智能扫码

SDK 文档

腾讯云

文档目录

SDK 文档
Demo 体验

最近更新时间：2024-09-04 16:38:52

智能扫码目前提供了 Android 端 App 让您体验扫码功能，欢迎安装体验。

Android

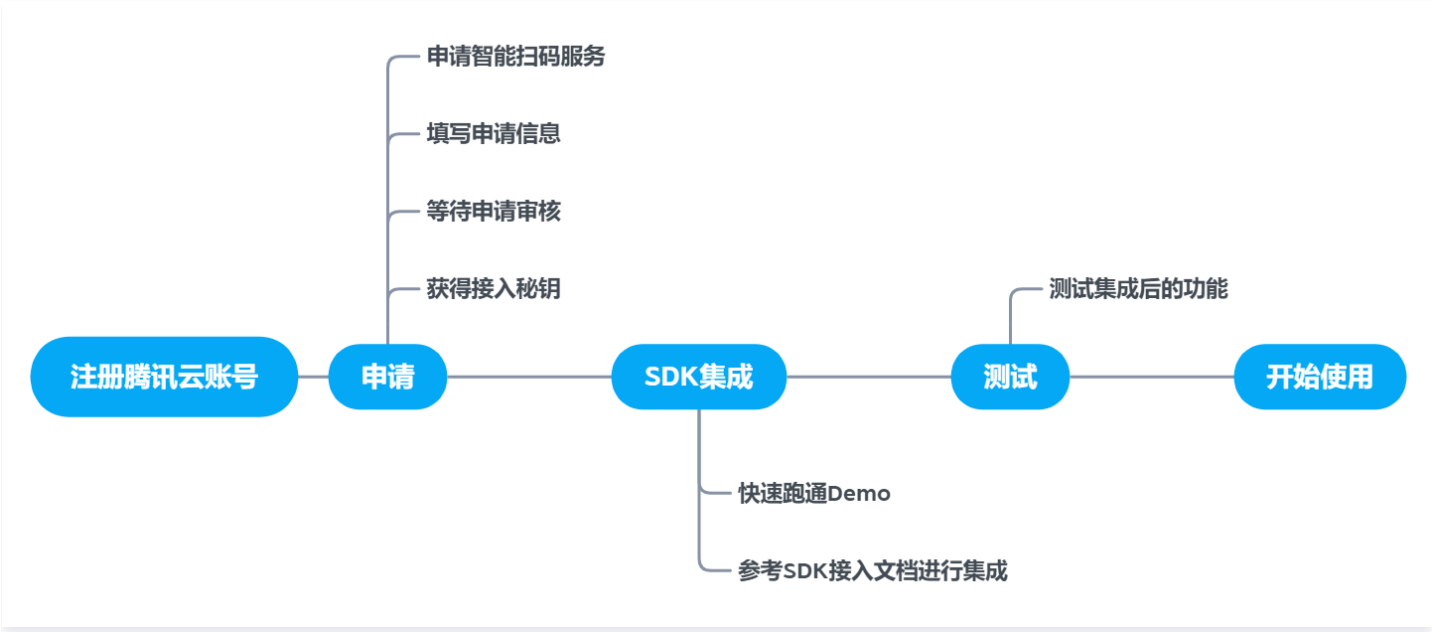
Android 端智能扫码页面

接入准备

最近更新时间：2024-09-04 17:28:06

您需要先 注册腾讯云账号，并 申请 开通智能扫码服务。等待审核通过后，您将获得专属密钥和 Demo 工程文件下载链接。您需要将 Demo 工程文件下载到您的本地环境。

接入流程



一分钟跑通 Demo
跑通 Demo（iOS）

最近更新时间：2024-09-04 15:38:01

准备工作

请先按照 接入准备 所述流程下载好 Demo 工程文件。

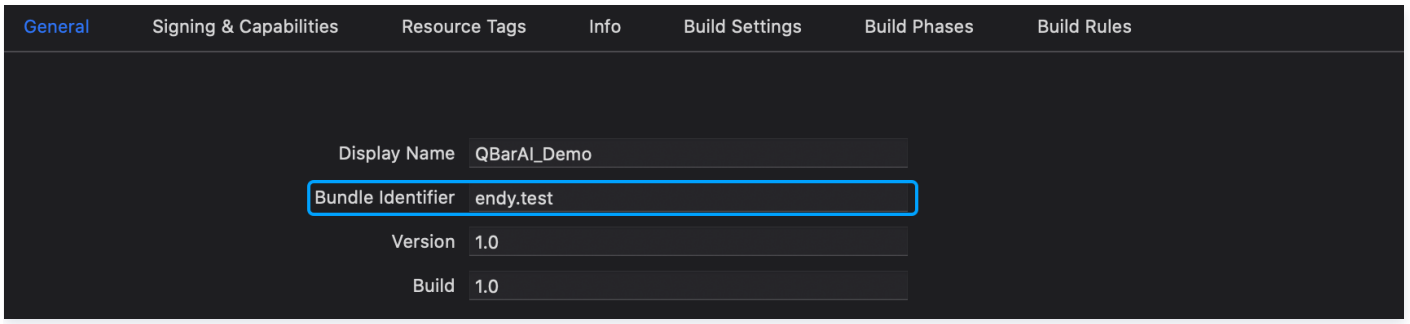
步骤1：设置密钥

使用 xcode 打开下载好的 Demo 工程文件，在 HomeViewController 的 SECRET_ID 和 SECRET_KEY 中填入您的专属密钥。

```
static NSString* const SECRET_ID     = @"";
static NSString* const SECRET_KEY    = @"";
```

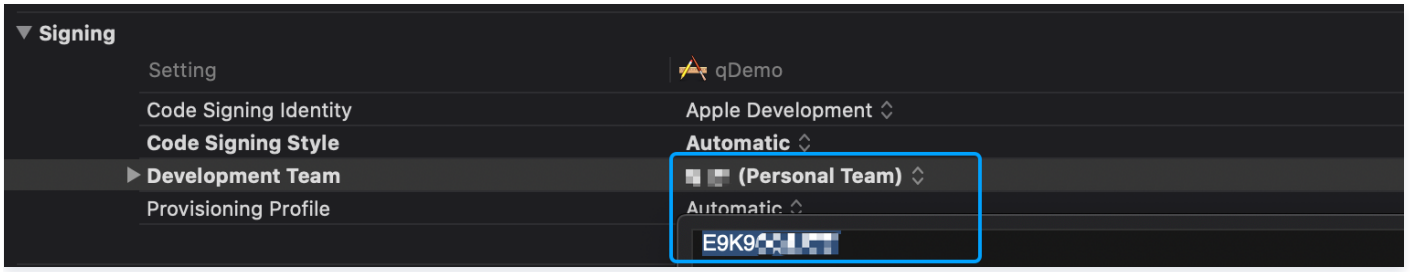
步骤2：更改 Bundle ID

找到 **TARGETS -> General ->Identity**，将其中 Bundle Identifier 的内容修改为申请密钥时填写的 iOS Bundle id。



步骤3：更改 Development Team

找到 **TARGETS -> Build Settings ->Signing**，将其中 Development Team 的内容修改为申请密钥时填写的 iOS DEVELOPMENT_TEAM。



步骤4：运行真机调试

跑通 Demo（Android）

最近更新时间：2024-09-04 15:37:54

准备工作

请先按照 接入准备 所述流程下载好 Demo 工程文件。

步骤1：设置密钥

使用 Android Studio 打开下载好的 Demo 工程文件，在 MainActivity 的 SECRET_ID和SECRET_KEY 中填入您的专属密钥。

```
//控制台申请秘钥
private static final String SECRET_ID = "";
private static final String SECRET_KEY = "";
```

步骤2：设置 App 签名文件

在 App 模块下的 build.gradle 文件中，设置 signingConfigs。storeFile file 里填入您的 App 签名文件的路径（**注意：该签名文件要与您在控制台申请密钥时用来获取哈希值的签名文件相一致**），并填入对应的 storePassword、keyAlias 和 keyPassword 的值。

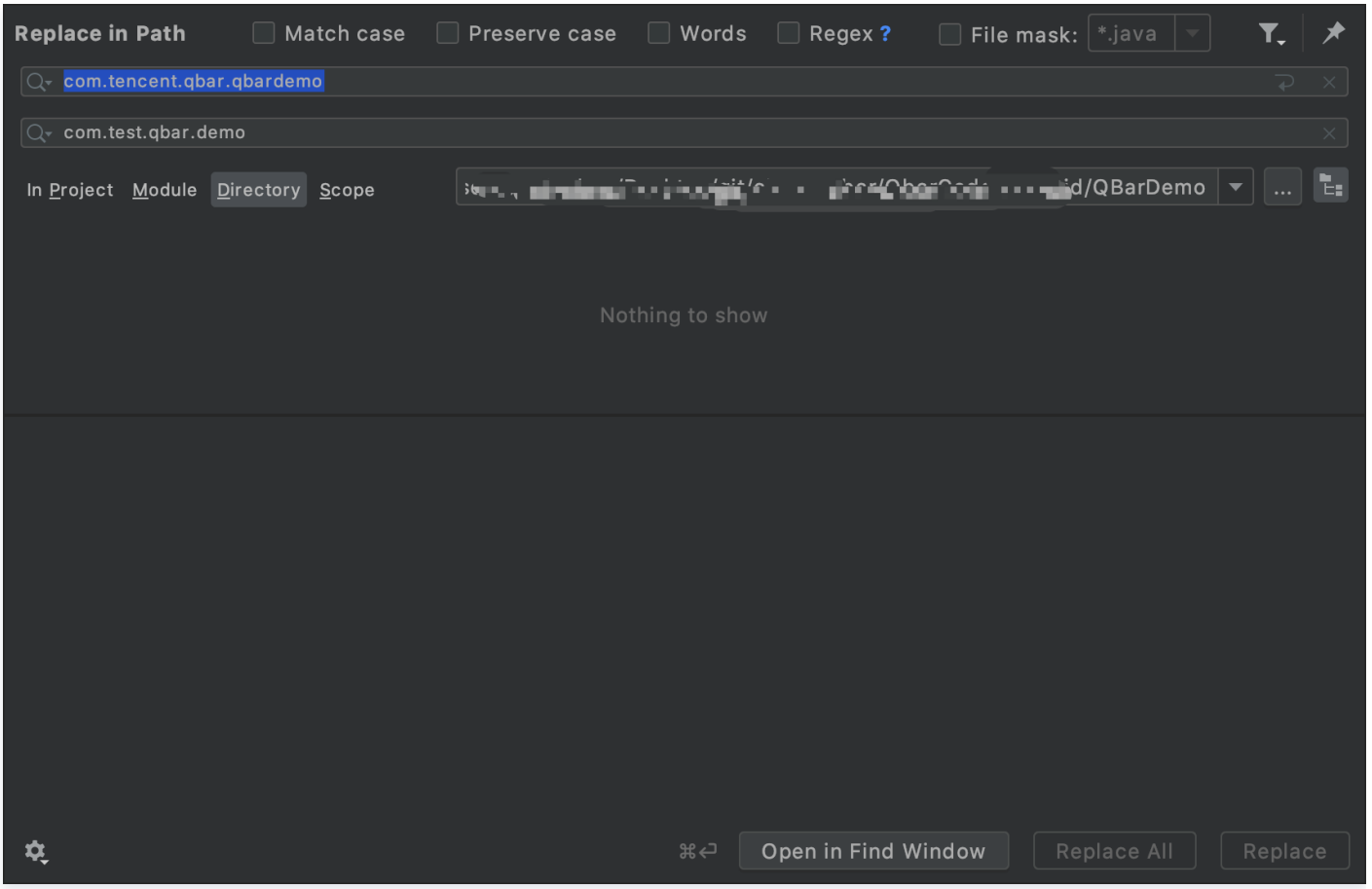
```
signingConfigs {
    mysign {
        storeFile file('')
        storePassword ''
        keyAlias = ''
        keyPassword ''
    }
    debug {
        storeFile file('')
        storePassword ''
        keyAlias = ''
        keyPassword ''
    }
}
```

步骤3：替换 App 包名

需要更改 Demo 工程文件的包名为您申请密钥时使用的包名，然后重新 build 工程。步骤如下：

1. 在 AndroidManifest.xml 更改 package 值为您申请密钥时填写的 App 包名。
2. 在工程的 java 目录下新建对应包名目录结构的目录，例如您的包名为 aaa.bbb.ccc 则在 java 下新建目录结构为：
    - java
        - aaa
            - bbb
                - ccc
3. 将 Demo 工程文件原始 java 目录下最底层的一个子目录的文件复制到上述新建目录的最底层目录里，即 ccc 下。
4. 确认复制过去的文件头部的包名是您的 App 包名，如果不是，可以在项目或者 module 右键，单击 replace in path，替换文件中的包名。

5. 删除掉原本 Demo 工程文件里 java 目录下的子目录及相关文件，重新 build 整个项目。



步骤4：运行真机调试

一分钟集成 SDK
快速集成（iOS）

最近更新时间：2024-12-13 09:38:32

开发准备

1. 注册腾讯云账号，提交智能扫码服务的申请，等待审核通过后，获得专属密钥。（申请地址：智能扫码申请）
2. 从 SDK 下载链接中下载智能扫码 SDK 到本地准备集成。

iOS 端智能扫码 SDK 接入流程

iOS 端智能扫码 SDK 介绍

SDK中包含了三个文件和一个文件夹，分别是 libQBarCode.a、ncnn.framework、openmp.framework、QbarCodeRes.bundle 和 include 包含的头文件。

- libQBarCode.a – 静态文件，是主要的扫码逻辑库。
- ncnn.framework与openmp.framework– 高性能神经网络前向计算框架。
- QbarCodeRes.bundle – 资源文件。
- include – 包含接入所需头文件。

环境依赖

- 当前 iOS 端智能扫码SDK版本适用于 IOS 9.0及以上的版本。
- Xcode 使用11.0及以上版本开发集成，建议使用最新版。

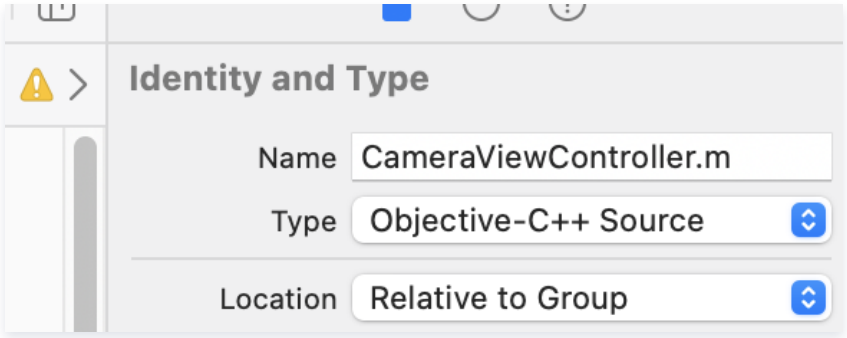
接入步骤

1. 将ncnn.framework、openmp.framework、libQBarCode.a、QbarCodeRes.bundle添加至项目中，并且 include 头文件也添加到项目中。
2. 引入系统库 framework。

```
├── CoreMedio.framework
├── AVFoundation.framework
├── UIKit.framework
├── AVFoundation.framework
└── UIKit.framework
```

3. 添加编译选项

- 调用 SDK 页面设置为 **Objective-C++ Source**。



- 将 **bitCode** 设置为 **NO(不支持bitcode)**。
- 将 **Other Linker Flags** 添加 **-ObjC**。

4. 权限设置

智能扫码 SDK 需要手机网络、 摄像头、访问相册的使用权限，请添加对应的权限声明。

```
<key>Privacy - Camera Usage Description</key>
<string>扫码需要开启您的摄像头权限，用于识别</string>
<key>Privacy - Photo Library Usage Description</key>
<string>扫码需要您开启相册权限，浏览您的照片</string>
```

SDK 接口说明

SDK 初始化：

用户初始化智能扫码 SDK，SECRET_ID 与 SECRET_KEY 传入云服务后台申请的密钥信息（申请地址：智能扫码申请），同时需要导入 QBarCodeKit.h、QBarSDKUIConfig.h 和 QbarCodeRes.bundle 资源文件。

```
#import "QBarCodeKit.h"

static NSString* const SECRET_ID     = @""; // SECRET_ID 信息
static NSString* const SECRET_KEY    = @""; // SECRET_KEY 信息
static const NSString *ERRCODE = @"errorcode"; // 结果错误码对应的Key
static const NSString *ERRMSG = @"errormsg"; // 错误信息对应的Key
static const NSString *CONTENT = @"content"; // 识别结果信息对应的Key


[sdk initQBarCodeKit:SECRET_ID secretKey:SECRET_KEY teamId:teamId resultHandle:^(NSDictionary * _Nonnull
resultDic) {
  NSNumber *errCode = resultDic[ERRCODE]; // errorCode 为0 授权验证通过
  int code = [errCode intValue];
  NSString *errMsg= resultDic[ERRMSG];  // 获取初始化失败信息
}];
```

摄像头数据实时识别：

智能扫码 SDK 提供摄像头实时数据扫码功能，满足您在自定义 UI 布局中的使用需求。

```
/**
  摄像头视频流数据解码
  sampleBuffer 桢数据
*/
[qBarCodeKit qBarDecodingWithSampleBuffer:sampleBuffer resuldHandle:^(NSDictionary * _Nonnull resultDic) {
  NSString *msg = resultDic[CONTENT];
  NSLog(@"qBarDecodingWithSampleBuffer result: %@",msg);
}];
```

默认扫描界面识别：（推荐）

如果您只关注扫码功能不需支持自定义 UI 界面，可以使用智能扫码 SDK 内自带的默认界面完成扫描操作。

```
[sdk startDefaultQBarScan:self withResult:^(NSDictionary * _Nonnull resultDic) {
       NSString *msg = [self convertToJsonData:resultDic];
 }];
```

传入图片识别：

除了主动扫描以外，智能扫码 SDK 还支持图片识别功能，只需传入需要识别的图像：

```
/**
   image  需解码图片 图片大小建议小于1M
*/
[qBarCodeKit decodeImageWithQBar:image resultHandler:^(NSDictionary * _Nonnull resultDic) {
   NSString *msg = resultDic[CONTENT];
   NSLog(@"decodeImageWithQBar result: %@",msg);
}];
```

数据声明

iOS 识别的数据结果也就是从 CONTENT 中获取的值为 Json 格式，先举例说明其内容格式：

```
{
  "charset":"ISO8859-1",
  "data":"https://cloud.tencent.com/",
  "typeName":"QR_CODE"
}
```

其中 charset 表示内容信息的字符集，data 表示扫码得到的内容信息，typeName 表示扫码类型如条形码、二维码。

快速集成（Android）

最近更新时间：2024-09-04 15:47:58

开发准备

1. 注册腾讯云账号，提交智能扫码服务的申请，等待审核通过后，获得专属密钥。（申请地址：智能扫码申请）
2. 从 SDK 下载链接中下载智能扫码 SDK 到本地准备集成。

Android 端智能扫码 SDK 接入流程

Android 端智能扫码 SDK 介绍

SDK 文件为 **QBarCode-v0.1.2.aar**，该文件里面封装了智能扫码接口、so 文件、检测与超分模型的资源文件。
功能：提供实时识别一维码、二维码和图片内一、二维码检测识别的服务。

环境依赖

当前 Android 端智能扫码 SDK 适用于 API 19 (Android 4.4) 及以上版本。

接入步骤

1. 将 **QBarCode-v0.1.2.aar** 包添加到您的工程文件中的**libs**目录下。
2. 配置 **build**
   在您的工程文件中的 **build.gradle** 中进行如下配置：

```
//使用 aar 时必须要设置
android {
    //**
    defaultConfig {
       /***/
        ndk {
            abiFilters "armeabi-v7a","arm64-v8a"
        }
    }
  sourceSets {
        main {
          jniLibs.srcDirs = ['libs/jniLibs']
        }
    }
}
dependencies {
    implementation files('libs/QBarCode-v0.1.2.aar')
    // 智能扫码依赖第三方库
    implementation 'commons-io:commons-io:2.6'
}
```

3. 权限申请
   需要在 AndroidManifest.xml 文件中声明权限

```
<!--摄像头使用权限-->
<uses-feature android:name="android.hardware.camera" />
<uses-permission
    android:name="android.permission.CAMERA"
    android:required="true" />
<!--文件存储使用权限-->
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERNET"/>
```

对于需要兼容 Android 6.0 以上的用户，以上权限除了需要在 AndroidManifest.xml 文件中声明权以外，还需使用代码动态申请权限。

SDK 接口说明

sdk 初始化：

用户初始化智能扫码 SDK，SECRET_ID 与 SECRET_KEY 传入云服务后台申请的密钥信息（申请地址：智能扫码申请）

```
    private QBarCodeKit qBarCodeKit;

    //Android 6及以上动态申请权限，权限通过后再初始化
    qBarCodeKit = QBarCodeKit.getInstance();
    qBarCodeKit.initQBarCodeKit(SECRET_ID, SECRET_KEY, MainActivity2.this, new
QBarCodeKit.OnSdkKitInitCallback (){
      @Override
      public void onInitResult(String errCode, String errMsg) {
        //initAuth 执行时间可能有1-2s，当返回SUCCESS(0) 说明授权成功，再进行后面的操作
        Log.d("onInitResult: ", "errCode:" + errCode + " errMsg:" + errMsg);
      }
});
```

摄像头数据实时识别：

扫描 SDK 提供了 ScanCodeDetectView，用来方便您在自定义的 UI 界面里使用智能扫描功能。首先您需要在 UI 界面的布局文件中添加 ScanCodeDetectView：

```
<com.tencent.scanlib.ui.ScanCodeDetectView
    android:id="@+id/scan_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:show="false"/><!--show 设置false-->
```

然后只需要在对应的界面代码中为 ScanCodeDetectView 设置结果回调即可。

```
scanView.setScanCallBack(new QBarSdkCallback() {
  @Override
  public void onIdentityResult(ScanResult result) {
    String data = result.getData(); // 识别内容
    String charset = result.getTypeName(); // 内容信息字符集
    String typeName = result.getCharset(); // 扫码类型
  }
});
scanView.onCreate(); // 构建ScanCodeDetectView
```

ScanCodeDetectView 的生命周期如下，请在对应的生命周期阶段进行调用。

```
//scanView Life cycle
scanView.onCreate();
scanView.onResume();
scanView.onPause();
scanView.onStop();
scanView.onDestroy();
```

默认扫描界面识别：

如果您只关注扫码功能不需要支持自定义 UI 界面，可以使用智能扫码 SDK 内自带的默认界面完成扫描操作。

```
qBarCodeKit.startDefaultQBarScan(MainActivity.this, new QBarSdkCallback() {
  @Override
```

```
  public void onIdentityResult(ScanResult result) {
    String data = result.getData(); // 识别内容
    String charset = result.getTypeName(); // 内容信息字符集
    String typeName = result.getCharset(); // 扫码类型
  }

  @Override
  public void onFail(final int errorCode, final String errorMsg) {
         // 智能扫码的错误
    runOnUiThread(new Runnable() {
        @Override
      public void run() {
          Toast.makeText(MainActivity.this, "code: " + errorCode + " msg: " + errorMsg,
                         Toast.LENGTH_SHORT).show();
      }
    });
});
```

传入图片识别：

除了主动扫描以外，智能扫码 SDK 还支持图片识别功能，只需传入需要识别的图像即可：

```
/*
 * bitmap 图片大小建议小于1M，避免OOM
 */
List<ScanResult> results = qBarCodeKit.decodeImageWithQBar(bitmap, MainActivity.this);
```

图片识别功能具体使用范例如下：

```
   /*
   * bitmap 获取
   * filePath 源图片路径
   */
    BitmapFactory.Options sizeOptions = new BitmapFactory.Options();
    sizeOptions.inJustDecodeBounds = true;
    BitmapFactory.decodeFile(filePath, sizeOptions);
    BitmapFactory.Options decodeOptions = new BitmapFactory.Options();
    if (sizeOptions.outWidth * sizeOptions.outHeight * 3 > 10 * 1024 * 1024) {
        Log.i("MainActivity2", String.format("bitmap too large %d x %d, sample",sizeOptions.outWidth,
sizeOptions.outHeight));
        decodeOptions.inSampleSize = 2;
    }
    Bitmap bitmap = BitmapFactory.decodeFile(filePath, decodeOptions);
    /*
    * bitmap 图片大小建议小于1M，避免OOM
    */
   List<ScanResult> results = qBarCodeKit.decodeImageWithQBar(bitmap, MainActivity.this);
   for (ScanResult result : results) {
     String data = result.getData(); // 识别内容
     String charset = result.getTypeName(); // 内容信息字符集
     String typeName = result.getCharset(); // 扫码类型
   }
```

混淆规则配置

如果您的应用开启了混淆功能，为确保扫描 SDK 的正常使用，请把以下部分添加到您的混淆文件中。

```
-keep public interface com.tencent.scanlib.camera.Auth$OnInitCallback { *; }
```

```
-keep public interface com.tencent.scanlib.decoder.FileDecodeQueue$FileDecodeCallBack { *; }
-keep public interface com.tencent.scanlib.ui.ScanCodeView$ScanCallBack { *; }
-keep class com.tencent.scanlib.decoder.FileDecodeQueue { *; }
-keep class com.tencent.scanlib.decoder.FileDecodeQueue$* { *; }
-keep class com.tencent.scanlib.decoder.QBar$QBarResult { *; }
-keep class com.tencent.scanlib.decoder.QBarConstants { *; }
-keep class com.tencent.scanlib.ui.ScanCodeView { *; }
-keep class com.tencent.cloud.auth.lib.Error {*;}
-keep class com.tencent.qbar.** {*;}
-keep class com.tencent.scanlib.kit.** {*;}
-keep class com.tencent.cloud.auth.lib.Jni$AuthResult {*;}
-keep class com.tencent.scanlib.model.ScanResult {*;}
-keep class com.tencent.scanlib.model.ScanResultWithDetect {*;}
```

客户端 API 文档
iOS API 概览

最近更新时间：2024-12-13 09:38:32

智能扫码的 API 主要涉及 QBarCodeKit 对象，下面对其支持的 API 作出说明。

QBarCodeKit

API	功能描述
sharedInstance	获得 QBarCodeKit 的单例对象
getVersion	获得 SDK 的版本信息
initQBarCodeKit	初始化 SDK，并完成鉴权认证
decodeImageWithQBar	可以识别传入图片中存在的二维码、条形码信息
qBarDecodingWithSampleBuffer	通过摄像头拍摄流进行二维码扫描
startDefaultQBarScan	启动 SDK 提供的默认界面进行扫码
setViewConfig	设置界面配置信息

sharedInstance

```
+ (instancetype) sharedInstance
```

- 功能描述：
  获得 QBarCodeKit 的单例对象静态方法。
- 返回结果：
  QBarCodeKit 的单例对象。

getVersion

```
- (NSString *) getVersion
```

- 功能描述：
  获得 SDK 的版本信息。
- 返回结果：
  当前 SDK 的版本信息。

initQBarCodeKit

```
- (void) initQBarCodeKit:(NSString *)secretId secretKey:(NSString *)secretkey teamId:(NSString *)teamId
resultHandle:(resultCodeContent)handle;
```

- 功能描述：
  初始化 SDK，并完成鉴权认证。
- 传入参数：
  - **secretId** 用户在后台申请的 secretId 信息
  - **secretkey** 用户在后台申请后获取的专属密钥信息
  - **teamId** 用户申请所填写的 development team id
  - **handle** 用于接收初始化与鉴权认证的结果回调，并将结果放到 QBarResultCodeContent 中。

decodeImageWithQBar

```
- (void) decodeImageWithQBar:(UIImage *)image resultHandler:(resultCodeContent)handle
```

- 功能描述：
  可以识别传入图片中存在的二维码、条形码信息。
- 传入参数：
- image 需要识别的图像信息
- handle 用于图像识别的结果回调，并将结果放到 QBarResultCodeContent 中。

qBarDecodingWithSampleBuffer

```
- (void) qBarDecodingWithSampleBuffer:(CMSampleBufferRef) sampleBuffer resuldHandle:
(QBarResultCodeContent)handle
```

- 功能描述：
  通过摄像头拍摄流进行二维码扫描。
- 传入参数：

```
**sampleBuffer** 摄像头当前拍摄的 Buffer 信息
```

- handle 用于摄像头拍摄扫码的结果回调，并将结果放到 QBarResultCodeContent 中。

startDefaultQBarScan

```
-(void)startDefaultQBarScan:(UIViewController *)vc withResult:(QBarResultCodeContent)resultDict;
```

- 功能描述：
  启动SDK提供的默认界面进行扫码。
- 传入参数：
  vc 启动默认扫码界面时当前界面的 ViewController

setViewConfig

```
- (void) setViewConfig:(QBarSDKUIConfig *)qBarSDKUIConfig
```

- 功能描述：
  设置界面的配置 Config 信息。
- 传入参数：
  QBarSDKUIConfig

属性	含义
naviBarHidde	导航栏是否隐藏
naviBarBgColor	导航栏背景色
naviBarTitle	导航栏标题
naviBarTitleColor	导航栏标题颜色
scanTips	提示语
scanTipsColor	提示语文字颜色
statusBarHidde	是否隐藏状态栏

回调说明

QBarResultCodeContent

```
typedef void (^qBarResultCodeContent)(NSDictionary *resultDic);
```

其中 resultDic 字典包含的字段为：

- **errorcode** number 类型 对应结果的错误码
- **errormsg** NSString 类型 对应结果异常的错误信息
- **content** NSArray 类型 对应存放扫码结果信息 (NSArray 内部是NSSting json 对象)

```
NSArray *content = resultDic[CONTENT];
NSLog(@"content %@",content);
```

Android API 概览

最近更新时间：2024-09-04 16:33:00

智能扫码的 API 主要涉及 QBarCodeKit、ScanCodeDetectView 以及回调接口类，下面对其支持的 API 作出说明。

QBarCodeKit

QBarCodeKit 是智能扫码的对外接口类，智能扫码对外的功能大多在此接口类中包括，基础 API 如下。

API	功能描述
getInstance()	创建 QBarCodeKit 的单例
getVersion()	获取 SDK 当前的版本号
initQBarCodeKit()	初始化 SDK，并完成鉴权认证
startDefaultQBarScan()	启动 SDK 提供的默认界面进行扫码
decodeImageWithQBar()	可以识别传入图片中存在的二维码、条形码信息

getInstance()

```
public static QBarCodeKit getInstance()
```

功能描述：
创建 QBarCodeKit 的单例。
返回结果：
QBarCodeKit 的单例对象。

getVersion()

```
public String getVersion()
```

功能描述：
获取 SDK 当前的版本号。
返回结果：
当前 SDK 的版本信息。

initQBarCodeKit()

```
public void initQBarCodeKit(String secretId, String secretKey, final Context context, final
OnSdkKitInitCallback callback
```

功能描述：
初始化 SDK，并完成鉴权认证操作。
传入参数：
secretId 用户在后台申请的 secretId 信息
secretKey 用户在后台申请后获取的专属密钥信息
context 当前环境的上下文信息
callback 初始化与鉴权的结果回调类 OnSdkKitInitCallback

startDefaultQBarScan()

```
public void startDefaultQBarScan(Activity context, QBarSdkCallback callback)
```

功能描述：
启动 SDK 提供的默认界面进行扫码，并通过回调获取扫码结果。

传入参数：
context 调用该函数界面的 Activity 对象用来启动默认扫码界面
callback 用来接收扫码结果的回调类 QBarSdkCallback 。

decodeImageWithQBar()

```
public List<ScanResult> decodeImageWithQBar(Bitmap bitmap, Context context)
```

功能描述：
识别传入图片中存在的二维码、条形码信息。
传入参数：
bitmap 需要识别的图像信息
context 当前应用的上下文信息
返回结果：
List ScanResult 图片识别结果的列表
ScanResult 包含属性及含义：
String data 识别到的内容信息
String charset 内容包含的字符集
String typeName 所扫描图片包含的二维码类型

ScanCodeDetectView

ScanCodeDetectView 是智能扫码 SDK 提供的一个支持扫码的 View 组件，主要目的是为了满足用户将扫码功能嵌入到自定义界面的需求。基本 API 介绍如下。

API	功能描述
setScanCallBack()	为 ScanCodeDetectView 设置扫码回调接收类
setScanTipsTVText()	主动在 ScanCodeDetectView 上显示 tips 信息
showNoContentResult()	主动在 ScanCodeDetectView 上显示无识别结果提示
onCreate()	生命周期 onCreate 对应方法
onResume()	生命周期 onResume 对应方法
onPause()	生命周期 onPause 对应方法
onStop()	生命周期 onStop 对应方法
onDestroy()	生命周期 onDestroy 对应方法

setScanCallBack()

```
public void setScanCallBack(QBarSdkCallback callback)
```

功能描述：
为 ScanCodeDetectView 设置扫码回调接收类。
传入参数：
callBack 识别结果接收回调类 QBarSdkCallback 。

setScanTipsTVText()

```
public void setScanTipsTVText(String text)
```

功能描述：
主动在 ScanCodeDetectView 上显示 tips 信息。
传入参数：
text 需要显示 tips 的信息。

showNoContentResult()

```
public void showNoContentResult()
```

功能描述：
主动在 ScanCodeDetectView 上显示无识别结果的提示。

onCreate()

```
public void onCreate()
```

功能描述：
生命周期 onCreate 对应方法，需在界面的对应生命周期函数内调用。

onResume()

```
public void onResume()
```

功能描述：
生命周期 onResume 对应方法，需在界面的对应生命周期函数内调用。

onPause()

```
public void onPause()
```

功能描述：
生命周期 onPause 对应方法，需在界面的对应生命周期函数内调用。

onStop()

```
public void onStop()
```

功能描述：
生命周期 onStop 对应方法，需在界面的对应生命周期函数内调用。

onDestroy()

```
public void onDestroy()
```

功能描述：
生命周期 onDestroy 对应方法，需在界面的对应生命周期函数内调用。

回调类说明

OnSdkKitInitCallback

智能扫码初始化接口的回调类，接收初始化鉴权认证的结果。

```
/**
 * The interface On sdk kit init callback.
 */
public interface OnSdkKitInitCallback {
    /**
     * On init result.
     *
     * @param errCode the error code
     * @param errMsg  the error msg
     */
```

```
    void onInitResult(String errCode, String errMsg);
}
```

QBarSdkCallback

智能扫码使用默认界面扫码的回调类，接收扫码结果信息。

```
/**
 * The interface Q bar sdk callback.
 */
public interface QBarSdkCallback {

    /**
     * 如果只一个码，解码结果回调
     *
     * @param result 数据
     */
    void onIdentityResult(ScanResult result);

    /**
     * 扫码过程中出现的异常
     * @param errorCode 错误码
     * @param errorMsg 错误信息
     */
    void onFail(int errorCode, String errorMsg);
}
```

ScanResult 包含属性及含义：

- **String data** 识别到的内容信息。
- **String charset** 内容包含的字符集。
- **String typeName** 所扫描图片包含的二维码类型。

错误码

最近更新时间：2024-09-04 17:29:08

错误码参考

errCode	errMsg	说明
0	CODE_SUCCESS	成功
1	CODE_LOCAL_TIME_NOT_CORRECT	当地时间不正确
2	CODE_GET_SERVER_TIME_FAIL	获取服务器时间失败
3	CODE_ONLINE_GET_LICENSE_FAIL	在线获取许可证失败
4	CODE_LOCAL_LICENSE_FILE_NOT_EXISTS	本地许可文件不存在
5	CODE_WRITE_LICENSE_FILE_ERROR	写许可文件错误
6	CODE_DECODE_LICENSE_ERROR	许可错误
7	CODE_READ_LICENSE_ERROR	读取许可错误
8	CODE_MAKE_LICENSE_REQUEST_ERROR	许可请求错误
9	CODE_MAKE_TIME_REQUEST_ERROR	时间请求错误
10	CODE_DEVICE_ID_NOT_MATCH	设备 ID 不匹配
11	CODE_PACKAGE_NAME_NOT_MATCH	包名称不匹配
12	CODE_PACKAGE_SIGNATURE_NOT_MATCH	包明或签名不匹配
13	CODE_LICENSE_OUT_OF_DATE	过期许可
14	CODE_NOT_INITED	未初始化