引用SDK，如下所示：

- 5.0开始的SDK支持clang module, 可以直接使用@import来引入

```
@import TXLiteAVSDK_UGC;
```

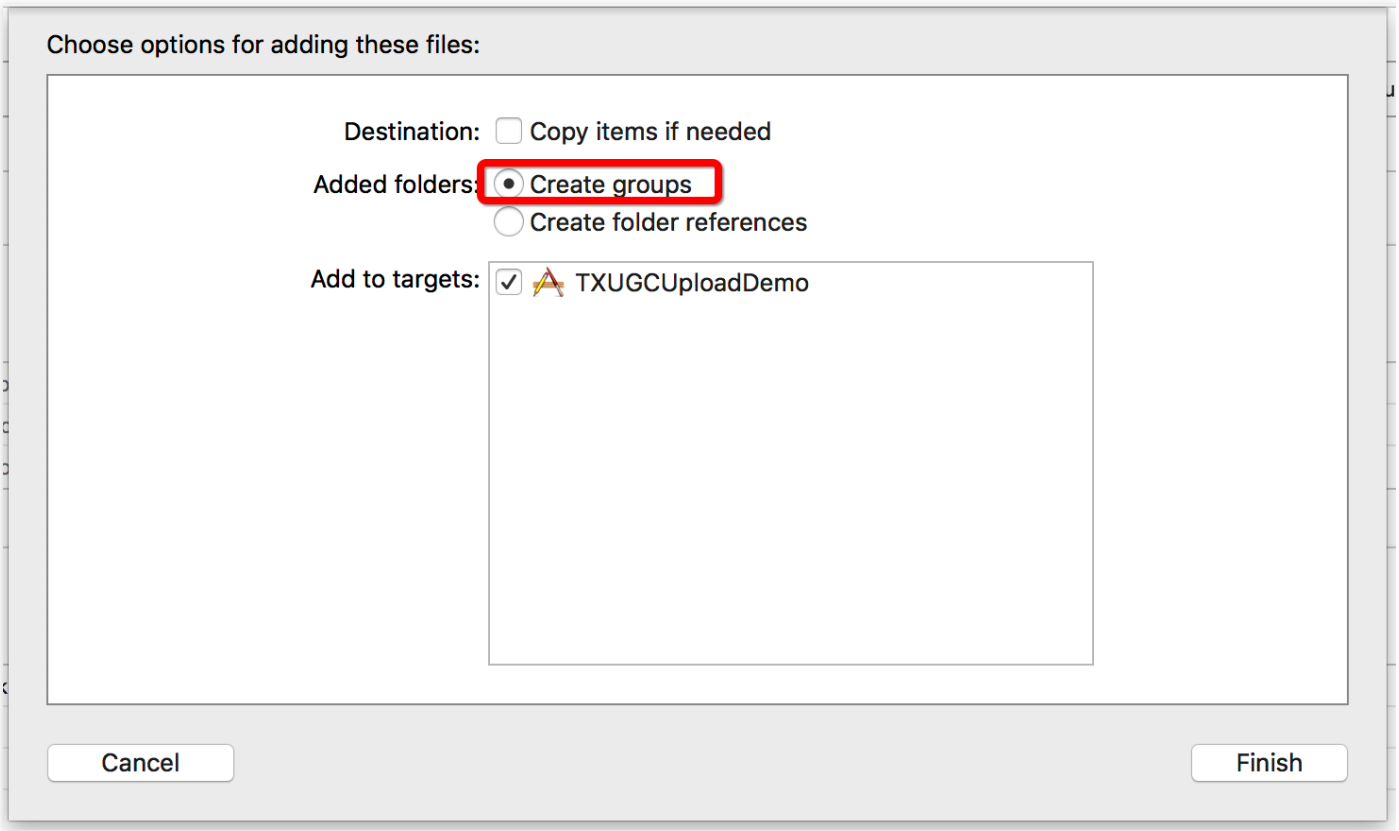
- 5.0之前的版本SDK需要单独引用使用到的头文件，比如

```
#import <TXLiteAVSDK_UGC/TXUGCBase.h>
```

短视频发布功能集成

短视频发布功能以源码形式对外提供，您需要手动集成源代码到您的工程中。

- 拷贝上传源代码目录 Demo/TXLiteAVDemo/VideoUpload 到您的工程目录中。
- 将 VideoUpload 目录拖拽到 xcode 工程中的合适位置，在弹出的对话框中选择 Added floders:Create groups，选择添加到的 target，然后单击 finish。



验证

下面在 HelloSDK 的代码中，调用 SDK 的接口，获取 SDK 版本信息，以验证工程设置是否正确。

引用头文件

在 ViewController.m 开头引用 SDK：

```
@import TXLiteAVSDK_UGC；
```

添加调用代码

在 viewDidLoad 方法中添加代码：

```
- (void)viewDidLoad {
[super viewDidLoad];
// 打印SDK的版本信息
NSLog(@"SDK Version = %@", [TXLiveBase getSDKVersionStr]);
}
```

编译运行

如果前面各个步骤都操作正确的话，HelloSDK 工程就可以顺利编译通过。在 Debug 模式下运行 App，Xcode 的 Console 窗格会打印出 SDK 的版本信息。

```
2017-09-26 16:16:15.767 HelloSDK[17929:7488566] SDK Version = 3.4.1761
```

至此，工程配置完成。

LOG 打印

在 TXLiveBase 中可以设置 log 是否在控制台打印以及 log 的级别，具体代码如下：

- **setConsoleEnabled**
  设置是否在 xcode 的控制台打印 SDK 的相关输出。

- **setLogLevel**
  设置是否允许 SDK 打印本地 log，SDK 默认会将 log 写到当前 App 的 **Documents/logs** 文件夹下。
  如果您需要我们的技术支持，建议将次开关打开，在重现问题后提供 log 文件，非常感谢您的支持。

- **Log 文件的查看**
  小直播 SDK 为了减少 log 的存储体积，对本地存储的 log 文件做了加密，并且限制了 log 数量的大小，所以要查看 log 的文本内容，需要使用 log 解压缩工具。

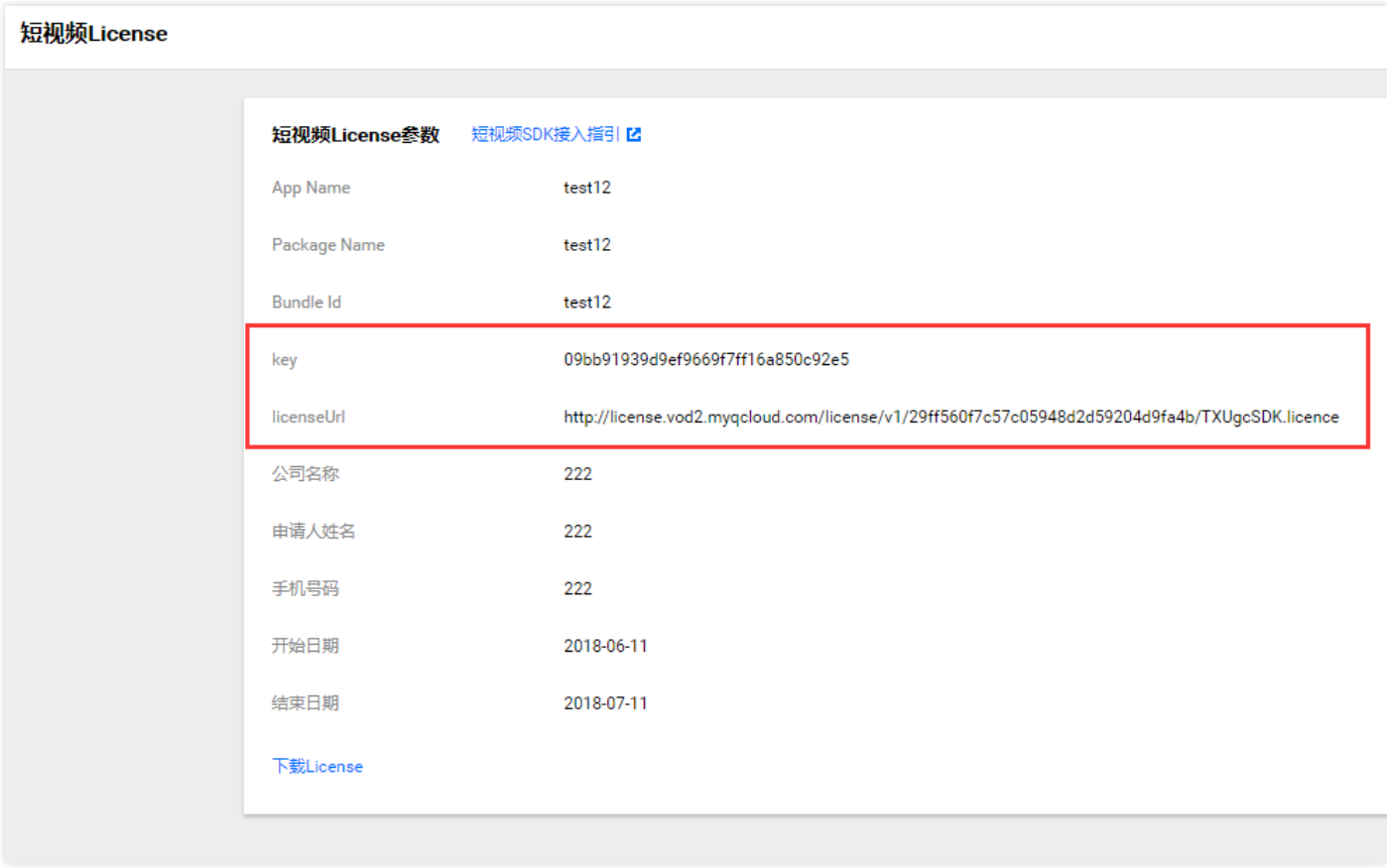
```
[TXLiveBase setConsoleEnabled:YES];
[TXLiveBase setLogLevel:LOGLEVEL_DEBUG];
```

Android Studio

最近更新时间：2018-07-19 10:18:41

短视频licence集成

- 在 控制台 填写完信息后，会拿到 key 和 url，见下图。



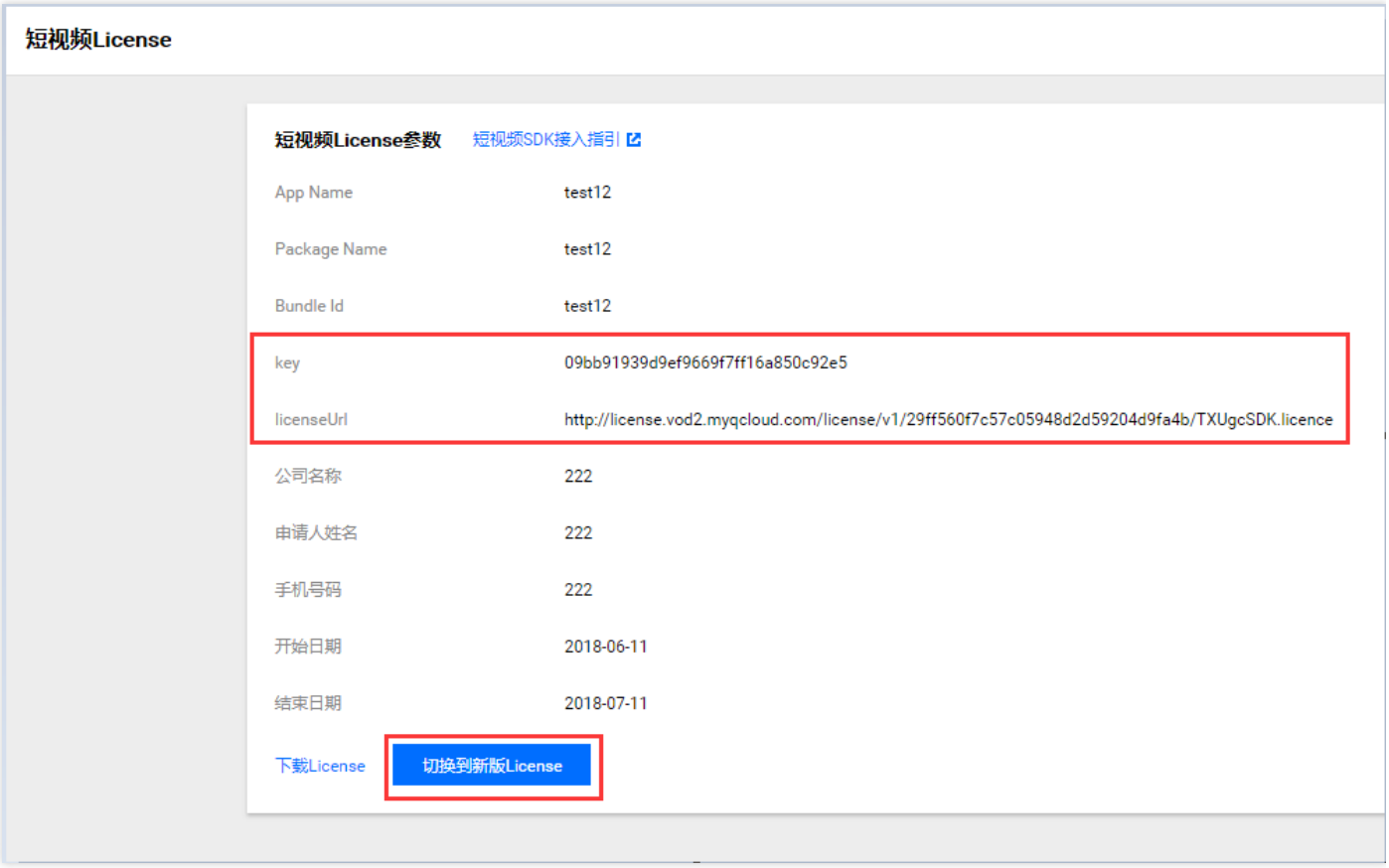
在您的应用中使用短视频功能之前（建议在 application 中）把拿到的 key 和 url 设置到下面接口中

```
TXUGCBase.getInstance().setLicence(context, url, key);
```

- 您可以选择是否打包 licence 到应用中：如果不选择打包，SDK 第一次使用需要访问网络；如果选择打包，把 TXUgcSDK.licence（名称要正确）放到 asset 根目录下即可。
- 当您的 licence 过期了，可以登录腾讯云点播控制台进行续费，SDK 会自动续期，不需要您的应用做任何操作。
- 如果您的 licence 校验失败，您可以调用下面代码来查看 licence 信息是否填写错误。

```
TXUGCBase.getInstance().getLicenceInfo();
```

- 对于使用 4.7 版本 licence 的用户，如果您升级了 SDK 到 4.9 版本了，您可以登录控制台，单击下图的 **切换到新版License** 按钮生成对应的 key 和 url，按照上述操作集成即可。



系统要求

SDK 支持 在 Android 4.0.3（API 15）及以上系统上运行，但只有（Android 4.3) API 18 以上的系统才能开启硬件编码。

开发环境

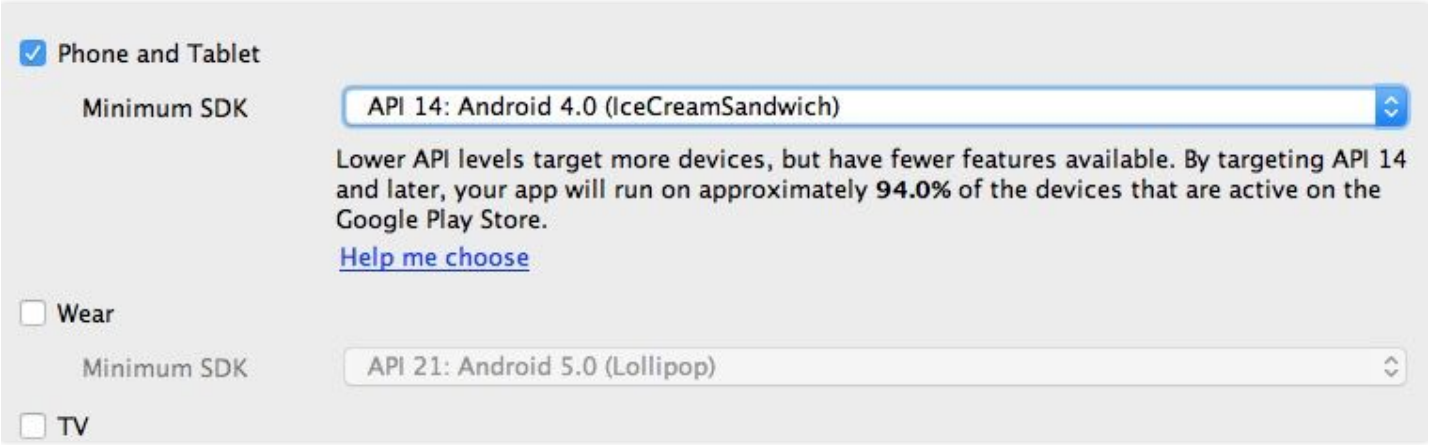
以下是 SDK 的开发环境，App 开发环境不需要与 SDK 一致，但要保证兼容：

- Android NDK: android-ndk-r12b
- Android SDK Tools: android-sdk_25.0.2
  - minSdkVersion: 15
  - targetSdkVersion: 21
- Android Studio（推荐您也使用 Android Studio，当然您也可以使用 Eclipse+ADT）

版权所有：腾讯云计算（北京）有限责任公司

集成攻略（aar）

1. 新建工程



2. 拷贝文件

将 aar 包放在工程 libs 目录下即可

3. 工程配置

- 在工程 App 目录下的 build.gradle 中，添加引用 aar 包的代码：

```
dependencies {
compile fileTree(dir: 'libs', include: ['*.jar'])
// 导入短视频SDK aar
compile(name: 'LiteAVSDK_UGC_3.9.2794', ext: 'aar')
}
```

- 在工程目录下的 build.gradle 中，添加 flatDir，指定本地仓库：

```
allprojects {
repositories {
jcenter()
flatDir {
dirs 'libs'
}
}
}
```

- 在工程目录下的build.gradle的defaultConfig里面，指定ndk兼容的架构：

```
defaultConfig {
applicationId "com.tencent.liteav.demo"
minSdkVersion rootProject.ext.minSdkVersion
targetSdkVersion rootProject.ext.targetSdkVersion
versionCode 1
versionName "2.0"

ndk {
abiFilters "armeabi", "armeabi-v7a"
// 如果您使用的是商业版，只能使用 armeabi 架构，即：
// abiFilters "armeabi",
}
}
```

- 最后编译一下工程 Rebuild Project。

集成攻略（jar）

1. 库说明

解压 zip 压缩包后得到 libs目录，里面主要包含 jar 文件和 so 文件，文件清单如下：

jar文件	说明
liteavsdk.jar	小直播 SDK android 核心库

so文件	说明
libliteavsdk.so	小直播 SDK 核心组件
libtxffmpeg.so	ffmpeg 基础库（ijk 版本），用于点播播放功能，解决一些视频格式的兼容问题
libtxplayer.so	ijkplayer 开源库，用于点播播放功能，解决一些视频格式的兼容问题
libtxsdl.so	ijkplayer 开源库，用于点播播放功能，解决一些视频格式的兼容问题

2. 拷贝文件

如果您的工程之前没有指定过 jni 的加载路径，推荐您将刚才得到的 jar 包和 so 库拷贝到 **Demo\app\src\main\jniLibs**目录下，这是android studio 默认的 jni 加载目录。

版权所有：腾讯云计算（北京）有限责任公司

如果您使用的是商业版，那么解压zip包后，除了 jar 包和 so 库增加了以外，还多了assets目录下的文件，这些是动效所需要的，需要全部拷贝到工程的assets目录下，参考 动效变脸->工程配置

3. 工程配置

在工程 App 目录下的 build.gradle 中，添加引用 jar 包和 so 库的代码。

```
dependencies {
compile fileTree(dir: 'libs', include: ['*.jar'])
// 导入腾讯云直播SDK jar
compile fileTree(dir: 'src/main/jniLibs', includes: ['*.jar'])
}
```

短视频发布功能集成

短视频发布是以源代码形式提供的。

1. 文件说明

短视频上传相关库，在**Demo\app\libs**目录下找到用于短视频上传的 jar 包，文件清单如下：

jar文件	说明
cos-xml-android-sdk-1.2.jar	腾讯云对象存储服务（COS）的文件上传包， 此组件用于短视频上传(TXUGCPublish)功能
qcloud-core-1.2.jar	腾讯云对象存储服务（COS）的文件上传包， 此组件用于短视频上传(TXUGCPublish)功能
okhttp-3.2.0	一款优秀的开源 HTTP 组件
okio-1.6.0	一款优秀的开源网络 I/O 组件
xstream-1.4.7.jar	一款优秀的开源序列化组件
fastjson-1.1.62.android.jar	一款优秀的开源 json 组件

短视频上传源代码，在 **Demo\app\src\main\java\com\tencent\liteav\demo\videoupload** 目录下

2.拷贝文件

将刚才得到的 jar 包拷贝到 **Demo\app\src\main\libs** 目录下，这是 android studio 默认的 jar lib 加载目录。
将刚才得到的源代码目录 **videoupload** 拷贝到自己的工程源码目录下，记得修改源代码里的 package 名。

3. 工程配置

在工程 App 目录下的 build.gradle 中，添加引用 jar 包的代码。

```
dependencies {
compile fileTree(dir: 'libs', include: ['*.jar'])
}
```

配置 App 权限

在 AndroidManifest.xml 中配置 App 的权限，音视频类 App 一般需要以下权限：

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_LOGS" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.Camera"/>
<uses-feature android:name="android.hardware.camera.autofocus" />
```

验证

在工程中调用 SDK 接口，获取 SDK 版本信息，以验证工程配置是否正确。

1. 引用SDK

在 MainActivity.java 中引用 SDK 的 class：

```
import com.tencent.rtmp.TXLiveBase;
```

2. 调用接口

在 onCreate 中调用 getSDKVersioin 接口获取版本号：

版权所有：腾讯云计算（北京）有限责任公司

```
String sdkver = TXLiveBase.getSDKVersionStr();
Log.d("liteavsdk", "liteav sdk version is : " + sdkver);
```

3.编译运行

如果前面各步骤都操作正确，demo 工程将顺利编译通过，运行之后将在 logcat 中看到如下 log 信息：
`09-26 19:30:36.547 19577-19577/ D/liteavsdk: liteav sdk version is : 3.9.2794`

减少 APK 体积

整个 SDK 的体积主要来自于 so 文件，这些 so 文件是 SDK 正常运行所依赖的音视频编解码库、图像处理库以及声学处理组件，如果短视频 SDK 的功能不是 App 的核心功能，您可以考虑采用在线加载的方式减少最终 apk 安装包的大小。

1. 上传 SO 文件

将 SDK 压缩包中的 so 文件上传到 COS，并记录下载地址，比如 `http://xxx-appid.cossh.myqcloud.com/so_files.zip` 。

2. 启动准备

在用户启动 SDK 相关功能前，比如开始播放视频之前，先用 loading 动画提示用户“正在加载相关的功能模块”。

3. 下载 SO 文件

在用户等待过程中，App 就可以到 `http://xxx-appid.cossh.myqcloud.com/so_files.zip` 下载 so 文件，并存入应用目录下（比如应用根目录下的 files 文件夹），为了确保这个过程不受运营商 DNS 拦截的影响，请在文件下载完成后校验 so 文件的完整性。

4. 加载 SO 文件

等待所有 so 文件就位以后，调用 TXLiveBase 的 setLibraryPath 将下载的目标 path 设置给 SDK， 然后再调用 SDK 的相关功能。之后，SDK 会到这些路径下加载需要的 so 文件并启动相关功能。

LOG 打印

在 TXLiveBase 中可以设置 log 是否在控制台打印以及 log 的级别，具体代码如下：

- **setConsoleEnabled**
  设置是否在 Android Studio 的控制台打印 SDK 的相关输出。
- **setLogLevel**
  设置是否允许 SDK 打印本地 log，SDK 默认会将 log 写到 sdcard 上的 **log / tencent / liteav** 文件夹下。

版权所有：腾讯云计算（北京）有限责任公司

如果您需要我们的技术支持，建议将次开关打开，在重现问题后提供 log 文件，非常感谢您的支持。

- **Log 文件的查看**
  小直播 SDK 为了减少 log 的存储体积，对本地存储的 log 文件做了加密，并且限制了 log 数量的大小，所以要查看 log 的文本内容，需要使用 log 解压缩工具。

```
TXLiveBase.setConsoleEnabled(true);
TXLiveBase.setLogLevel(TXLiveConstants.LOG_LEVEL_DEBUG);
```

常见问题排查

如果您将 SDK 导入到您的工程，编译运行出现类似以下错误：

```
Caused by: android.view.InflateException:
Binary XML file #14:Error inflating class com.tencent.rtmp.ui.TXCloudVideoView
```

可以按照以下流程来排查问题：

- 确认是否已经将 SDK 中的 jar 包和 so 库放在 jniLibs目录下。

- 如果您使用 aar 集成方式的完整版本，在工程目录下的 build.gradle 的 defaultConfig 里面确认下是否将 x64 架构的 so 库过滤掉。因为完整版本中连麦功能所使用的声学组件库暂时不支持 x64 架构的手机。

```
defaultConfig {
applicationId "com.tencent.liteav.demo"
minSdkVersion rootProject.ext.minSdkVersion
targetSdkVersion rootProject.ext.targetSdkVersion
versionCode 1
versionName "2.0"

ndk {
abiFilters "armeabi", "armeabi-v7a"
// 如果您使用的是商业版，只能使用 armeabi 架构，即：
// abiFilters "armeabi",
}
}
```

- 检查下混淆规则，确认已将 SDK 的相关包名加入了不混淆名单。

```
-keep class com.tencent.** { *; }
```