金融级身份认证

身份证 OCR 识别接入

产品文档

腾讯云

文档目录

身份证 OCR 识别接入
SDK 接入
生成签名
Android SDK 接入
开发准备
配置流程
接口调用
错误码描述
接入示例
iOS SDK 接入
配置流程
接口调用
错误码与多重告警码
公众号接入
生成签名
公众号启动 H5 身份识别
H5 身份证识别结果跳转
小程序接入
小程序接入身份证 OCR 识别
启动小程序身份证识别
小程序身份证 OCR 识别结果跳转
验证结果
方式一：前端获取结果验证签名
方式二：合作伙伴服务端查询结果

身份证 OCR 识别接入
SDK 接入
生成签名

最近更新时间：2018-07-27 17:19:29

准备步骤

- **前置条件：请合作方确保 NONCE ticket 已经正常获取，获取方式见 NONCE ticket 获取 。**
- 合作方为身份证 OCR 识别服务生成签名，需要具有以下参数：
- 参与签名的数据需要和使用该签名的sdk中的请求参数保持一致。

参数名	说明	来源
appId	腾讯云线下对接分配的 App ID	腾讯云线下对接分配
userId	用户唯一标识	合作方自行分配（与 SDK 里面定义的 userId 值保持一致）
version	参数值为：1.0.0	-
ticket	合作伙伴服务端缓存的 ticket，注意是 NONCE 类型	获取方式见 NONCE ticket 获取（所用的 userId 参数值需要和 SDK 里面定义的 userId 值保持一致）
nonceStr	必须是 32 位随机数	合作方自行生成（与 SDK 里面定义的随机数保持一致）

基本步骤

1. 生成一个 32 位的随机字符串 nonceStr（其为字母和数字，登录时也要用到）。
2. 将 appId、userId、version 连同 ticket、nonceStr 共五个参数的值进行字典序排序。
3. 将排序后的所有参数字符串拼接成一个字符串。
4. 将排序后的字符串进行 SHA1 编码，编码后的 40 位字符串作为签名（sign）。

注意：

签名算法可参考 签名算法说明 。

参考示例

请求参数：

参数名	参数值
appId	TIDA0001
userId	userID19959248596551
nonceStr	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
ticket	XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

字典排序后的参数为：

[1.0.0, TIDA0001, XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS ， kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551]

拼接后的字符串为：

1.0.0TIDA0001XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7TuserID19959248596551

计算 SHA1 得到签名：

4AE72E6FBC2E9E1282922B013D1B4C2CBD38C4BD

该字串就是最终生成的签名（40 位），不区分大小写。

下一步：

Android SDK 接入

iOS SDK 接入

Android SDK 接入
开发准备

最近更新时间：2018-06-20 17:44:49

权限检测

SDK 需要用到相机权限 / 读取手机信息 / 写 SD 卡权限。

1. Android 6.0 以上系统
   SDK 运行时检测权限，需要用户授权。
2. Android 6.0 以下系统
   Android 没有运行时权限，检测权限只能靠开关相机进行。考虑到 SDK 的使用时间很短，快速频繁开关相机可能会导致手机抛出异常，故 SDK 内对 Android 6.0 以下手机没有做权限的检测。为了进一步提高用户体验，在 Android 6.0 以下系统上，**我们建议合作方在拉起 SDK 前，帮助 SDK 做相机 / 读取手机信息 / 写 SD 卡权限权限检测，提示用户确认打开了这三项权限后再进行身份证 OCR 识别**，可以使整个身份证识别体验更快更好。
3. armeabi-v7a 平台
   目前 SDK 只支持 armeabi-v7a 平台，为了防止在其他 CPU 平台上 SDK Crash，我们建议在您的 App 的 build.gradle 里加上 abiFilter，如下图中红框所示：

> **注意：**
> 有且只有 armeabi-v7a 平台。

```
defaultConfig {
    applicationId "com.webank.testcloud.cloudfacetest"
    minSdkVersion 14
    targetSdkVersion 23
    versionCode 1
    versionName "1.0"

    ndk {
        // 设置支持的SO库架构
        abiFilters 'armeabi-v7a'
    }
}
```

下一步：配置流程

配置流程

最近更新时间：2019-01-11 16:06:07

1. 接入配置

云 OCR SDK（WbCloudOcr）最低支持到 ** Android API 14: Android 4.0(ICS) ，**请在构建项目时注意。**
云 OCR SDK 将以 AAR 文件的形式提供。另外 OCR SDK 同时需要依赖云公共组件 WBCloudNormal，同样也是以 AAR 文件的形式提供。
需要添加下面文档中所示的依赖（将提供的 AAR 文件加入到 App 工程的 `libs` 文件夹下面，
并且在 **build.gradle 中添加下面的配置）：**

```
android{
//...
repositories {
flatDir {
dirs 'libs' //this way we can find the .aar file in libs folder
}
}
}
//添加依赖
dependencies {
//0. appcompat-v4
compile 'com.android.support:appcompat-v4:23.1.1'
//1. 云OCR SDK
compile(name: 'WbCloudOcrSdk', ext: 'aar')
//2.云公共组件
compile(name: 'WbCloudNormal', ext: 'aar') }
```

2. 混淆配置

云 OCR 产品的混淆规则分为三部分，分别是云 OCR SDK 的混淆规则，云公共组件的混淆规则及依赖的第三方库混淆规则。

云 OCR SDK 的混淆规则

```
#########################云 ocr 混淆规则 ocr-BEGIN##############################
-keepattributes InnerClasses
-keep public class com.webank.mbank.ocr.WbCloudOcrSDK{
```

```
public <methods>;
public static final *;
}
-keep public class com.webank.mbank.ocr.WbCloudOcrSDK$*{
*;
}

-keep public class com.webank.mbank.ocr.tools.ErrorCode{
*;
}

-keep public class com.webank.mbank.ocr.net.*$*{
*;
}
-keep public class com.webank.mbank.ocr.net.*{
*;
}
##########################云 ocr 混淆规则 ocr-END#################################
```

您可以将如上代码拷贝到您的混淆文件中，也可以将 SDK 中的 webank-cloud-ocr-proguard-rules.pro 拷贝到主工程根目录下,然后通过 -include webank-cloud-ocr-rules.pro 加入到您的混淆文件中。

云公共组件的混淆规则

```
##########################webank normal混淆规则-BEGIN#################################
#不混淆内部类
-keepattributes InnerClasses
-keepattributes *Annotation*
-keepattributes Signature

-keep, allowobfuscation @interface com.webank.normal.xview.Inflater
-keep, allowobfuscation @interface com.webank.normal.xview.Find
-keep, allowobfuscation @interface com.webank.normal.xview.BindClick

-keep @com.webank.normal.xview.Inflater class *
-keepclassmembers class * {
@com.webank.normal.Find *;
@com.webank.normal.BindClick *;
}

-keep public class com.webank.normal.net.*$*{
*;
}
-keep public class com.webank.normal.net.*{
*;
```

```
}
-keep public class com.webank.normal.thread.*{
*;
}
-keep public class com.webank.normal.thread.*$*{
*;
}
-keep public class com.webank.normal.tools.WLogger{
*;
}

#webank normal 包含的第三方库 bugly
-keep class com.tencent.bugly.webank.**{
*;
}

#wehttp 混淆规则
-dontwarn com.webank.mbank.okio.**

-keep class com.webank.mbank.wehttp.**{
public <methods>;
}
-keep interface com.webank.mbank.wehttp.**{
public <methods>;
}
-keep public class com.webank.mbank.wehttp.WeLog$Level{
*;
}
-keep class com.webank.mbank.wejson.WeJson{
public <methods>;
}


##########################webank normal混淆规则-END###############################
```

您可以将如上代码拷贝到您的混淆文件中，也可以将 SDK 中的 `webank-cloud-normal-proguard-rules.pro` 拷贝到主工程根目录下，然后通过 `-include webank-cloud-normal-rules.pro` 加入到您的混淆文件中。

云 OCR 依赖的第三方库的混淆规则

云 OCR 依赖的第三方库的混淆规则全部内容变更为：

```
#######################云OCR依赖的第三方库 混淆规则-BEGIN#############################
#

## support:appcompat-v7
```

```
-keep public class android.support.v7.widget.** { *; }
-keep public class android.support.v7.internal.widget.** { *; }
-keep public class android.support.v7.internal.view.menu.** { *; }

-keep public class * extends android.support.v4.view.ActionProvider {
public <init>(android.content.Context);
}
############################云OCR依赖的第三方库 混淆规则-END#########################
#######
```

您可以根据您现有的混淆规则，将缺少的第三库混淆规则拷贝到您的混淆文件中。

上一步：开发准备

下一步：接口调用

接口调用

最近更新时间：2019-01-09 17:49:52

SDK 接口调用方法

SDK 代码调用的入口为 com.webank.mbank.ocr.WbCloudOcrSDK 这个类。

```
public class WbCloudOcrSDK{

/**
* 该类为一个单例，需要先获得单例对象再进行后续操作
*/
public static synchronized WbCloudOcrSDK getInstance() {
// ...
}
/**
* 在使用SDK前先初始化，传入需要的数据data
*由 OcrLoginListener返回是否登录SDK成功
* 关于传入数据data见后面的说明
*/
public void init(Context context,Bundle data,OcrLoginListener loginListerner){
// ...
}
/**
* 登录成功后，调用此函数拉起sdk页面
* @param context 拉起SDK的上下文
* @param idCardScanResultListener 返回到第三方的接口
* @param type 进入SDK的模式，参数是枚举类型
*/
Public void startActivityForOcr(Context context,IDCardScanResultListener,WBOCRTYPEMODE type){
// ...
}
/**
* 登录回调接口
*/
public interface OcrLoginListener {
void onLoginSuccess();
void onLoginFailed(String errorCode, String errorMsg);
}

/**
* 退出SDK,返回第三方的回调,同时返回ocr识别结果
```

```
*/
public interface IDCardScanResultListener{
/**
* @RARAM exidCardResult SDK返回的识别结果的错误码
* @RARAM exidCardResult SDK返回的识别结果的错误信息
*/
void onFinish(String errorCode, String errorMsg);
}
```

WbCloudOcrSdk.init() 的第二个参数用来传递数据，可以将参数打包到 data(Bundle) 中，必须传递的参数包括：

```
//这些都是WbCloudOcrSdk.InputData对象里的字段，是需要传入的数据信息
String orderNo; //订单号
String clientIp; //用户 ip 信息,格式为”ip=xxx.xxx.xxx.xxx;”
// 示例：”ip=58.60.124.0”
String gps; //用户 gps 信息, 格式为”lgt=xxx;lat=xxx;”
//示例：“lgt=22.5044;lat=113.9537“
String openApiAppId; //APP_ID
String openApiAppVersion; //openapi Version
String openApiNonce; //32位随机字符串
String openApiUserId; //user id
String openApiSign; //签名信息
```

⚠ 注意：

以上参数被封装在 WbCloudOcrSdk.InputData 对象中（它是一个 Serializable 对象）。

WbCloudOcrSdk.startActivityForOcr() 的第三个参数 type 是个枚举类 WBOCRTYPEMODE ，决定第三方登录成功后以哪种模式进行身份证识别。

```
/**
* 调OCR 的模式
*/
public enum WBOCRTYPEMODE {
WBOCRSDKTypeNormal,//标准模式，先进入准备界面再进入扫描身份证界面
WBOCRSDKTypeFrontSide,//人像面模式，直接进入身份证人像面识别
WBOCRSDKTypeBackSide,//国徽面模式，直接进入身份证国徽面识别
}
```

登录接口

```
/**
* 登录回调接口
*/
public interface OcrLoginListener {
void onLoginSuccess();//登录成功
/**
* @PARAM errorCode 登录失败错误码
* @PARAM errorMsg 登录失败错误信息
*/
void onLoginFailed(String errorCode, String errorMsg);
}
```

返回第三方接口

```
/**
* 退出SDK,返回第三方的回调,同时返回ocr识别结果
*/
public interface IDCardScanResultListener{
/**
* 退出SDK,返回第三方的回调,同时返回ocr识别结果
* @param errorCode 返回错误码，识别成功返回 0
* @param errorMsg 返回错误信息，和错误码相关联 */
void onFinish(String errorCode, String errorMsg);
}
```

身份证识别结果类

身份证识别结果, 封装在 EXIDCardResult 类中，通过 WbCloudOcrSDK.getInstance().getResultReturn() 获得，该类属性如下所示：

```
public int type;//拉起SDK的模式所对应的int 值，也就是startActivityForOcr 方法中WBOCRTYPEMODE type的枚举值value
// 识别人像面返回的信息
public String cardNum; //身份证号码
public String name;//姓名
public String sex;//性别
public String address;//住址
public String nation;//民族
public String birth;//出生年月日
public String frontFullImageSrc;// 人像面图片存放路径
public String frontWarning;//人像面告警码
```

```
//识别国徽面返回的信息
public String office;//签发机关
public String validDate;//有效期限
public String backFullImageSrc;//国徽面图片存放路径
public String backWarning;//国徽面告警码

//每次网络请求都会返回的信息
public String multiWarning;//多重告警码
public String clarity;//清晰度得分
public String sign;//签名
public String orderNo; //订单号
public String ocrId;//识别的唯一标识
```

接口参数说明

InputData 是用来给 SDK 传递一些必须参数所需要使用的对象（ WbCloudOcrSdk.init() 的第二个参数），合作方需要往里塞入 SDK 需要的一些数据以便启动 OCR SDK。
其中 InputData 对象中的各个参数定义如下表，请合作方按下表标准传入对应的数据。

参数	说明	类型	长度	是否必填
orderNo	订单号，合作方订单的唯一标识	String	32	是
openApiAppId	腾讯服务分配的 AppID	String	腾讯服务分配	是
openApiAppVersion	接口版本号，默认填：1.0.0	String	20	是
openApiNonce	32位随机字符串，每次请求需要的一次性 nonce	String	32	是
openApiUserId	用户 ID，每个用户唯一的标识	String	30	是
openApiSign	合作方后台服务器通过 ticket 计算出来的签名信息	String	40	是
clientIp	用户 ip 信息 格式为 “ip=xxx.xxx.xxx.xxx;” 示例：“ip=58.60.124.0;”	String	30	是
gps	用户 GPS 信息 格式为 “lgt=xxx;lat=xxx;” 示例：“lgt=22.5044;lat=113.9537;”	String	30	是

个性化参数设置

WbCloudOcrSdk.init() 里的 Bundle data ，除了必须要传的 InputData 对象之外，还可以由合作方方传入一些个性化参数，量身打造更契合自己 App 的 SDK。如果合作方未设置这些参数，则以下所有参数按默认值设置。

设置 SDK 的界面标题栏背景色

合作方可以设置进入 SDK 的准备界面的标题栏背景色（仅对标准模式此设置才有效）。 SDK 默认显示准备界面的标题栏背景颜色是白色（ #ffffff ），但第三方可对其个性化设置。设置代码如下：

```
# 在 MainActivity 中单击某个按钮的代码逻辑：
//先将必填的 InputData 放入 Bundle 中
data.putSerializable(WbCloudFaceVerifySdk.INPUT_DATA, inputData);
//设置标题栏背景色，如果不设置则默认展示；设置了则以设置为准
//此处设置进入 SDK 的第一个界面的标题栏背景色为蓝色(#409eff)
data.putString(WbCloudOcrSDK.TITLE_BAR_COLOR, "#409eff");
```

设置 SDK 的界面标题栏内容

合作方可以设置进入 SDK 的准备界面的标题栏文字内容（仅对标准模式此设置才有效）。SDK 默认显示第一个界面的标题栏文字内容是身份证识别，但第三方可对其个性化设置。设置代码如下：

```
# 在 MainActivity 中单击某个按钮的代码逻辑：
//先将必填的 InputData放入 Bundle 中
data.putSerializable(WbCloudFaceVerifySdk.INPUT_DATA, inputData);
//设置标题栏文字内容，如果不设置则默认展示；设置了则以设置为准
//此处设置进入 SDK 的第一个界面的标题栏文字内容为居民身份证识别
data.putString(WbCloudOcrSDK.TITLE_BAR_CONTENT, "居民身份证识别");
```

设置 SDK 的水印文字内容

合作方可以设置进入 SDK 的第一个界面上的水印文字内容。 SDK 默认显示第一个界面的水印文字内容是仅供内部业务使用，但第三方可对其个性化设置。设置时需要注意：水印文字长度不超过8，且只支持汉字，若长度超过8，SDK 会自动截取前8个汉字。设置代码如下：

```
# 在MainActivity 中单击某个按钮的代码逻辑：
//先将必填的 InputData 放 入Bundle 中
data.putSerializable(WbCloudFaceVerifySdk.INPUT_DATA, inputData);
//设置水印文字内容，如果不设置则默认展示；设置了则以设置为准
//此处设置进入 SDK 的第一个界面的水印文字内容为仅供本次业务使用 data.putString(WbCloudOcrSDK.WATER_MASK_TEXT, "仅供本次业务使用");
```

设置 SDK 的扫描识别的时间上限

合作方可以设置 SDK 的扫描识别时间的上限。 SDK 打开照相机进行扫描识别的时间上限默认是20秒，20秒内若扫描识别成功则返回到 SDK 的第一个界面，否则直到20秒直接退出扫描界面。第三方可对其个性化设置，设置的时间上限不能超过60秒，建议第三方采用默认值，不要修改这个参数。设置代码如下：

```
# 在 MainActivity 中单击某个按钮的代码逻辑：
//先将必填的 InputData 放入 Bundle 中
data.putSerializable(WbCloudFaceVerifySdk.INPUT_DATA, inputData);
//设置 SDK 扫描识别身份证的时间上限，如果不设置则默认 20 秒；设置了则以设置为准
//此处设置 SDK 的扫描识别时间的上限为 20 秒
data.putLong(WbCloudOcrSDK.SCAN_TIME, 20000);
```

设置 SDK 识别是否校验身份证正反面

合作方可以设置 SDK 是否校验正反面都识别成功，此设置只适用于标准模式。在标准模式下传“2”则会对身份证正反面识别进行强校验，即正反面都识别成功了才能单击【完成】。不传或传其他则默认不会对身份证正反面识别进行强校验。设置代码如下：

```
# 在 MainActivity 中单击某个按钮的代码逻辑：
//先将必填的 InputData 放入 Bundle 中
data.putSerializable(WbCloudFaceVerifySdk.INPUT_DATA, inputData);
//设置 SDK 是否校验身份证正反面都识别成功，如果不设置则默认不校验；设置了则以设置为准
//此处设置 SDK在标准模式下对身份证人像面、国徽面识别进行强校验
OCR_FLAG参数值为 “1” 、null 时，人像面必须识别，国徽面可选识别
OCR_FLAG参数值为 “2” 时，人像面、国徽面都必须识别）
data.putString(WbCloudOcrSDK.OCR_FLAG, "2");
```

个性化设置接入示例

```
# 在 MainActivity 中单击某个按钮的代码逻辑：
//先将必填的 InputData 放入 Bundle 中
data.putSerializable(WbCloudOcrSDK.INPUT_DATA, inputData);
//个性化参数设置,此处均设置为与默认不同
//设置 SDK 标题栏背景颜色，默认白色，此处设置为蓝色（仅对标准模式有效） data.putString(WbCloudO
crSDK.TITLE_BAR_COLOR, "#409eff");
//设置 SDK 标题栏文字内容，默认展示身份证识别,此处设置为居民身份证识别（仅对标准模式有效）
data.putString(WbCloudOcrSDK.TITLE_BAR_CONTENT, "居民身份证识别");
//设置 SDK 水印文字内容，默认仅供内部业务使用，此处设置为仅供本次业务使用（仅对标准模式有效）
data.putString(WbCloudOcrSDK.WATER_MASK_TEXT, "仅供本次业务使用");
//设置扫描识别的时间上限,默认 20 秒，此处设置为 20 秒
data.putLong(WbCloudOcrSDK.SCAN_TIME, 20000);
//此处设置 SDK在标准模式下对身份证人像面、国徽面识别进行强校验
```

```
OCR_FLAG参数值为 “1” 、null 时，人像面必须识别，国徽面可选识别
OCR_FLAG参数值为 “2” 时，人像面、国徽面都必须识别）
data.putString(WbCloudOcrSDK.OCR_FLAG, "2");
```

错误码描述

最近更新时间：2018-01-24 15:27:59

终端返回错误码

错误码	错误描述
IDOCR_LOGIN_PARAMETER_ERROR = "-20000";	传入参数有误
IDOCR_USER_CANCEL="200101";	用户取消操作
IDOCR__ERROR_USER_NO_NET="100101";	无网络
IDOCR_USER_2G="100102";	不支持 2G 网络
IDOCR_ERROR_PERMISSION_CAMERA="100103";	无相机权限
IDOCR_ERROR_PERMISSION_READ_PHONE="100103";	READ PHONE 未权限
IDOCR_ERROR_PERMISSION="100103";	权限异常
IDOCR_LOGIN__ERROR="-10000";	登录错误
SERVER_FAIL="-30000";	内部服务错误

后台返回错误码

错误码	错误描述
INTERNAL_SERVER_ERROR="999999"	网络不给力，请稍后再试
FRONT_INTERNAL_SERVER_ERROR="999998"	网络不给力，请稍后再试
SERVICE_TIME_OUT="999997"	网络不给力，请稍后再试
OAUTH_INVALID_REQUEST="400101"	不合法请求
OAUTH_INVALID_LOGIN_STATUS="400102"	不合法请求
OAUTH_ACCESS_DENIED="400103"	服务器拒绝访问此接口
OAUTH_INVALID_PRIVILEGE="400104"	无权限访问此请求
OAUTH_REQUEST_VALIDATE_ERROR="400105"	身份验证不通过
OAUTH_TPS_EXCEED_LIMIT="400501"	请求超过最大限制

错误码	错误描述
OAUTH_INVALID_VERSION="400502"	请求上送版本参数错误
OAUTH_INVALID_FILE_HASH="400503"	文件校验值错误
OAUTH_REQUEST_RATE_LIMIT="400504"	请求访问频率过高

接入示例

最近更新时间：2019-01-08 17:26:56

```
# 在 MainActivity 中单击某个按钮的代码逻辑：
//先填好数据
Bundle data = new Bundle();
WbCloudOcrSDK.InputData inputData = new WbCloudOcrSDK.InputData(
orderNo,
appId,
openApiAppVersion,
nonce,
userId,
sign);
data.putSerializable(WbCloudOcrSDK.INPUT_DATA, inputData);
//个性化参数设置,可以不设置，不设置则为默认选项。
//此处均设置为和默认设置不同
data.putString(WbCloudOcrSDK.TITLE_BAR_COLOR, "#409eff");
//设置 SDK 标题栏文字内容，默认展示身份证识别,此处设置为居民身份证识别data.putString(WbCloudOcrS
DK.TITLE_BAR_CONTENT, "居民身份证识别");
//设置 SDK 水印文字内容，默认仅供内部业务使用，此处设置为仅供本次业务使用
data.putString(WbCloudOcrSDK.WATER_MASK_TEXT, "仅供本次业务使用");
//设置扫描识别的时间上限,默认 20 秒，建议默认
data.putLong(WbCloudOcrSDK.SCAN_TIME, 20000);
data.putString(WbCloudOcrSDK.OCR_FLAG, "2");
//初始化 SDK，得到是否登录 SDK 成功的结果
WbCloudOcrSDK.getInstance().init(MainActivity.this, data, new WbCloudOcrSDK.OcrLoginListener() {
@Override
public void onLoginSuccess() { //登录成功，拉起 SDL 页面
WbCloudOcrSDK.getInstance().startActivityForOcr(MainActivity.this,
new WbCloudOcrSDK.IDCardScanResultLisrener(){ //返推出 SDK 回调接口
@Override
public void onFinish(String resultCode, String resultMsg) {
// resultCode为0，则识别成功；否则识别失败
if ("0".equals(resultCode)) {
// TODO:2017/10/30
WLogger.d(TAG, "识别成功,识别身份证的结果是:"+WbCloudOcrSDK.getInstance().getResultReturn().toSt
ring());
}else{ // TODO:2017/10/30
WLogger.d(TAG, "识别失败:"+resultCode+”  --”  +resultMsg);
}
}
},WbCloudOcrSDK.WBOCRTYPEMODE.WBOCRSDKTypeNormal);
}
```

```
@Override
public void onLoginFailed(String errorCode, String errorMsg) {
if(errorCode.equals(ErrorCode.IDOCR_LOGIN_PARAMETER_ERROR)) {
Toast.makeText(MainActivity.this, "传入参数有误！" + errorMsg, Toast.LENGTH_SHORT).show();
} else {
Toast.makeText(MainActivity.this, "登录 OCR SDK 失败！" + "errorCode= " + errorCode + " ;errorMsg
=" + errorMsg, Toast.LENGTH_SHORT).show();
}
}
})
```

iOS SDK 接入
配置流程

最近更新时间：2018-11-02 17:07:30

> **注意：**
> **接入之前，请详细阅读 SDK 中的 readme 和接入指引。**

以下为接入配置的步骤：

1. 将 SDK 文件导入到目标 project 中去,并确定 **“add to target”** 被勾选.

```
├── WBOCRService.bundle
|
├── WBOCRService.framework
|
├── include
| └── recdetect.h
├── librecdetect.a
└── opencv2.framework
```

2. 在 【build phases】 >【link with libraries】下加入如下依赖：

CoreTelephony.framework
AssetsLibrary.framework
CoreMedia.framework
AVFoundation.framework
WebKit.framework
SystemConfiguration. **framework**
libc++.tbd

3. 【Build Setting】>【Enable Bitcode】 设置为 NO。
4. 【Build Setting】>【 Linking】 > 【other linker flag】设置增加 `-ObjC` 和 `-lz linker flag`
5. SDK 中需要使用 camera，需要在 `Info.plist` 中添加 `NSCameraUsageDescription` 为 key 的键值对。

接口调用

最近更新时间：2019-01-15 19:06:59

调用详情参见 Demo 工程以及头文件 WBOCRService.h

在使用 OCR SDK 的类中引入 WBOCRService.h

```
#import <WBOCRService/WBOCRService.h>
@interface ViewController ()
// TODO:
@end
```

实例化 SDK

WBOCRService 是一个单例类 ，通过如下方法来实例化这个类：

```
#import <WBOCRService/WBOCRService.h>
```

启动 OCR SDK 服务

实例化 WBOCRService 之后，调用 start 方法启动 SDK，start 方法的头文件声明如下：

```
- (void)startOCRServiceWithConfig:(nullable WBOCRConfig *)config
version:(nonnull NSString *)version
appId:(nonnull NSString *)appId
nonce:(nonnull NSString *)nonce
userId:(nonnull NSString *)userId
sign:(nonnull NSString*)sign
orderNo:(nonnull NSString *)orderNo
tartSucceed:(nonnull WBOCRServiceStartSucceedBlock)startSucceed
recognizeSucceed:(nonnull WBOCRServiceRecognizeSuccessBlock)recognizeSucceed
failed:(nonnull WBOCRServiceFailedBlock)failed;
```

start 方法参数说明：

- config 是一个配置参数，nullable，用来对 SDK 做一些配置，后文的 config。
- version 表示 webank OpenAPI 的版本号，由腾讯服务分配，没有特别说明请输入：1.0.0。
- appId 腾讯服务分配的 app_id。
- nonce SDK 鉴权用参数，一次性有效。
- userId 表示每个用户唯一的标识，由腾讯服务分配。
- sign SDK 鉴权用参数，从接入方后台获取。

- orderNo 订单号，这个参数用来查询识别结果用，接入方必须保证这个参数的唯一性，该参数长度不得超过 32 位，订单号中尽量不要包含特殊字符。
- startSucceed SDK 启动成功回调，当用户鉴权通过的时候会走到这个回调。
- recognizeSucceed SDK 识别成功回调，可以通过这个回调来获取识别结果信息。
- failed SDK 发生异常，退出时候的回调，可以通过这个回调获取失败信息。

config 参数说明（“身份证 / 银行卡识别”的配置参数）
config 是个 WBOCRConfig 实例，通过 `WBOCRConfig *config = [WBOCRConfig sharedConfig]` 来实例化，这里面有几个参数需要着重讲一下：

- SDKType 这个参数决定本次使用的是哪种类型的 OCR 服务，SDK 目前提供身份证人像面识别、身份证国徽面识别、身份证两面识别和银行卡识别四种类型的服务，比如可以通过 `config.SDKType = WBOCRSDKTypeIDCardFrontSide;` 来选择使用身份证人像面识别服务。

```
// WBOCRSDKTypeNoraml : 标准模式，SDK调起成功后，先进入拍摄准备页面，待正反两面识别完成之后，将本次识别结果返回到第三方APP
// WBOCRSDKTypeFontSide: 人像面识别模式，SDK调起成功后，直接进入拍摄识别页面，识别身份证人像面，识别完成之后，将本次识别结果返回第三方APP
// WBOCRSDKTypeBackSide: 国徽面识别模式，SDK调起成功后，直接进入拍摄识别页面，识别身份证国徽面，识别完成之后，将本次识别结果返回第三方APP
// WBOCRSDKTypeBankCard:银行卡识别模型
```

- needBothSidesRecognized，只有当上面的 SDKType 参数选择的是 WBOCRSDKTypeIDCardNormal 的时候， needBothSidesRecognized 参数才起作用，它的功能是控制 SDK 识别完成的逻辑。
  - 当 needBothSidesRecognized 参数为 YES 的时候，用户需要全部完成人像面和国徽面识别后，才能单击 “成功退出 SDK ”。
  - 当 needBothSidesRecognized 参数为 NO 的时候，用户仅需要完成人像面识别，就可以 “成功退出 SDK”。这里 “成功退出 SDK” 的含义，是指 SDK 通过 recognizeSucceed 回调退出。
- config 里面还有很多 readonly 参数，这些参数和本次识别相关，开发者可以按需在这些接口获取。

处理 SDK 的回调信息

开发者通过 start 方法的 startSucceed、recognizeSucceed 和 failed 三个 block 来接收 SDK 回调信息。
startSucceed（成功进入 SDK 回调）
进入这个回调，说明当前用户已经通过SDK鉴权，应用成功进入SDK界面了。

recognizeSucceed（识别成功，即将退出SDK回调）
进入这个回调，说明 SDK 已经识别成功，即将退出，回到 App 中的界面，这里面有两个参数 resultModel 和 extension。

- resultModel 是对识别结果的封装，如果当前识别的是身份证，就会返回一个 WBIDCardInfoModel 类型的实例；如果当前识别的是银行卡，返回的是一个 WBBankCardInfoModel 类型的实例。关于每个字段的详细含

义，请参考 WBOCRService.h 头文件。

- extension 是一个扩展字段，备用，目前版本为空，不需要处理。

failed（SDK 异常，即将退出 SDK 回调）

进入这个回调，说明 SDK 发生异常，SDK 即将退出，可以通过这个回调获取失败信息，这里面有两个参数 error 和 extension。

- error 是一个 NSError 类型的实例，里面会封装错误码和错误描述，下面代码展示了一条错误码为 200101 的 error 信息，详情请参见 错误码描述。

```
NSError *error = [NSError errorWithDomain:@"com.webank.ocr.error" code:200101 userInfo:@{NSLocalizedDescriptionKey:@"用户取消操作"}];
```

- extension 是一个扩展字段，备用，目前版本为空，不需要处理。

错误码与多重告警码

最近更新时间：2019-01-08 16:45:49

错误码

返回码	返回信息	处理措施
100100	传入 SDK 参数不合法	检查传入参数是否合法
100101	无网络，请确认	确认网络正常
100102	不支持 2G 网络	更换网络环境
100103	无相机权限	-
200101	用户取消操作	用户主动退出操作
200102	识别超时	用户在身份证正反面识别过程中超过设定的阈值（20s）无法识别，提示超时

多重告警码

在 OCR SDK 2.2.0 版本，引入了多重告警码：

- 银行卡识别时，在 WBBankCardInfoModel 类的头文件中，增加了三个字段：
  i. 清晰度 clarity 字段。
  ii. 多重警告码 multiWarningCode 字段。
  iii. 多重警告码描述 multiWarningMsg 字段。
- 身份证人相面识别时，在 WBIDCardInfoModel 类的头文件中，增加了两个字段：
  i. 清晰度 frontClarity 字段。
  ii. 多重警告码 frontMultiWarning 字段。
- 身份证国国徽面识别时时，在 WBIDCardInfoModel 类的头文件中，增加了两个字段：
  i. 清晰度 backClarity 字段。
  ii. 多重警告码 backMultiWarning 字段。

识别成功时，在 recognizeSucceed 的回调中，通过获取 resultModel 中的相应字段来获取多重告警码。

公众号接入
生成签名

最近更新时间：2018-07-27 17:23:02

准备步骤

- **前置条件：请合作方确保 NONCE ticket 已经正常获取，获取方式见 NONCE ticket 获取 。**
- 合作方根据本次OCR识别的如下参数生成签名,需要签名的参数信息如下：
- 参与签名的数据需要和使用该签名的接口中的请求参数保持一致。

参数	说明	来源
appId	腾讯云线下对接分配的 App ID	腾讯云线下对接分配
orderNo	订单号，本次身份证验证合作伙伴上送的订单号，唯一标识	合作方自行分配
userId	用户 ID ，用户的唯一标识（不要带有特殊字符）	合作方自行分配（与接口中定义的 userId 保持一致）
version	参数值为：1.0.0	-
api ticket	合作伙伴服务端缓存的 tikcet，注意是 NONCE 类型	获取方式见 NONCE ticket 获取（所用的 userId 参数值需要和接口里面的 userId 值保持一致）
nonce	随机数：32 位随机串（字母 + 数字组成的随机数）	合作方自行生成（与接口里面定义的随机数保持一致）

基本步骤

1. 生成一个 32 位的随机字符串 nonce（其为字母和数字，登录时也要用到）。
2. 将 webankAppId、userId、orderNo、version、连同 ticket、nonce 共 6 个参数的值进行字典序排序。
3. 将排序后的所有参数字符串拼接成一个字符串。
4. 将排序后的字符串进行 SHA1 编码，编码后的 40 位字符串作为签名（sign）。

注意：

签名算法可参考 签名算法说明 。

参考示例

请求参数：

参数名	参数值
webankAppId	appId001
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
orderNo	aabc1457895464
ticket	zxc9Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

字典排序后的参数为：

[1.0.0, aabc1457895464, appId001, kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T , userID19959248596551, zxc9Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS]

拼接后的字符串为：

1.0.0aabc1457895464appId001kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T **userID19959248596551zxc9Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS**

计算 SHA1 得到签名：

5E034EF71E90E5F5FB072CDBB259FFF25A938B03

该字串就是最终生成的签名（40位），不区分大小写。

下一步：公众号启动 H5 身份证识别

公众号启动 H5 身份识别

最近更新时间：2018-01-24 15:28:27

合作方公众号上送 sign，后台校验 sign 通过之后重定向到身份证识别 H5。

请求 URL：

```
https://ida.webank.com/api/h5/ocrlogin
```

请求方法：GET

请求参数：

参数	说明	类型	长度	是否必填
webankAppId	腾讯云线下对接分配的 App ID	String	腾讯云线下对接决定	是
version	接口版本号，默认参数值：1.0.0	String	20	是
nonce	随机数 32 位随机串（字母 + 数字组成的随机数）	String	32	是
orderNo	订单号，由合作方上送，每次唯一，此信息为本次人脸验证上送的信息。	字符串	32	是
url	OCR 识别完成后回调第三方 URL，**需要第三方提供完整 URL 且做 URL Encode。** **完整 URL Encode 示例:** **原 URL（https://idaop.webank.com）** **Encode 后：** **(http://idaop.webank.com)**	string		是
userId	用户 ID ，用户的唯一标识（不要带有特殊字符）	string		是
sign	签名：使用上面生成的签名	string	40	是
ocrFlag	人像面、国徽面识别配置 参数值为 “1”、null 时，人像面必须识别，国徽面可选识别 参数值为 “2” 时，人像面、国徽面都必须识别）	string		否

上一步：合作方生成签名

下一步：H5 身份证识别结果跳转

H5 身份证识别结果跳转

最近更新时间：2018-01-24 15:28:30

用于 H5 身份证 OCR 识别结果返回跳转第三方 URL，其带参数：返回码，订单号和签名。合作伙伴接根据返回码判断识别是否成功完成，同时需要根据订单号等信息获取身份证识别结果（识别结果包含姓名、性别、民族、出生日期、身份证号、住址、签发机关、证件有效期）请通过查询接口获取。（见 服务端查询结果）

请求

请求URL：

```
https://xxx.com/xxx?code=xxxx&orderNo=xxxx&signature=xxxx
```

注意：

1. xxxx.com 为合作方上送的 URL。
2. 合作方根据 前端获取结果验证签名 说明进行签名校验，确保返回结果的安全性。

请求方法：GET

响应

响应参数：

参数	说明	类型	长度
code	身份证识别结果的返回码，0 表示识别成功，其他错误码标识失败	字符串	
orderNo	订单号，由合作方上送，每次唯一，此信息为本次身份证识别上送的信息	字符串	32
signature	对 URL 参数 App ID、oderNo 和 SIGN ticket 的签名。具体见的签名生成和校验规则	字符串	40

注意：

合作方通过查询身份证识别结果查询接口获取身份证识别信息（姓名、身份证等）及身份证照片，请参考身份证识别结果章节 方式一：前端获取结果验证签名 和 方式二：合作伙伴服务端查询结果。

上一步：公众号启动 H5 身份证识别

小程序接入
小程序接入身份证 OCR 识别

最近更新时间：2018-12-05 17:41:25

步骤一：关联人脸验证小程序

公众号关联小程序：登录公众号后台后，进入【小程序】>【小程序管理】>【添加】>【关联小程序】。
通过 appid：wxb9b6a64ddce80154 搜索 WeFaceX2 小程序并关联。

步骤二：授权小程序关联

联系腾讯工作人员授权小程序关联，并确认小程序关联成功。

启动小程序身份证识别

最近更新时间：2018-12-11 11:15:43

生成签名

准备步骤

- 前置条件：请合作方确保 NONCE ticket 已经正常获取，获取方式见 NONCE ticket 获取 。
- 合作方根据本次身份证 OCR 识别的如下参数生成签名，需要签名的参数信息如下：

参数	说明	来源
appid	腾讯云线下对接分配的 App ID	腾讯云线下对接分配
version	接口版本号1.0.0	-
nonce	随机数32位随机串（字母 + 数字组成的随机数）	合作方自行生成
orderNo	订单号，由合作方上送，每次唯一，此信息为本次人脸验证上送的信息。	合作方自行分配
userId	用户 ID ，用户的唯一标识（不要带有特殊字符）	合作方自行分配
api ticket	合作伙伴服务端缓存的 tikcet，注意是 NONCE 类型	获取方式见 NONCE ticket 获取

基本步骤

1. 生成一个32位的随机字符串（字母和数字） nonce（接口请求时也要用到）。
2. 将 webankAppId、userId、orderNo、version 连同 ticket、nonce 共6个参数的值进行字典序排序。
3. 将排序后的所有参数字符串拼接成一个字符串。
4. 将排序后的字符串进行 SHA1 编码，编码后的40位字符串作为签名（sign）。

> ⚠ **注意：**
>
> 签名算法详情请参见 签名算法说明 。

参考示例

请求参数：

参数名	参数值
webankAppId	appId001
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T （必须为32位）
version	1.0.0
orderNo	aabc1457895464
ticket	zxc9Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

字典排序后的参数为：

```
[1.0.0, aabc1457895464, appId001, kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551, zxc9Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS]
```

拼接后的字符串为：

```
1.0.0aabc1457895464appId001kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7TuserID19959248596551zxc9Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS
```

计算 SHA1 得到签名：

```
5E034EF71E90E5F5FB072CDBB259FFF25A938B03
```

该字串就是最终生成的签名（40位），不区分大小写。

启动小程序身份证 OCR 识别

合作方小程序上送 sign，调用微信打开小程序 API 身份证 OCR 小程序。
请求方法：使用 navigator 组件打开，navigatot 使用方法示例：

```
<navigator target="miniProgram" open-type="navigate" app-id="" path="" extra-data="" version="release">打开绑定的小程序</navigator>
```

此方法是微信小程序提供的 API，详细信息请参考 微信小程序官方文档 。

- navigator 组件无回调函数，无法得知跳转结果。且该组件只能触发跳转事件，在跳转前应先获取到 sign 等参数。

- 由于 sign 存在有效期，若跳转到小程序后用户长时间未进行操作，合作方小程序应再次获取 sign 后执行跳转（每次调起 OCR 小程序都应获取一次）。
- 由于 navigator 组件目前无回调函数，所以执行跳转后合作方小程序应清除当前缓存，保证异常退出后用户可以再次获取 sign 后跳转。

⚠ 注意：

- 由于微信修改了小程序调起第三方小程序方式，老的方式 wx.navigateToMiniProgram（OBJECT）将于7月5日废弃，届时在基础库2.0.7以上（对应微信6.6.6）的微信将无法使用此接口调起小程序。
- 由于 navigator 组件仅在基础库2.0.7（对应微信6.6.6）以上可用，合作方跳转的逻辑仍需兼容旧方式，即低版本的微信。可以使用 wx.getSystemInfo（OBJECT）获取到当前微信的基础库版本，判断其小于2.0.7版本使用旧的跳转方式。详细信息请参考 wx.getSystemInfo（OBJECT）文档。

请求参数：

参数	类型	默认值	是否必填	说明
target	String	-	是	在哪个目标上发生跳转，默认当前小程序
open-type	String	navigate	是	跳转方式
app-id	String	wxb9b6a64ddce80154	是	要打开的小程序 AppId
path	String	pages/index	是	打开小程序的页面路径，如果为空则打开首页
extra-data	Object	-	是	需要传递给目标小程序的数据，目标小程序可在 App.onLaunch()，App.onShow() 中获取到这份数据
version	String	release	是	要打开的小程序版本，有效值 develop（开发版），trial（体验版），release（正式版），仅在当前小程序为开发版或体验版时此参数有效；如果当前小程序是体验版或正式版，则打开的小程序必定是正式版。

extra-data 中需要传递的参数：

参数	说明	类型	长度	是否必填

参数	说明	类型	长度	是否必填
webankAppId	WebankAppId，由腾讯指定	String	由腾讯指定腾讯服务分配	是
version	接口版本号	String	20	必填，默认值：1.0.0
nonce	随机数 32位随机串（字母+数字组成的随机数）	String	32	是
orderNo	订单号，由合作方上送，每次唯一，此信息为本次识别上送的信息	String	32	是
userId	用户 ID ，用户的唯一标识（不要带有特殊字符）	String	-	是
sign	签名：使用上面生成的签名	String	40	是
ocrFlag	人像面、国徽面识别配置 参数值为 “1”、null 时，人像面必须识别，国徽面可选识别 参数值为 “2” 时，人像面、国徽面都必须识别）	String	-	否

小程序身份证 OCR 识别结果跳转

最近更新时间：2018-12-11 11:20:15

1. OCR 小程序完成识别流程后，会携带唯一标识、订单号、验证结果、签名跳转回第三方小程序。
2. 第三方小程序需在【小程序生命周期函数】>【监听小程序】显示 onShow 方法中监听结果返回，获得识别结果。

> ⚠ **注意：**
>
> - 第三方小程序在 onShow（options）中可以通过 options.referrerInfo.extraData 拿到返回的结果参数。此部分为微信小程序提供的 API，详情可以参考 微信小程序官方文档。
> - 合作方根据进行签名校验，确保返回结果的安全性。
> - 2.0.4以上的基础库跳转时，需要在全局配置中加声明合作方参照 微信文档加声明。

3. **返回结果参数：**

参数	说明	类型	长度（字节）
code	身份证识别结果的返回码，0表示识别成功，其他错误码表示失败	String	-
orderNo	订单号，由合作方上送，每次唯一，此信息为本次身份证识别上送的信息	String	32
Signature	对 URL 参数 App ID、oderNo 和 SIGN ticket 的签名。具体见 签名生成 和校验规则	String	40

验证结果

方式一：前端获取结果验证签名

最近更新时间：2018-05-10 16:11:18

此方式用于：

1. 合作伙伴 App 端或 H5 收到远程身份证识别 SDK 返回以及签名结果。
2. 合作伙伴 App 端或 H5 调用其服务端接口进行签名认证，接口认证成功后继续业务流程。

1. 合作方后台生成签名

准备步骤

- **前置条件：请合作方确保 SIGN ticket 已经正常获取，获取方式见 SIGN ticket 获取 。**
- 合作方为身份证 OCR 识别服务生成签名，需要具有以下参数：

参数	说明	来源
app_id	腾讯云线下对接分配的 App ID	腾讯云线下对接分配
order_no	订单号，本次人脸验证合作伙伴上送的订单号，唯一标识	合作方自行分配
api ticket	合作伙伴服务端缓存的 tikcet，注意是 SIGN 类型	获取方式见 SIGN ticket 获取

基本步骤

1. 将 appId, orderNo, ticket（SIGN 类型）共三个参数的值进行字典序排序。
2. 将排序后的所有参数字符串拼接成一个字符串。
3. 将排序后的字符串进行 SHA1 编码，编码后的 40 位字符串作为签名（sign）。

> **注意：**
>
> 签名算法可参考 签名算法说明 。

参考示例

请求参数：

参数名	参数值

参数名	参数值
app_id	appId001
order_no	test1480921551481
ticket	duSz9ptwyW1Xn7r6gYltxz3feMdJ8Na5x7JZuoxurE7RcI5TdwCE4KT2eEeNNDoe

字典排序后的参数为：

```
[appId001, duSz9ptwyW1Xn7r6gYltxz3feMdJ8Na5x7JZuoxurE7RcI5TdwCE4KT2eEeNNDoe, test1480921551481]
```

拼接后的字符串为：

```
appId001duSz9ptwyW1Xn7r6gYltxz3feMdJ8Na5x7JZuoxurE7RcI5TdwCE4KT2eEeNNDoetest1480921551481
```

计算 SHA1 得到签名：

```
B02CEBEB07F792B2F085E8CB1E7BA9EC19284F54
```

该字串就是最终生成的签名（40位），不区分大小写。

2. 比对签名

合作方服务端生成的签名与 SDK 返回的签名比对，如果相同即可信任 SDK 的身份证 OCR 识别结果。

> **注意：**
>
> 合作方必须定时刷新 ticket（SIGN）保证远程身份认证后台缓存有该合作方的 ticket（SIGN），否则远程身份认证后台无法生成签名值。

方式二：合作伙伴服务端查询结果

最近更新时间：2018-10-09 15:24:30

此方式用于：

合作伙伴服务端生成签名，并调用身份证识别服务端查询结果，鉴权完成后返回结果（服务端上送 order_no 和 app_id 查询）。

1. 合作方后台生成签名

准备步骤

- **前置条件：请合作方确保 SIGN ticket 已经正常获取，获取方式见 SIGN ticket 获取 。**
- 合作方为身份证 OCR 识别服务生成签名，需要具有以下参数：

参数	说明	来源
app_id	腾讯云线下对接分配的 App ID	腾讯云线下对接分配
order_no	订单号，本次人脸验证合作伙伴上送的订单号，唯一标识	合作方自行分配
version	默认值：1.0.0	
api ticket	合作伙伴服务端缓存的 tikcet，注意是 SIGN 类型	获取方式见 SIGN ticket 获取
nonceStr	32 位随机字符串，字母和数字	合作方自行生成

基本步骤

1. 生成一个 32 位的随机字符串 nonceStr（其为字母和数字，登录时也要用到）。
2. 将 app_id、order_no、version 连同 ticket、nonceStr 共五个参数的值进行字典序排序。
3. 将排序后的所有参数字符串拼接成一个字符串。
4. 将排序后的字符串进行 SHA1 编码，编码后的 40 位字符串作为签名（sign）。

> **注意：**
>
> 签名算法可参考 签名算法说明 。

2. 身份证 OCR 识别结果查询接口

请求

请求URL：

```
https://idasc.webank.com/api/server/getOcrResult
```

请求方法：GET

请求参数：

参数	说明	类型	长度（字节）	是否必填
app_id	腾讯服务分配的 App ID	字符串	腾讯服务分配	是
order_no	订单号，合作方订单的唯一标识	字符串	32	是
get_file	是否需要获取身份证 OCR 图片文件。 值为1则返回文件；其他则不返回	字符串	1	否，非必填
nonce	随机数	字符串	32	是
version	版本号，默认值：1.0.0	字符串	20	是
sign	签名值，使用本页第一步生成的签名	字符串	40	是

请求示例：

```
https://idasc.webank.com/api/server/getOcrResult?app_id=xxx&nonce=xxx&order_no=xxx&version=1.0.0&sign=xxx&get_file=xxxx
```

响应

响应参数：

参数	类型	说明
frontCode	String	“0” 说明人像面识别成功
backCode	String	“0” 说明国徽面识别成功
orderNo	String	订单编号
name	String	frontCode 为 0 返回：证件姓名

参数	类型	说明
sex	String	frontCode 为 0 返回：性别
nation	String	frontCode 为 0 返回：民族
birth	String	frontCode 为 0 返回：出生日期
address	String	frontCode 为 0 返回：地址
idcard	String	frontCode 为 0 返回：身份证号
validDate	String	backCode 为 0 返回：证件的有效期
authority	String	backCode 为 0 返回：发证机关
frontPhoto	Base 64 String	人像面照片，转换后为 JPG 格式
backPhoto	Base 64 String	国徽面照片，转换后为 JPG 格式
frontCrop	Base 64 String	人像面切边照片
backCrop	Base 64 String	国徽面切边照片
headPhoto	Base 64 String	身份证头像照片
frontWarnCode	String	人像面告警码，在身份证有遮挡、缺失、信息不全时会返回告警码；当 frontCode 为 0 时才会出现告警码，告警码的含义请参考【通用响应码列表 】>【身份证 OCR 识别响应码】
backWarnCode	String	国徽面告警码，在身份证有遮挡、缺失、信息不全时会返回告警码；当 backCode 为 0 时才会出现告警码，告警码的含义请参考【通用响应码列表 】>【身份证 OCR 识别响应码】
operateTime	String	做 OCR 的操作时间
frontMultiWarning	String	正面多重告警码，含义请参考【通用响应码列表】

参数	类型	说明
backMultiWarning	String	反面多重告警码，含义请参考【通用响应码列表】
frontClarity	String	正面图片清晰度
backClarity	String	反面图片清晰度

注意：

- 身份证照片信息作为存证，合作伙伴可以通过此接口拉取识别结果和文件，需要注意请求参数的 `get_file` 需要设置为 1；如果不上送参数或者参数为空，默认不返回照片信息。为确保用户操作整体流程顺利完成，部分情况下获取照片会有1秒左右的延迟。
- 照片均为 base64 位编码，其中照片解码后格式一般为 JPG。
- 对于身份证 OCR 识别有部分遮挡、缺失、信息不全的情况，请参考 frontWarnCode 和 backWarnCode 告警码。（参考【通用响应码列表 】>【身份证识别响应码】）