人脸核身

含 UI 集成方案

腾讯云

文档目录

- 含 UI 集成方案
  - Plus 版人脸核身
    - 开始集成
    - App SDK
      - 合作方后台上送身份信息
      - 生成 SDK 接口调用步骤使用签名
      - Android人脸核身
        - 开发准备
        - 配置流程
        - 接口调用
        - 核身结果返回
        - Android 错误码
        - 接入示例
      - iOS 人脸核身
        - 配置流程
        - 接口调用
        - 接入流程与说明
        - 核身结果返回
        - iOS 错误码
      - Harmony NEXT人脸核身
        - 开发准备
        - 配置流程
        - 接口调用
        - 核身接口返回
        - 错误码描述
        - 接入示例
      - 人脸查询核身结果
        - 查询核身结果
        - 核身结果-多张照片返回
      - 登录鉴权
        - 获取 Access Token
        - 获取 SIGN ticket
        - 获取 NONCE ticket
        - 签名算法说明
    - 微信小程序
      - 微信小程序接入指引
      - 微信小程序 uni-app 接入流程
      - 微信小程序资质文件列表
      - 人脸核身对账指引
    - H5（微信公众号）
      - 微信原生 H5（微信公众号）
      - 微信原生 H5（微信公众号）资质
      - 人脸核身对账指引
    - H5（微信浏览器）
      - 微信浮层 H5（微信浏览器）
      - 人脸核身对账指引
    - H5 (移动端浏览器）
      - H5 人脸核身介绍
      - H5 人脸核身接入步骤
      - 人脸核身 App 调用 H5 兼容性配置指引
      - 合作方后台上传身份信息
      - 启动 H5 人脸核身
      - H5 人脸核身结果跳转

人脸核身结果查询
查询核身结果
人脸认证多张照片查询接口
增强版人脸核身
开始集成
App SDK
合作方后台上传身份信息
生成 SDK 接口调用步骤使用签名
Android 人脸核身
开发准备
配置流程
接口调用
核身结果返回
Android 错误码
接入示例
iOS 人脸核身
配置流程
接口调用
接入流程与说明
核身结果返回
iOS 错误码
Harmony NEXT 人脸核身
开发准备
配置流程
接口调用
核身接口返回
错误码描述
接入示例
人脸核身结果查询
查询核身结果
核身结果-多张照片返回
登录鉴权
获取 Access Token
获取 SIGN ticket
获取 NONCE ticket
签名算法说明
微信小程序
微信小程序接入指引
微信小程序 uni-app 接入流程
微信小程序资质文件列表
人脸核身对账指引
H5（微信公众号）
微信原生 H5（微信公众号）
微信原生 H5（微信公众号）资质文件列表
人脸核身对账指引
H5（微信浏览器）
微信浮层 H5（微信浏览器）
人脸核身对账指引
H5（移动端浏览器）
H5 人脸核身介绍
H5 人脸核身接入步骤
人脸核身 App 调用 H5 兼容性配置指引
合作方后台上传身份信息
启动 H5 人脸核身
H5 人脸核身结果跳转

人脸核身结果查询
查询核身结果
人脸认证多张照片查询接口
基础版人脸核身
App SDK
合作方后台上传身份信息
生成 SDK 接口调用步骤使用签名
Android 人脸核身
开发准备
配置流程
接口调用
核身结果返回
人脸 Android 错误码
接入示例
iOS 人脸核身
配置流程
接口调用
接入流程与说明
核身结果返回
人脸 iOS 错误码
Harmony NEXT人脸核身
开发准备
配置流程
接口调用
核身结果返回
错误码描述
接入示例
人脸核身结果查询
查询核身结果
核身结果-多照片查询接口
OCR SDK 接入
身份证 OCR SDK 接入
OCR 生成签名
OCR Android SDK 接入
OCR iOS SDK 接入
OCR Harmony SDK 接入
OCR 验证结果
身份证 OCR 错误码
银行卡 OCR SDK 接入
OCR 生成签名
OCR Android SDK 接入
OCR iOS SDK 接入
OCR Harmony SDK 接入
OCR 验证结果
银行卡 OCR 错误码
登录鉴权
获取 Access Token
获取 SIGN ticket
获取 NONCE ticket
获取 WBappid
签名算法说明
H5（微信浏览器）
微信普通 H5 配置流程
H5（PC浏览器）
合作方后台上传身份信息

启用 H5 人脸认证
PC 端 H5 人脸核身结果返回及跳转
人脸核身结果查询
查询核身结果
核身结果-多张照片返回
E证通
开始集成
微信小程序
接入流程说明
服务端集成
小程序集成
错误码列表
uni-app（小程序）
H5（移动端浏览器）
意愿核身
微信小程序
接入准备
意愿核身小程序接入流程
意愿核身 uni-app 小程序接入流程
E证通
意愿核身 E证通接入流程
App SDK
合作方初始化上传信息
生成 SDK 接口调用步骤使用签名
Android SDK 接入
iOS SDK 接入
Harmony NEXT 意愿核身接入
意愿核身查询结果
H5（微信浏览器）
微信H5（微信浏览器）
H5（移动端浏览器）
意愿核身移动 H5 接入步骤
合作方初始化上传信息
启动意愿核身移动 H5
H5 结果返回及跳转
意愿核身查询结果
App 调用 H5 兼容性配置
实证 NFC
App SDK
合作方初始化上传信息
生成 SDK 接口调用步骤使用签名
Android SDK 接入
iOS SDK 接入
实证 NFC 查询结果
计费错误码
Plus 版人脸核身服务错误码
增强版人脸核身服务错误码
基础版人脸核身服务错误码
E 证通服务错误码
意愿核身服务错误码
实证 NFC 服务错误码
核身及要素 API 服务错误码

含 UI 集成方案
Plus 版人脸核身
开始集成

最近更新时间：2025-06-12 14:35:51

> **说明：**
> 如果您在接入过程中遇到问题，欢迎您 联系我们 进行询问。

Plus 版人脸核身支持通过 微信小程序、微信原生H5（微信公众号）、微信浮层H5（微信浏览器）、Plus 版 App SDK、移动浮层H5（移动端浏览器）渠道接入，接入流程如下：

接入方式	接入流程	特殊说明
H5（微信公众号）	1. 申请人脸核身服务 2. 确认公司主体资质 3. 申请业务流程 RuleId ，提交行业对应资质 4. 管理员微信号，扫描 授权二维码 5. 实名核身鉴权 6. 获取实名核身结果信息	有行业类目限制，审核时间3 – 5天 详细可查阅 微信原生H5（微信公众号）资质
H5（微信浏览器）	1. 申请人脸核身服务 2. 申请业务流程 RuleId 3. 实名核身鉴权 4. 获取实名核身结果信息	微信公众号、企业微信
微信小程序	1. 申请人脸核身服务 2. 确认公司主体资质 3. 申请业务流程 RuleId 下载微信小程序 SDK 4. 管理员微信号，扫描 授权二维码 5. 微信小程序接入 6. 实名核身鉴权 7. 获取实名核身结果信息	有行业类目限制，审核时间3 – 5天 详细可查阅 微信小程序资质文件列表
App SDK（Plus版）	1. 人脸核身 SDK 2. 合作方后台上传身份信息 3. Android SDK 4. iOS SDK 5. 人脸验证结果	–
H5（移动端浏览器）	1. App 调用 H5 兼容性配置（如 App 调用） 2. 合作方后台上传身份信息 3. 启动 H5 人脸核身 4. 人脸核身结果返回及跳转 5. 人脸核身结果查询	支持多渠道接入，包括 App 调用 H5.浏览器等

接入方式和活体检测模式

Plus 版人脸核身不同的接入方式对应的活体检测模式不同，详细如下表所示：

接入方式	活体检测模式（任选其一）	备注
微信小程序	一闪活体	有行业类目限制，审核时间3 – 5天
H5（微信公众号）	一闪活体	有行业类目限制，审核时间3 – 5天
H5（微信浏览器）	一闪活体、远近活体	–

App SDK（增强版）	一闪活体	-
H5（移动端浏览器）	一闪活体、远近活体	-

App SDK
合作方后台上送身份信息

最近更新时间：2025-01-07 10:21:42

生成签名

准备步骤

- 下载 SDK，请到控制台自助接入列表页获取密码。
- 前置条件：请合作方确保 SIGN ticket 已经正常获取，获取方式见 获取 SIGN ticket 。
- SIGN 类型的 ticket，其有效期为60分钟，且可多次使用。
- 合作方为人脸核身服务生成签名，需要具有以下参数：

参数	说明	来源
appId	业务流程唯一标识，即 WBappid	参考 获取 WBappid 指引在人脸核身控制台内申请
userId	用户唯一标识，同一个用户的 userId 请保持一致，不同用户请不要使用同一个 userId，我们会根据 userId 来做防重复点击优化以及登录态校验	合作方自行分配，需跟用户绑定，32位以内，不包含特殊字符
version	参数值为：1.0.0	-
ticket	合作伙伴服务端获取的 ticket，注意是 SIGN 类型	获取方式请参见 获取 SIGN ticket
nonce	必须是32位随机数	合作方自行生成

> **注意：**
> 参与签名的数据需要与使用该签名的 SDK 中的请求参数保持一致。

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nonce、userId、appId连同ticket、version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	参数值
appId	IDAXXXXX
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
ticket	XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

- 字典排序后的参数为：

```
[1.0.0, IDAXXXXX, XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS,
kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551]
```

- 拼接后的字符串为：

```
1.0.0IDAXXXXXXO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7
TuserID19959248596551
```

- 计算 SHA1 得到签名：

该字符串就是最终生成的签名（40位），不区分大小写。

```
D7606F1741DDCF90757DA924EDCF152A200AC7F0
```

合作方后台上传身份信息

请求

- **请求 URL：**https://kyc1.qcloud.com/api/server/getPlusFaceId?orderNo=xxx

> ⚠ **注意：**
> 为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。

- **请求方法：**POST
- **报文格式：**Content-Type: application/json
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
orderNo	订单号，字母/数字组成的字符串，由合作方上传，每次唯一，不能超过32位	String	不能超过 32 位	是
name	姓名	String	-	• 使用权威源比对时：姓名+证件号必须输入 • 使用自带源比对时：姓名+证件号 可不输入
idNo	证件号码	String	-	• 使用权威源比对时：姓名+证件号必须输入 • 使用自带源比对时：姓名+证件号 可不输入
userId	用户 ID，用户的唯一标识（不能带有特殊字符），需要跟生成签名的 userId 保持一致。同一个用户的 userId 请保持一致，不同用户请不要使用同一个 userId，我们会根据 userId 来做防重复点击优化以及登录态校验	String	不能超过 32 位	是
sourcePhotoStr	比对源照片，注意：原始图片不能超过500k，且必须为 JPG 或 PNG、BMP 格式。参数有值：使用合作伙伴提供的比对源照片进行比对，必须注意是正脸可信照片，照片质量由合作方保证参数为空：根据身份证号 + 姓名使用权威数据源比对	BASE64 String	1048576	否，非必填请使用标准的图片转base64方法，base64编码不可包含换行符，不需要加前缀
sourcePhotoType	比对源照片类型参数值为1 时是：水纹正脸照参数值为 2 时是：高清正脸照重要提示：照片上无水波纹的为高清照，请勿传错，否则影响比对准确率。如有疑问，请联系腾讯云技术支持线下确认	String	1	否，提供比对源照片需要传
version	默认参数值为：1.0.0	String	20	是
sign	签名：使用上面生成的签名	String	40	是
nonce	随机数	String	32	是

水纹照示例

响应

响应参数:

参数	类型	说明
code	String	0：成功 非0：失败 详情请参见 Plus 版人脸核身服务错误码
msg	String	请求结果描述
bizSeqNo	String	请求业务流水号
orderNo	String	订单编号
faceId	String	此次刷脸用户标识，调 SDK 时传入

响应示例:

```
{
    "code": 0,
    "msg": "成功",
    "result": {
        "bizSeqNo":"业务流水号",
        "orderNo":"合作方订单号",
        "faceId":"cc1184c3995c71a731357f9812aab988"
     }
}
```

说明:
success：false 无意义，合作伙伴可忽略，无需处理。
faceId 有效期为5分钟，每次进行人脸核身都需要重新获取。

生成 SDK 接口调用步骤使用签名

最近更新时间：2024-12-20 15:23:02

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 。
- NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。
- 合作方为人脸核身服务生成签名，需要具有以下参数：

参数名	说明	来源
appId	业务流程唯一标识，即WBappid	参考 获取 WBappid 指引在人脸核身控制台内申请
userId	用户唯一标识	合作方自行分配（和 SDK 里面定义的 userId 保持一致）
version	参数值为：1.0.0	-
ticket	合作伙伴服务端获取的 ticket，注意是 NONCE 类型	获取方式见 NONCE ticket 获取
nonce	必须是32位随机数	合作方自行生成（和 SDK 里面定义的随机数保持一致）

> **注意：**
> 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。

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 appId、userId、version 连同 ticket、nonce 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	IDAXXXXX
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
ticket	XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

- 字典排序后的参数为：

```
[1.0.0, IDAXXXXX, XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS ,
kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551]
```

- 拼接后的字符串为：

```
1.0.0IDAXXXXXXO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7
```

```
TuserID19959248596551
```

- **计算 SHA1 得到签名：**

```
D7606F1741DDCF90757DA924EDCF152A200AC7F0
```

该字符串就是最终生成的签名（ 40 位），不区分大小写。

Android人脸核身
开发准备

最近更新时间：2024-12-20 15:23:02

权限检测

- **Android 6.0 以上系统**

  SDK 需要用到相机，对 Android 6.0 以上的系统会做权限的运行时检测。
- **Android 6.0 以下系统**
- 由于 Android 6.0 以下系统 Android 并没有运行时权限，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出现异常，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，建议合作方在拉起 SDK 前，帮助 SDK 做相机权限检测，提示并确认用户打开了这项权限后，再进行人脸核身，可以使整个人脸核身体验更快更好。

CPU 平台设置

目前 SDK 只支持 armeabi-v7a、armeabi-v8a 平台，为了防止在其他 CPU 平台上 SDK crash，建议在您的 App 的 build.gradle 里加上 abiFilter，如下代码所示，您也可以根据您的 App CPU 平台情况进行相应的设置和过滤：

> ⚠ **注意：**
> SDK 支持 armeabi-v7a 和 armeabi-v8a 两个平台，合作方可以根据自身情况设置需要的 CPU 平台。

```
defaultConfig {
    ndk {
          //设置支持的so库框架
          abiFilters 'armeabi-v7a', 'arm64-v8a'
     }
}
```

配置流程

最近更新时间：2024-12-20 15:23:02

注意事项

- 人脸核身 SDK（WbCloudFaceLiveSdk）最低支持到 Android API 16: Android 4.1.0（ICS），请在构建项目时注意。
- 人脸核身 SDK 将以 AAR 文件的形式提供，默认黑色皮肤，无需格外设置。
- 人脸核身 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。
- 需要为人脸核身 SDK 添加依赖，方式如下： 将提供的 AAR 文件加入到 App 工程的 libs 文件夹下，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-v7
 compile 'com.android.support:appcompat-v7:23.0.1'
 //1. 云刷脸SDK，
//aar的名称，例如：WbCloudFaceLiveSdk-v6.0.0-1234567.aar，填入'WbCloudFaceLiveSdk-v6.0.0-1234567'
 compile(name: 'aar的名称', ext: 'aar')
//2. 云normal SDK，
//aar的名称，例如：WbCloudNormal-v5.1.10-123456789.aar，填入 'WbCloudNormal-v5.1.10-123456789.aar'
 compile(name: 'aar的名称', ext: 'aar')
}
    }
```

混淆配置

人脸核身 SDK 的混淆规则

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

```
#####################云刷脸混淆规则    faceverify-BEGIN#######################
####
#不混淆内部类
-keepattributes InnerClasses

#不混淆jni调用类
-keepclasseswithmembers class *{
    native <methods>;
}

##################### faceverify-BEGIN ##########################
-ignorewarnings
-keep public class com.tencent.ytcommon.**{*;}
-keep class com.tencent.turingfd.sdk.mfa.TNative$aa { public *; }
-keep class com.tencent.turingfd.sdk.mfa.TNative$aa$bb { public *;}
-keep class com.tencent.turingcam.** {*;}
-keep class com.tencent.turingfd.** {*;}

-keep public class com.tencent.youtu.ytagreflectlivecheck.jni.**{*;}
-keep public class com.tencent.youtu.ytagreflectlivecheck.YTAGReflectLiveCheckInterface{
    public <methods>;
}
-keep public class com.tencent.youtu.ytposedetect.jni.**{*;}
```

```
-keep public class com.tencent.youtu.ytposedetect.data.**{*;}
-keep public class com.tencent.youtu.liveness.YTFaceTracker{*;}
-keep public class com.tencent.youtu.liveness.YTFaceTracker$*{*;}

-keep public class com.tencent.cloud.huiyansdkface.facelight.net.*$*{
    *;
}
-keep public class com.tencent.cloud.huiyansdkface.facelight.net.**{
    *;
}
-keep public class com.tencent.cloud.huiyansdkface.facelight.config.cdn.WbUiTips{
    *;
}

#================数据上报混淆规则 start==========================
#实体类
-keep class com.tencent.cloud.huiyansdkface.analytics.EventSender{
    *;
}
-keep class com.tencent.cloud.huiyansdkface.analytics.EventSender$*{
    *;
}
-keep class com.tencent.cloud.huiyansdkface.analytics.WBSAEvent{
     *;
}
-keep class com.tencent.cloud.huiyansdkface.analytics.WBSAParam{
     *;
}
#================数据上报混淆规则 end==========================

######################faceverify-END##############################

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

-keep public class com.tencent.cloud.huiyansdkface.normal.net.*$*{
    *;
}
-keep public class com.tencent.cloud.huiyansdkface.normal.net.*{
    *;
}
#bugly
-keep class com.tencent.bugly.idasc.**{
    *;
}
#wehttp混淆规则
-dontwarn com.tencent.cloud.huiyansdkface.okio.**
-keep class com.tencent.cloud.huiyansdkface.okio.**{
    *;
}
-dontwarn com.tencent.cloud.huiyansdkface.okhttp3.OkHttpClient$Builder

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

接口调用

最近更新时间：2025-06-06 17:12:52

SDK 接口调用方法

SDK 代码调用的入口为 WbCloudFaceVerifySdk 这个类。

```
public class WbCloudFaceVerifySdk {

/**
* 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 WbCloudFaceVerifySdk getInstance(){
//     ...
}
/**
* 在使用 SDK 前先初始化，传入需要的数据 data
* 由 WbCloudFaceVerifyLoginListener 返回是否登录 SDK 成功
* 关于传入数据 data 见后面的说明
*/
public void initPlusSdk (Context context,Bundle data,
WbCloudFaceVerifyLoginListener loginListerner){
//     ...
}
/**
* 登录成功后，调用此函数拉起 sdk 页面。
* 由 startWbFaceVerifySdk 返回刷脸结果。
*/
public void startWbFaceVerifySdk(Context ctx,
WbCloudFaceVerifyResultListener listener) {     // ...
}
/**
 * 释放资源，防止内存泄漏。收到刷脸结果后即可调用
 */
public void release() {
}
```

- NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。
- WbCloudFaceVerifySdk.initPlusSdk()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以下内容（参数详情请参见 接口参数说明 ）：

> **注意：**
> 以上参数被封装在 WbCloudFaceVerifySdk.InputData 对象中（它是一个 Serializable 对象）。

接入示例

详情请参见 Android 光线活体 + 人脸比对 接入示例 。

接口参数说明

InputData 对象说明

InputData 是用来给 SDK 传递一些必须参数所需要使用的对象（WbCloudFaceVerifySdk.initPlusSdk() 的第二个参数），合作方需要传入SDK 需要的一些数据以便启动刷脸 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（字节）	是否必填
faceId	刷脸 ID 号，获取方式请参见 合作方上传身份信息	String	-	是
agreementNo	订单号，合作方订单的唯一标识	String	32	是

openApiAppId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	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	User Id，每个用户唯一的标识	String	32	是
openApiSign	获取方式请参考 生成 SDK 接口调用步骤使用签名	String	40	是
verifyMode	刷脸类型：分级模式 FaceVerifyStatus.Mode.GRADE	FaceVerifyStatus.Mode	-	是
keyLicence	在 人脸核身控制台 内申请的SDKlicense	String	以实际申请为准	是

个性化参数设置（可选）

WbCloudFaceVerifySdk.initPlusSdk()里Bundle data，除了必须要传的 InputData 对象之外，还可以由合作方为其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 支持境外调用

SDK 默认不支持境外调用。如果合作方需支持境外调用，可以通过该字段进行设置。设置代码如下：

```
# 优先使用境外域名，默认false，不使用。
# 请注意：如果合作方可以分辨国内或者境外ip，建议境外ip开启这个配置，国内ip访问不进行配置
data.putBoolean(WbCloudFaceContant.IS_ABROAD, true);
```

选择 SDK 提示语言类型

合作方可以选择 SDK 显示提示语的语言类型，默认简体中文，可以选择繁体中文，英语，日语，韩语，泰语和印尼语。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(WbCloudFaceContant.INPUT_DATA, inputData);
  //简体中文  WbCloudFaceContant.LANGUAGE_ZH_CN
  //繁体中文  WbCloudFaceContant.LANGUAGE_ZH_HK
  //英语  WbCloudFaceContant.LANGUAGE_EN
  //印尼语  WbCloudFaceContant.LANGUAGE_ID
  //日语  WbCloudFaceContant.LANGUAGE_JA
  //韩语  WbCloudFaceContant.LANGUAGE_KO
  //泰语 WbCloudFaceContant.LANGUAGE_TH
  //默认设置为简体中文，此处设置为英文
  data.putString(WbCloudFaceContant.LANGUAGE,WbCloudFaceContant.LANGUAGE_EN);
```

特别注意，在设置了国际化语言的情况下，SDK 不会播放语音，自定义退出框也无效。同时合作方设定的自定义提示语也需要合作方传入相应语种的提示语。

SDK 样式选择

合作方可以选择 SDK 样式。目前 SDK 有黑色模式和白色模式两种，默认显示白色模式。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(WbCloudFaceContant.INPUT_DATA, inputData);
  //对 sdk 样式进行设置，默认为白色模式
  //此处设置为白色模式
  data.putString(WbCloudFaceContant.COLOR_MODE, WbCloudFaceContant.WHITE);
```

SDK 还支持自定义皮肤，支持定制刷脸过程中各个组件的色值，因涉及可设置元素较多，此处可以参考接入 demo。使用自定义皮肤时需要设置：

```
# 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(WbCloudFaceContant.INPUT_DATA, inputData);
  //对 sdk 样式进行设置，默认为白色模式
  //此处设置为自定义模式
  data.putString(WbCloudFaceContant.COLOR_MODE, WbCloudFaceContant.CUSTOM);
```

可配置的颜色值如下：

```
# 在App的colors.xml中设置：
    <!--供客户定制颜色值，可修改仅自己需要修改的颜色，其余复制粘贴demo示例即可-->
     <!--请注意，根据客户定制的需求，有些色值除了需要在用户app的colors里加上以下所有配置之外，还需要同步增加mipmaps和drawable的
xml文件-->
     <!--详情见下面说明，所有的定制res都以wbcf_custom开头，用户可以参考demo使用-->

     <!--授权页面还有一张背景图需要设置,请在mipmap文件夹里替换-->
     <!--授权页面titlebar背景颜色-->
     <color name="wbcf_custom_auth_title_bar">#ffffff</color>
     <!--授权页面titlebar返回键颜色-->
     <color name="wbcf_custom_auth_back_tint">#ffffff</color>
     <!--授权页面背景颜色-->
     <color name="wbcf_custom_auth_bg">#ffffff</color>
     <!--授权页面标题颜色-->
     <color name="wbcf_custom_auth_title">#1d2232</color>
     <!--授权页面文字颜色-->
     <color name="wbcf_custom_auth_text">#a5a7ad</color>
     <!--授权页面协议名称颜色-->
     <color name="wbcf_custom_auth_name_text">#1d2232</color>
     <!--授权页面按钮文字颜色-->
     <color name="wbcf_custom_auth_btn_text">#80ffffff</color>
     <!--授权页面被选中按钮背景色，需要配合drawable/wbcf_custom_auth_btn_checked.xml使用-->
     <color name="wbcf_custom_auth_btn_checked_bg">#1e2642</color>
     <!--授权页面未选中按钮背景色，需要配合drawable/wbcf_custom_auth_btn_unchecked.xml使用-->
     <color name="wbcf_custom_auth_btn_unchecked_bg">#9ca1ae</color>

     <!--识别页面背景色-->
     <color name="wbcf_custom_verify_bg">#ffffff</color>
     <!--识别页面初始化圆框-->
     <color name="wbcf_custom_initial_border">#33ffffff</color>
     <!--识别页面有脸时圆框颜色-->
     <color name="wbcf_custom_border">#ffb155</color>
     <!--识别页面有脸时提示文字颜色-->
     <color name="wbcf_custom_tips_text">#1d2232</color>
     <!--识别页面人脸不符合条件时圆框颜色-->
     <color name="wbcf_custom_border_error">#fa5a5a</color>
     <!--识别页面人脸不符合条件时提示文字颜色-->
     <color name="wbcf_custom_tips_text_error">#fa5a5a</color>
     <!--识别页面用户自定义提示文字颜色-->
     <color name="wbcf_custom_customer_tip_text">#5d646d</color>
     <!--识别页面长提示文字颜色-->
     <color name="wbcf_custom_long_tip_text">#777A84</color>
     <!--识别页面长提示背景颜色,需要配合drawable/wbcf_custom_long_tip_bg.xml使用-->
     <color name="wbcf_custom_long_tip_bg">#fafafa</color>

     <!--弹窗背景颜色-->
     <color name="wbcf_custom_dialog_bg">#ffffff</color>
     <!--弹窗标题文字颜色-->
     <color name="wbcf_custom_dialog_title_text">#363C62</color>
     <!--弹窗内容文字颜色-->
     <color name="wbcf_custom_dialog_text">#363C62</color>
     <!--弹窗右边按钮文字颜色-->
     <color name="wbcf_custom_dialog_right_text">#363C62</color>
```

```
    <!--弹窗左边按钮文字颜色-->
    <color name="wbcf_custom_dialog_left_text">#363C62</color>
```

合作方个性化提示定制（国际化语言模式下不支持）

合作方可以设置定制的提示语，通过配置传给 SDK。自定义提示语分为短提示（不长于17个字符）和长提示（不长于70个字符）。如果不设置，默认无。如果超过字数限制，将会被截断显示。自定义短提示分为验证阶段提示和上传阶段提示，可以分开设置。位置可以选择位于预览圆框的上方或者下方。默认自定义短提示位于预览圆框下方。若设置自定义短提示位于预览圆框上方的话，SDK 过程提示将自动调整到圆框下方。 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(WbCloudFaceContant.INPUT_DATA, inputData);
  //定制合作方个性化提示语
  //此处将设置人脸采集时的个性化提示语
  data.putString(WbCloudFaceContant.CUSTOMER_TIPS_LIVE, “扫描人脸后与您身份证进行对比”);
  //此处将设置上传人脸时的个性化提示语
  data.putString(WbCloudFaceContant. CUSTOMER_TIPS_UPLOAD, “已提交审核，请等待结果”);
  //设置合作方定制提示语的位置，默认为识别框的下方
  //识别框的下方： WbCloudFaceContant.CUSTOMER_TIPS_LOC_BOTTOM
  //识别框的上方：WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP
  data.putInt(WbCloudFaceContant.CUSTOMER_TIPS_LOC,WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP);
```

自定义长提示位于整个验证页面的下方，存在于整个刷脸流程中，不能设置位置和时机。长提示的显示除了设置代码外，还需邮件申请开通，具体 联系小助手 。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(WbCloudFaceContant.INPUT_DATA, inputData);
//整个识别过程中合作方定制长提示语（不超过70个字符）
  data.putString(WbCloudFaceContant.CUSTOMER_LONG_TIP, customerLongTip);
```

合作方还可以自定义 SDK 退出二次确认弹窗的文字内容，包括提示标题，提示内容，确认键文案和取消键文案。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(WbCloudFaceContant.INPUT_DATA, inputData);
//退出确认弹窗定制提示标题
  data.putString(WbCloudFaceContant.DIALOG_TITLE, dialogTitle);
  //退出确认弹窗定制提示内容
  data.putString(WbCloudFaceContant.DIALOG_TEXT, dialogText);
  //退出确认弹窗定制确认键文案
  data.putString(WbCloudFaceContant.DIALOG_YES, dialogYes);
   //退出确认弹窗定制取消键文案
   data.putString(WbCloudFaceContant.DIALOG_NO, dialogNo);
```

比对类型选择

SDK 提供权威数据源比对、自带比对源比对，合作方可以选择传入参数控制对比类型。

- **权威库网纹图片比对类型**

  合作方必须要先在获取 faceId 的接口里送入用户的姓名与身份证号信息，得到相应的 faceId 后，再送入 SDK，供 SDK 与权威库网纹图片进行比对。不需要额外设置对比类型。
- **自带比对源比对类型**

合作方在获取 faceId 的接口里送入用户提供对比源数据，获取 faceId 后送入 SDK 进行对比。合作方可以上传两类照片，一类是网纹照，一类是高清照；不需要额外设置对比类型。

是否录制视频存证

SDK 默认不录制视频存证。如果合作方需要视频存证，可以通过该字段进行设置。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(WbCloudFaceContant.INPUT_DATA, inputData);
  //设置是否录制视频进行存证，默认不录制存证。
  //此处设置为录制存证
  data.putBoolean(WbCloudFaceContant.VIDEO_UPLOAD, true);
```

是否对录制视频进行检查

> **说明：**
> 如果在 “是否录制视频存证” 步骤的设置为录制存证，则目前该字段有效；否则，设置为不录制存证，也不会对视频进行检查，设置无效。

- 在 SDK 使用过程中，发现视频录制在性能不太好的手机上可能会报错，导致刷脸中断，影响用户体验。
- 为了减少录制视频导致的人脸核身中断问题，SDK 默认设置对录制的视频不作检测。
- 如果合作方对人脸核身安全有更加严格的要求，可以考虑打开这一选项。但打开这个字段可能导致某些低性能手机上用户人脸核身不能进行，请慎重考虑。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(WbCloudFaceContant.INPUT_DATA, inputData);
  //设置是否对录制的视频进行检测，默认不检测
  //此处设置为检测
  data.putBoolean(WbCloudFaceContant.VIDEO_CHECK, true);
```

是否播放语音提示（国际化语言模式下不支持）

SDK 默认不打开语音提示。合作方可以根据自身需求选择是否开启，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);
  //设置是否打开语音提示，默认关闭
  //此处设置为打开
  data.putBoolean(WbCloudFaceContant. PLAY_VOICE, true);
```

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(WbCloudFaceContant.INPUT_DATA, inputData);
  //个性化参数设置，此处均设置为与默认相反
   //默认设置为简体中文，此处设置为英文
  data.putString(WbCloudFaceContant.LANGUAGE,WbCloudFaceContant.LANGUAGE_EN);
  //sdk样式设置，默认为白色
  //此处设置为白色
  data.putString(WbCloudFaceContant.COLOR_MODE, WbCloudFaceContant.WHITE);
  //定制合作方个性化提示语
  //此处将设置人脸采集时的个性化提示语
  data.putString(WbCloudFaceContant.CUSTOMER_TIPS_LIVE, “扫描人脸后与您身份证进行对比”);
  //此处将设置上传人脸时的个性化提示语
  data.putString(WbCloudFaceContant.CUSTOMER_TIPS_UPLOAD, “已提交审核，请等待结果”);
  //设置合作方定制提示语的位置，默认为识别框的下方
  //识别框的下方： WbCloudFaceContant.CUSTOMER_TIPS_LOC_BOTTOM
  //识别框的上方：WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP
  //此处设置为识别框的上方
  data.putInt(WbCloudFaceContant.CUSTOMER_TIPS_LOC,WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP);
  //设置选择的比对类型  默认为权威数据源对比
  //此处设置权威数据源比对
  data.putString(WbCloudFaceContant.COMPARE_TYPE, WbCloudFaceContant.ID_CARD);
```

```
//是否需要录制上传视频 默认不需要，此处设置为不需要
data.putBoolean(WbCloudFaceContant.VIDEO_UPLOAD, false);
//是否对录制视频进行检查，默认不检查，此处设置为不检查
data.putBoolean(WbCloudFaceContant.VIDEO_CHECK, false);
//设置是否打开语音提示，默认关闭，此处设置为关闭
data.putBoolean(WbCloudFaceContant. PLAY_VOICE, false);
```

核身结果返回

最近更新时间：2025-01-07 10:21:42

Android SDK 结果分别由以下两个回调返回：
如果您只需要获取刷脸结果，不需要拉取视频和照片，您可以信任前端 SDK 核身结果，返回中的 isSuccess 字段 True 代表人脸核身对比成功；false 代表人脸核身失败。核身 SDK 和核身后台服务之间通信采用加密方式，可以有效防止结果篡改。

```
/**
  * 登录回调接口  返回登录 sdk 是否成功
  */
public interface WbCloudFaceVerifyLoginListener {
        void onLoginSuccess();
        void onLoginFailed(WbFaceError error);
    }

/**
  * 刷脸结果回调接口
  */
public interface WbCloudFaceVerifyResultListener {
        void onFinish(WbFaceVerifyResult result);
```

WbFaceVerifyResult 对象说明

WbFaceVerifyResult 是 SDK 用来给合作方传递身份识别结果的对象，在 WbCloudFaceVerifyResultListener回调中作为参数返回给合作方 App。
WbFaceVerifyResult 对象的各个字段意义如下表所示：

字段名	类型	字段含义	说明
isSuccess	boolean	人脸核身是否成功	True 代表人脸核身对比成功。 false 代表人脸核身失败，具体的失败原因请参考 WbFaceError 对象说明 。
sign	String	签名	供 App 校验人脸核身结果的安全性。
liveRate	String	活体检测分数	-
similarity	String	人脸比对分数	“仅活体检测”类型不提供此分数。
RiskInfo	自定义对象	后台返回的刷脸风险信息	请注意：部分情况下此字段不返回，请不要做强校验。
WbFaceError	自定义对象	人脸核身错误	人脸核身成功时为 null。

RiskInfo 对象说明

RiskInfo 是 SDK 用来给合作方传递刷脸风险信息的对象，在 WbCloudFaceVerifyLoginListener 回调的WbFaceVerifyResult 对象中作为参数返回给合作方 App。
其中 RiskInfo 对象的各个字段意义如下表所示：

字段名	类型	字段含义	说明
deviceInfoLevel	String	设备风险等级	设备风险级别，如果命中风险则显示
deviceInfoTag	String	设备风险标签	设备风险等级描述
riskInfoLevel	String	身份风险等级	如果命中身份风险则显示风险等级
riskInfoTag	String	身份风险标签	身份风险等级描述

详情见 Plus 版人脸核身错误码风险标签说明 。

WbFaceError 对象说明

WbFaceError 是 SDK 用来给合作方传递人脸核身错误信息的对象，在 WbCloudFaceVerifyLoginListener 回调和WbFaceVerifyResult 对象中作为参数返回给合作方 App。WbFaceError 对象的各个字段意义如下表所示，各个字段的内容取值详情请参见 错误码。

字段名	类型	字段含义	说明
domain	String	错误发生的阶段	只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，后端无法查询到刷脸结果。
code	String	错误码	-
desc	String	错误描述	如有需求，可以展示给用户。
reason	String	错误信息内容	错误的详细实际原因，主要用于定位问题。

Android 错误码

最近更新时间：2024-12-20 15:23:02

SDK 在登录以及返回人脸服务结果时，如果发生错误或者识别失败会返回 WBFaceError 对象，其中各个字段的内容如下：

WBFaceErrorDomainParams

错误码	说明	原因
11000	传入参数为空	传入的 xx 为空
11001	传入的 keyLicence 不可用	传入的 keyLicence 不可用
11002	报文加解密失败	报文加解密失败

WBFaceErrorDomainLoginNetwork

错误码	说明	原因
21100	网络异常	登录时网络异常（请求未到达后台）
21200	网络异常	登录时后台返回参数有误（请求已到达后台）

WBFaceErrorDomainLoginServer

错误码	说明	原因
其他错误码	透传后台错误码	例如签名问题等

WBFaceErrorDomainGetInfoNetwork

错误码	说明	原因
31100	网络异常	获取活体类型/光线资源，网络异常（请求未到达后台）
31200	网络异常	获取活体类型/光线资源，后台返回参数有误（请求到达后台）

WBFaceErrorDomainNativeProcess

错误码	说明	原因
41000	用户取消	回到后台/单击 home/左上角/上传时左上角取消
41001	无法获取唇语数据	获取数字活体的数字有问题
41002	权限异常，未获取权限	相机
41003	相机运行中出错	-
41004	视频录制中出错	不能存/启动失败/结束失败
41005	请勿晃动人脸,保持姿势	未获取到最佳图片
41006	视频大小不满足要求	视频大小不满足要求
41007	超时	预检测/动作活体
41008	检测中人脸移出框外	活体/数字/反光
41009	光线活体本地错误	-
41010	风险控制超出次数	用户重试太多次
41011	没有检测到读数声音	数字活体过程中没有发声
41012	初始化模型失败，请重试	初始化算法模型失败

41013	初始化 SDK 异常	WbCloudFaceVerifySdk 未被初始化
41014	简单模式本地加密失败	编码转换异常/加解密编码失败

WBFaceErrorDomainCompareNetwork

错误码	说明	原因
51100	网络异常	对比时，网络异常（请求未到达后台）
51200	网络异常	对比时，后台返回参数有误（请求已到达后台）

WBFaceErrorDomainCompareServer

错误码	说明	原因
其他错误码	透传后台错误码	–

接入示例

最近更新时间：2024-12-20 15:23:02

权威库网纹图片比对、自带对比源对比接入示例：

```
# 在 MainActivity 中单击某个按钮的代码逻辑：
//先填好数据
Bundle data = new Bundle();
WbCloudFaceVerifySdk.InputData inputData = new WbCloudFaceVerifySdk.InputData(
                faceId,
                agreementNo,
                openApiAppId,
                openApiAppVersion,
                openApiNonce,
                userId,
                userSign,
                verifyMode,
                keyLicence);
  data.putSerializable(WbCloudFaceContant.INPUT_DATA, inputData);

//个性化参数设置,可以不设置，不设置则为默认选项。
//默认设置为简体中文，此处设置为英文
data.putString(WbCloudFaceContant.LANGUAGE,WbCloudFaceContant.LANGUAGE_EN);
//sdk样式设置，默认为白色
//此处设置为白色
data.putString(WbCloudFaceContant.COLOR_MODE, WbCloudFaceContant.WHITE);
//定制合作方个性化提示语，默认不设置
//此处将设置人脸采集时的个性化提示语data.putString(WbCloudFaceContant.CUSTOMER_TIPS_LIVE, “扫描人脸后与您身份证进行对
比”);
//此处将设置上传人脸时的个性化提示语
data.putString(WbCloudFaceContant. CUSTOMER_TIPS_UPLOAD, “已提交审核，请等待结果”);
//设置合作方定制提示语的位置，默认为识别框的下方
//识别框的下方： WbCloudFaceContant.CUSTOMER_TIPS_LOC_BOTTOM
//识别框的上方：WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP
//此处设置为识别框的上方
data.putInt(WbCloudFaceContant.CUSTOMER_TIPS_LOC,WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP);
//设置选择的比对类型  默认为权威库网纹图片比对
//此处设置权威数据源对比
data.putString(WbCloudFaceContant.COMPARE_TYPE, WbCloudFaceContant.ID_CARD);
//是否需要录制上传视频 默认不需要，此处设置为不需要
data.putBoolean(WbCloudFaceContant.VIDEO_UPLOAD, false);
//是否对录制视频进行检查，默认不检查，此处设置为不检查
data.putBoolean(WbCloudFaceContant.VIDEO_CHECK, false);
//设置是否打开语音提示，默认关闭，此处设置为关闭
data.putBoolean(WbCloudFaceContant. PLAY_VOICE, false);

//初始化 SDK，得到是否登录 SDK 成功的结果，由 WbCloudFaceVerifyLoginListener  返回登录结果
//【特别注意】建议对拉起人脸识别按钮做防止重复点击的操作
//避免用户快速点击导致二次登录，二次拉起刷脸等操作引起问题
WbCloudFaceVerifySdk.getInstance().
    initPlusSdk (DemoActivity .this, data, new WbCloudFaceVerifyLoginListener() {
        @Override
        public void onLoginSuccess () {
            //登录成功，拉起 sdk 页面，由 FaceVerifyResultListener  返回刷脸结果
            WbCloudFaceVerifySdk.getInstance().startWbFaceVerifySdk(DemoActivity.this, new
WbCloudFaceVerifyResultListener() {
                @Override
                public void onFinish(WbFaceVerifyResult result) {
                    if (result != null) {
                        if (result.isSuccess()) {
                            Log.d(TAG, "刷脸成功！");
```

```
                    } else {
                        Log.d(TAG, "刷脸失败！");
                    }
                }
                //刷脸结束后，及时释放资源
                WbCloudFaceVerifySdk.getInstance().release();
            }
        });
    }
    @Override
    public void onLoginFailed (WbFaceError error){
        Log.d(TAG, "登录失败！");

        //刷脸结束后，及时释放资源
        WbCloudFaceVerifySdk.getInstance().release();
    }
});
```

iOS 人脸核身
配置流程

最近更新时间：2025-06-06 17:12:52

使用 Cocoapod 集成

SDK 最低支持到 iOS11.0，请在构建项目时候注意，目前仅支持 Xcode11.0 及更高版本编译。如需特别支持 iOS8.0 版本，请联系技术支持。 以下为接入配置的步骤：

1. 将 TencentCloudHuiyanSDKFace_framework 文件夹拷贝到自己项目的 podfile 文件所在的同一目录。
2. 在 podfile 使用如下配置：

> ⚠ **注意：**
> target 后面内容根据自己项目配置，demo 可参考 SDK，请到控制台自助接入列表页获取密码。

```
target 'WBCloudReflectionFaceVerify-Demo' do
pod 'TencentCloudHuiyanSDKFace_framework', :path=> './TencentCloudHuiyanSDKFace_framework'
end
```

3. 使用 pod install 命令。
4. SDK 需要使用相机权限，请在 info.plist 中添加：

```
Privacy - Camera Usage Description
```

直接引用 Framework

SDK 最低支持到 iOS11.0，请在构建项目时候注意。 以下为接入配置的步骤：

1. 引用以下资源文件到项目：

```
TencentCloudHuiyanSDKFace.framework
TuringShieldCamRisk.framework
YTFaceTrackerLiveness.framework
YTCommonLiveness.framework
YTPoseDetector.framework
YTFaceAlignmentTinyLiveness.framework
YTFaceDetectorLiveness.framework
tnnliveness.framework
YTFaceLiveReflect.framework
TencentCloudHuiyanSDKFace.bundle
face-tracker-003.bundle
YTCv.framework
YtSDKKitFrameworkTool.framework
```

2. SDK 依赖以下系统框架，需要在BuildPhases > Link Binary With Libraries中添加，可以参考 Demo，具体依赖的系统库如下：

```
UIKit.framewrok
AVFoundation.framework
CoreVideo.framework
Security.framework
SystemConfiguration.framework
CoreMedia.framework
CoreTelephony.framework
ImageIO.framework
VideoToolbox.framework
Accelerate.framework
webkit.framework
libc++.tbd
```

```
libz.tbd
```

3. SDK 需要使用相机权限，请在 info.plist 中添加：

```
Privacy - Camera Usage Description
```

4. 需要在 **BuildSettings > Other Linker Flags** 中设置：-ObjC。

接口调用

最近更新时间：2024-12-20 15:23:02

SDK 接口调用方法

SDK 的功能通过WBFaceVerifyCustomerService 这个类的方法进行调用，**其中 SDK 中使用的 nonce，sign 等重要信息，需要合作方从自己后台拉取，并且两者不能缓存，使用后即失效**，详细接口说明如下，其他的操作请参考 Demo 中的登录接口的参数说明：

版本号及宏定义说明

```
#import <UIKit/UIKit.h>
#ifndef WBFaceVerifyConst_h
#define WBFaceVerifyConst_h
#define WBCloudReflectionFaceVerifyVersion

UIKIT_EXTERN NSString *const WBCloudFaceVerifySDKVersion;

/**
 SDK使用的主题风格

 - WBFaceVerifyThemeDarkness: 暗黑色系主题
 - WBFaceVerifyThemeLightness: 明亮色系主题
 - WBFaceVerifyThemeOrange: 橙色主题
 - WBFaceVerifyThemeCustom: 自定义主题，通过修改 bundle 中的 custom.json 实现自定义

 */
typedef NS_ENUM(NSInteger, WBFaceVerifyTheme) {
    WBFaceVerifyThemeDarkness = 0,
    WBFaceVerifyThemeLightness,
    WBFaceVerifyThemeOrange,
    WBFaceVerifyThemeCustom,
};

typedef NS_ENUM(NSInteger, WBFaceVerifyLanguage) {
    WBFaceVerifyLanguage_ZH_CN = 0, //简体中文
    WBFaceVerifyLanguage_ZH_HK,     //繁体中文
    WBFaceVerifyLanguage_EN,        //英语
    WBFaceVerifyLanguage_ID,        //印尼语
    WBFaceVerifyLanguage_JA,        //日语
    WBFaceVerifyLanguage_KO,        //韩语
    WBFaceVerifyLanguage_TH         //泰语
};

typedef NS_ENUM(NSInteger, WBFaceCustomTipsLoc) {
    WBFaceCustomTipsLoc_Bottom = 0, //提示语在下
    WBFaceCustomTipsLoc_Top,
};
#endif /* WBFaceVerifyConst_h */
```

入口方法说明

NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。

faceID +活体检测+人脸比对服务：

```
/*
Plus版SDK核身入口，注意传入的 faceId 不能为空，且必须为Plus版 faceId，否则会报 failure

@param userid 用户唯一标识，由合作方自行定义
@param nonce  满足接入要求的32位随机数
@param sign 获取方式请参考 [生成 SDK 接口调用步骤使用签名](https://cloud.tencent.com/document/product/1007/63358)
```

```
@param appid 业务流程唯一标识，即 wbappid，可参考 获取 WBappid 指引在人脸核身控制台内申请
@param orderNo 每次人脸身份认证请求的唯一订单号：建议为32位字符串（不超过32位）
@param apiVersion 后台 api 接口版本号（不是 SDK 的版本号），默认请填写@"1.0.0"
@param licence 在人脸核身控制台内申请（该 licence 同 App 当前使用的 bundle id 绑定）
@param faceId 合作方必须要先获取*Plus版*faceId，再送入sdk，不允许为空，比对or活体检测服务由 sdkConfig.useAdvanceCompare
设置
@param sdkConfig SDK 基础配置项目
@param success 服务登录成功回调，登录成功以后开始进行活体和检测服务
@param failure 服务登录失败回调，具体参考错误码文档
*/
-(void) initPlusSDKWithUserId:(NSString *)userid
                        nonce:(NSString *)nonce
                         sign:(NSString *)sign
                        appid:(NSString *)appid
                      orderNo:(NSString *)orderNo
                   apiVersion:(NSString *)apiVersion
                      licence:(NSString *)licence
                       faceId:(nonnull NSString *)faceId
                    sdkConfig:(WBFaceVerifySDKConfig *)sdkConfig
                      success:(void (^)())success
                      failure:(void (^)(WBFaceError * _Nonnull))failure;
/**
 以上一次的登录结果拉起刷脸页面，必须先登录再拉起刷脸页面

 @return 拉起是否成功
 */
- (BOOL)startWbFaceVeirifySdk;
```

个性化参数设置

SDK 登录接口 initPlusSDK 方法中需要传入 WBFaceVerifySDKConfig 字段，通过该对象可以配置 SDK 中其他基础配置：包括设置主题风格，资源路径等，仅活体 or 比对服务，通过 useAdvanceCompare 属性区分，务必根据自己的业务进行设置，具体参考头文件。

```
/**
 人脸识别 SDK 基础配置类
 */
@interface WBFaceVerifySDKConfig : NSObject

#pragma mark - common
/**
 sdk中拉起人脸活体识别界面中使用UIWindow时的windowLevel配置，默认配置是1 + UIWindowLevelNormal

 如果接入放app中有其他自定义UIWindow，为了防止界面覆盖，可以酌情设置该参数
 */
@property (nonatomic, assign) NSUInteger windowLevel;

/**
 是否境外用户
 默认：NO
 请注意：如果合作方可以分辨国内或者境外ip，建议境外ip开启这个配置，国内ip访问不进行配置
 */
@property (nonatomic, assign) BOOL isAbroad;

/**
 人脸识别服务是否进行通过录像，从而进行视频存证

 default: NO
 */
@property (nonatomic, assign) BOOL recordVideo;

/**
 是否由 SDK 内部处理 sdk 网络请求的 cookie
```

```
 默认值：YES
 */
@property (nonatomic, assign) BOOL manualCookie;

/**
 人脸识别页面中的主题风格，需要配合不同资源包使用：
 WBFaceVerifyThemeDarkness - 暗灰主题
 WBFaceVerifyThemeLightness - 明亮主题（默认）
 */
@property (nonatomic, assign) WBFaceVerifyTheme theme;
/**
 多语言配置
 默认中文，当使用其他语言时，强制静音
 */
@property (nonatomic, assign) WBFaceVerifyLanguage language;
/**
 是否静音
 默认值：YES
 */
@property (nonatomic, assign) BOOL mute;
/*
 送入自定义提示文案的位置
 默认：WBFaceCustomTipsLoc_Bottom
 */
@property (nonatomic, assign) WBFaceCustomTipsLoc tipsLoc;

/*
 检测过程中展示的文案
 默认为空
 */
@property (nonatomic, copy) NSString *customTipsInDetect;

/*
 上传过程中展示的文案
 默认为空
 */
@property (nonatomic, copy) NSString *customTipsInUpload;

/*
 底部提示文案，长度不超过70字
 */
@property (nonatomic, copy) NSString *bottomCustomTips;
/*
 退出二次确认UI配置
 */
@property (nonatomic, copy) NSString *exitAlertTitle; //标题
@property (nonatomic, copy) NSString *exitAlertMessage; //内容
@property (nonatomic, copy) NSString *exitAlertRight; //右边按钮
@property (nonatomic, copy) NSString *exitAlertLeft; //左边按钮
/*
 如果有使用苹果分屏模式（UIWindowScene），打开此开关
 Xcode11新建工程有使用 Scene，可以参考资料自行调整
 默认为 NO
 */
@property (nonatomic, assign) BOOL useWindowSecene;

/**
 资源路径，不包含bundle本身（仅当需要自己下发资源时配置，本地资源无需配置）
 ！！！重要：此目录下必须包含face-tracker-v001.bundle和WBCloudReflectionFaceVerify.bundle两个文件，否则无法拉起SDK
 */
@property (nonatomic, copy) NSString *bundlePath;

/**
```

```
 是否采用Plus版比对服务，仅Plus版接口生效，仅活体服务设置为 NO
 默认为 NO，注意：如果需要使用完整活体+比对服务请设置为 YES
 */
@property (nonatomic, assign) BOOL useAdvanceCompare;
#pragma mark - simple //非标特有字段，标准模式无需设置
/**
 是否返回录制的视频

 default: NO
 */
@property (nonatomic, assign) BOOL returnVideo;

/**
 返回视频加密的公钥，如果不配置则不加密

 需要recordVideo returnVideo同时为YES，才返回加密的视频内容
 */
@property (nonatomic, copy) NSString *publicKey;

/**
 AES 加密需要用到的 IV
 */
@property (nonatomic, copy) NSString *aesIV;
/**
 默认 sdk 配置
 */
+(instancetype)sdkConfig;

@end
```

自定义皮肤设置

可以通过修改 JSON 文件，自定义刷脸的部分 UI。首先把 Theme 设置为 WBFaceVerifyThemeCustom，再修改 JSON 文件，路径：WBCloudReflectionFaceVerify_framework/Resource/WBCloudReflectionFaceVerify.bundle/custom.json。

```
{
        "name": "custom",
    "statusBarStyle" :   "light",  /** 状态栏颜色 */
    "navbarTintColor":   "0xffffff", /** 导航栏色调 */
    "baseNavBarColor" :  "0x0", /** 导航栏默认颜色 */
    "authNavBarColor" :  "0x22252A", /** 授权详情页导航栏颜色 */

    "imageBgColor" :     "0x23262b", /** 刷脸页背景色 */
    "faceNormalColor": "0x33FFFFFF", /** 默认刷脸框颜色 */
    "faceSatisfyColor": "0x5065FF", /** 人脸框识别成功颜色 */
    "faceErrorColor": "0xFF6034", /** 人脸框识别失败颜色 */

    "guideLabelColor" :  "0xffffff", /** 提示语默认颜色 */
    "guideLabelErrorColor" : "0xff5240", /** 识别错误时提示语颜色 */
    "customTipsColor":   "0xFFFFFF", /** 自定义提示语颜色 */
    "bottomTipsBgColor": "0x0DFFFFFF", /** 底部提示背景色 */
    "bottomTipsTextColor": "0x80FFFFFF", /** 底部提示文案颜色 */

    "backButtonImage":"backbutton@dark", /** 刷脸页面返回按钮图片 */
    "authBackButtonImage":"backbutton@dark", /** 授权页面返回按钮图片 */
    "authBodyImage": "wbcf_auth_face@dark", /** 授权页人脸框图片 */
    "authCheckboxUnSelectImage":"wbcf_checkbox_unselect", /** 授权页勾选框未勾选图片 */
    "authCheckboxSelectImage":"wbcf_checkbox_select", /** 授权页勾选框勾选图片 */
    "authCheckboxTipsColor": "0xFFFFFF", /** 授权页勾选提示语颜色 */
    "authCheckboxLinkColor": "0x6F80FF", /** 授权页详情链接提示颜色 */
    "authAgreeButtonNormalColor": "0xB7C0FF", /** 授权页同意按钮默认颜色 */
    "authAgreeButtonHighlightColor": "0x5065FF", /** 授权页同意按钮高亮颜色 */
    "authAgreeTextNormalColor":"0xFFFFFF", /** 授权页同意按钮文案默认颜色 */
```

```
    "authAgreeTextHighlightColor":"0xFFFFFF", /** 授权页同意按钮文案高亮颜色 */

    "alertTitleColor":"0x000000", /** 弹框主题颜色 */
    "alertMessageColor":"0x000000", /** 弹框详细信息颜色 */
    "alertLeftBtnColor":"0x5065FF", /** 弹框左边按钮文案颜色 */
    "alertRightBtnColor":"0x5065FF", /**弹框右边按钮文案颜色 */
    "alertBackgroundColor":"0xFFFFFF", /** 弹框背景色 */
}
```

注意：

1. 需要替换返回按钮图片时，需要修改对应 navbarTintColor 到按钮颜色。
2. 有透明色时，透明度放在前面，例如 80FFFFFF，FFFFFF 代表白色，80代表透明度。
3. 需要替换图片的目录在 JSON 文件同级目录 common 文件夹下，替换时注意保证相同尺寸大小，否则显示会异常。

接入流程与说明

最近更新时间：2024-12-20 15:23:02

整体工作流程

人脸识别主要流程包括以下步骤：

1. 登录人脸识别 SDK，即 initPlusSDKWithUserId 方法。
2. 登录结果返回，在第一步登录成功以后，再调用 startWbFaceVerifySdk方法拉起人脸识别页面，如果登录失败，会通过 failure block 回调返回一个 WBFaceError 对象。
3. 人脸识别的具体过程，当中各种失败情况以及用户主动退出，直接跳到第5步。
4. 人脸识别前端 SDK 检测结果上传后台进行活体检测以及人脸比对等服务。
5. 结果返回，通过 delegate 回调方法或者注册 NSNotification 的方式获取活体检测/人脸比对的结果，结果封装在 WBFaceVerifyResult。

登录接口说明

iOS 人脸识别接口大部分参数说明请参考头文件注释。

> **注意：**
> userid、nonce、sign、orderNo 等参数建议通过接入方后台透传给前端，前端无需在本地生成。

方法功能一：

如果使用活体检测服务+人脸比对服务（身份证的网纹照片进行对比）方式，则需要传入 faceID，其中 faceID 需要合作方后台调用 合作方后台上传身份信息 生成 faceID，其中 userid、nonce、sign、orderNo 要与生成 sign 时使用的相同。

方法功能二：

如果使用活体检测服务+人脸比对服务（合作方提供的比对源图片进行比对），则需要传入 faceID，其中 faceID 需要合作方后台调用 合作方后台上传身份信息 生成 faceID，其中 userid、nonce、sign、orderNo 要与生成 sign 时使用的相同，在生成 faceID 时传入比对源图片。

识别界面拉起方式说明

默认情况 SDK 会创建一个全屏幕大小的UIWindow，覆盖在当前屏幕上方，并且默认 windowLevel=UIWindowLevelNormal+1。

核身结果返回

最近更新时间：2024-12-20 15:23:02

iOS SDK 中，人脸识别结果返回给接入方有两种方式，两种结果回调都在 Main Thread 完成： 如果您只需要获取刷脸结果，不需要拉取视频和照片，您可以信任前端 SDK 核身结果，返回中的 isSuccess 字段 YES 代表人脸核身对比成功；NO 代表人脸核身失败。核身 SDK 和核身后台服务之间通信采用加密方式，可以有效防止结果篡改。

1. 通过实现 WBFaceVerifyCustomerServiceDelegate 的 WBFaceVerifyCustomerServiceDelegate 方法，通过该方法返回 WBFaceVerifyResult 对象。
2. 无需实现 delegate 方法，通过注册WBFaceVerifyCustomerServiceDidFinishedNotification 通知，在接受到该通知时，进行结果处理。

WBFaceVerifyCustomerServiceDidFinishedNotification通知中，通过获取该通知的 userInfo，获取字典中 key 为 faceVerifyResult对应的 value 对象就是 WBFaceVerifyResult。

WBFaceVerifyResult 说明

字段名	类型	字段含义	说明
isSuccess	BOOL	人脸核身是否成功	YES 代表认证成功，NO 代表认证失败，具体原因参考 人脸核身 iOS 错误码
sign	NSString	签名	供 App 校验人脸核身结果的安全性
liveRate	NSString	活体检测分数	-
similarity	NSString	人脸比对分数	“仅活体检测” 类型不提供此分数
WbFaceError	自定义对象	人脸核身错误	人脸核身成功时为 nil

WBFaceError 对象说明

字段名	类型	字段含义	说明
domain	NSString	错误发生的阶段	只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，后端无法查询到刷脸结果
code	NSString	错误码	-
desc	NSString	错误描述	如有需求，可以展示给用户
reason	NSString	错误信息内容	错误的详细实际原因，主要用于定位问题
RiskInfo	自定义对象	后台返回的刷脸风险信息	请注意：部分情况下此字段不返回，请不要做强校验

RiskInfo 对象说明

RiskInfo 是 SDK 用来给合作方传递刷脸风险信息的对象，在结果回调的 WBFaceVerifyResult 对象中作为参数返回给合作方 App。其中 RiskInfo 对象的各个字段意义如下表所示：

字段名	类型	字段含义	说明
deviceInfoLevel	String	设备风险等级	设备风险级别，如果命中风险则显示
deviceInfoTag	String	设备风险标签	设备风险等级描述
riskInfoLevel	String	身份风险等级	如果命中身份风险则显示风险等级
riskInfoTag	String	身份风险标签	身份风险等级描述

详情见 SaaS 错误码 风险标签说明。

```
@interface WBFaceSimpleModeResult : NSObject

/**
 结果对应的订单号
 */
```

```
@property (nonatomic, copy, readonly) NSString *orderNo;

/**
 接下来用于人脸对比的安全性参数
 */
@property (nonatomic, copy, readonly) NSString *encryptAESKey;

/**
 视频编码
 */
@property (nonatomic, copy, readonly) NSString *userVideoString;

/**
 使用传入 publickey 加密过的 AES
 */
@property (nonatomic, copy, readonly) NSString *videoEncryptAESKey;
/**
 用于后面进行人脸比对的数据参数
 */
@property (nonatomic, copy, readonly) NSString *identifyStr;

@end

/*
 增强级结果，具体参数含义参考后台返回字段，结果为透传
*/
@interface WBFaceRiskInfo : NSObject
@property (nonatomic, copy, readonly) NSString *deviceInfoLevel;
@property (nonatomic, copy, readonly) NSString *deviceInfoTag;
@property (nonatomic, copy, readonly) NSString *riskInfoLevel;
@property (nonatomic, copy, readonly) NSString *riskInfoTag;
@end

/**
 人脸服务返回结果对象
 */
@interface WBFaceVerifyResult : NSObject
/**
 人脸比对结果是否通过：

 YES：表示人脸服务通过
 NO：表示人脸服务不通过
 */
@property (nonatomic, assign, readonly) BOOL isSuccess;
/**
 结果对应的订单号
 */
@property (nonatomic, copy, readonly) NSString *orderNo;
/**
isSuccess == YES 时，sign 有值，可以去后台拉取本次人脸服务的照片，视频存证
 isSuccess == NO  时，sign 无意义
 */
@property (nonatomic, copy, readonly) NSString * sign;
/**
 活体检测服务得分
 isSuccess == YES 时，liveRate 有值：
         1.liveRate 可能是 @"分数为空"，这种情况具体咨询合作方
         2.float类型的字符串，请调用 [liveRate floatValue] 获取具体分数
 isSuccess == NO  时，liveRate 无意义
 */
@property (nonatomic, copy, readonly) NSString * liveRate;
/**
 人脸比对服务得分
```

```
 isSuccess == YES 时, similarity 有值:
        1.similarity 可能是 @"分数为空", 这种情况具体咨询合作方
        2.float类型的字符串, 请调用 [similarity floatValue] 获取具体分数
 isSuccess == NO  时, similarity 无意义
 */
@property (nonatomic, copy, readonly) NSString * similarity;

/**
 人脸比对图片之一

 isSuccess == YES 时，该属性是上送比对图片之一UIImage的base64编码字符串(图片格式是jpg)

 isSuccess == NO  时，如果是 SDK 返回的错误码，该属性为nil，如果是后端返回的错误，该属性是上送比对图片之一UIImage的base64编码
字符串(图片格式是jpg)
 */
@property (nonatomic, copy, readonly) NSString * userImageString;

/**
 isSuccess == YES 时候, error 无意义
 isSuccess == NO  时, error中存储的具体错误信息,参考 WBFaceError.h
 */
@property (nonatomic, strong, readonly) WBFaceError * error;

#pragma mark - 非标专用返回参数

@property (nonatomic, strong, readonly) WBFaceSimpleModeResult *simpleModeResult;

#pragma mark - 增强SDK专用参数
@property (nonatomic, strong, readonly) WBFaceRiskInfo *riskInfo;

#pragma mark -

-(NSString *)description;
@end
```

iOS 错误码

最近更新时间：2024-12-20 15:23:02

SDK 在登录以及返回人脸服务结果时，如果发生错误或者识别失败会返回 WBFaceError 对象，该对象的结构如下,并且在判断具体错误时，需要先根据 domain 字段判断错误发生在 SDK 服务运行中的位置，然后根据 code 判断具体的错误：

```
NS_ASSUME_NONNULL_BEGIN
/*
 错误 domain 划分成两类：

 出现在登录时，通过调用 startXXXX 方法的 failure block 进行回调返回：
 WBFaceErrorDomainInputParams、WBFaceErrorDomainLoginNetwork、WBFaceErrorDomainLoginServer


 人脸服务结果返回(封装在 WBFaceVerifyResult 中)：
 WBFaceErrorDomainGetInfo, WBFaceErrorDomainNativeProcess, WBFaceErrorDomainCompareNetwork,
WBFaceErrorDomainCompareServer
 */

/* 登录时传入参数有误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainInputParams;
/* 登录时网络请求错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainLoginNetwork;
/* 登录时后台拒绝登录，具体参考后台 word 版本错误码，这里直接透传 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainLoginServer;
/* 拉取有效信息时候网络错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainGetInfo;
/* native 本地在活体检测中，某些原因导致错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainNativeProcess;
/* 上送后台比对时,网络错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainCompareNetwork;
/* 后台比对完成,返回比对结果的错误原因*/
UIKIT_EXTERN NSString *const WBFaceErrorDomainCompareServer;

@interface WBFaceError: NSObject
/**
 错误发生的地方，具体的发生地方由上面定义的 WBFaceErrorDomainXXXX 指定
 */
@property (nonatomic, readonly, copy) NSString *domain;
/**
 每个domain中有相应的错误代码，具体的错误代码见
 */
@property (nonatomic, readonly, assign) NSInteger code; //错误代码
@property (nonatomic, readonly, copy) NSString *desc; //获取本地化描述
@property (nonatomic, readonly, copy) NSString *reason; // 错误出现的真实原因
@property (nonatomic, readonly, copy) NSDictionary * _Nullable otherInfo;// 预留接口
@end
```

若 Domain 为：WBFaceErrorDomainInputParams

错误码	描述	详细实际原因
12000	传入参数为空	传入的参数为空
12001	传入的 keyLicence 不可用	传入的 keyLicence 不可用
12002	身份证格式不正确	身份证格式不正确
12003	使用自带对比源，传入参数错误，非 base64	传入的 srcPhotoString 不是 base64
12004	使用自带对比源，传入参数错误，超过1MB	传入的 srcPhotoString 超过1MB

12005	SDK 资源引入版本不匹配	没有引入资源包或者引入的资源包版本与当前 SDK 版本不匹配
12006	订单号不能为0或者超过32位	-
12007	nonce 字符串位数不为32位	-

Domain 为：WBFaceErrorDomainLoginNetwork

错误码	描述	详细实际原因
22100	网络异常	登录时网络异常（请求未到达后台）
22200	网络异常	登录时后台返回参数有误（请求已到达后台）

Domain 为：WBFaceErrorDomainLoginServer

错误码	描述	详细实际原因
其他错误码	透传后台错误码	例如签名问题等

Domain 为：WBFaceErrorDomainGetInfo

错误码	描述	详细实际原因
32100	网络异常	获取活体类型或光线阈值，网络异常(请求未到达后台)
32200	网络异常	获取活体类型或光线阈值，后台返回参数有误（请求已到达后台）

Domain 为：WBFaceErrorDomainNativeProcess

错误码	描述	详细实际原因
42000	用户取消	回到后台或单击 home 或左上角或上传时左上角取消
42001	网络环境不满足认证需求	无网络或2G 网络
42002	权限异常，未获取权限	相机或麦克风或 read phone 或 external 或 storage
42003	相机运行中出错	-
42004	视频录制中出错	不能存或启动失败或结束失败
42005	请勿晃动人脸，保持姿势	未获取到最佳图片
42006	视频大小不满足要求	视频大小不满足要求
42007	超时	预检测/动作活体
42008	检测中人脸移出框外	活体或反光
42009	光线活体本地错误	-
42010	风险控制超出次数	用户重试太多次
42011	没有检测到读数声音	活体过程中没有发声
42012	模型初始化失败	-
42015	报文解密失败	请求报文解密失败

Domain 为：WBFaceErrorDomainCompareNetwork

错误码	描述	详细实际原因
52100	网络异常	对比时，网络异常（请求未到达后台）
52200	网络异常	对比时，后台返回参数有误（请求已到达后台）

Domain 为：WBFaceErrorDomainCompareServer

错误码	描述	详细实际原因
其他错误码	透传后台错误码	-

Harmony NEXT人脸核身

开发准备

最近更新时间：2024-12-20 15:23:03

权限

SDK 需要用到相机权限和网络权限，如下示例：

```
"requestPermissions": [
      {
        "name": "ohos.permission.INTERNET",
      },
      {
        "name": "ohos.permission.GET_NETWORK_INFO"
      },
      {
        "name": "ohos.permission.CAMERA",
        "reason": "face verify",
        "usedScene": {
          "abilities": [
            "EntryAbility",
          ],
          "when": "inuse"
        }
      }
    ]
```

CPU 平台设置

当前 Harmony NEXT 手机都是64位手机，目前 SDK 只支持armeabi-v8a 平台。

配置流程

最近更新时间：2025-06-06 17:12:52

接入配置

SDK（WbCloudFaceLiveSdk-xx.har,具体版本号以 SDK 交付件为准）最低支持到 5.0.0(12)，请在构建项目时注意。
刷脸 SDK 将以 HAR 文件的形式提供。
将提供的 HAR 文件添加到工程的 libs 目录下，并且在 oh-package.json5 中添加下面的配置：

```
"dependencies": {
"wbcloudfaceverifysdk": "file:../libs/ WbCloudFaceVerifySdk.har"
}
```

接口调用

最近更新时间：2025-01-15 17:14:52

本章将展示 SDK 核心接口的调用方法。具体的代码示例参考交付 Demo(demo/module/src/main/ets/pages/HuiYanModePage.ets)。

> ⚠ **注意：**
> 以下内容出现 “刷脸” 一词均可以表示为 “人脸核身”。

SDK 接口调用方法

SDK 代码调用的入口为：WbCloudFaceVerifySdk 这个类。

```
export class WbCloudFaceVerifySdk {

  /**
   * 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 getInstance(): WbCloudFaceVerifySdk {
    …
  }

  /**
   * 在使用SDK前先初始化，传入需要的数据WbFaceVerifyConfig
   * 由 WbCloudFaceVerifyLoginCallback返回是否登录SDK成功
   * 关于传入WbFaceVerifyConfig见后面的说明
   */
  public initPlusSdk(context: Context, config: WbFaceVerifyConfig, loginCallback:
WbCloudFaceVerifyLoginCallback) {
        …
  }

  /**
   * 登录成功后，调用此函数拉起sdk页面。
   * 由 WbCloudFaceVerifyResultCallback返回刷脸结果。
   */
  public startWbFaceVerifySdk(context: Context, resultCallback: WbCloudFaceVerifyResultCallback) {
    …
  }

  /**
   * 拿到刷脸结果后，释放资源， 防止内存泄漏
   * 【注意】请务必在拿到刷脸结果后释放资源，否则可能会发生丢失回调的情况!
   */
  public release() {
    …
  }
}
```

WbCloudFaceVerifySdk.initPlusSdk()的第二个参数 config: WbFaceVerifyConfig 用来传递 sdk 初始化必需数据和sdk 配置项参数。

```
export class WbFaceVerifyConfig {
      //sdk初始化必需数据
  public inputData: InputData;
  //是否开启日志，默认不打开sdk日志，
  //【注意】上线请务必关闭sdk日志!
  public isEnableLog: boolean = false;
}
```

必须传递的参数包括（参考要求详见本页接口参数说明的描述）：

```
//这些都是InputData对象里的字段，是需要传入的数据信息
export class InputData {
  //此次刷脸用户标识，合作方需要向人脸识别后台拉取获得，详见获取faceId接口
  public faceId: string;
//订单号
  public orderNo: string;
//APP_ID
  public appId: string;
  //openapi Version
public version: string;
//32位随机字符串
  public nonce: string;
  //User id
  public userId: string;
  //签名信息
  public sign: string;
  ////在人脸核身控制台内申请
  public licence: string;
}
```

关于接口调用的示例可参考 接入示例 。

接口参数说明

InputData 对象说明

InputData 是用来给 SDK 传递必须参数所需要使用的对象，合作方需要往里塞入SDK 需要的一些数据以便启动刷脸 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（字节）	是否必填
faceId	刷脸id号，由合作方向人脸识别后台拉取获得	String	-	是
orderNo	订单号，合作方订单的唯一标识	String	32	是
appId	业务流程唯一标识，即wbappid，可参考 获取WBappid 指引在人脸核身控制台内申请	String	8	是
version	接口版本号，默认填1.0.0	String	20	是
nonce	32 位随机字符串，每次请求需要的一次性 nonce	String	32	是
userId	User Id，每个用户唯一的标识	String	30	是
sign	获取方式请参考 生成SDK接口调用步骤使用签名	String	40	是
licence	在 人脸核身控制台 内申请的 SDKlicense	String	以实际申请为准	是

个性化参数设置（可选）

WbCloudFaceVerifySdk.initPlusSdk() 里 WbFaceVerifyConfig，除了必须要传的 InputData 对象之外，还可以由合作方为其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。
合作方可以选择 SDK 样式。目前 SDK 有黑色模式和白色模式两种，默认显示白色模式。设置代码如下：

```
//对 sdk 样式进行设置，默认为白色模式
//此处设置为白色模式
wbFaceVerifyConfig.themeMode = ThemeMode.Light
SDK 还支持自定义皮肤，支持定制刷脸过程中各个组件的色值。使用自定义皮肤时需要设置：
wbFaceVerifyConfig.themeMode = ThemeMode.Custom
并在App的resources/rawfile目录中新建wbcf_custom.json主题配置文件：
可配置的颜色值参考 json 文件如下：
{
"name": "custom",
"statusBarStyle": "light",
```

```
"navbarTintColor": "#ffffff",
"baseNavBarColor": "#0",
"authNavBarColor": "#22252A",
"imageBgColor": "#22252A",
"faceNormalColor": "#33FFFFFF",
"faceSatisfyColor": "#5065FF",
"faceErrorColor": "#EB4140",
"guideLabelColor": "#ffffff",
"guideLabelErrorColor": "#EB4140",
"customTipsColor": "#FFFFFF",
"bottomTipsBgColor": "#0DFFFFFF",
"bottomTipsTextColor": "#80FFFFFF",
"backButtonImage": "app.media.wbcf_back_white",
"authBackButtonImage": "app.media.wbcf_back_white",
"authBodyImage_651": "app.media.wbcf_protocal_img",
"authBodyImage_will": "app.media.wbwf_auth_head_image",
"authCheckboxHighlightColor": "#5065FF",
"authCheckboxTipsColor": "#FFFFFF",
"authCheckboxLinkColor": "#6F80FF",
"authAgreeButtonNormalColor": "#B7C0FF",
"authAgreeButtonHighlightColor": "#5065FF",
"authAgreeTextNormalColor": "#FFFFFF",
"authAgreeTextHighlightColor": "#FFFFFF",
"authTipBackgroundColor_651": "#292C31",
"authTipTitileTextColor_651": "#ADB1C6",
"authTipDetailTextColor_651": "#80FFFFFF",
"alertTitleColor": "#000000",
"alertMessageColor": "#000000",
"alertLeftBtnColor": "#5065FF",
"alertRightBtnColor": "#5065FF",
"alertBackgroundColor": "#FFFFFF"
}
```

合作方个性化提示定制（国际化语言模式下不支持）

合作方可以设置定制的提示语，通过配置传给 SDK。自定义提示语分为短提示（不长于17个字符）和长提示（不长于70个字符）。如果不设置，默认无。如果超过字数限制，将会被截断显示。 自定义短提示分为验证阶段提示和上传阶段提示，可以分开设置。位置可以选择位于预览圆框的上方或者下方。默认自定义短提示位于预览圆框下方。若设置自定义短提示位于预览圆框上方的话，SDK 过程提示将自动调整到圆框下方。 设置代码如下：

```
let uiConfig: WbUiConfig = new WbUiConfig();
//设置合作方定制提示语的位置，默认为识别框的下方
//识别框的下方： CustomTipsLocation.Bottom
//识别框的上方： CustomTipsLocation.Top
uiConfig.tipsLocation = CustomTipsLocation.Bottom
//此处将设置人脸采集时的个性化提示语
uiConfig.customTipsInDetect = "扫描人脸后与您身份证进行对比"
//此处将设置上传人脸时的个性化提示语
uiConfig.customTipsInUpload = "已提交审核，请等待结果"
uiConfig.bottomCustomTips = cache.bottomCustomTips
wbFaceVerifyConfig.uiConfig = uiConfig
```

自定义长提示位于整个验证页面的下方，存在于整个刷脸流程中，不能设置位置和时机。长提示的显示除了设置代码外，还需邮件申请开通，详情请 联系我们 。设置代码如下：

```
uiConfig.bottomCustomTips = customerLongTip
```

核身接口返回

最近更新时间：2025-01-07 10:21:42

SDK 结果分别由以下两个回调返回：
如果您只需要获取刷脸结果，不需要拉取视频和照片，您可以信任前端 SDK 核身结果，返回中的 isSuccess 字段 True 代表人脸核身对比成功；false 代表人脸核身失败。核身 SDK 和核身后台服务之间通信采用加密方式，可以有效防止结果篡改。

```
/**
  * 登录回调接口 返回登录 sdk 是否成功
  */
export interface WbCloudFaceVerifyLoginCallback {
  onLoginSuccess: () => void;
  onLoginFail: (error: WbFaceError) => void;
}
/**
  * 刷脸结果回调接口
  */
export interface WbCloudFaceVerifyResultCallback {
  onFinish: (result: WbFaceVerifyResult) => void;
}
```

WbFaceVerifyResult 对象说明

WbFaceVerifyResult 是 SDK 用来给合作方传递身份识别结果的对象，在 WbCloudFaceVerifyResultListener 回调中作为参数返回给合作方 App。
WbFaceVerifyResult 对象的各个字段意义如下表所示：

字段名	类型	字段含义	说明
isSuccess	boolean	人脸核身是否成功	True 代表人脸核身对比成功；false 代表人脸核身失败，具体的失败原因请参考 WbFaceError 对象说明
sign	String	签名	供 App 校验人脸核身结果的安全性
liveRate	String	活体检测分数	-
similarity	String	人脸比对分数	“仅活体检测” 类型不提供此分数
WbFaceError	自定义对象	人脸核身错误	人脸核身成功时为 null

WbFaceError 对象说明

WbFaceError 是 SDK 用来给合作方传递人脸核身错误信息的对象，在 WbCloudFaceVerifyLoginListener 回调和 WbFaceVerifyResult 对象中作为参数返回给合作方 App。WbFaceError 对象的各个字段意义如下表所示，各个字段的内容取值详情请参见 Plus 版人脸核身服务错误码 。

字段名	类型	字段含义	说明
domain	String	错误发生的阶段	只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，后端无法查询到刷脸结果
code	String	错误码	-
desc	String	错误描述	如有需求，可以展示给用户
reason	String	错误信息内容	错误的详细实际原因，主要用于定位问题

错误码描述

最近更新时间：2024-12-20 15:23:03

SDK 在登录以及返回人脸服务结果时，如果发生错误或者识别失败会返回 WBFaceError 对象，该对象的字段结构意义见 核身接口返回，其中各个字段的内容如下：

WBFaceErrorDomainParams

Code（错误码）	Description（描述）	Reason（详细实际原因）
13000	传入参数为空	传入的 xx 为空
13001	传入的 keyLicence 不可用	传入的 keyLicence 不可用
13002	报文加解密失败	报文加解密失败

WBFaceErrorDomainLoginNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
23100	网络异常	登录时网络异常（请求未到达后台）
23200	网络异常	登录时后台返回参数有误（请求到达后台）

WBFaceErrorDomainLoginServer

Code（错误码）	Description（描述）	Reason（详细实际原因）
其他错误码	透传后台错误码	例如签名问题等等

WBFaceErrorDomainGetInfoNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
33100	网络异常	获取活体类型/光线资源，网络异常（请求未到达后台）
33200	网络异常	获取活体类型/光线资源，后台返回参数有误（请求到达后台）

WBFaceErrorDomainNativeProcess

Code（错误码）	Description（描述）	Reason（详细实际原因）
43000	用户取消	回到后台/单击 home /左上角/上传时左上角取消
43001	无法获取唇语数据	获取数字活体的数字有问题
43002	权限异常，未获取权限	相机
43003	相机运行中出错	–
43004	视频录制中出错	不能存/启动失败/结束失败
43005	请勿晃动人脸,保持姿势	未获取到最佳图片
43006	视频大小不满足要求	视频大小不满足要求
43007	超时	预检测/动作活体
43008	检测中人脸移出框外	活体/数字/反光
43009	光线活体本地错误	–
43010	风险控制超出次数	用户重试太多次
43011	没有检测到读数声音	数字活体过程中没有发声
43012	初始化模型失败，请重试	初始化算法模型失败

43013	初始化 sdk 异常	WbCloudFaceVerifySdk 未被初始化
43014	简单模式本地加密失败	编码转换异常/加解密编码失败

WBFaceErrorDomainCompareNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
53100	网络异常	对比时，网络异常（请求未到达后台）
53200	网络异常	对比时，后台返回参数有误（请求到达后台）

WBFaceErrorDomainCompareServer

Code（错误码）	Description（描述）	Reason（详细实际原因）
其他错误码	透传后台错误码	-

接入示例

最近更新时间：2024-12-20 15:23:03

权威库网纹图片比对、自带对比源对比接入示例：

```
#在DemoPage中单击某个按钮的代码逻辑：
//先填好数据
let inputData = new InputData(
      orderNo,
      this.appId,
      '1.0.0',
      this.nonce,
      this.userId,
      sign,
      this.licence,
      faceId
    )

    //设置必需参数
    let wbFaceVerifyConfig = new WbFaceVerifyConfig(inputData);
    //是否打开 sdk 日志开关
    wbFaceVerifyConfig.isEnableLog = true;

    WbCloudFaceVerifySdk.getInstance().initPlusSdk(getContext(), wbFaceVerifyConfig, {
      onLoginSuccess: () => {
        DemoLog.i(this.TAG, `onLoginSuccess:`)
       WbCloudFaceVerifySdk.getInstance().startWbFaceVerifySdk(getContext(), {
          onFinish: (_result: WbFaceVerifyResult) => {
            DemoLog.i(this.TAG, `WbCloudFaceVerifySdk onFinish`)
            //todo 处理刷脸结果
              ……
            //处理完后释放 sdk
            //【特别注意】请在拿到 sdk 结果后对 sdk 进行释放，不要在页面结束时释放，避免未能获取刷脸回调结果的情况
           WbCloudFaceVerifySdk.getInstance().release();
          }
        })
      },
      onLoginFail: (error: WbFaceError) => {
        DemoLog.e(this.TAG, `onLoginFailed:JSON.stringify(error)`)
        //todo 处理登录错误逻辑
       ……
        //【特别注意】请在拿到 sdk 结果后对 sdk 进行释放，不要在页面结束时释放，避免未能获取刷脸回调结果的情况
       WbCloudFaceVerifySdk.getInstance().release();
      }
    })
```

人脸查询核身结果
查询核身结果

最近更新时间：2025-01-15 17:14:52

当您的用户完成核身认证后，如果您需要拉取人脸核身的视频和图片用于存证等其他需要，您可以调用查询核身结果接口来获取。
重要提示：
1. 您需要在前端完成刷脸的回调后，再来调用查询核身结果接口获取刷脸视频和照片。
2. 人脸核身完成后的相关业务数据请尽快拉取，超过人脸核身服务必须最短缓存时间的业务数据将完全清理。

> ⚠ 注意：
> 当您的 App 接入的是我们的基础版或增强版 SDK以及Plus版服务时，SDK 回调中只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过查询核身结果接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，查询核身结果接口无法查询到刷脸结果。

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，获取方式请参见 获取 SIGN ticket 。
- 合作方为人脸核身服务生成签名，需要具有以下参数：

参数	说明	来源
appId	业务流程唯一标识，即 WBappid	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，字母/数字组成的字符串，本次人脸核身合作伙伴上送的订单号，唯一标识	合作方自行分配，不要带有特殊字符
version	默认值：1.0.0	-
api ticket	合作伙伴服务端缓存的 ticket，注意是 SIGN 类型	获取方式请参见 获取 SIGN ticket
nonce	32位随机字符串，由字母和数字组成	合作方自行生成，不要带有特殊字符

基本步骤

1. 生成一个32位的随机字符串 nonce（由字母和数字组成，登录时也要用到）。
2. 将 appId、orderNo、version、api ticket、nonce 共5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 。

人脸核身结果查询接口(升级)

请求

- **请求 URL**：`https://kyc1.qcloud.com/api/v2/base/queryfacerecord?orderNo=xxx`

> ⚠ 注意：
> 为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。

- **请求方法**：POST
- **报文格式**：Content-Type: application/json
- **请求参数**：

参数	说明	类型	长度（字节）	是否必填

appId	人脸核身控制台 申请的 WBappid	String	8	是
version	版本号，默认值：1.0.0	String	20	是
nonce	随机数	String	32	是
orderNo	订单号，字母/数字组成的字符串,是您需要查询结果的人脸核身订单号	String	32	是
sign	签名值，使用本页第一步生成的签名	String	40	是
getFile	是否需要获取人脸识别的视频和文件，值为1则返回视频和照片、值为2则返回照片、值为3则返回视频；其他则不返回	String	1	否
queryVersion	查询接口版本号(传1.0则返回 sdk 版本号和 trtc 标识)	String	8	否

响应

- 响应参数：

参数	类型	说明
code	String	0：表示身份验证成功且认证为同一人
msg	String	返回结果描述
bizSeqNo	String	业务流水号
transactionTime	String	请求接口的时间
orderNo	String	订单编号
liveRate	String	活体检测得分
riskInfo	object	后台返回的刷脸风险信息
similarity	String	人脸比对得分
occurredTime	String	进行刷脸的时间
photo	Base 64 string	人脸核身时的照片，base64 位编码
video	Base 64 string	人脸核身时的视频，base64 位编码
sdkVersion	String	人脸核身时的 sdk 版本号
trtcFlag	String	Trtc 渠道刷脸则标识"Y"
appId	String	腾讯云控制台申请的 WBappid

- 响应示例：


```
{"code":"0",
"msg":"请求成功",
"bizSeqNo":"2204152000118444241549140859447 4",
"result":{
 "orderNo":"testReflect1650008613761",
 "liveRate":"99",
 "riskInfo":{
 "deviceInfoLevel":"1",
 "deviceInfoTag":"",
 "riskInfoLevel":"",
 "riskInfoTag":""
  },
 "similarity":"97.0",
 "occurredTime":"20220415154341",
 "appId":"IDAXXXXX",
 "photo":"*",
 "sdkVersion":"v4.5.2.1",
```

```
  "video":"*",
  "bizSeqNo":"2204152000118444241549140859447 4"
   },
"transactionTime":"20220415154914"
}
```


code 非 0 时，有时不返回图片和视频。

注意事项

- 照片和视频信息作为存证，合作伙伴可以通过此接口拉取视频等文件，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，默认不返回视频和照片信息。
- 由于照片和视频信息有可能超过 1M，考虑传输的影响，建议合作伙伴在使用时注意，建议获取比对结果用于后续流程处理和存证使用分开调用。避免网络传输带来的影响。
- 如果合作方是完成人脸核身流程后马上去拉取视频/照片，建议在查询接口加上一个查询机制：判断图片是否存在，轮询3次，每次2s。
- 照片和视频均为 base64 位编码，其中照片解码后格式一般为 JPG 和 PNG。视频格式解码后一般为 MP4。
- 服务端验证结果接口返回66660011无此查询结果的可能原因：

  1.1 66660017 验证次数超限后被风控，风控后的订单为无效，查询结果为无此查询结果。

  1.2 用户中途退出刷脸，没有完成人脸验证的流程，没有比对，查询结果为无此查询结果。

  1.3 66660018 操作超时，请退出重试 无此 ID 的用户身份信息，H5faceid 5分钟有效期，失效后无法进入刷脸流程，查询结果为无此查询结果。

  1.4 66660016 视频格式或大小不合法 文件或视频不合法，无法进行比对，查询结果为无此查询结果。

  1.5 400604 上传的视频非实时录制,被时间戳校验拦截，查询结果为无此查询结果。

  1.6 查询超过3天的订单返回无此查询结果。
- 响应参数请勿做强校验，后续可能会有扩展。

RiskInfo 对象说明

RiskInfo 是用来给合作方传递刷脸风险信息的对象。其中 RiskInfo 对象的各个字段意义如下表所示：

字段名	类型	字段含义	说明
deviceInfoLevel	String	设备风险等级	设备风险级别，如果命中风险则显示
deviceInfoTag	String	设备风险标签	设备风险等级描述
riskInfoLevel	String	身份风险等级	如果命中身份风险则显示风险等级
riskInfoTag	String	身份风险标签	身份风险等级描述
LivenessInfoTag	String	AI防护盾标签	大模型命中风险描述

大模型命中风险描述

序号	Plus 版 Tag 说明
01	用户全程闭眼
02	用户未完成指定动作
03	疑似翻拍攻击
04	疑似合成攻击
05	疑似黑产模版
06	疑似存在水印
07	反光校验未通过
08	出现多张人脸
09	人脸质量过差

10	距离校验不通过
11	疑似对抗样本攻击
12	脸部区域疑似存在攻击痕迹

详情见 Plus 版人脸核身错误码风险标签说明 。

核身结果-多张照片返回

最近更新时间：2024-12-20 15:23:03

人脸认证多张照片查询接口：获取人脸认证结果多张照片的接口。

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 获取 SIGN ticket 。
- 合作方为人脸验证服务生成签名，需要具有以下参数：

参数	说明	来源
appId	业务流程唯一标识，即 WBappid	请参见 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，字母/数字组成的字符串，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，不要带有特殊字符
version	默认值：1.0.0	-
api ticket	合作伙伴服务端缓存的 ticket，注意是 SIGN 类型	获取方式请参见 获取 SIGN ticket
nonce	32位随机字符串，字母和数字	合作方自行生成，不要带有特殊字符

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 appId、orderNo、version 连同 api ticket、nonce 共5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 。

人脸认证多张照片查询接口

请求

请求URL： `https://kyc1.qcloud.com/api/v2/base/queryphotoinfo?orderNo=xxx`

> ⚠ **注意：**
> 为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。

请求方法： POST
HTTP 请求 header：

参数名	是否必选	类型	说明
Content-Type	是	String	application/json

请求参数：

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考获取 WBappid 指引在人脸核身控制台内申请	String	8	是
version	版本号，默认值：1.0.0	String	20	是
nonce	随机数	String	32	是
orderNo	订单号，字母/数字组成的字符串，合作方订单的唯一标识	String	32	是

sign	签名值，使用本页第一步生成的签名	String	40	是

响应

返回参数说明：

参数名	类型	说明
code	int	0：成功 非0：失败
msg	String	请求结果描述
bizSeqNo	String	请求业务流水号
orderNo	String	订单编号
occurredTime	String	刷脸时间（yyyyMMddHHmmss）
photoList	List	Base64 图像列表（返回1-3张照片），若照片不存在，则返回 null

返回示例：

```
{
"code": 0,
"msg": "请求成功",
"bizSeqNo": "业务流水号",
"result": {
"orderNo": "AAAAAA001",
"occurredTime": "20180907142653",
"photoList": ["第一个base64photo字符串", "第二个base64photo字符串", "第三个base64photo字符串"]
}
}
```

登录鉴权
获取 Access Token

最近更新时间：2025-01-07 10:21:42

注意事项

- 所有场景默认采用 UTF-8 编码。
- Access Token 必须缓存在磁盘，并定时刷新，且不能并发刷新，建议每20分钟请求新的 Access Token，获取之后立即使用最新的 Access Token。旧的只有一分钟的并存期。
- 如果未按照上述做定时刷新，可能导致鉴权不通过，影响人脸服务正常调用。
- 每次用户登录时必须重新获取NONCE ticket。

请求

- **请求 URL：** `https://kyc1.qcloud.com/api/oauth2/access_token`
- **请求方法：**GET
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
secret	WBappid 对应的密钥，申请 WBappid 时得到，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	64	是
grant_type	授权类型，默认值为：client_credential（必须小写）	String	20	是
version	版本号，默认值为：1.0.0	String	20	是

- **请求示例**

```
https://kyc1.qcloud.com/api/oauth2/access_token?
appId=xxx&secret=xxx&grant_type=client_credential&version=1.0.0
```

响应

响应示例：

```
{
"code":"0",
"msg":" 请求成功 ",
"transactionTime":"20151022043831",
"access_token":"accessToken_string",
"expire_time":"20151022043831",
"expire_in":"7200"
}
```

注意：

1. code 不为0则表示获取失败，可以根据 code 和 msg 字段进行定位和调试，code 详情请参见 Plus 版人脸核身服务错误码 。
2. expire_in 为 access_token 的最大生存时间，单位：秒，合作伙伴在判定有效期时以此为准。
3. expire_time 为 access_token 失效的绝对时间，由于各服务器时间差异，不能使用作为有效期的判定依据，只展示使用。
4. 修改 secret 之后，该 appId 生成的 access_token 和 ticket 都失效。

获取 SIGN ticket

最近更新时间：2025-01-07 10:21:42

- **前置条件**：请合作方确保 Access Token 已经正常获取，获取方式见 Access Token 获取 。
- **用途**：SIGN ticket是合作方后台服务端业务请求生成签名鉴权参数之一，用于后台查询验证结果、调用其他业务服务等。
- API ticket 的 SIGN 类型，必须缓存在磁盘，并定时刷新，刷新的机制如下：
- **由于 API ticket 的生命周期依赖于 Access Token 为了简单方便，建议 API ticket 的刷新机制与 Access Token 定时机制原理一致，建议按照每20分钟和 Access Token 绑定定时刷新，且不能并发刷新。**
- **获取新的之后立即使用最新的，旧的有一分钟的并存期。**
- **如果未按照上述做定时刷新，可能导致鉴权不通过，影响人脸服务正常调用。**

请求

- **请求 URL**：`https://kyc1.qcloud.com/api/oauth2/api_ticket`
- **请求方法**：GET
- **请求参数**：

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
access_token	请根据 获取 Access Token 指引进行获取	String	64	是
type	ticket 类型，默认值：SIGN（必须大写）	String	20	是
version	版本号，默认值：1.0.0	String	20	是

- **请求示例**：

```
https://kyc1.qcloud.com/api/oauth2/api_ticket?appId=xxx&access_token=xxx&type=SIGN&version=1.0.0
```

响应

响应示例：

```
{
 "code":"0",
 "msg":" 请求成功 ",
 "transactionTime":"20151022044027",
 "tickets":[
{
 "value":"ticket_string",
 "expire_in":"3600",
 "expire_time":"20151022044027"}
]
}
```

注意：

1. code 不为 0 则表示获取失败，可以根据 code 和 msg 字段进行定位和调试。code 详情请参见 Plus 版人脸核身服务错误码 。
2. expire_in 为 SIGN ticket 的最大生存时间，单位：秒，合作伙伴在判定有效期时以此为准。
3. expire_time 为 SIGN ticket 失效的绝对时间，由于各服务器时间差异，不能使用作为有效期的判定依据，只展示使用。
4. access_token 失效时，该 access_token 生成的 ticket 都失效。
5. tickets 只有一个。

获取 NONCE ticket

最近更新时间：2025-01-07 10:21:42

注意事项

- **前置条件**：确保 Access Token 已经正常获取，获取方式见 Access Token 获取。
- **用途**：NONCE ticket 是合作方 前端包含 App 和 H5 等 生成签名鉴权参数之一，启动 H5 或 SDK 人脸核身。
- API ticket 的 NONCE 类型，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket。

请求

- **请求 URL**：`https://kyc1.qcloud.com/api/oauth2/api_ticket`
- **请求方法**：GET
- **请求参数**：

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
access_token	请根据 Access Token 获取 指引进行获取	String	64	是
type	ticket 类型，默认值：NONCE（必须大写）	String	20	是
version	版本号	String	20	是
user_id	当前使用用户的唯一标识，需合作伙伴自行定义注意：合作伙伴必须保证 user_id 的全局唯一性，不要带有特殊字符	String	32	是

- **请求示例**：

```
https://kyc1.qcloud.com/api/oauth2/api_ticket?
appId=xxx&access_token=xxx&type=NONCE&version=1.0.0&user_id=xxx
```

响应

响应示例：

```
{
 "code":"0",
 "msg":" 请求成功 ",
 "transactionTime":"20151022044027",
 "tickets":[
{
 "value":"ticket_string",
 "expire_in":"120",
 "expire_time":"20151022044027"}
]
}
```

注意：

1. code 不为 0 则表示获取失败，可以根据 code 和 msg 字段定位和调试。code 详情请参见 Plus 版人脸核身服务错误码 。
2. expire_in 为 access_token 的最大生存时间，单位：秒，合作伙伴在**判定有效期时以此为准**。
3. expire_time 为 access_token 失效的绝对时间，由于各服务器时间差异，不能使用作为有效期的判定依据，只展示使用。
4. access_token 失效时，该 access_token 生成的 ticket 都失效。

签名算法说明

最近更新时间：2024-12-20 15:23:03

Java 签名算法

以下是生成签名算法，合作伙伴可以直接使用。

```
public static String sign(List<String> values, String ticket) { //values传ticket外的其他参数
if (values == null) {
throw new NullPointerException("values is null");
}
values.removeAll(Collections.singleton(null));// remove null
values.add(ticket); java.util.Collections.sort(values);
StringBuilder sb = new StringBuilder();
for (String s : values) { sb.append(s);
}
return Hashing.sha1().hashString(sb,
Charsets.UTF_8).toString().toUpperCase();
}
```

> **注意：**
> Hashing 使用的是 com.google.common.hash.Hashing，版本是 guava-18.0。

调用方法

获取 token 和 ticket 之后再用 ticket 对数据进行签名，相关代码如下所示：

```
String accessToken = client.getAccessToken();//http 请求
String ticket = client.getTicket(accessToken);//http 请求
String sign = SignUtils.sign(data, ticket);
```

微信小程序
微信小程序接入指引

最近更新时间：2025-03-14 15:33:12

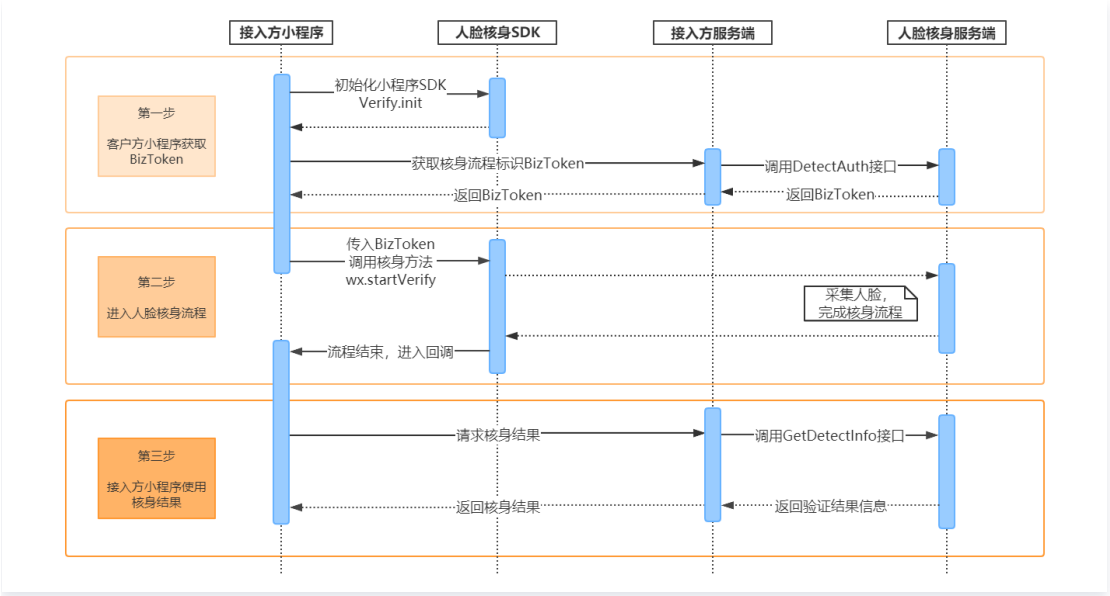
小程序在微信使用场景中越来越广泛，腾讯云人脸核身针对小程序提供嵌入式 SDK，开发人员可以将相应的 SDK 添加到小程序开发工具中，从而直接调用小程序 SDK 中提供的 OCR 识别、活体检测和 1：1人脸比对服务。
基于微信小程序原生体验，人脸核身微信小程序接入渠道提供一闪活体检测模式，要求符合以下行业要求且具备相关资质的客户才能申请，接入前请确认公众号的主体和类目是否符合以下范围，并准备好对应的资质文件，资质审核需要5 – 7个工作日，请预留好上线时间。所需对应的资质文件请参见 微信小程序资质文件列表 。

- 政务：机构或事业单位。
- 金融：银行、保险、信托、基金、证券/期货、持牌消费金融、非金融机构自营小额贷款、汽车金融/金融租赁。
- 医疗：公立医疗机构、互联网医院、三级私立医疗机构。
- 运营商：基础电信运营商、虚拟运营商（含转售移动通信）。
- 教育：学历教育（学校）、公立学校、私立学校。
- 出行与交通：网约车（快车/出租车/专车/其他网约车）、航空、公交/地铁、水运、骑车、火车/高铁/动车、长途汽车、租车、高速服务。
- 生活服务：生活缴费。
- 旅游：酒店。
- 商业服务：公证。
- 社交：直播。
- 物流：收件/派件、货物运输。

说明：
本模式需要先扫码授权：打开二维码 ，小程序管理员扫码后，单击**自定义权限**，只勾选人脸核身权限，将该权限授权给慧眼第三方平台。

授权指引

接入流程



1. 登录 人脸核身控制台，单击**自助接入>创建业务流程**，完成**微信 H5/小程序服务**的业务流程创建，获取 RuleID 和小程序 SDK 下载链接。
2. 下载小程序 SDK，并在小程序代码中引入，调用 init 方法进行初始化。
3. 接入方服务端调用实名核身鉴权 DetectAuth 接口，获取到核身流程标识(BizToken)。
4. 客户后端将 BizToken 返回给客户小程序端，然后小程序调用核身方法 startVerify 进入核身流程。
5. 用户完成人脸核身后，会以回调函数形式返回 BizToken ，接入方小程序将 BizToken 传给接入方服务端，接入方服务端即可凭借 BizToken 参数调用获取实名核身结果信息增强版 GetDetectInfoEnhanced 接口去获取本次核身的详细信息，最后将核身结果返回给接入方小程序。

SDK 接入

开发准备

下载 SDK

登录 人脸核身控制台 下载小程序 SDK，并在小程序代码中引入，调用 init 方法进行初始化。

安装 SDK

将小程序 SDK 文件夹放在小程序根目录下，使用 require 函数引入。

```
const Verify = require('/verify_mpsdk/main');
```

调试 SDK

请在微信开发者工具中使用手机“预览”模式进行调试，请勿使用“真机调试”。

卸载 SDK

卸载时删除 `verify_mpsdk` 文件夹，移除相应 require 代码即可。

快速入门

1. 将 verify_mpsdk 文件夹放到小程序项目根目录。
2. 初始化慧眼实名核身 SDK。
   在 App.js 的 onLaunch() 中加入相应代码，在 App.json 文件里添加活体验证页面 `verify_mpsdk/index/index` 。

```
App({
 onLaunch: function () {
     // 初始化慧眼实名核身组件
     const Verify = require('/verify_mpsdk/main');
     Verify.init();
 }
})
// app.json
{
 "pages":[
     "verify_mpsdk/index/index"
 ]
}
```

3. 调用 SDK 功能函数 wx.startVerify()。
   在需要实名认证的地方调用 wx.startVerify() 进入实名认证页面，认证完成会触发对应的回调函数。

```
  // 单击某个按钮时，触发该函数
gotoVerify: function () {
 let BizToken = getBizToken();// 该函数为客户自定义函数，去客户后端调用 DetectAuth 接口获取 BizToken
 // 调用实名核身功能
 wx.startVerify({
     data: {
         token: BizToken // BizToken
     },
     success: (res) => { // 验证成功后触发
         // res 包含验证成功的token，这里需要加500ms延时，防止iOS下不执行后面的逻辑
         setTimeout(() => {
             // 验证成功后，拿到token后的逻辑处理，具体以客户自身逻辑为准
         }, 500);
     },
     fail: (err) => {  // 验证失败时触发
         // err 包含错误码，错误信息，弹窗提示错误
         setTimeout(() => {
             wx.showModal({
                 title: "提示",
                 content: err.ErrorMsg,
                 showCancel: false
             })
         }, 500);
     }
 });
}
```

4. 添加域名服务器白名单。
   您需要在小程序上线前进入：微信公众号管理平台 > 管理 > 开发管理 > 开发设置 > 服务器域名将以下域名添加至白名单，小程序前端接口请求有域名白名单限制，未添加白名单的域名只能在调试模式下运行。

```
// request 合法域名、uploadFile 合法域名、downloadFile 合法域名这三种都要添加
faceid.qq.com、faceid.qcloud.com
// socket合法域名 （v1.0.20及以上版本需要添加以下socket域名）

// v1.0.17及以上版本身份校验环节如需NFC方式读取证件，需要添加以下socket域名
wss://idcloudread.eidlink.com
```

基本 API 描述

- `Verify.init(options)`：初始化插件。
  - `options`：Object required 初始化的参数。
- `wx.startVerify(options)`：进入实名认证页面。
  - `options`：Object required 初始化的参数。
  - `options.data.token`：String required 客户后端调用 DetectAuth 接口获取的 BizToken。
  - `options.success`：Function(res) required 验证成功的回调。res 包含验证成功的 token。
  - `options.fail`：Function(err) required 验证失败的回调。err 包含错误码、错误信息。

操作步骤

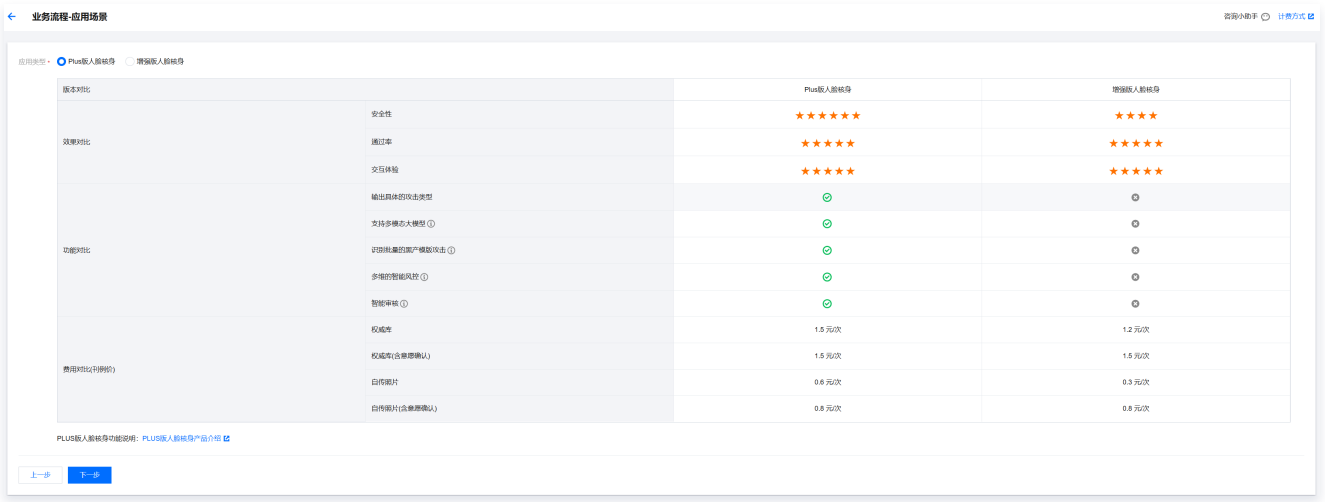
步骤1：创建 RuleId

RuleId 用于调用配置的业务流程，创建步骤如下：

1. 登录腾讯云人脸核身控制台，在 自助接入 页面，单击**创建业务流程**。

2. 选择应用场景：**微信小程序**，并根据需求填写相关信息，填写完成后单击**下一步**。微信小程序需要上传对应的资质文件，我们会在3 – 5个工作日完成审核。

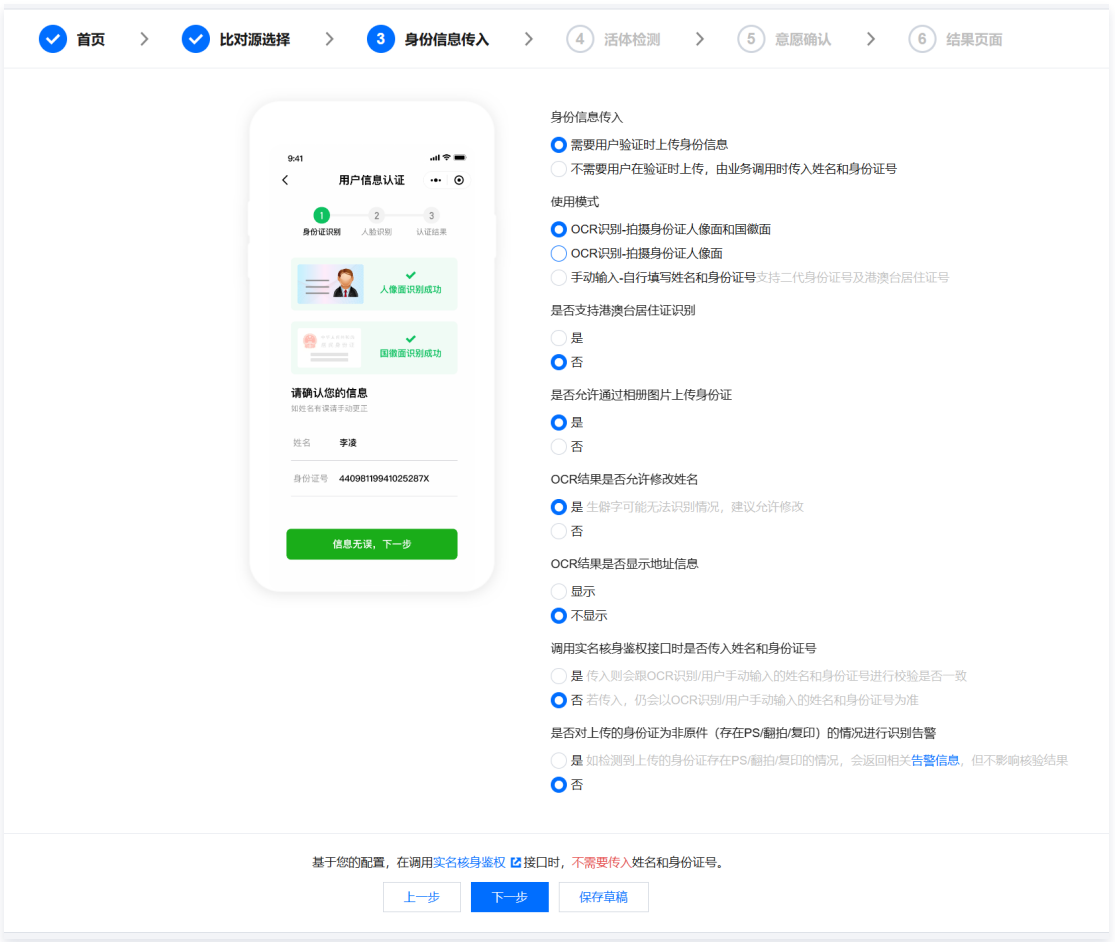
3. 选择应用类型：选择 Plus 版人脸核身，然后单击**下一步**。



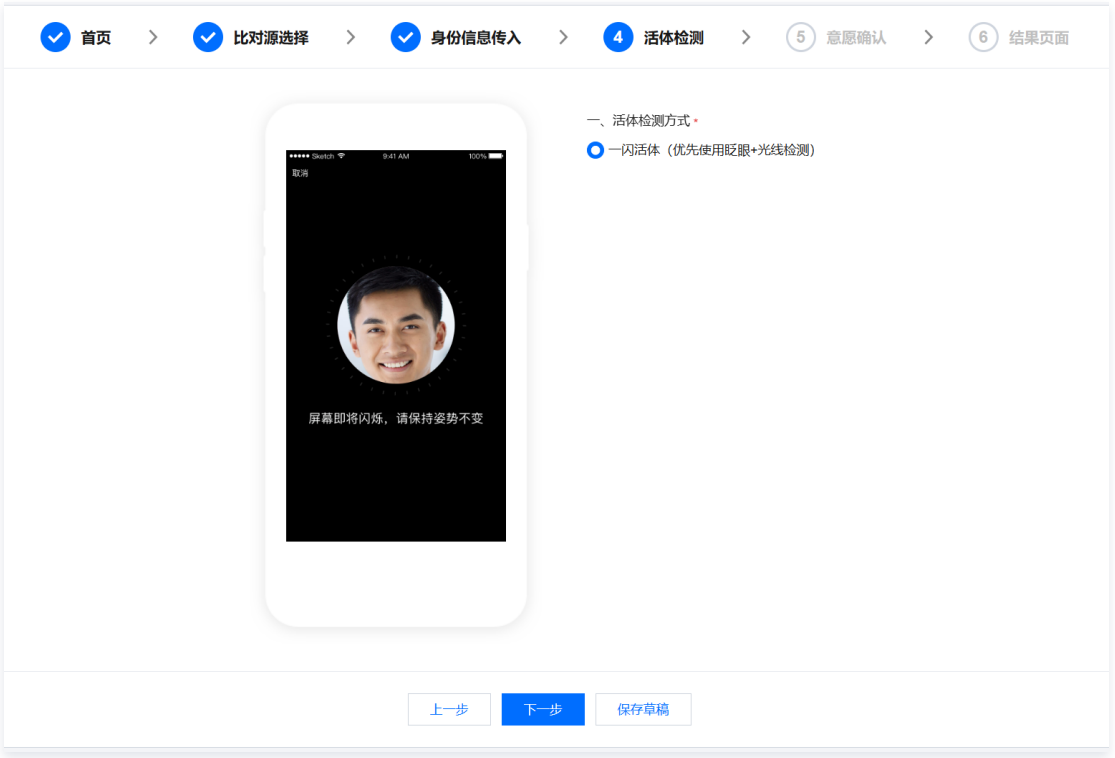
4. 进入接入配置，根据您业务的实际场景填写页面标题、业务名称、业务描述信息、业务联系方式后单击**下一步**。

5. 选择人脸比对库源。人脸核身支持两种方式：**跟权威库比对**和**跟上传照片比对**。
   - 其中**跟权威库比对**的收费价格为：Plus版人脸核身（权威库）– 微信的价格。
   - **跟上传照片比对**的收费价格为：Plus版人脸核身（自传照片）– 微信的价格。OCR 不再单独收费。

6. 配置身份证 OCR 功能，如果不需要则勾选“**不需要用户在验证时上传**”，然后单击**下一步**。

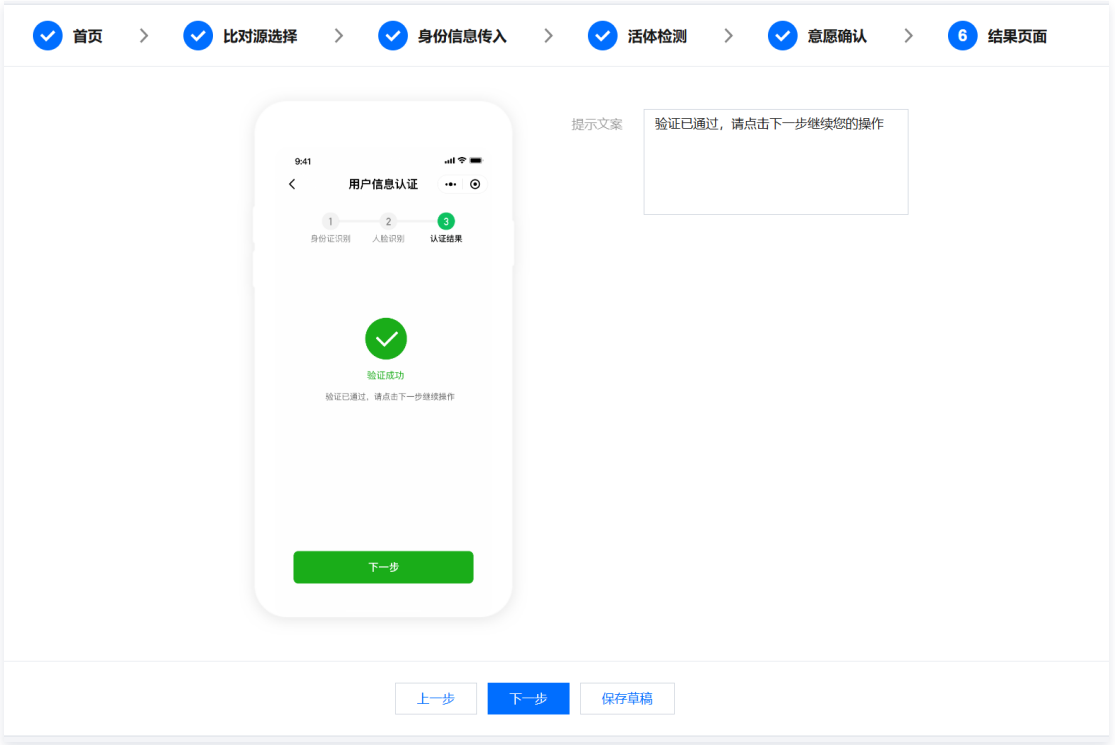


7. 配置活体检测方式，勾选后单击**下一步**。



8. 配置意愿页面，根据您的业务情况勾选：是否增加用户意愿确认流程，然后单击**下一步**。请注意，Plus 版人脸核身默认是不增加意愿确认流程的，如配置了增加意愿确认，则将按**意愿核身**价格计费，详情请参见 价格说明 。

9. 配置结果页的文案描述，然后单击**下一步**。



10. 业务信息填写完成后，确认您的配置信息和计费信息，然后单击**确认并提交审核**。

11. 流程审核通过后，系统会自动创建流程并分配业务 ID（RuleId）。

步骤2：获取 BizToken

完成 RuleId 创建后，需获取 BizToken，用于调用您配置的人脸核身验证的流程。
调用 实名核身鉴权接口，传入 步骤1 生成的 RuleId，和认证结束后重定向的回调链接地址 RedirectUrl，得到核验流程唯一密钥（BizToken）和用于发起核身流程的 URL。

- 在线调试

> **注意：**
> BizToken 是仅一次核身流程的标识，有效时间为7,200秒；用户完成核身后，开发者可用该标识获取验证结果信息，结果信息仅保留3天。
> 如果输入参数 RedirectUrl 为空，则用户完成核验后默认跳转腾讯云人脸核身产品介绍页。

步骤3： 回调小程序SDK

请参考该文档的 快速入门 章节， 回调到小程序 SDK 后，用户进入核身流程，完成身份证拍摄识别、活体检测等操作完成身份核验。

步骤4： 查询核验结果信息

用户完成人脸核身后，页面会跳转到 wx.startVerify的回调地址（该回调地址是业务方自定义的绝对路径地址）之后，您在服务端即可凭借 BizToken 参数调用 获取实名核身结果信息 接口去获取本次核身的详细信息。获取到用户验证过程数据，包括文本信息、识别分数、照片和视频。也可以通过访问 腾讯云人脸核身控制台 查看服务调用情况。

> **说明：**
> wx.startVerify 的回调地址请参考：该文档 快速入门 章节 的第3小节示例中对 success 和 fail 回调的处理方法。

> **注意：**
> - 为了保证用户的隐私，结果信息人脸核身侧仅保留3天，请您及时拉取。

- 在开发、产品使用、费用、合同等问题有任何疑问，欢迎您 联系我们 进行询问。

微信小程序 uni-app 接入流程

最近更新时间：2025-04-27 15:05:01

接入准备

授权指引及接入准备参见 微信小程序接入指引 。
小程序前端接口请求有域名白名单限制，如果不添加只能再调试模式下运行，上线前需要将如下两个域名在小程序后台添加服务器域名。

```
https://faceid.qq.com;
https://faceid.qcloud.com;
```

uni-app接入

步骤一：注册并创建 uni-app 开发环境

uni-app 开发接入具体参照 uni官网 。

步骤二：下载 SDK

控制台下载最新版本的 SDK。

步骤三：并配置 verify_mpsdk

本 SDK 仅支持 uni 微信小程序端。

1. **将 verify-mpsdk 文件夹拷贝到项目根目录的 wxcomponents 文件夹下。**
2. **创建一个空页面调用组件。**

   pages.json 注册组件地址，如pages/verify/index 。

```
{
  "pages": [
    ...
    {
      "path": "pages/verify/index",
      "style": {
        "navigationBarTitleText": "uni-app",
        "usingComponents": {
              "verify-mp": "/wxcomponents/verify_mpsdk/indexCom"
          }
      }
    }
  ]
}
```

   在注册的地址生成小程序SDK页，调用verify-mp组件。

> **注意：**
> 该页面为空内容组件调用页，不需要其他逻辑代码。

```
<template>
    <verify-mp></verify-mp>
</template>

<script>
export default {
  data() {
    return {   }
  },
  onLoad() {},
  methods: {},
}
</script>
```

3. **初始化。**

- **方法一：可以在 App.vue 中全局初始化。**

```
export default {
  onLaunch: function() {
    const verify = require('/wxcomponents/verify_mpsdk/main');
    verify.init();
  },
};
```

- 方法二：在需要调用到的页面方法之前初识化即可。

4. 调用 startVerify。

> **注意：**
> startVerify 调用需要在 init 初始化之后。

```
//业务发起调用页面
wx.startVerify({
  data: {
    token: '', // 必要参数，BizToken
    startPath: 'pages/verify/index' // 必要参数，配置了verify核身组件的页面地址
  },
  success: (res) => { // 验证成功后触发
    // res 包含验证成功的token，这里需要加500ms延时，防止iOS下不执行后面的逻辑
    // 验证成功后，拿到token后的逻辑处理，具体以客户自身逻辑为准
  },
  fail: (err) => {  // 验证失败时触发
    // err 包含错误码，错误信息，弹窗提示错误
  }
});
```

基本 API 描述

- Verify.init(options)：初始化插件。
  - options：Object required 初始化的参数。
- wx.startVerify(options)：进入实名认证页面。
  - options：Object required 初始化的参数。
  - options.data.startPath：组件初始化页面，必填。
  - options.data.token：String required 客户后端调用 DetectAuth 接口获取的 BizToken。
  - options.success：Function(res) required 验证成功的回调。res 包含验证成功的 token。
  - options.fail：Function(err) required 验证失败的回调。err 包含错误码、错误信息。

获取实名核身结果信息

用户完成人脸核身后，页面会跳转到 RedirectUrl 上，地址中会带上此次验证流程使用的 BizToken，您在服务端即可凭借 BizToken 参数调用 获取实名核身结果信息 接口去获取本次核身的详细信息。获取到用户验证过程数据，包括文本信息、识别分数、照片和视频。也可以通过访问 腾讯云人脸核身控制台 查看服务调用情况。

卸载 SDK

卸载时删除 verify_mpsdk 文件夹，移除相应代码即可。

示例 demo

示例 demo 下载

微信小程序资质文件列表

最近更新时间：2025-02-20 10:46:52

微信小程序是基于微信原生的体验，客户体验好，但有明确的主体行业限制。接入前请确认小程序的主体和类目是否符合以下范围，并准备好对应的资质文件，资质审核需要5 – 7个工作日，请预留好上线时间。

微信小程序的主体类目

微信小程序支持行业类目及资质文件要求如下：

小程序一级类目	小程序二级类目	小程序三级类目	使用人脸核验接口所需资质
快递与邮政	寄件/收件	/	《快递业务经营许可证》
物流服务	货物运输	/	《道路运输经营许可证》（经营范围需含网络货运）
教育	学历教育（学校）	/	（2选1）： 1. 公立学校：由教育行政部门出具的审批设立证明 或《事业单位法人证书》 2. 私立学校：《民办学校办学许可证》与《民办非企业单位登记证书》
医疗	公立医疗机构	/	《医疗机构执业许可证》与《事业单位法人证书》
	互联网医院	/	仅支持公立医疗机构互联网医院（2选1）： 1. 卫生健康部门的《设置医疗机构批准书》 2.《医疗机构执业许可证》（范围均需含“互联网诊疗”或名称含“互联网医院”等相关内容）
医疗服务	三级私立医疗机构	/	仅支持三级以上私立医疗机构，提供《医疗机构执业许可证》、《营业执照》及《医院等级证书》
政务民生	所有二级类目	/	仅支持政府/事业单位，提供《组织机构代码证》或《统一社会信用代码证》
金融业	银行	/	（2选1）： 1.《金融许可证》 2.《金融机构许可证》
	保险	/	（8选1）： 1.《保险公司法人许可证》 2.《经营保险业务许可证》 3.《保险营销服务许可证》 4.《保险中介许可证》 5.《经营保险经纪业务许可证》 6.《经营保险公估业务许可证》 7.《经营保险资产管理业务许可证》 8.《保险兼业代理业务许可证》
	信托	/	（2选1）： 1.《金融许可证》 2.《金融机构许可证》
	基金	公募基金	（4选1）： 1.《经营证券期货业务许可证》且业务范围必须包含“基金” 2.《基金托管业务许可证》 3.《基金销售业务资格证书》 4.《基金管理资格证书》
	证券/期货	/	《经营证券期货业务许可证》
	消费金融	/	银监会核准开业的审批文件与《金融许可证》与《营业执照》
	非金融机构自营小额贷款	/	仅支持省金融办监管的网络小贷主体，同时提供： 1.《小额贷款公司经营许可证》或《小额贷款机构经营许可证》 2. 申请主体资质承诺函

	汽车金融	/	仅支持汽车金融主体，同时提供： 1.《营业执照》（公司名称包含“汽车金融”；营业范围包含“汽车金融”业务） 2.《金融许可证》或银保监会及其派出机构颁发的开业核准批复文件
交通服务	打车(网约车)	快车/出租车/专车/其他网约车	自营性网约车：提供《网络预约出租汽车经营许可证》 网约车平台：提供与网约车公司的合作协议以及合作网约车公司的《网络预约出租汽车经营许可证》
	航空	/	1. 航司：提供《公共航空运输企业经营许可证》 2. 机场：提供《民用机场使用许可证》或《运输机场使用许可证》
	地铁	/	提供地铁公司《营业执照》
	水运	/	1. 船企：提供《水路运输许可证》 2. 港口：提供《港口经营许可证》
	骑车	/	仅支持共享单车，提供共享单车公司《营业执照》
	火车/高铁/动车	/	仅支持铁路局/公司官方，提供铁路局/公司《营业执照》
	公交	公交公司	提供公交公司《营业执照》
	长途汽车	/	（2选1）： 1.《道路运输经营许可证》（经营范围需含客运） 2. 官方指定联网售票平台（授权或协议或公开可查询文件）
	租车	/	运营公司提供《备案证明》与对应公司《营业执照》，且营业执照中包含汽车租赁业务
	高速服务	/	仅支持 ETC 发行业务，（2选1）： 1. 事业单位主体，需提供《事业单位法人证书》 2. 官方指定的发行单位（一发单位），需提供“官方授权或协议，或公开可查询的文件”
生活服务	生活缴费	/	（供电类）提供《电力业务许可证》与《营业执照》，且《营业执照》经营范围含供电。 （燃气类）提供《燃气经营许可证》与《营业执照》，且《营业执照》且经营范围含供气。 （供水类）提供《卫生许可证》与《营业执照》。
IT 科技	基础电信运营商	/	（2选1）： 1. 基础电信运营商：提供《基础电信业务经营许可证》 2. 运营商分/子公司：提供营业执照（含相关业务范围）
	转售移动通信	/	仅支持虚拟运营商，提供《增值电信业务许可证》（业务种类需含通过转售方式提供移动通信业务）
旅游服务	住宿服务	/	仅支持酒店，提供《酒店业特种行业经营许可证》
商业服务	公证	/	仅支持公证处，提供《公证处执业许可证》
社交	直播	/	（2选1）： 1.《信息网络传播视听节目许可证》 2.《网络文化经营许可证》（经营范围含网络表演）

人脸核身对账指引

最近更新时间：2025-01-15 17:14:52

本文将为您描述使用腾讯云慧眼人脸核身服务中的微信渠道产品时（如使用人脸核身微信小程序、微信原生 H5、微信普通 H5 的人脸核身 SaaS 服务），如何与慧眼人脸核身进行对账操作。

对账工作主要分为2部分：

1. 收集使用慧眼业务时的明细信息。
2. 根据明细信息计算出账单。

下文将为您提供两种方式进行收集明细和计算账单。

方式1

在业务流程中保存明细

用户完成核验后，接入方调用 DetectAuth 接口生成的 `BizToken` 是一次核身流程的流水号。在业务完成后，接入方需要调用 GetDetectInfoEnhanced 接口来获取 `DetectInfo`，`DetectInfo` 就是描述本次核身的详细信息，包括核身结果，收费详情，照片视频信息等。**一个 BizToken 对应且只对应一个 DetectInfo。**

收集完整明细信息的关键点在于：

1. 标记 `BizToken`，确保不遗漏任何一个 `BizToken`。
   在调用 DetectAuth 接口获取 `BizToken` 后，必须将其存储起来，并标记该 `BizToken` 是否已经拉取到 `DetectInfo`。
2. 建立机制，确保每一个 `BizToken` 都能拉取到 `DetectInfo`。
   一般情况下慧眼侧是建议接入方在自己的回调接口中（DetectAuth 接口传入的 `RedirectUrl` 参数）调用 GetDetectInfoEnhanced 接口 去获取 `DetectInfo`。但由于用户侧核身场景环境比较复杂，并不能保证每一次的核身流程都能回调到 `RedirectUrl`。所以，接入方需要建立一个机制，来检查出未拉取 `DetectInfo` 的 `BizToken`，并重新拉取一遍。
3. 确保拉取到完成状态时的 `DetectInfo`。
   在人脸核身业务中，每个 `BizToken` 都有自己的状态，当 `BizToken` 到达完成状态时，所有的接口均不允许调用，其对应的 `DetectInfo` 也不能再更新。也就是说，若 `BizToken` 未达到完成状态的情况下，用户有可能使用此 `BizToken` 进行重试。未达到完成状态的 `BizToken` 都须重新拉取 `DetectInfo` 来确保拉取到的 `DetectInfo` 是完成状态时的。
   满足以下条件中的任意一个，即可认为 `BizToken` 处在完成状态：
   - GetDetectInfoEnhanced 接口返回的 `Response.Text.ErrCode` 为0。
   - 距离 Token 生成时间已超过2小时。

> ⚠ **注意：**
> 慧眼人脸核身支持配置最大可重试次数，该次数仅影响活体一比一的次数，并不影响 OCR、短信等接口的调用。

计算账单

`DetectInfo` 中与计费相关的信息全部都位于 `Response.Text.LivenessDetail` 数组中，该数组存放的是该 `BizToken` 进行的活体一比一的次数与每次活体一比一的详情。**遍历该数组，统计 `IsNeedCharge` 为 `true` 的个数即是此 BizToken 的收费次数。**

> ⚠ **注意：**
> `Response.Text.ErrCode` 是用于判断本次核验是否通过的字段，不用于判断计费与否。

为方便对账，建议将 `LivenessDetail` 数组中的每一项均保存下来。

```
let totalCount = 0; // 活体一比一的总记录数
let chargeCount = 0; // 活体一比一的收费记录数
let GetDetectInfoResponse = GetDetectInfoEnhanced(xxx, xxx, ...); // 调用GetDetectInfoEnhanced接口获取DetectInfo
for (let detail of GetDetectInfoResponse.Text.LivenessDetail){
    totalCount = totalCount + 1;
    if (detail.IsNeedCharge == true) {
        chargeCount = chargeCount + 1; // 当IsNeedCharge为true时，收费记录数加1
    }
    saveChargeDetailToDB(detail); // 保存至数据库，以便日后查看
}
```

方式2

使用 GetWeChatBillDetails 接口每日拉取

为了方便接入方更快捷的获取明细信息，慧眼人脸核身提供了一个接口（GetWeChatBillDetails）用于拉取每日的明细信息。该接口可以拉取子账号 + RuleId 维度的某一天的所有 token 数据。
具体使用方式如下：
1. 建立定时任务，每日早上6点后开始执行。**早于6点调用 GetWeChatBillDetails 接口会报参数异常。**
2. 调用 GetWeChatBillDetails 接口，拉取前一天所有的验证明细信息。

```
    let WeChatBillDetails = []; // 当日全量的账单详情
    let cursor = 0; // 游标位，拉取第一页时传0；剩下页面根据上一次请求回包的结果传入。
    let hasNextPage = true; // 是否还有下一页的标志位
    while (hasNextPage) { // 如果还有下一页，则继续拉取
        const result = GetWeChatBillDetails(Date /* 传入某一天日期*/, RuleId /* 传入RuleId*/, cursor); // 调用
GetWeChatBillDetails接口，并获取返回。
        WeChatBillDetails = WeChatBillDetails.concat(result.WeChatBillDetails); // 将拉取到的数据合并到全量的数组中
        cursor = result.NextCursor; // 将回包中返回的下一页的游标赋值给游标位，准备下一次拉取
        hasNextPage = result.HasNextPage; // 将是否还有下一页的标志位重新赋值，用于决定是否进行下一次拉取
    }
    // 此时WeChatBillDetails中就是某一天全量的账单信息
```

计算账单

统计 WeChatBillDetail 数组中的 ChargeCount 总数，即为当日计费总数。同时记录 ChargeDetail 内容，可用于校验：ChargeCount 值 = ChargeDetail 中 IsNeedCharge 为 true 的条目总数 = ErrorCode 为收费错误码的条目总数。

```
 let totalCount = 0; // 活体一比一的总记录数
let chargeCount = 0; // 活体一比一的收费记录数
for (let WeChatBillDetail of WeChatBillDetails) { // 遍历WeChatBillDetails数组
    let ChargeDetails = WeChatBillDetail.ChargeDetails; // 取计费详情属性
    totalCount += ChargeDetails.length; // 总记录数为计费详情的总条目数
    chargeCount += WeChatBillDetail.ChargeCount; // 收费记录数为ChargeCount的总和
    for (let ChargeDetail of ChargeDetails) {
        saveChargeDetailToDB(ChargeDetail); // 遍历ChargeDetails数组，保存至数据库，以便日后查看
    }
}
```

常见问题

1. **什么时候进行计费上报?**
   计费上报时机是以**一比一请求时间为准**，即 `Response.Text.LivenessDetail[].ReqTime` 、GetWeChatBillDetails 接口中的 `ChargeDetail.ReqTime` 。例如，BizToken 是在7月1日的23:59:59生成的，但是进行比对的时机是在7月2日的0点，那一比一的请求时间也会落在7月2日的0点，同时这次调用也会上报至7月2日的账单中。
2. **LivenessDetail 中有些字段为 null 是什么原因?**
   当本次核身没有进行一比一或者其他原因造成一比一失败时，有可能导致 ReqTime、Seq、Idcard、Name、Sim、IsNeedCharge、Comparestatus、Comparemsg、CompareLibType 字段是个空值，此时本条明细是不会进行计费的。可以理解为，只有 IsNeedCharge 字段为 true 的时候是计费的，该字段为 false、null 都不计费。

附录

GetWeChatBillDetails 接口介绍

输入参数

参数名	是否必填	类型	描述
Date	Y	Date	拉取的日期（YYYY-MM-DD）。最大可追溯到365天前。当天6点后才能拉取前一天的数据。
RuleId	Y	String	业务对应的 ruleid
Cursor	Y	Integer	游标。用于分页，取第一页时传0，取后续页面时，传入本接口响应中的 NextCursor 字段的值。

输出参数

参数名	类型	描述
HasNextPage	Boolean	是否还有下一页。该字段为 true 时，需要将 NextCursor 的值作为入参继续调用本接口。
NextCursor	Integer	下一页的游标。用于分页。
WeChatBillDetails	WeChatBillDetail 数组	数据，明细如下表。

WeChatBillDetail

属性名	类型	说明
BizToken	String	token。
ChargeCount	Integer	本 token 收费次数。
ChargeDetails	ChargeDetail数组	本 token 计费详情，明细如下表。

ChargeDetail

属性名	类型	说明
ReqTime	String	一比一请求时间戳，13位。
Seq	String	一比一请求的唯一标记。
Idcard	String	一比一时使用的、脱敏后的身份证号。
Name	String	一比一时使用的、脱敏后的姓名。
Sim	String	一比一的相似度。0-100，保留2位小数。
IsNeedCharge	Boolean	本次详情是否收费。
ChargeType	String	收费类型，比对、核身、混合部署。
ErrorCode	String	本次活体一比一最终结果。
ErrorMessage	String	本次活体一比一最终结果描述。

H5（微信公众号）
微信原生 H5（微信公众号）

最近更新时间：2025-02-10 10:30:33

本文档以接入微信原生 H5 的操作流程为例，描述通过微信 HTML5 方式接入人脸核身的完整操作流程。

接入说明

1. 主体行业要求
   微信原生 H5 有明确的主体行业要求，在控制台申请 RuleId 时需上传对应的行业资质。所以在接入微信原生 H5 前，请确认公众号的主体和类目是否符合范围，并准备好对应的资质文件，详情可查阅 微信原生H5（微信公众号）资质文件列表 。
2. 审核时间
   微信原生 H5 有主体行业限制，资质审核需要3 - 5个工作日，请预留好上线时间。
3. 活体检测模式
   微信原生 H5 支持一闪活体检测（眨眼 + 光线）、随机双动作检测。
4. 公众号管理员授权
   微信原生 H5 需要公众号管理员授权，打开二维码，将该权限授权给人脸核身第三方平台。
5. 认证页面流程
6. 微信原生 H5 用户进行人脸核身流程：

前提条件

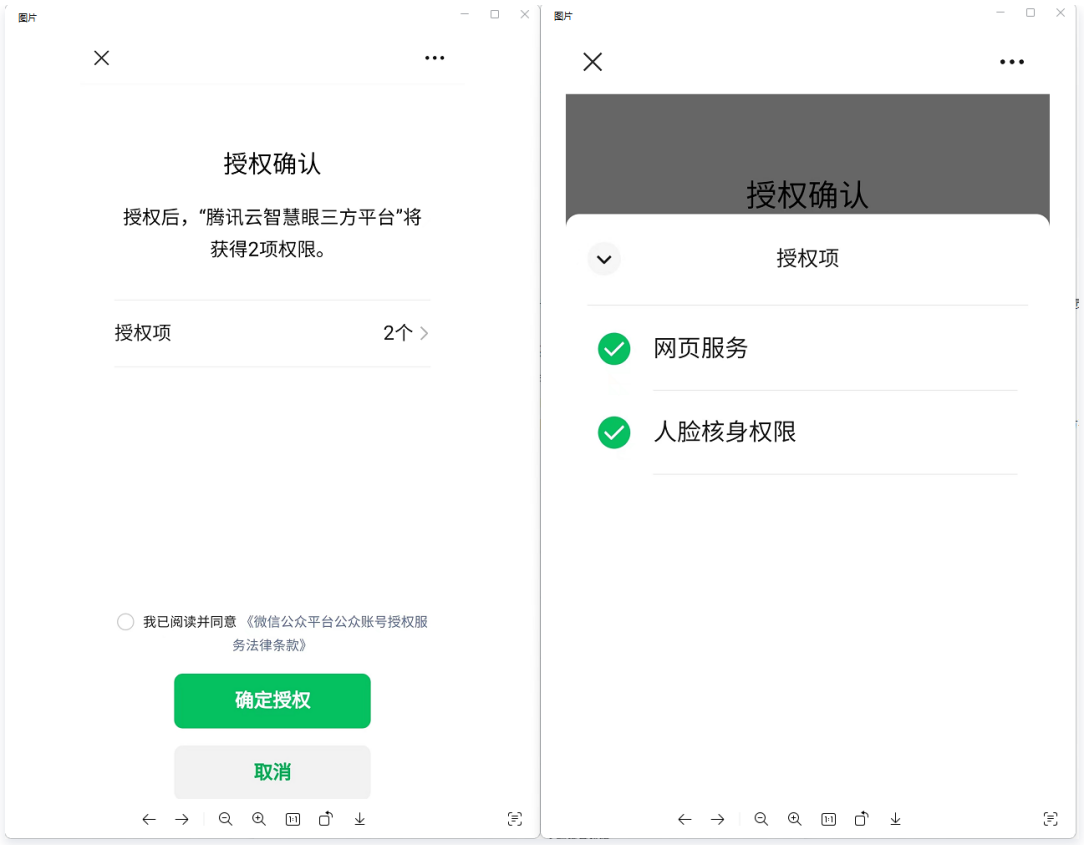
1. 已注册腾讯云账号，并完成实名认证。
2. 已申请通过腾讯云人脸核身。

如果还未完成以上操作，可参考 流程指引 完成操作。

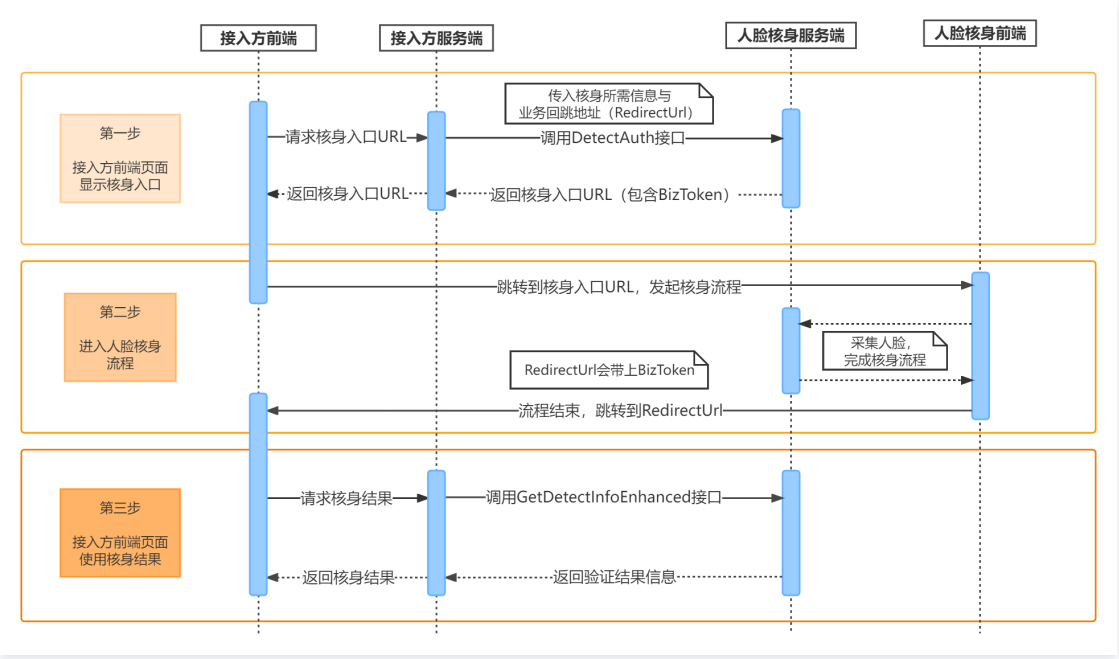
准备事项

结合您的实际业务场景，确认以下事项：

1. **确认接入方式**：选择微信原生 H5 模式，微信原生 H5 的接入方式有行业限制，且资质文件中主体与需要接入公众号主体一致，详细行业类目和资质材料请查阅 微信原生H5（微信公众号）资质文件列表 。
2. **确认人脸比对库源**：是需要人脸核身跟权威库比对还是跟上传照片比对，两个详细价格可参阅 计费概述 。
3. 登录官网控制台 创建 API 密钥（SecretId 和 SecretKey）。
4. **确认微信公众号已权限授权给人脸核身**：微信原生 H5 模式需要公众号管理员授权，步骤如下：
   打开二维码，公众号管理员扫码后，确定授权。



接入时序图



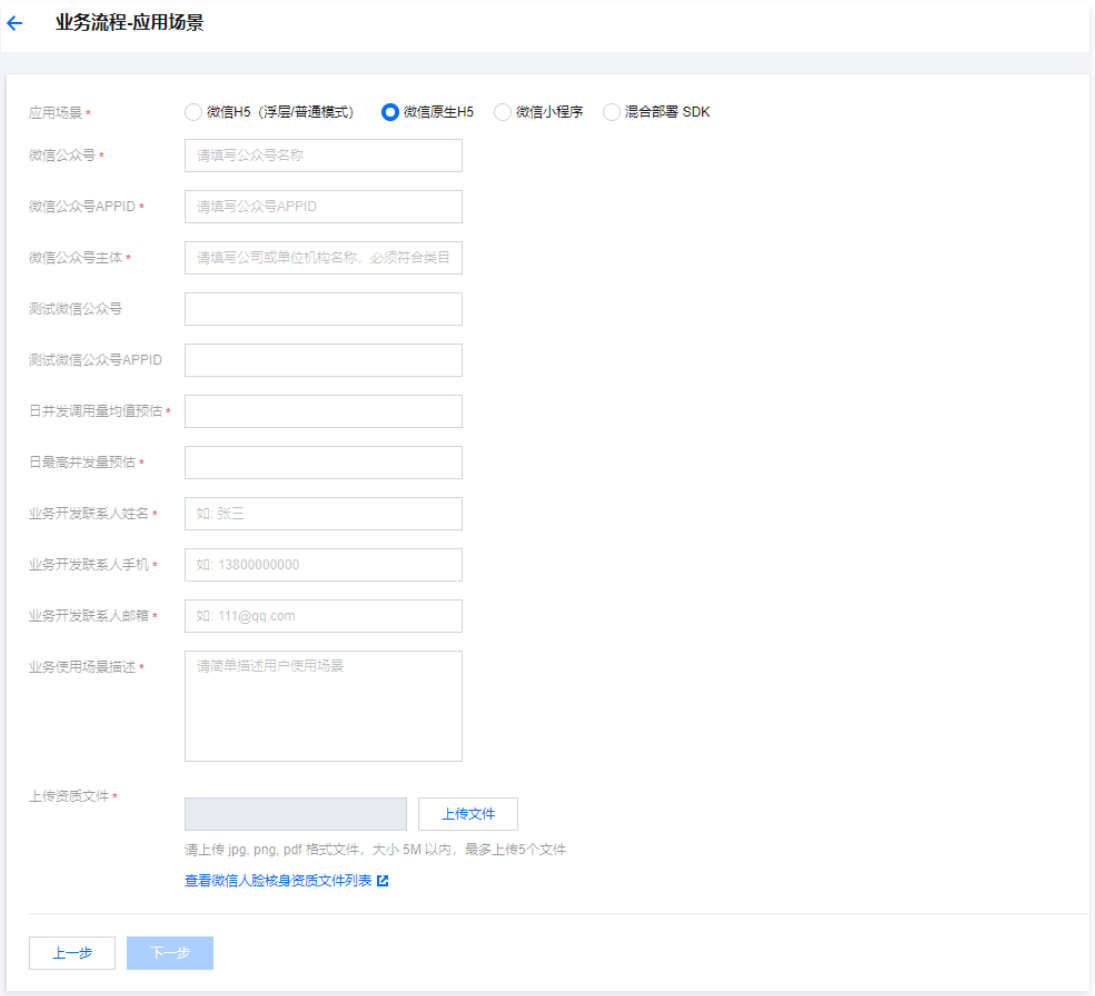
操作步骤

步骤1：创建 RuleId

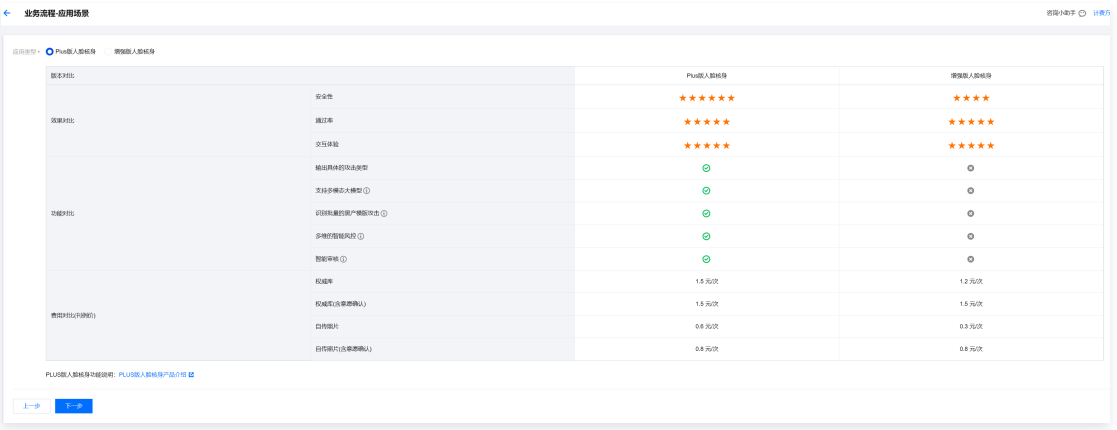
RuleId 用于调用配置的业务流程，创建步骤如下：

1. 登录腾讯云人脸核身控制台，在 自助接入 页面，单击**创建业务流程**。

2. 选择应用场景：**微信原生 H5**，并根据需求填写相关信息，填写完成后单击**下一步**。**微信原生 H5** 需要上传对应的 资质文件，我们会在3 – 5个工作日完成审核。



3. 选择应用类型：选择**Plus版人脸核身**，然后单击**下一步**。

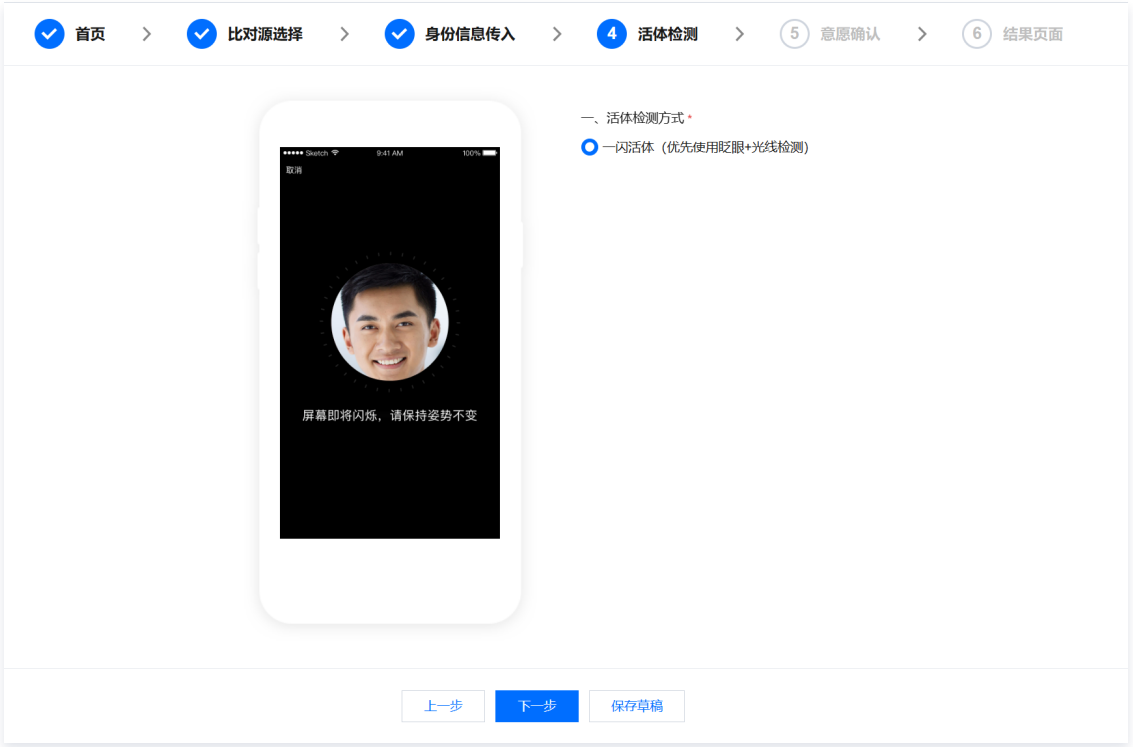


4. 进入接入配置，根据您业务的实际场景填写页面标题、业务名称、业务描述信息、业务联系方式后单击**下一步**。

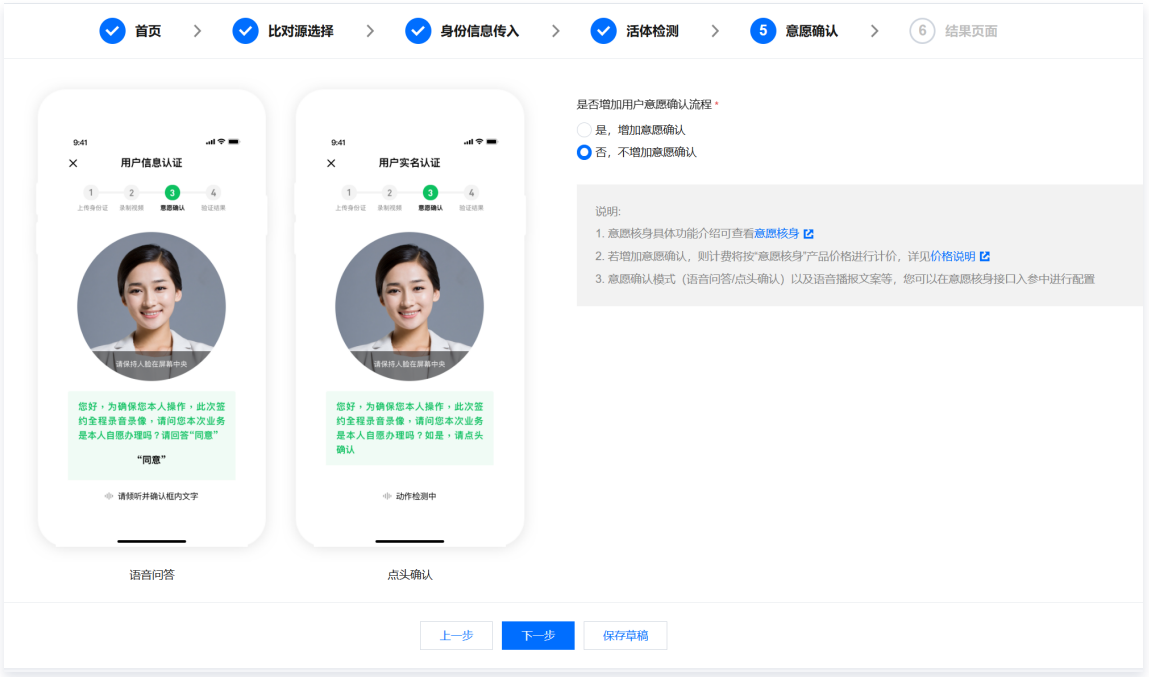
5. 选择人脸比对库源。人脸核身支持两种方式：**跟权威库比对**和**跟上传照片比对**。
   - 其中**跟权威库比对**的收费价格为：Plus 版人脸核身（权威库）- 微信的价格。
   - **跟上传照片比对**的收费价格为：Plus 版人脸核身（自传照片）- 微信的价格。OCR 不再单独收费。

6. 配置身份证 OCR 功能，如果不需要则勾选“**不需要用户在验证时上传**”，然后单击**下一步**。

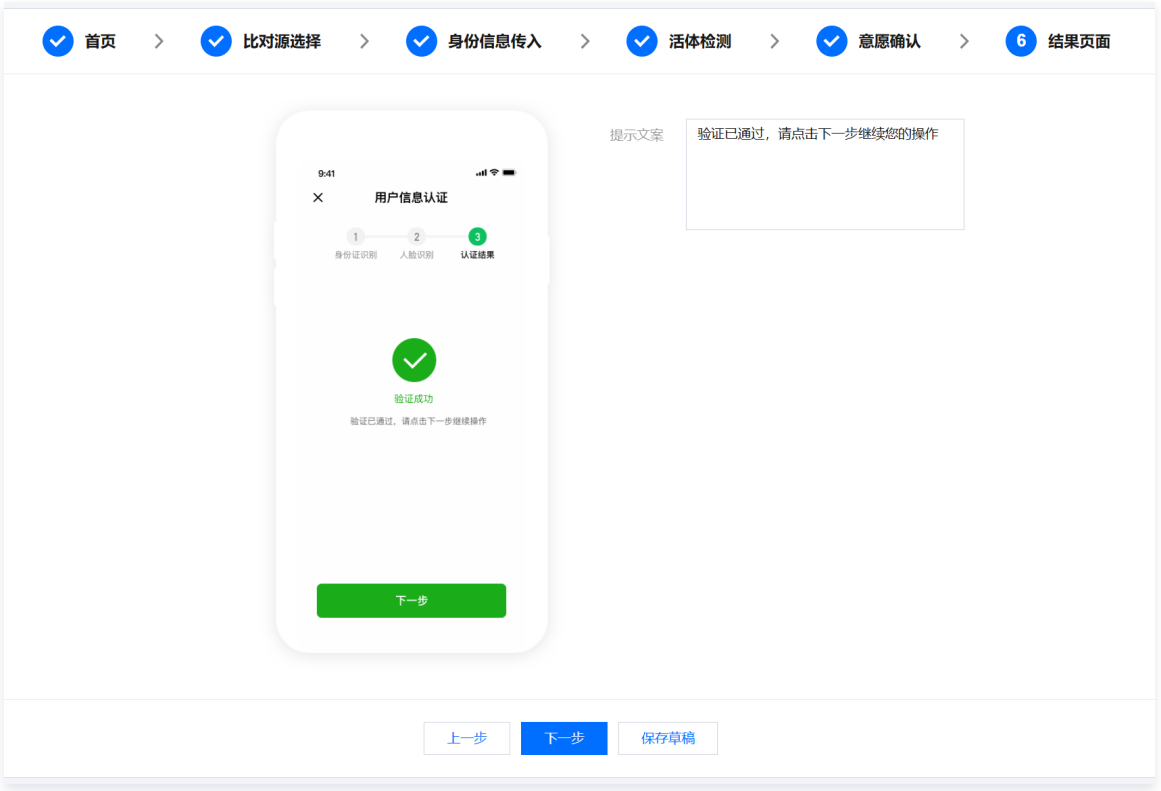
7. 配置活体检测方式，勾选后单击**下一步**。



8. 配置意愿页面，根据您的业务情况勾选：是否增加用户意愿确认流程，然后单击**下一步**。请注意，Plus 版人脸核身默认是不增加意愿确认流程的，如配置了增加意愿确认，则将按**意愿核身**价格计费，详情请参见 价格说明 。



9. 配置结果页的文案描述，然后单击**下一步**。



10. 业务信息填写完成后，确认您的配置信息和计费信息，然后单击**确认并提交审核**。

11. 流程审核通过后，系统会自动创建流程并分配业务 ID（RuleId）。

步骤2：获取 BizToken

完成 RuleId 创建后，需获取 BizToken，用于调用您配置的人脸核身验证的流程。
调用 **实名核身鉴权接口**，传入**步骤1**生成的 RuleId，和认证结束后重定向的回调链接地址 RedirectUrl，得到核验流程唯一密钥（BizToken）和用于发起核身流程的 URL。

- **在线调试**

> **注意：**
> BizToken 是仅一次核身流程的标识，有效时间为7,200秒；用户完成核身后，开发者可用该标识获取验证结果信息，结果信息仅保留3天。
> 如果输入参数 RedirectUrl 为空，则用户完成核验后默认跳转腾讯云人脸核身产品介绍页。

步骤3： 跳转人脸核身 URL 完成核验

您在前端通过地址跳转方式重定向至**步骤2**中获取的核身入口 URL，用户进入核身流程，完成身份证拍摄识别、录制视频等操作完成身份核验。

步骤4： 查询核验结果信息

用户完成人脸核身后，页面会跳转到 RedirectUrl 上，地址中会带上此次验证流程使用的 BizToken，您在服务端即可凭借 BizToken 参数调用 **获取实名核身结果信息** 接口去获取本次核身的详细信息。获取到用户验证过程数据，包括文本信息、识别分数、照片和视频。也可以通过访问 **腾讯云人脸核身控制台** 查看服务调用情况。

> **注意：**
> 为了保证用户的隐私，结果信息人脸核身侧仅保留3天，请您及时拉取。
> 在开发、产品使用、费用、合同等问题有任何疑问，欢迎您 **点此链接** 扫描二维码添加腾讯云助手进行询问。

微信原生 H5（微信公众号）资质

最近更新时间：2024-11-15 17:07:12

微信原生 H5 浮层模式是基于微信原生的体验，客户体验好，但有明确的主体行业限制。
接入前请确认公众号的主体和类目是否符合以下范围，并准备好对应的资质文件，资质审核需要5 - 7个工作日，请预留好上线时间。

微信原生 H5 的主体类目

微信公众号支持行业类目及资质文件要求如下：

一级类目	二级类目	使用人脸核验接口所需资质
教育	学历教育（学校）	公立学校：由教育行政部门出具的审批设立证明或《事业单位法人证书》。
医疗	公立医疗机构	《医疗机构执业许可证》与《事业单位法人证书》。
	互联网医院	仅支持公立医疗机构互联网医院（2选1）： 1. 卫生健康部门的《设置医疗机构批准书》。 2. 《医疗机构执业许可证》（范围均需含“互联网诊疗”或名称含“互联网医院”等相关内容）。
政务民生	所有二级类目	仅支持政府/事业单位，提供《组织机构代码证》或《统一社会信用代码证》。

人脸核身对账指引

最近更新时间：2024-11-15 17:07:12

本文将为您描述使用腾讯云慧眼人脸核身服务中的微信渠道产品时（如使用人脸核身微信小程序、微信原生 H5、微信普通 H5 的人脸核身 SaaS 服务），如何与慧眼人脸核身进行对账操作。

对账工作主要分为2部分：

1. 收集使用慧眼业务时的明细信息。
2. 根据明细信息计算出账单。

下文将为您提供两种方式进行收集明细和计算账单。

方式1

在业务流程中保存明细

用户完成核验后，接入方调用 DetectAuth 接口生成的 `BizToken` 是一次核身流程的流水号。在业务完成后，接入方需要调用 GetDetectInfoEnhanced 接口来获取 `DetectInfo`，`DetectInfo` 就是描述本次核身的详细信息，包括核身结果，收费详情，照片视频信息等。**一个 BizToken 对应且只对应一个 DetectInfo。**

收集完整明细信息的关键点在于：

1. 标记 `BizToken`，确保不遗漏任何一个 `BizToken`。
   在调用 DetectAuth 接口获取 `BizToken` 后，必须将其存储起来，并标记该 `BizToken` 是否已经拉取到 `DetectInfo`。
2. 建立机制，确保每一个 `BizToken` 都能拉取到 `DetectInfo`。
   一般情况下慧眼侧是建议接入方在自己的回调接口中（DetectAuth 接口传入的 `RedirectUrl` 参数）调用 GetDetectInfoEnhanced 接口 去获取 `DetectInfo`。但由于用户侧核身场景环境比较复杂，并不能保证每一次的核身流程都能回调到 `RedirectUrl`。所以，接入方需要建立一个机制，来检查出未拉取 `DetectInfo` 的 `BizToken`，并重新拉取一遍。
3. 确保拉取到完成状态时的 `DetectInfo`。
   在人脸核身业务中，每个 `BizToken` 都有自己的状态，当 `BizToken` 到达完成状态时，所有的接口均不允许调用，其对应的 `DetectInfo` 也不能再更新。也就是说，若 `BizToken` 未达到完成状态的情况下，用户有可能使用此 `BizToken` 进行重试。未达到完成状态的 `BizToken` 都须重新拉取 `DetectInfo` 来确保拉取到的 `DetectInfo` 是完成状态时的。
   满足以下条件中的任意一个，即可认为 `BizToken` 处在完成状态：
   - GetDetectInfoEnhanced 接口返回的 `Response.Text.ErrCode` 为0。
   - 距离 Token 生成时间已超过2小时。

> ⚠ **注意：**
> 慧眼人脸核身支持配置最大可重试次数，该次数仅影响活体一比一的次数，并不影响 OCR、短信等接口的调用。

计算账单

`DetectInfo` 中与计费相关的信息全部都位于 `Response.Text.LivenessDetail` 数组中，该数组存放的是该 `BizToken` 进行的活体一比一的次数与每次活体一比一的详情。**遍历该数组，统计 `IsNeedCharge` 为 `true` 的个数即是此 BizToken 的收费次数。**

> ⚠ **注意：**
> `Response.Text.ErrCode` 是用于判断本次核验是否通过的字段，不用于判断计费与否。

为方便对账，建议将 `LivenessDetail` 数组中的每一项均保存下来。

```
let totalCount = 0; // 活体一比一的总记录数
let chargeCount = 0; // 活体一比一的收费记录数
let GetDetectInfoResponse = GetDetectInfoEnhanced(xxx, xxx, ...); // 调用GetDetectInfoEnhanced接口获取DetectInfo
for (let detail of GetDetectInfoResponse.Text.LivenessDetail){
    totalCount = totalCount + 1;
    if (detail.IsNeedCharge == true) {
        chargeCount = chargeCount + 1; // 当IsNeedCharge为true时，收费记录数加1
    }
    saveChargeDetailToDB(detail); // 保存至数据库，以便日后查看
}
```

方式2

使用 GetWeChatBillDetails 接口每日拉取

为了方便接入方更快捷的获取明细信息，慧眼人脸核身提供了一个接口（GetWeChatBillDetails）用于拉取每日的明细信息。该接口可以拉取子账号 + RuleId 维度的某一天的所有 token 数据。
具体使用方式如下：

1. 建立定时任务，每日早上6点后开始执行。**早于6点调用 GetWeChatBillDetails 接口会报参数异常。**
2. 调用 GetWeChatBillDetails 接口，拉取前一天所有的验证明细信息。

```
    let WeChatBillDetails = []; // 当日全量的账单详情
    let cursor = 0; // 游标位，拉取第一页时传0；剩下页面根据上一次请求回包的结果传入。
    let hasNextPage = true; // 是否还有下一页的标志位
    while (hasNextPage) { // 如果还有下一页，则继续拉取
        const result = GetWeChatBillDetails(Date /* 传入某一天日期*/, RuleId /* 传入RuleId*/, cursor); // 调用
GetWeChatBillDetails接口，并获取返回。
        WeChatBillDetails = WeChatBillDetails.concat(result.WeChatBillDetails); // 将拉取到的数据合并到全量的数组中
        cursor = result.NextCursor; // 将回包中返回的下一页的游标赋值给游标位，准备下一次拉取
        hasNextPage = result.HasNextPage; // 将是否还有下一页的标志位重新赋值，用于决定是否进行下一次拉取
    }
    // 此时WeChatBillDetails中就是某一天全量的账单信息
```

计算账单

统计 WeChatBillDetail 数组中的 ChargeCount 总数，即为当日计费总数。同时记录 ChargeDetail 内容，可用于校验：ChargeCount 值 = ChargeDetail 中 IsNeedCharge 为 true 的条目总数 = ErrorCode 为收费错误码的条目总数。

```
let totalCount = 0; // 活体一比一的总记录数
let chargeCount = 0; // 活体一比一的收费记录数
for (let WeChatBillDetail of WeChatBillDetails) { // 遍历WeChatBillDetails数组
    let ChargeDetails = WeChatBillDetail.ChargeDetails; // 取计费详情属性
    totalCount += ChargeDetails.length; // 总记录数为计费详情的总条目数
    chargeCount += WeChatBillDetail.ChargeCount; // 收费记录数为ChargeCount的总和
    for (let ChargeDetail of ChargeDetails) {
        saveChargeDetailToDB(ChargeDetail); // 遍历ChargeDetails数组，保存至数据库，以便日后查看
    }
}
```

常见问题

1. **什么时候进行计费上报?**
   计费上报时机是以**一比一请求时间**为准，即 Response.Text.LivenessDetail[].ReqTime、GetWeChatBillDetails 接口中的ChargeDetail.ReqTime。例如，BizToken 是在7月1日的23:59:59生成的，但是进行比对的时机是在7月2日的0点，那一比一的请求时间也会落在7月2日的0点，同时这次调用也会上报至7月2日的账单中。
2. **LivenessDetail 中有些字段为 null 是什么原因?**
   当本次核身没有进行一比一或者其他原因造成一比一失败时，有可能导致 ReqTime、Seq、Idcard、Name、Sim、IsNeedCharge、Comparestatus、Comparemsg、CompareLibType 字段是个空值，此时本条明细是不会进行计费的。可以理解为，只有 IsNeedCharge 字段为 true 的时候是计费的，该字段为 false、null 都不计费。

附录

GetWeChatBillDetails 接口介绍

输入参数

参数名	是否必填	类型	描述
Date	Y	Date	拉取的日期（YYYY-MM-DD）。最大可追溯到365天前。当天6点后才能拉取前一天的数据。
RuleId	Y	String	业务对应的 ruleid。
Cursor	Y	Integer	游标。用于分页，取第一页时传0，取后续页面时，传入本接口响应中得 NextCursor 字段的值。

输出参数

参数名	类型	描述
HasNextPage	Boolean	是否还有下一页。该字段为 true 时，需要将 NextCursor 的值作为入参继续调用本接口。
NextCursor	Integer	下一页的游标。用于分页。
WeChatBillDetails	WeChatBillDetail 数组	数据，明细如下表。

WeChatBillDetail

属性名	类型	说明
BizToken	String	token。
ChargeCount	Integer	本 token 收费次数。
ChargeDetails	ChargeDetail数组	本 token 计费详情，明细如下表。

ChargeDetail

属性名	类型	说明
ReqTime	String	一比一请求时间戳，13位。
Seq	String	一比一请求的唯一标记。
Idcard	String	一比一时使用的、脱敏后的身份证号。
Name	String	一比一时使用的、脱敏后的姓名。
Sim	String	一比一的相似度。0-100，保留2位小数。
IsNeedCharge	Boolean	本次详情是否收费。
ChargeType	String	收费类型，比对、核身、混合部署。
ErrorCode	String	本次活体一比一最终结果。
ErrorMessage	String	本次活体一比一最终结果描述。

H5（微信浏览器）
微信浮层 H5（微信浏览器）

最近更新时间：2025-01-15 17:14:52

本文档以接入微信浮层 H5 的操作流程为例，描述通过微信 HTML5 方式接入人脸核身的完整操作流程。

> **说明：**
> 若需同时在微信场景及非微信场景（如企业微信、支付宝、移动 H5 等）使用，可根据此文档完成接入后 联系我们 配置支持。

接入说明

1. 主体行业要求
   无行业类目没有限制。
2. 兼容性要求
   浮层 H5 能够兼容大部分用户的微信浏览器环境，仅有少部分浏览器不支持（低于5%），当检测到浏览器不兼容时会自动切换为普通 H5 的模式，以保证人脸核身的正常进行。

手机平台	应用端	兼容性要求
iOS	微信浏览器	iOS 系统版本 14.3+，微信版本 6.5+
	企微微信浏览器	iOS 系统版本 14.3+，企微版本4.0.8+
Android	微信浏览器	都支持
	企微微信浏览器	都支持

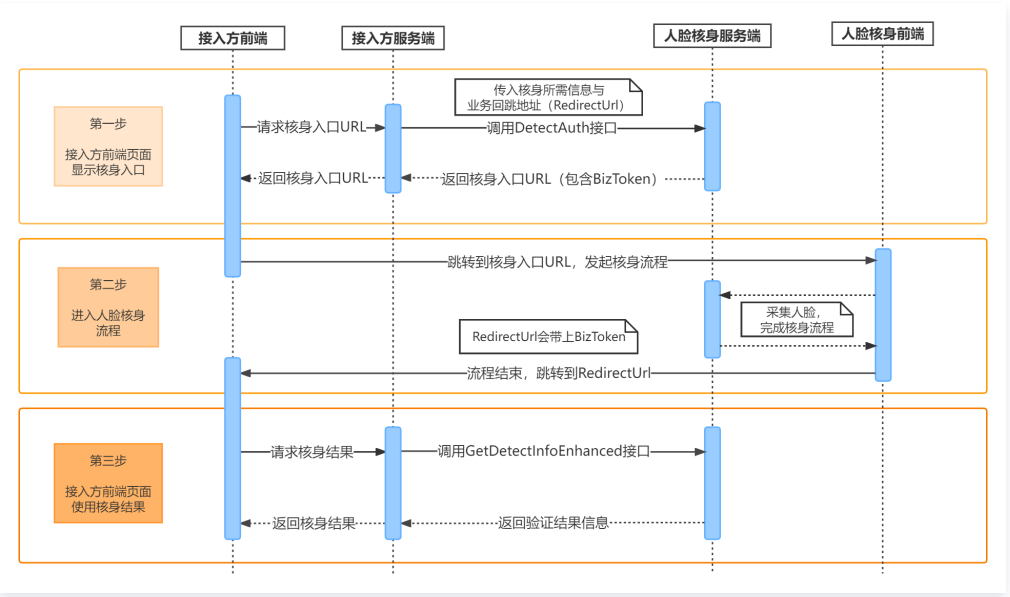
3. 活体检测模式
   微信浮层 H5 支持一闪活体检测（眨眼+光线）、随机双动作检测。
4. 认证页面流程

前提条件

1. 已注册腾讯云账号，并完成实名认证。
2. 已申请通过腾讯云人脸核身。

如果还未完成以上操作，可参考 流程指引 完成操作。

接入时序图

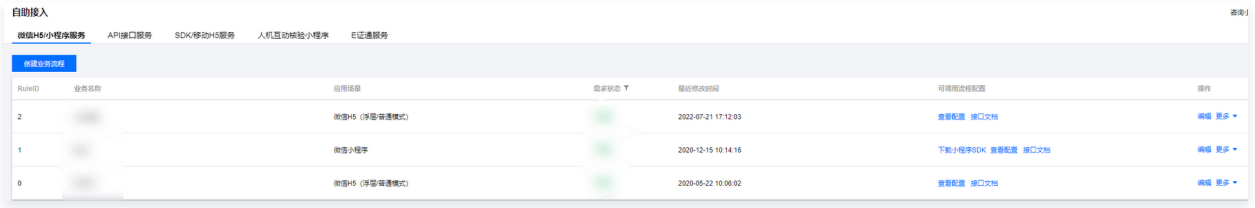


操作步骤

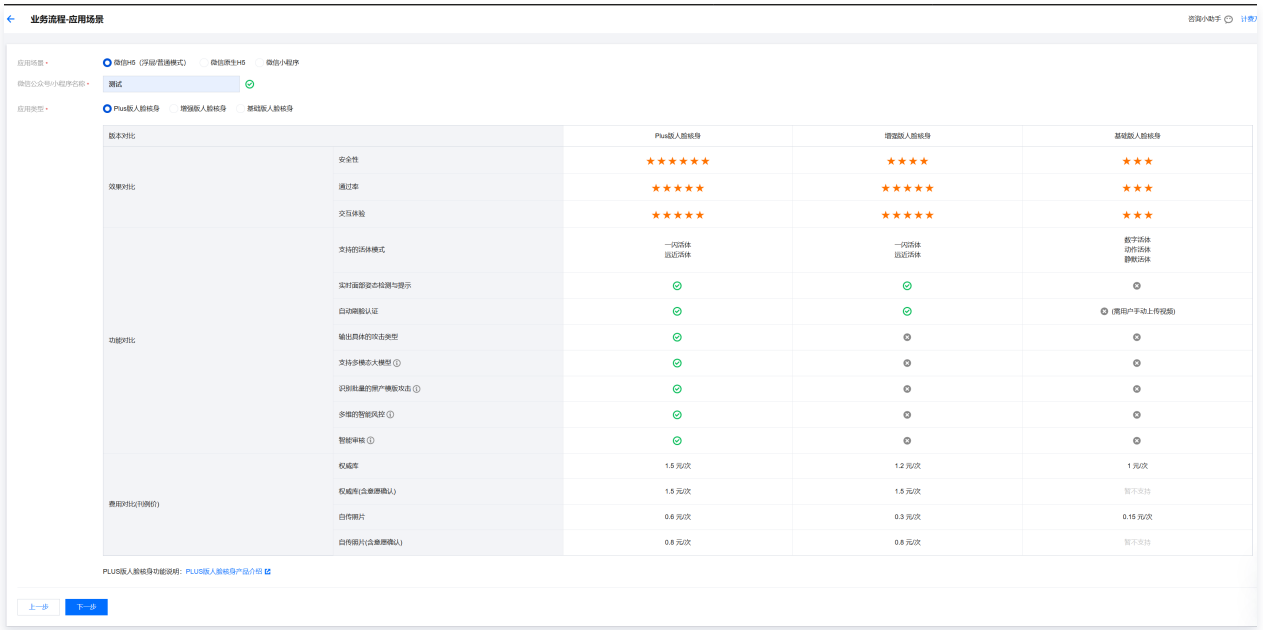
步骤1：创建 RuleId

RuleId 用于调用配置的业务流程，创建步骤如下：

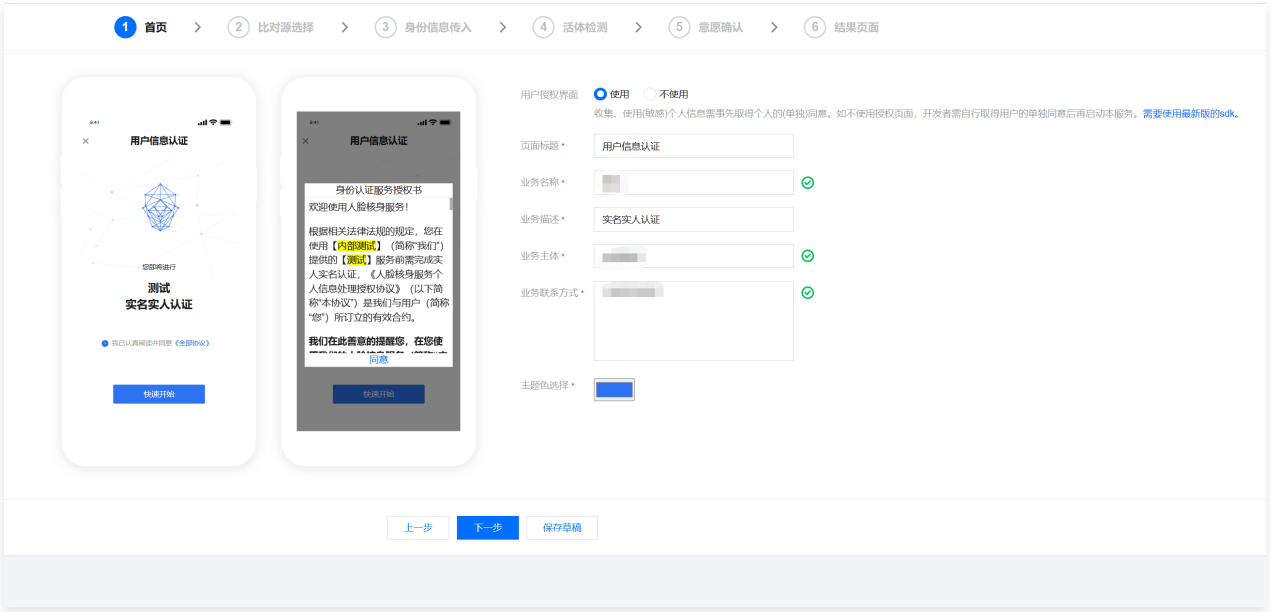
1. 登录腾讯云人脸核身控制台，在 **自主接入** 页面，单击**创建业务流程**。



2. 选择应用场景选择：**微信 H5（浮层/普通模式）**，应用类型选择：**Plus版人脸核身**，然后单击**下一步**。

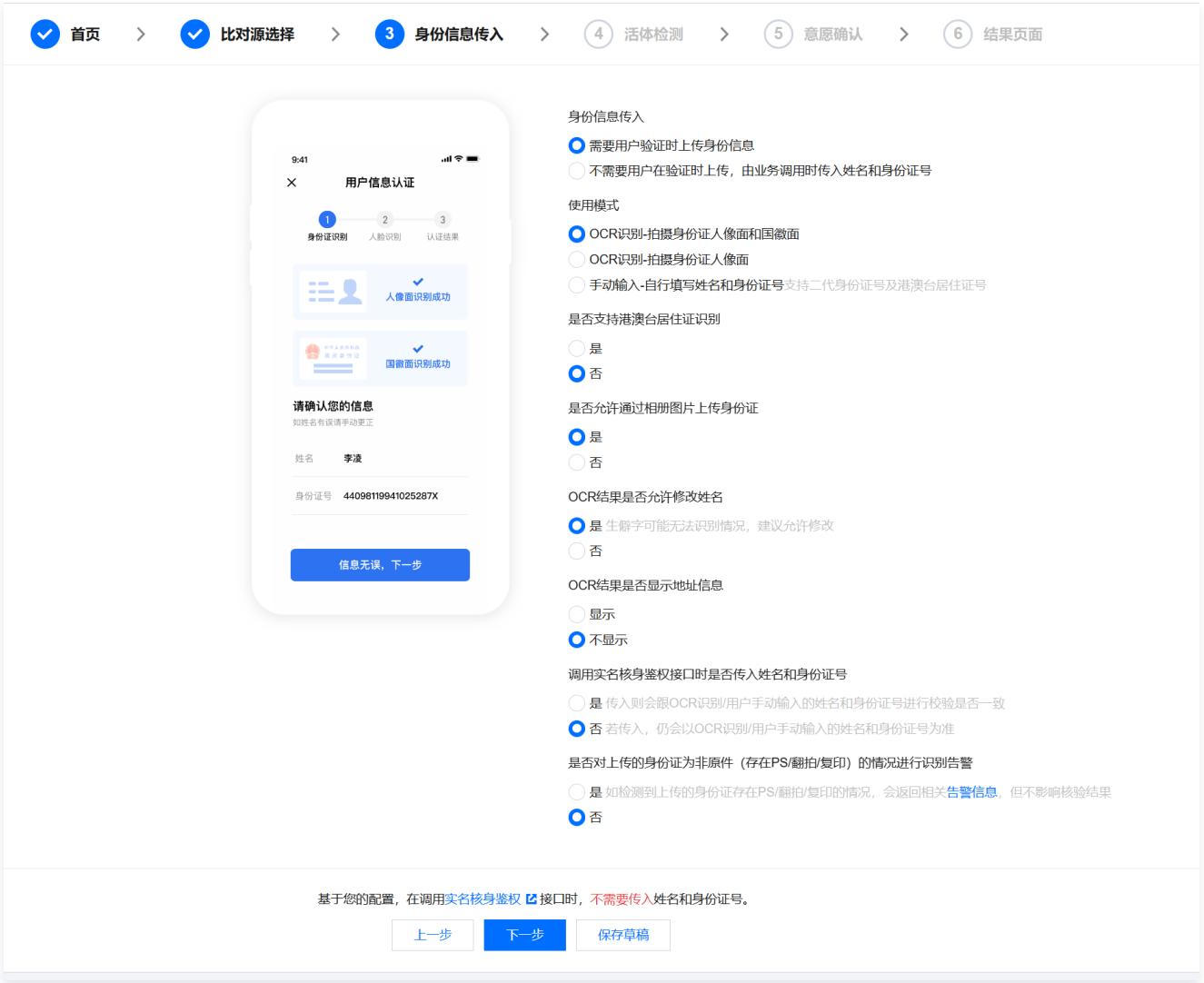


3. 进入接入配置，根据您业务的实际场景填写页面标题、业务名称、业务描述信息后单击**下一步**。



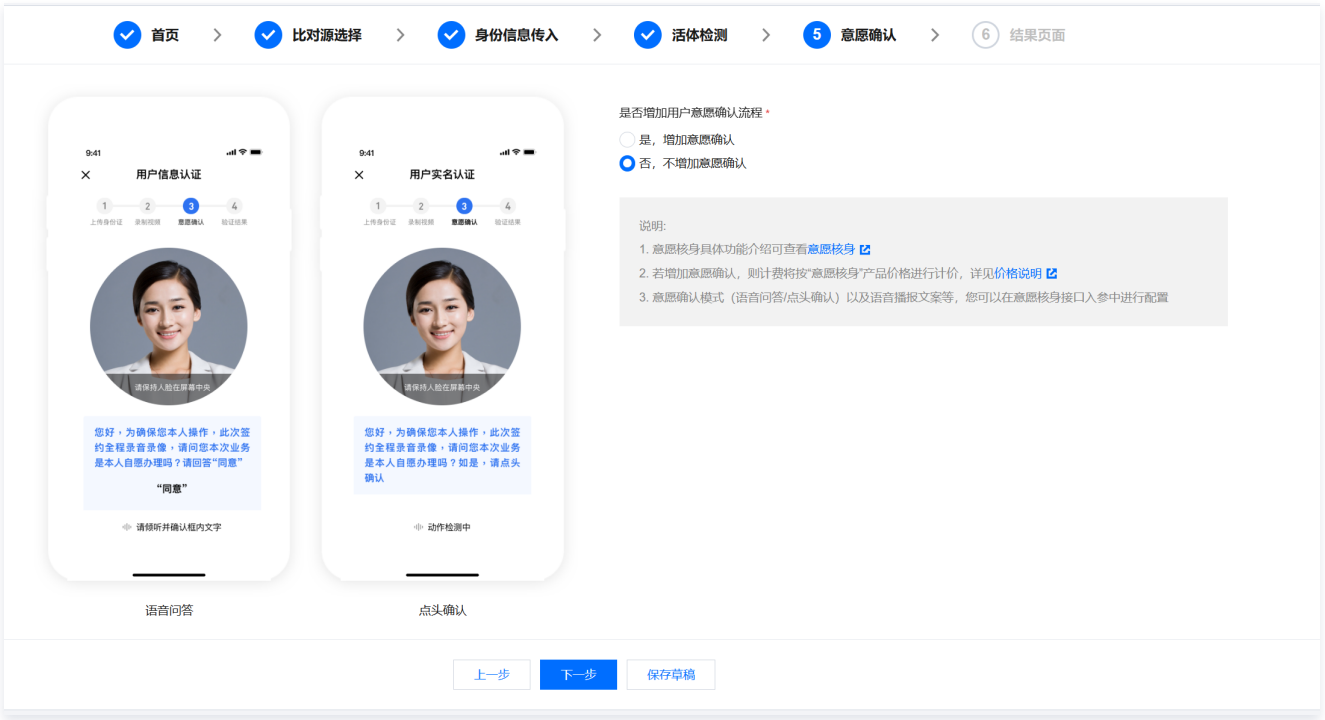
4. 选择人脸比对库源。人脸核身支持两种方式：**跟权威库比对**和**跟上传照片比对**。
   - 其中**跟权威库比对**收费价格为：Plus版人脸核身（权威库）– 浮层 H5 的价格。
   - **跟上传照片比对**收费的价格为：Plus版人脸核身（自传照片）– 浮层 H5 的价格。OCR 不再单独收费。

5. 配置身份证 OCR 功能，如果不需要则勾选“**不需要用户在验证时上传**”，然后单击**下一步**。

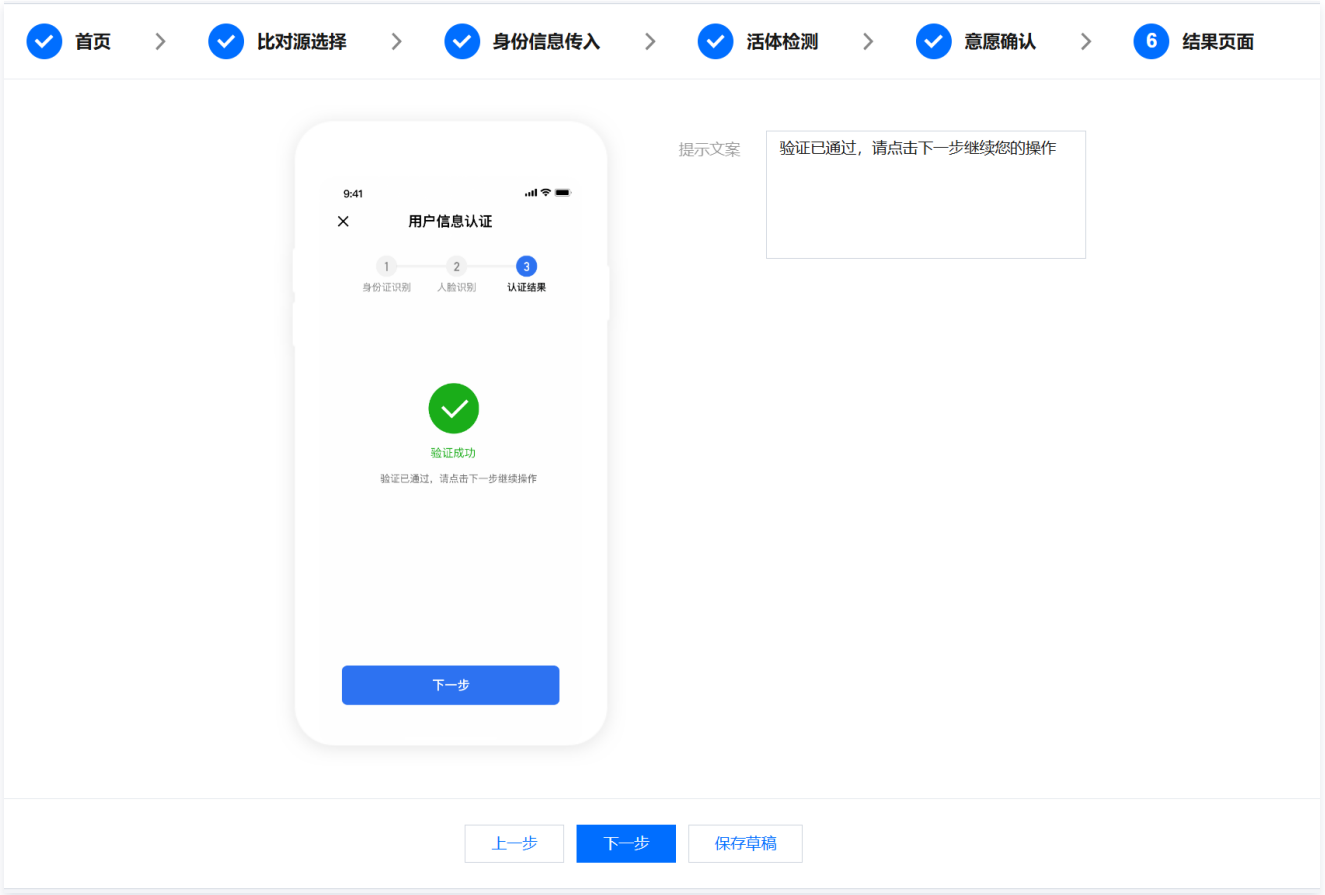


6. **配置活体检测方式，勾选后单击下一步。**

7. **配置意愿页面，根据您的业务情况勾选：是否增加用户意愿确认流程，然后单击下一步。请注意，Plus 版人脸核身默认是不增加意愿确认流程的，如配置了增加意愿确认，则将按意愿核身价格计费，详情请参见** 价格说明 。



8. 配置结果页的文案描述，然后单击**下一步**。



9. 业务信息填写完成后，确认您的配置信息然后单击**确认并提交审核**。

步骤2：获取 BizToken

- 完成 RuleId 创建后，需获取 BizToken，用于调用您配置的人脸核身验证的流程。
- 调用 实名核身鉴权，传入 步骤1 生成的 RuleId，和认证结束后重定向的回调链接地址 RedirectUrl，得到核验流程唯一密钥（BizToken）和用于发起核身流程的 URL。
- 在线调试 。

> ⚠ 注意：
> - BizToken 是仅一次核身流程的标识，有效时间为7,200秒；用户完成核身后，开发者可用该标识获取验证结果信息，结果信息仅保留3天。
> - 如果输入参数 RedirectUrl 为空，则用户完成核验后默认跳转腾讯云人脸核身产品介绍页。

步骤3： 跳转人脸核身 URL 完成核验

您在前端通过地址跳转方式重定向至 步骤2 中获取的核身入口 URL，用户进入核身流程，完成身份证拍摄识别、录制视频等操作完身份核验。

步骤4： 查询核验结果信息

用户完成人脸核身后，页面会跳转到 RedirectUrl 上，地址中会带上此次验证流程使用的 BizToken，您在服务端即可凭借 BizToken 参数调用 获取实名核身结果信息 接口去获取本次核身的详细信息。获取到用户验证过程数据，包括文本信息、识别分数、照片和视频。也可以通过访问 人脸核身控制台 查看服务调用情况。

> ⚠ 注意：
> - 为了保证用户的隐私，结果信息人脸核身侧仅保留3天，请您及时拉取。
> - 在开发、产品使用、费用、合同等问题有任何疑问，欢迎您 联系我们 进行询问。

人脸核身对账指引

最近更新时间：2024-11-15 17:07:12

本文将为您描述使用腾讯云慧眼人脸核身服务中的微信渠道产品时（如使用人脸核身微信小程序、H5（微信公众号）、H5（微信浏览器）的人脸核身 SaaS 服务），如何与慧眼人脸核身进行对账操作。

对账工作主要分为2部分：

1. 收集使用慧眼业务时的明细信息。
2. 根据明细信息计算出账单。

下文将为您提供两种方式进行收集明细和计算账单。

方式1

在业务流程中保存明细

用户完成核验后，接入方调用 DetectAuth 接口生成的 `BizToken` 是一次核身流程的流水号。在业务完成后，接入方需要调用 GetDetectInfoEnhanced 接口来获取 `DetectInfo`，`DetectInfo` 就是描述本次核身的详细信息，包括核身结果，收费详情，照片视频信息等。**一个 BizToken 对应且只对应一个 DetectInfo。**

收集完整明细信息的关键点在于：

1. 标记 `BizToken`，确保不遗漏任何一个 `BizToken`。
   在调用 DetectAuth 接口获取 `BizToken` 后，必须将其存储起来，并标记该 `BizToken` 是否已经拉取到 `DetectInfo`。
2. 建立机制，确保每一个 `BizToken` 都能拉取到 `DetectInfo`。
   一般情况下慧眼侧是建议接入方在自己的回调接口中（DetectAuth 接口传入的 `RedirectUrl` 参数）调用 GetDetectInfoEnhanced 接口 去获取 `DetectInfo`。但由于用户侧核身场景环境比较复杂，并不能保证每一次的核身流程都能回调到 `RedirectUrl`。所以，接入方需要建立一个机制，来检查出未拉取 `DetectInfo` 的 `BizToken`，并重新拉取一遍。
3. 确保拉取到完成状态时的 `DetectInfo`。
   在人脸核身业务中，每个 `BizToken` 都有自己的状态，当 `BizToken` 到达完成状态时，所有的接口均不允许调用，其对应的 `DetectInfo` 也不能再更新。也就是说，若 `BizToken` 未达到完成状态的情况下，用户有可能使用此 `BizToken` 进行重试。未达到完成状态的 `BizToken` 都须重新拉取 `DetectInfo` 来确保拉取到的 `DetectInfo` 是完成状态时的。
   满足以下条件中的任意一个，即可认为 `BizToken` 处在完成状态：
   - GetDetectInfoEnhanced 接口返回的 `Response.Text.ErrCode` 为0。
   - 距离 Token 生成时间已超过2小时。

> ⚠ **注意：**
> 慧眼人脸核身支持配置最大可重试次数，该次数仅影响活体一比一的次数，并不影响 OCR、短信等接口的调用。

计算账单

`DetectInfo` 中与计费相关的信息全部都位于 `Response.Text.LivenessDetail` 数组中，该数组存放的是该 `BizToken` 进行的活体一比一的次数与每次活体一比一的详情。**遍历该数组，统计 `IsNeedCharge` 为 `true` 的个数即是此 BizToken 的收费次数。**

> ⚠ **注意：**
> `Response.Text.ErrCode` 是用于判断本次核验是否通过的字段，不用于判断计费与否。

为方便对账，建议将 `LivenessDetail` 数组中的每一项均保存下来。

```
let totalCount = 0; // 活体一比一的总记录数
let chargeCount = 0; // 活体一比一的收费记录数
let GetDetectInfoResponse = GetDetectInfoEnhanced(xxx, xxx, ...); // 调用GetDetectInfoEnhanced接口获取DetectInfo
for (let detail of GetDetectInfoResponse.Text.LivenessDetail){
    totalCount = totalCount + 1;
    if (detail.IsNeedCharge == true) {
        chargeCount = chargeCount + 1; // 当IsNeedCharge为true时，收费记录数加1
    }
    saveChargeDetailToDB(detail); // 保存至数据库，以便日后查看
}
```

方式2

使用 GetWeChatBillDetails 接口每日拉取

为了方便接入方更快捷的获取明细信息，慧眼人脸核身提供了一个接口（GetWeChatBillDetails）用于拉取每日的明细信息。该接口可以拉取子账号 + RuleId 维度的某一天的所有 token 数据。
具体使用方式如下：
1. 建立定时任务，每日早上6点后开始执行。**早于6点调用 GetWeChatBillDetails 接口会报参数异常。**
2. 调用 GetWeChatBillDetails 接口，拉取前一天所有的验证明细信息。

```
    let WeChatBillDetails = []; // 当日全量的账单详情
    let cursor = 0; // 游标位，拉取第一页时传0；剩下页面根据上一次请求回包的结果传入。
    let hasNextPage = true; // 是否还有下一页的标志位
    while (hasNextPage) { // 如果还有下一页，则继续拉取
        const result = GetWeChatBillDetails(Date /* 传入某一天日期*/, RuleId /* 传入RuleId*/, cursor); // 调用
GetWeChatBillDetails接口，并获取返回。
        WeChatBillDetails = WeChatBillDetails.concat(result.WeChatBillDetails); // 将拉取到的数据合并到全量的数组中
        cursor = result.NextCursor; // 将回包中返回的下一页的游标赋值给游标位，准备下一次拉取
        hasNextPage = result.HasNextPage; // 将是否还有下一页的标志位重新赋值，用于决定是否进行下一次拉取
    }
    // 此时WeChatBillDetails中就是某一天全量的账单信息
```

计算账单

统计 WeChatBillDetail 数组中的 ChargeCount 总数，即为当日计费总数。同时记录 ChargeDetail 内容，可用于校验：ChargeCount 值 = ChargeDetail 中 IsNeedCharge 为 true 的条目总数 = ErrorCode 为收费错误码的条目总数。

```
let totalCount = 0; // 活体一比一的总记录数
let chargeCount = 0; // 活体一比一的收费记录数
for (let WeChatBillDetail of WeChatBillDetails) { // 遍历WeChatBillDetails数组
    let ChargeDetails = WeChatBillDetail.ChargeDetails; // 取计费详情属性
    totalCount += ChargeDetails.length; // 总记录数为计费详情的总条目数
    chargeCount += WeChatBillDetail.ChargeCount; // 收费记录数为ChargeCount的总和
    for (let ChargeDetail of ChargeDetails) {
        saveChargeDetailToDB(ChargeDetail); // 遍历ChargeDetails数组，保存至数据库，以便日后查看
    }
}
```

常见问题

1. **什么时候进行计费上报?**
   计费上报时机是以一比一请求时间为准，即 `Response.Text.LivenessDetail[].ReqTime` 、GetWeChatBillDetails 接口中的 `ChargeDetail.ReqTime` 。例如，BizToken 是在7月1日的23:59:59生成的，但是进行比对的时机是在7月2日的0点，那一比一的请求时间也会落在7月2日的0点，同时这次调用也会上报至7月2日的账单中。
2. **LivenessDetail 中有些字段为 null 是什么原因?**
   当本次核身没有进行一比一或者其他原因造成一比一失败时，有可能导致 ReqTime、Seq、Idcard、Name、Sim、IsNeedCharge、Comparestatus、Comparemsg、CompareLibType 字段是个空值，此时本条明细是不会进行计费的。可以理解为，只有 IsNeedCharge 字段为 true 的时候是计费的，该字段为 false、null 都不计费。

附录

GetWeChatBillDetails 接口介绍

输入参数

参数名	是否必填	类型	描述
Date	Y	Date	拉取的日期（YYYY-MM-DD）。最大可追溯到365天前。当天6点后才能拉取前一天的数据。
RuleId	Y	String	业务对应的 ruleid。
Cursor	Y	Integer	游标。用于分页，取第一页时传0，取后续页面时，传入本接口响应中得 NextCursor 字段的值。

输出参数

参数名	类型	描述
HasNextPage	Boolean	是否还有下一页。该字段为 true 时，需要将 NextCursor 的值作为入参继续调用本接口。
NextCursor	Integer	下一页的游标。用于分页。
WeChatBillDetails	WeChatBillDetail 数组	数据，明细如下表。

WeChatBillDetail

属性名	类型	说明
BizToken	String	token。
ChargeCount	Integer	本 token 收费次数。
ChargeDetails	ChargeDetail数组	本 token 计费详情，明细如下表。

ChargeDetail

属性名	类型	说明
ReqTime	String	一比一请求时间戳，13位。
Seq	String	一比一请求的唯一标记。
Idcard	String	一比一时使用的、脱敏后的身份证号。
Name	String	一比一时使用的、脱敏后的姓名。
Sim	String	一比一的相似度。0-100，保留2位小数。
IsNeedCharge	Boolean	本次详情是否收费。
ChargeType	String	收费类型，比对、核身、混合部署。
ErrorCode	String	本次活体一比一最终结果。
ErrorMessage	String	本次活体一比一最终结果描述。

H5（移动端浏览器）
H5 人脸核身介绍

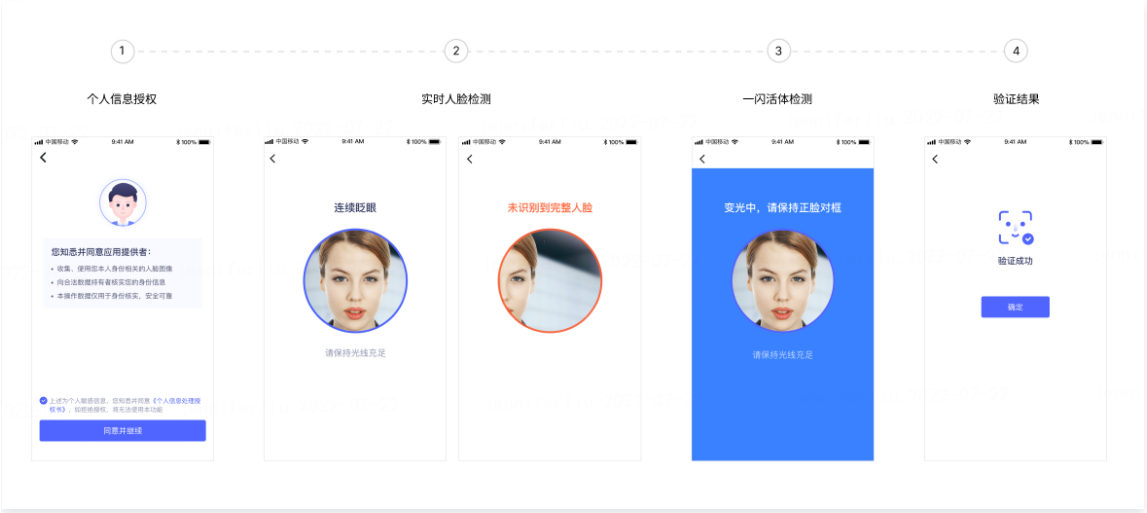
最近更新时间：2025-06-16 17:02:32

方案介绍

移动 H5 人脸核身全新升级，支持人脸实时检测和一闪活体模式，特点如下：

- 支持人脸实时检测，用户交互体验更好。
- 支持动作 + 一闪活体，活体安全性更高。
- 优先实时检测模式，如遇到不兼容的情况，则会平滑切换为视频录制模式。
- 支持多场景接入，包括 App 调用 H5、微信公众号、企业微信、浏览器等。

> **注意：**
> 接入前务必注意：请参见 移动 H5 人脸核身接入步骤 进行接入，如是 App 调用，务必按照 兼容性配置 指引进行兼容性适配，否则将影响正常使用。



兼容性说明

因网页端实时音视频技术是在2017年初次提出，对浏览器和手机系统存在兼容性要求，经过大规模测试，腾讯云H5实时检测人脸核身的兼容性情况如下：

手机平台	应用端	应用端说明	兼容性要求	机型支持率*
iOS	App	在客户自有 App 中使用，适用于在客户 App 中触达用户的业务场景	iOS 系统版本 14.3+ 需按照 兼容性配置 指引做适配	90%
	公众号	在微信公众号中使用，适用于在微信中触达用户的业务场景	iOS系统版本 14.3+ 微信版本 6.5+	95%
	浏览器	在手机浏览器使用，适用于通过短信或其他方式发送链接触达用户的业务场景	iOS系统版本 11.1.2+ 支持Safari浏览器，浏览器版本 11+	100%
Android	App	在客户自有 App 中使用，适用于在客户 App 中触达用户的业务场景	Android系统版本 7+ 需按照 兼容性配置 指引做适配	95%
	公众号	在微信公众号中使用，适用于在微信中触达用户的业务场景	Android系统版本 7+ 微信版本 6.5+	95%
	自带浏览器	在手机系统自带浏览器使用，适用于通过短信或其他方式发送链接触达用户的业务场景	Android系统版本 7+ 华为、OPPO、VIVO、魅族、荣耀、三星等自带浏览器兼容性较好（支持率80%） 小米自带浏览器兼容性一般（支持率30%）	
	QQ浏览器	在QQ手机浏览器使用，适用于通过短信或其他方式发送链接触达用户的业务场景	Android系统版本 7+ 支持QQ浏览器，不支持UC、百度浏览器	100%

- 采用达到版本要求的具有一定市场占有率的主流手机型号，测试机型共计829款。
- Android 手机测试品牌名单（按拼音排序）：HTC、OPPO、OPPO_REALME、VIVO、VIVO_IQOO、锤子-坚果、黑鲨、华硕、华为、荣耀、金立、联想、魅族、魅族-魅蓝、努比亚、努比亚-红魔、三星-GALAXY、小米、小米-红米、一加、中兴。

H5 人脸核身接入步骤

最近更新时间：2024-12-20 15:23:03

> **注意：**
> 接入前务必注意：请按照以下接入步骤进行接入，否则将影响人脸核身的正常使用。

序号	接入步骤	描述
1	App 调用 H5 兼容性配置	如是 App 调用，请务必按照兼容性配置指引进行 iOS 及 Android 手机的兼容性适配，否则将影响正常使用。
2	合作方后台上送身份信息	合作方后台上送人脸核身信息，包括 appid、orderNo、name、idNo、userID、liveInterType。
3	启动 H5 人脸核身	合作方上送 h5faceId 以及 sign，后台校验 sign 通过之后重定向到 H5 人脸核身。
4	人脸核身结果返回及跳转	刷脸完成后，认证结果页会回调启动 H5 人脸核身入参中指定的回调 URL 地址，使用回调参数中的 code判断是否核身通过。
5	人脸核身结果查询	当用户完成核身认证后，如果合作方需要拉取人脸核身的视频用于存证等其他需要，可以调用查询核身结果接口来获取。

人脸核身 App 调用 H5 兼容性配置指引

最近更新时间：2024-12-20 15:23:03

请按照下文 **兼容性配置** 指引进行 iOS 及安卓手机的兼容性适配。

iOS 接入

iPhone 的兼容性适配，需在配置里加上摄像头和麦克风的使用权限。App 的 info.plist 中加入：

```
.NSMicrophoneUsageDescription
.NSCameraUsageDescription
```

使用 WKWebView 时，需要通过 WKWebViewConfiguration 配置允许使用相机：

```
 WKWebViewConfiguration *config = [[WKWebViewConfiguration alloc] init];
config.allowsInlineMediaPlayback = YES;
```

Android 接入

由于 Android 机器碎片化严重，用系统 WebView 调起系统摄像头完成视频录制可能存在很多兼容性问题，如部分机器出现调不起摄像头、调起摄像头无法录制视频等。因此整理了接入指引。H5 刷脸包括 TRTC 和录制模式，合作方需要对这两种模式都做兼容性配置。

请合作方务必按照如下步骤顺序，实现兼容性处理：

1. 引入工具类。

   将 WBH5FaceVerifySDK.java 文件拷贝到项目中。该文件下载地址：人脸核身控制台（请到控制台自助接入列表页获取密码）。

2. 申请权限。

- 在 Manifest.xml 文件中增加申请以下权限：

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.INTERNET" />
```

- 动态申请权限：
  - 如果第三方编译的 targetSdkVersion >= 23，则需要动态申请权限。
  - 如果第三方编译的 targetSdkVersion < 23，则不需要动态申请权限。
  - 权限代码申请处理逻辑，demo 仅供参考，合作方可根据自身业务需求自行处理。
  - 一定要在动态权限申请成功后，才能去调用 enterOldModeFaceVerify() 录制模式或enterTrtcFaceVerify() trtc 模式体验 h5 刷脸功能。

3. 设置 WebSettings：

   调用 WebView.loadUrl(String url) 前一行添加如下代码设置 WebSettings。

```
/**
*对 WebSettings 进行设置：添加 ua 字段上送kyc/h5face;kyc/2.0
*/

webSetting.setUserAgentString(ua+";kyc/h5face;kyc/2.0");//设置ua包含;kyc/h5face;kyc/2.0
```

4. 重写 WebChromeClient：

   调用 WebView.loadUrl(String url) 前，WebView 必须调用 setWebChromeClient(WebChromeClient webChromeClient)，并重写 WebChromeClient 的如下5个函数：

```
/**
     *TRTC 刷脸模式配置，这里负责处理来自H5页面发出的相机权限申请
先申请终端的相机权限，再授权h5的trtc刷脸请求
     * @param request 来自H5页面的权限请求
     */
    @Override
    public void onPermissionRequest(PermissionRequest request) {
```

```
        if
(request!=null&&request.getOrigin()!=null&&WBH5FaceVerifySDK.getInstance().isTencentH5FaceVerify(request.get
Origin().toString())){  //判断是腾讯h5刷脸的域名
            Log.d(TAG,"onPermissionRequest 收到腾讯h5刷脸页面的相机授权");
            this.request=request;
            if (activity!=null){
//申请终端的相机权限，trtc模式一定需要申请相机权限。申请权限的代码demo仅供参考，合作方可根据自身业务定制
                activity.requestCameraPermission(true,false);
            }
        }
    }

    /**
     * 终端相机权限申请成功后，拉起TRTC刷脸模式进行实时刷脸验证

  */
    public void enterTrtcFaceVerify(){
  if (Build.VERSION.SDK_INT>Build.VERSION_CODES.LOLLIPOP){  // android sdk 21以上
            if (request!=null&&request.getOrigin()!=null){
               if (WBH5FaceVerifySDK.getInstance().isTencentH5FaceVerify(request.getOrigin().toString())){
//判断是腾讯h5刷脸的域名，如果合作方对授权域名无限制的话，这个if条件判断可以去掉，直接进行授权即可。
                    //授权
                    request.grant(request.getResources());
                    request.getOrigin();
                }
            }else {
                if (request==null){
                    Log.d(TAG,"enterTrtcFaceVerify request==null");
                    if (webView!=null&&webView.canGoBack()){
                        webView.goBack();
                    }
                }
      }
        }
     }

     // For Android >= 4.1  录制模式中，点击h5页面的【开始录制】按钮后触发的系统方法
    public void openFileChooser(ValueCallback<Uri> uploadMsg, String acceptType, String capture) {
if (WBH5FaceVerifySDK.getInstance().isTencentH5FaceVerify(null,null,acceptType)){ //判断是腾讯h5刷脸的域名
            this.uploadMsg=uploadMsg;
            this.acceptType=acceptType;
WBH5FaceVerifySDK.getInstance().setmUploadMessage(uploadMsg);
            if (activity!=null){
                //申请系统相机权限
                activity.requestCameraPermission(false,true);
            }
        }
     }

    // For Lollipop 5.0+ Devices  录制模式中，点击h5页面的【开始录制】按钮后触发的系统方法
    @TargetApi(21)
    @Override
    public boolean onShowFileChooser(WebView webView, ValueCallback<Uri[]> filePathCallback,
FileChooserParams fileChooserParams) {
        if (WBH5FaceVerifySDK.getInstance().isTencentH5FaceVerify(webView,fileChooserPara

ms,null)){ //判断是腾讯h5刷脸的域名
            this.webView=webView;
            this.filePathCallback=filePathCallback;
            this.fileChooserParams=fileChooserParams;
WBH5FaceVerifySDK.getInstance().setmUploadCallbackAboveL(filePathCallback);
            if (activity!=null){
 //申请系统相机权限
```

```
                activity.requestCameraPermission(false,false);
            }
        }
        return true;
 }

    //录制模式中，拉起系统相机进行录制视频
    public boolean enterOldModeFaceVerify(boolean belowApi21){
        if (belowApi21){ // For Android < 5.0
            if (WBH5FaceVerifySDK.getInstance().recordVideoForApiBelow21(uploadMsg, acceptType, activity)) {
                return true;
            }else{
// todo 合作方如果其他的h5页面处理，则再次补充其他页面逻辑
}
        }else { // For Android >= 5.0
            if (WBH5FaceVerifySDK.getInstance().recordVideoForApi21(webView, filePathCallback,
activity,fileChooserParams)) {
                return true;
            } else{
// todo 合作方如果其他的h5页面处理，则再次补充其他页面逻辑
}

        }
        return false;
    }
```

> **注意：**
> 如果第三方已重写以上函数，只要将如上述所示的函数体内容添加至第三方的对应函数体首行即可。
> 如果第三方没有重写以上函数，则直接按上述所示重写。
> WebView 不要使用 layerType 属性，否则导致刷脸界面白屏。

5. 重写 Activity：
   WebView 所属的 Activity 必须重写如下函数：

> **注意：**
> 如果第三方 WebView 所属的 Activity 已重写以下函数，则将如下所示的函数体内容添加至第三方的对应函数体首行即可。
> 如果第三方 WebView 所属的 Activity 没有重写以下函数，则直接按以下代码重写。

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        Log.d(TAG, "onActivityResult --------"+requestCode);
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == VIDEO_REQUEST) {//录制模式中，调用系统相机录制完视频后再回到当前app的回调
            if (WBH5FaceVerifySDK.getInstance().receiveH5FaceVerifyResult(requestCode, resultCode, data)) {
                return;
            }
        }else if (requestCode==PERMISSION_QUEST_TRTC_CAMERA_VERIFY){ //trtc模式中，申请相机权限时，从系统设置页面跳
转回当前app页面的处理。由于权限申请逻辑demo仅供参考，合作方自己处理即可。
            requestCameraPermission(true,belowApi21);
        }else if (requestCode==PERMISSION_QUEST_OLD_CAMERA_VERIFY){//录制模式中，申请权限时，从系统设置页面跳转回当
前app页面的处理。由于权限申请逻辑demo仅供参考，合作方自己处理即可。
            requestCameraPermission(false,belowApi21);
        }
    }
```

6. 权限拒绝的处理：
   在录制模式中，当用户点击了**开始录制**后，如果没有授权相机和录制权限，会弹框让用户授权。如果用户拒绝授权的话，一定要调用下面的方法：

```
WBH5FaceVerifySDK.getInstance().resetReceive();//https://www.teachcourse.cn/2224.html
```

可参看 demo 的调用和备注，demo 下载地址：人脸核身控制台（请到控制台自助接入列表页获取密码）。

Harmony Next 接入

demo下载地址：人脸核身控制台（请到控制台自助接入列表页获取密码）。

1. 申请权限

在model.json5文件中增加申请以下权限：

```
  "requestPermissions": [
      {
        "name": "ohos.permission.INTERNET",
      },
      {
        "name" : "ohos.permission.CAMERA",
        "reason": "$string:app_name",
        "usedScene": {
          "abilities": [
            "EntryAbility"
          ],
  "when":"inuse"
        }
      }
    ]
```

2. WebviewController 的设置

调用 Web 前添加如下代码给 WebviewController 设置 ua。

```
@State controller:webview.WebviewController=new webview.WebviewController();
```

3. 创建 Web 组件：

```
Web({
        src: 'https://kyc.qcloud.com/s/web/h5/#/entry', //设置加载的实时模式刷脸h5页面的url地址
        controller: this.controller  //给web设置controller
      })
        .onControllerAttached(() => {
          this.controller.setCustomUserAgent(this.controller.getCustomUserAgent() +
';kyc/h5face;hoskyc/2.0') //设置ua
        })
        .fileAccess(true)//web的配置，不能少
        .javaScriptAccess(true)//web的配置，不能少
```

4. Web 组件配置实时模式：

```
onPermissionRequest((event) => { // 终端页面收到h5刷脸实时模式的请求
          if (event) {
            this.checkPermission().then(() => { //1. 终端申请相机权限成功
              event.request.grant(event.request.getAccessibleResource()) // 2. 授权h5页面
            }).catch((error: BusinessError) => { // 终端申请相机权限失败
              console.error(TAG, "申请权限异常" + error.message)
              event.request.deny() //2.告诉h5页面没有权限
            })
          }
        })
  // 注意：checkersmission()是申请相机的方法，接入方可以根据自身业务需求发挥，demo的这个方法仅供参考。
```

5. Web 组件配置传统录制模式：

```
.onShowFileSelector((event) => { //终端页面收到h5刷脸传统录制模式的请求
          if (event) {
            let result = picCamera().then(result => { //1.打开系统相机
              let str: string[] = [result.resultUri]
              event.result.handleFileList(str) // 2. 把录制的视频传给h5页面
            }).catch((error: BusinessError) => { //打开系统相机异常
              console.error(TAG, "打开相机异常" + error.message)
            });
          }
          return true;
        })
```

Uniapp 接入（ Android ）

1. 注册并创建 uni-app 开发环境。

   uni-app 开发接　具体参照 uni 官　。
2. 下载 demo 并根据指引配置插件。

   demo下 载地址：人脸核身控制台（请到控制台自助接入列表页获取密码）。
   - 在 uni-app 工程 nativeplugins 目录下，放置 only android 插件以及插件的配置文件。

   - 在 uni-app 页面中调用插件方法，实现 H5 刷脸功能。

```
const h5FaceVerifyPlugin  = uni.requireNativePlugin('DC-WBH5FaceVerifyService');
  export default {
          methods: {
                  enterH5FaceVerify() {
                          let url="https://kyc1.qcloud.com/s/web/h5/#/entry";//拉起h5刷脸的url
                          let thirdurl="https://www.qq.com/";//h5刷脸完成后要跳转的接入方的url，这个接入方填写自
己的url
                          h5FaceVerifyPlugin.startH5FaceVerify({h5faceurl:url,
                          h5thirdurl:thirdurl},result => {
                                  console.log(result,"H5刷脸后跳转到thirdurl所在h5页面的回调");
                                  h5FaceVerifyPlugin.destroyH5Activity(null);//调用关闭插件的webView.
                                  //uniapp todo 接入方自己的逻辑

                          },result=>{
                                  //这里是终端接受h5页面的消息回调。uniapp与h5页面两者通信可通过这个回调作为中间桥接
实现。
                                  //注意：约定h5页面和webView通信通过JavaScriptInterface接口和JavaScript进行交
互。
```

```
                                //在H5页面中使用window.tencentApi.postMessage的方式来调用这个方法，参数为
String类型。
                                //如果是jsonobject需要转String
                                console.log(result,"自定义回调");
                                //uniapp todo 接入方自己的逻辑
                            });
                            console.log("click=======意愿性刷脸====>startH5FaceVerify");
                        }
                    }
                }
```

合作方后台上传身份信息

最近更新时间：2025-01-03 16:04:42

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 获取 SIGN ticket 。
- SIGN 类型的 ticket，其有效期为60分钟，且可多次使用。
- 合作方为人脸核身服务生成签名，需要具有以下参数：

参数	说明	来源
appId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
userId	用户唯一标识	合作方自行分配，需跟用户绑定，32位以内，不包含特殊字符
version	参数值为：1.0.0	-
ticket	合作伙伴服务端获取的 ticket，注意是 SIGN 类型	获取方式请参见 获取 SIGN ticket
nonce	必须是32位随机数	合作方自行生成

> **注意：**
> 参与签名的数据需要与使用该签名的接口中的请求参数保持一致。

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 nonce、userId、appId 连同 ticket、version 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	参数值
appId	IDAXXXXX
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
ticket	XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

- 字典排序后的参数为：

```
[1.0.0, IDAXXXXX, XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS,
kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551]
```

- 拼接后的字符串为：

```
1.0.0IDAXXXXXXO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7
TuserID19959248596551
```

- **计算 SHA1 得到签名：**

该字符串就是最终生成的签名（40位），不区分大小写。

```
D7606F1741DDCF90757DA924EDCF152A200AC7F0
```

合作方后台上送身份信息

请求

- **请求 URL：**https:// kyc1.qcloud.com/api/server/ getPlusFaceId?orderNo=xxx

> ⚠ **注意：**
> 为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。

- **请求方法：**POST
- **报文格式：**Content-Type: application/json
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 wbappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
orderNo	订单号，字母/数字组成的字符串，由合作方上送，每次唯一，不能超过32位	String	不能超过 32 位	是
name	姓名	String	-	使用权威源比对时：姓名+证件号 必须输入使用自带源比对时：姓名+证件号 可不输入
idNo	证件号码	String	-	使用权威源比对时：姓名+证件号 必须输入使用自带源比对时：姓名+证件号 可不输入
userId	用户 ID，用户的唯一标识（不能带有特殊字符），需要跟生成签名的 userId 保持一致	String	不能超过 32 位	是
sourcePhotoStr	比对源照片，**注意：原始图片不能超过500k，且必须为 JPG 或 PNG、BMP 格式。**参数有值：使用合作伙伴提供的比对源照片进行比对，必须注意是正脸可信照片，照片质量由合作方保证参数为空：根据身份证号 + 姓名使用权威数据源比对	BASE64 String	1048576	否，非必填请使用标准的图片转base64方法，base64编码不可包含换行符，不需要加前缀
sourcePhotoType	比对源照片类型参数值为1 时是：水纹正脸照参数值为 2 时是：高清正脸照 **重要提示：照片上无水波纹的为高清照，请勿传错，否则影响比对准确率。如有疑问，请联系腾讯云技术支持线下确认**	String	1	否，提供比对源照片需要传
liveInterType	活体交互模式 参数值为1时，表示仅使用实时检测模式，不兼容的情况下回调错误码3005； 参数值非1或不入参，表示优先使用实时检测模式，如遇不兼容情况，自动降级为视频录制模式。	String	1	否
version	默认参数值为：1.0.0	-	20	是
sign	签名：使用上面生成的签名	String	40	是

nonce	随机数	-	32	是

水纹照示例

响应

响应示例


```
 {
"code":"0",
"msg":"请求成功",
"bizSeqNo":"2106212000118443841832290801029 7",
"result":{
    "bizSeqNo":"2106212000118443841832290801029 7",
    "transactionTime":"20210621183229",
    "orderNo":"161709188560917432576916585 0",
    "faceId":"tx0375fa5243984381ea7b7013f13795",
    "optimalDomain":"kyc1.qcloud.com",
    "success":false
    },
"transactionTime":"20210621183229"
```


说明：
success：false 无意义，合作伙伴可忽略，无需处理。
faceId 有效期为5分钟，每次进行人脸核身都需要重新获取。

启动 H5 人脸核身

最近更新时间：2024-12-20 15:23:03

前置条件

合作方如果使用 App 内调起 H5 人脸核身，需要 App 平台的 webkit/blink 等组件支持调用摄像头录视频，方可正常使用人脸核身功能。

> **注意：**
> 1. 内调用 H5 人脸核身或者短信链接调用 H5 人脸核身等场景，若当前设备无法正常调用摄像头进行视频录制，前端将返回特定的错误码（如下）告知合作方，合作方拿到错误码后可选择用备用方案核身处理。
> 2. 在 App、微信公众号、浏览器中调用 H5 实时检测人脸核身，需要用户允许使用摄像头权限（授权操作形式包括弹窗、动作菜单、应用设置等，具体形式视手机型号而异）。如用户误操作导致拒绝授权，需要退出人脸核身并重新进入和允许授权。部分浏览器会缓存用户之前的授权操作，故如果退出人脸核身并重新进入时仍然出现无摄像头权限的情况，可以尝试清除浏览器缓存。

返回码	返回信息	处理措施
3001	该浏览器不支持视频录制	请使用其它验证方案
3002	登录态异常，cookie参数缺失	重新进入
3003	人脸核身中途中断	重新进入
3004	无摄像头权限	重新进入并授权摄像头
3005	该浏览器不支持实时检测模式	请使用其它浏览器
300101	报文包体问题	重新进入

摄像头授权操作示例：

形式1： 弹窗（popup）

形式2： 动作菜单（action sheet）

形式3：应用设置（setting）

生成签名

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 。

> **说明：**
> - 合作方根据本次人脸核身的如下参数生成签名，需要签名的参数信息如下表。

- 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	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，本次人脸核身合作伙伴上送的订单号，字母/数字组成的字符串,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（与接口中使用的 userId 保持一致）
version	参数值为：1.0.0	-
faceId	getAdvFaceId 接口返回的唯一标识	-
ticket	合作伙伴服务端获取的 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（与接口中的随机数保持一致）

基本步骤

1. 生成一个 32 位的随机字符串（字母和数字）nonce（接口请求时也要用到）。
2. 将 appId、userId、orderNo、version、faceId连同 ticket、nonce 共 7 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	appId001
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
faceId	bwiwe1457895464
orderNo	aabc1457895464
ticket	zxc9Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

字典排序后的参数为：

```
[1.0.0, aabc1457895464, appId001, bwiwe1457895464, kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551,
zxc9Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS]
```

拼接后的字符串为：

```
1.0.0aabc1457895464appId001bwiwe1457895464kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7TuserID19959248596551zxc9Qfxlti9iTVgH
AjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS
```

计算 SHA1 得到签名：

```
4E9DFABF938BF37BDB7A7DC25CCA1233D12D986B
```

该字串就是最终生成的签名（40 位），不区分大小写。

启动 H5 人脸核身

合作方上送 h5faceId 以及 sign，后台校验 sign 通过之后重定向到 H5 人脸核身。为了保证服务的高可用，全面消除单点风险，我们启用了多域名服务。启动 H5 人脸核身需要使用 合作方后台上送身份信息 接口返回内容中 optimalDomain 字段的域名。

请求 URL：

```
https://{optimalDomain}/api/web/login
```

> **注意：**
> 1. optimalDomain使用 合作方后台上送身份信息 接口返回的optimalDomain域名，如果 optimalDomain字段返回为空或者取不到，默认使用域名 kyc1.qcloud.com。
> 2. 该跳转url不能直接暴露在前端html页面的<a>标签中。某些浏览器会预加载<a>标签中的url，导致用户点击访问该url时，因url已经被预加载访问过，于是签名失效，页面报错“签名不合法”。

请求方法：GET

请求参数：

参数	说明	类型	长度	是否必填
appId	参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	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，字母/数字组成的字符串,每次唯一，此信息为本次人脸核身上送的信息，不能超过 32 位	String	32	是
faceId	getAdvFaceId接口返回的唯一标识。	String	32	是
url	**H5 人脸核身完成后回调的第三方 URL，需要第三方提供完整 URL 且做 URL Encode。**完整 URL Encode 示例:原 URL(https://cloud.tencent.com) Encode 后:(https%3a%2f%2fcloud.tencent.com)	String	-	是
resultType	是否显示结果页面，参数值为“1”时直接跳转到 url 回调地址，null 或其他值跳转提供的结果页面	String	-	否，非必填
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
from	browser：表示在浏览器启动刷脸 app：表示在 app 里启动刷脸默认值为 app	String	-	是
redirectType	跳转模式，参数值为“1”时，刷脸页面使用 replace 方式跳转，不在浏览器 history 中留下记录；不传或其他值则正常跳转	String	-	否，非必填

请求示例：

```
https://kyc1.qcloud.com/api/web/login?
appId=appId001&version=1.0.0&nonce=4bu6a5nv9t678m2t9je5819q46y9hf93&orderNo=161709188560917432576916585&faceId=
tx04f10695c3651ce155fea7070b74c9&url=https%3a%2f%2fcloud.tencent.com&from=browser&userId=23333333333333&sign=5D
D4184F4FB26B7B9F6DC3D7D2AB3319E5F7415F&redirectType=1
```

H5 人脸核身结果跳转

最近更新时间：2024-12-20 15:23:03

认证结束后，认证结果页会回调合作方后台上送身份信息中指定的回调 url 地址。并带有 code、orderNo、faceId 等参数。
您可以根据我们刷脸完成后的回调请求参数中的 code 判断是否核身通过，code 0 表示人脸核身成功，其他错误码表示失败。如果您需要二次验证结果准确性，您可以通过 服务端获取验证结果 获取最终认证结果。服务端获取验证结果 能够返回人脸核身的结果以及相应的视频和图片用于您存证等其他需要。（请注意：务必在收到完成刷脸的回调后，再来调用服务端获取验证结果。）

参数	说明	类型	长度（字节）
code	人脸核身结果的返回码，0 表示人脸核身成功，其他错误码标识失败。	字符串	32
orderNo	订单号 ，本次人脸核身上送的订单号。	字符串	32
faceId	本次请求返回的唯一标识，此信息为本次人脸核身上送的信息。	字符串	32
newSignature	对 URL 参数 App ID、orderNo 和 SIGN ticket、code 的签名。	字符串	40
liveRate	本次人脸核身的活体检测分数	字符串	40

注意：

- 若 H5 浏览器不支持调用摄像头录制视频，则直接返回错误 code（详见 启动 H5 人脸核身 的前置条件）。
- 在 H5 实时检测模式中，若人脸核身过程中发生中断，例如 App 被切换到后台等情况，前端会返回特定的错误码3003（人脸核身中途中断）。

人脸核身结果查询
查询核身结果

最近更新时间：2024-12-20 15:23:03

当您的用户完成核身认证后，如果您需要拉取人脸核身的视频和图片用于存证等其他需要，您可以调用查询核身结果接口来获取。

重要提示：

1. 您须要在前端完成刷脸的回调后，再来调用查询核身结果接口获取刷脸视频和照片。
2. 人脸核身完成后的相关业务数据，我们临时为您缓存3天，请根据自身业务所需，务必在3天时效内及时取回，超过3天我们会将业务数据完全清理。

> **注意：**
> 当您的 App 接入的是我们的基础版或增强版 SDK 服务时，SDK 回调中只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过查询核身结果接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，查询核身结果接口无法查询到刷脸结果。

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，获取方式请参见 SIGN ticket获取 。
- **合作方为人脸核身服务生成签名，需要具有以下参数：**

参数	说明	来源
app_id	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
order_no	订单号，字母/数字组成的字符串,是您需要查询结果的人脸核身订单号	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 。

人脸核身结果查询接口(升级)

请求

- **请求 URL：**https://kyc1.qcloud.com/api/v2/base/queryfacerecord?orderNo=xxx

> **注意：**
> 为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。

- **请求方法：**POST
- **报文格式：**Content-Type: application/json
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填
appId	腾讯云控制台申请的appid	String	8	是

version	版本号，默认值：1.0.0	String	20	是
nonce	随机数	String	32	是
orderNo	订单号，字母/数字组成的字符串,是您需要查询结果的人脸核身订单号	String	32	是
sign	签名值，使用本页第一步生成的签名	String	40	是
getFile	是否需要获取人脸识别的视频和文件，值为1则返回视频和照片、值为2则返回照片、值为3则返回视频；其他则不返回	String	1	否，非必填
queryVersion	查询接口版本号(传1.0则返回sdk版本号和trtc标识)	String	8	否

响应

- **响应参数：**

参数	类型	说明
code	String	0：表示身份验证成功且认证为同一人
msg	String	返回结果描述
bizSeqNo	String	业务流水号
transactionTime	String	请求接口的时间
orderNo	String	订单编号
liveRate	String	活体检测得分
similarity	String	人脸比对得分
occurredTime	String	进行刷脸的时间
photo	Base 64 string	人脸核身时的照片，base64 位编码
video	Base 64 string	人脸核身时的视频，base64 位编码
sdkVersion	String	人脸核身时的sdk版本号
trtcFlag	String	Trtc渠道刷脸则标识"Y"
appId	String	腾讯云控制台申请的appid

- **响应示例：**


```
{"code":"0","msg":"请求成功","bizSeqNo":"2203292000118445321117401579089 4","result":
{"orderNo":"16170918856091743257691658 52","liveRate":"99","similarity":"88.01","occurredTime":"2022032910471
7","appId":"IDAXXXXX","photo":"******","video":"******","bizSeqNo":"2203292000118445321117401579089 4","sdkVe
rsion":"1.12.12","trtcFlag":"Y"},"transactionTime":"20220329111740"}
```


code 非 0 时，有时不返回图片和视频。

注意事项：

- 照片和视频信息作为存证，合作伙伴可以通过此接口拉取视频等文件，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，默认不返回视频和照片信息。为确保用户操作整体流程顺利完成，部分情况下获取视频和照片会有1秒左右的延迟。
- 由于照片和视频信息有可能超过 1M，考虑传输的影响，建议合作伙伴在使用时注意，建议获取比对结果用于后续流程处理和存证使用分开调用。避免网络传输带来的影响。目前我们的查询接口是异步查询，返回的文件可能会有1/2 s的延迟。
- 如果合作方是完成人脸核身流程后马上去拉取视频/照片，建议在查询接口加上一个查询机制：判断图片是否存在，轮询3次，每次2s。

1.1 照片和视频均为 base64 位编码，其中照片解码后格式一般为 JPG 和 PNG。视频格式解码后一般为 MP4。

1.2 服务端验证结果接口返回66660011无此查询结果的可能原因：

1.2.1 66660017 验证次数超限后被风控，风控后的订单为无效，查询结果为无此查询结果。

1.2.2 用户中途退出刷脸，没有完成人脸验证的流程，没有比对，查询结果为无此查询结果。

1.2.3 66660018 操作超时，请退出重试 无此 ID 的用户身份信息，H5faceid 5分钟有效期，失效后无法进入刷脸流程，查询结果为无此查询结果。

1.2.4 66660016 视频格式或大小不合法 文件或视频不合法，无法进行比对，查询结果为无此查询结果。

1.2.5 400604 上传的视频非实时录制,被时间戳校验拦截，查询结果为无此查询结果(6).查询超过3天的订单返回无此查询结果。

1.3 响应参数请勿做强校验，后续可能会有扩展。

RiskInfo 对象说明

RiskInfo 是用来给合作方传递刷脸风险信息的对象。其中 RiskInfo 对象的字段意义如下表所示：

字段名	类型	字段含义	说明
LivenessInfoTag	String	AI防护盾标签	大模型命中风险描述

大模型命中风险描述：

序号	Plus 版 Tag 说明
01	用户全程闭眼
02	用户未完成指定动作
03	疑似翻拍攻击
04	疑似合成攻击
05	疑似黑产模版
06	疑似存在水印
07	反光校验未通过
08	出现多张人脸
09	人脸质量过差
10	距离校验不通过
11	疑似对抗样本攻击
12	脸部区域疑似存在攻击痕迹

人脸认证多张照片查询接口

最近更新时间：2024-12-20 15:23:03

人脸核身多张照片查询接口：获取人脸核身结果多张照片的接口。

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 。
- **合作方为人脸验证服务生成签名，需要具有以下参数：**

参数	说明	来源
webankAppId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请。
orderNo	订单号，字母/数字组成的字符串,是您需要查询结果的人脸核身订单号	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 。
nonce	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 。

核身结果-多照片查询接口

请求

请求URL： https://kyc1.qcloud.com/api/v2/base/queryphotoinfo?orderNo=xxx

> ⚠ **注意：**
> 为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。

请求方法：POST
HTTP 请求 header:

参数名	必选	类型	说明
Content-Type	是	string	application/json

请求参数：

参数	说明	类型	长度（字节）	是否必填
webankAppId	腾讯服务分配的AppId	字符串	腾讯服务分配	是
version	版本号，默认值：1.0.0	字符串	20	是
nonce	随机数	字符串	32	是
orderNo	订单号，字母/数字组成的字符串,是您需要查询结果的人脸核身订单号	字符串	32	是

sign	签名值，使用本页第一步生成的签名	字符串	40	是

响应

响应示例

```
{
     "code": 0,
     "msg":"请求成功",
       "bizSeqNo":"业务流水号",
   "result": {
       "orderNo": "AAAAAA001",
       "occurredTime": "20180907142653",
       "photoList":["第一个base64photo字符串", "第二个base64photo字符串", "第三个base64photo字符串"]
   }
 }
```

响应参数

参数名	类型	说明
code	int	0：成功，非0失败
msg	string	请求结果描述
bizSeqNo	string	请求业务流水号
orderNo	string	订单编号
occurredTime	string	刷脸时间（yyyyMMddHHmmss）
photoList	List	base64图像列表（返回1-3张照片）

增强版人脸核身
开始集成

最近更新时间：2024-06-20 10:49:31

> **说明**
> 如果您在接入过程中遇到问题，欢迎您 联系我们 进行询问。

增强版人脸核身支持通过 微信小程序、微信原生 H5（微信公众号）、微信浮层 H5（微信浏览器）、增强版 App SDK 和 移动浮层 H5（移动端浏览器） 渠道接入，接入流程如下：

接入方式	接入流程	特殊说明
H5（微信公众号）	1. 申请人脸核身服务 2. 确认公司主体资质 3. 申请业务流程 RuleId ，提交行业对应资质 4. 管理员微信号，扫描 授权二维码 5. 实名核身鉴权 6. 获取实名核身结果信息	有行业类目限制，审核时间3 – 5天 详细可查阅 微信原生H5（微信公众号）资质文件列表
H5（微信浏览器）	1. 申请人脸核身服务 2. 申请业务流程 RuleId 3. 实名核身鉴权 4. 获取实名核身结果信息	微信公众号、企业微信
微信小程序	1. 申请人脸核身服务 2. 确认公司主体资质 3. 申请业务流程 RuleId 下载微信小程序 SDK 4. 管理员微信号，扫描 授权二维码 5. 微信小程序接入 6. 实名核身鉴权 7. 获取实名核身结果信息	有行业类目限制，审核时间3 – 5天 详细可查阅 微信小程序资质文件列表
App SDK（增强版）	人脸核身 SDK 1. 合作方后台上传身份信息 2. Android SDK 3. iOS SDK 4. 人脸验证结果	–
H5（移动端浏览器）	1. App 调用 H5 兼容性配置（如 App 调用） 2. 合作方后台上传身份信息 3. 启动 H5 人脸核身 4. 人脸核身结果返回及跳转 5. 人脸核身结果查询	支持多渠道接入，包括 App 调用 H5、浏览器等

接入方式和活体检测模式

增强版人脸核身不同的接入方式对应的活体检测模式不同，详细如下表所示：

接入方式	活体检测模式（任选其一）	备注
微信小程序	一闪活体检测	有行业类目限制，审核时间3 – 5天
H5（微信公众号）	一闪活体检测	有行业类目限制，审核时间3 – 5天
H5（微信浏览器）	一闪活体检测	–
App SDK（增强版）	一闪活体检测	–

H5（移动端浏览器）	一闪活体检测	-

App SDK
合作方后台上传身份信息

最近更新时间：2025-01-07 10:21:42

生成签名

准备步骤

- 下载 SDK，请到控制台自助接入列表页获取密码。
- 前置条件：请合作方确保 SIGN ticket 已经正常获取，获取方式见 获取 SIGN ticket 。
- SIGN 类型的 ticket，其有效期为60分钟，且可多次使用。
- 合作方为人脸核身服务生成签名，需要具有以下参数：

参数	说明	来源
appId	业务流程唯一标识，即 WBappid	参考 获取 WBappid 指引在人脸核身控制台内申请
userId	用户唯一标识，同一个用户的 userId 请保持一致，不同用户请不要使用同一个 userId，我们会根据 userId 来做防重复点击优化以及登录态校验	合作方自行分配，需跟用户绑定，32位以内，不包含特殊字符
version	参数值为：1.0.0	-
ticket	合作伙伴服务端获取的 ticket，注意是 SIGN 类型	获取方式请参见 获取 SIGN ticket
nonce	必须是32位随机数	合作方自行生成

> **注意**
> 参与签名的数据需要与使用该签名的 SDK 中的请求参数保持一致。

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 nonce、userId、appId 连同 ticket、version 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**
> Java 签名算法可参考 签名算法说明 。

参考示例

- 请求参数：

参数	参数值
appId	IDAXXXXX
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
ticket	XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfu FBPIucaMS

- 字典排序后的参数为：

```
[1.0.0, IDAXXXXX, XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS,
kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551]
```

- 拼接后的字符串为：

```
1.0.0IDAXXXXXXO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7
TuserID19959248596551
```

- **计算 SHA1 得到签名：**
  该字符串就是最终生成的签名（40位），不区分大小写。

```
D7606F1741DDCF90757DA924EDCF152A200AC7F0
```

合作方后台上传身份信息

请求

- **请求 URL：**https://kyc1.qcloud.com/api/server/getAdvFaceId?orderNo=xxx

> ⚠ **注意**
> 为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。

- **请求方法：**POST
- **报文格式：**Content-Type: application/json
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
orderNo	订单号，字母/数字组成的字符串，由合作方上传，每次唯一，不能超过32位	String	不能超过32 位	是
name	姓名	String	-	使用权威源比对时：姓名+证件号 必须输入使用自带源比对时：姓名+证件号 可不输入
idNo	证件号码	String	-	使用权威源比对时：姓名+证件号 必须输入使用自带源比对时：姓名+证件号 可不输入
userId	用户 ID，用户的唯一标识（不能带有特殊字符），需要跟生成签名的 userId 保持一致。同一个用户的 userId 请保持一致，不同用户请不要使用同一个 userId，我们会根据 userId 来做防重复点击优化以及登录态校验	String	不能超过32 位	是
sourcePhotoStr	比对源照片，注意：原始图片不能超过500k，且必须为 JPG 或 PNG、BMP 格式。参数有值：使用合作伙伴提供的比对源照片进行比对，必须注意是正脸可信照片，照片质量由合作方保证参数为空：根据身份证号 + 姓名使用权威数据源比对	BASE64 String	1048576	否，非必填请使用标准的图片转base64方法，base64编码不可包含换行符，不需要加前缀
sourcePhotoType	比对源照片类型参数值为1 时是：水纹正脸照参数值为 2 时是：高清正脸照重要提示：照片上无水波纹的为高清照，请勿传错，否则影响比对准确率。如有疑问，请联系腾讯云技术支持线下确认	String	1	否，提供比对源照片需要传
version	默认参数值为：1.0.0	String	20	是
sign	签名：使用上面生成的签名	String	40	是
nonce	随机数	String	32	是

水纹照示例

响应

响应参数：

参数	类型	说明
code	String	0：成功 非0：失败 详情请参见增强版人脸核身服务错误码
msg	String	请求结果描述
bizSeqNo	String	请求业务流水号
orderNo	String	订单编号
faceId	String	此次刷脸用户标识，调 SDK 时传入

响应示例：

```
{
    "code": 0,
    "msg": "成功",
    "result": {
        "bizSeqNo":"业务流水号",
        "orderNo":"合作方订单号",
        "faceId":"cc1184c3995c71a731357f9812aab988"
     }
}
```

说明：
success：false 无意义，合作伙伴可忽略，无需处理。
faceId 有效期为5分钟，每次进行人脸核身都需要重新获取。

生成 SDK 接口调用步骤使用签名

最近更新时间：2025-01-15 17:14:52

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 。
- NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。
- 合作方为人脸核身服务生成签名，需要具有以下参数：

参数名	说明	来源
appId	业务流程唯一标识，即WBappid	参考 获取 WBappid 指引在人脸核身控制台内申请
userId	用户唯一标识	合作方自行分配（和 SDK 里面定义的 userId 保持一致）
version	参数值为：1.0.0	-
ticket	合作伙伴服务端获取的 ticket，注意是 NONCE 类型	获取方式见 NONCE ticket 获取
nonce	必须是32位随机数	合作方自行生成（和 SDK 里面定义的随机数保持一致）

> **注意**
> 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。

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 appId、userId、version 连同 ticket、nonce 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	IDAXXXXX
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
ticket	XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

- 字典排序后的参数为：

```
[1.0.0, IDAXXXXX, XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS ,
kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551]
```

- 拼接后的字符串为：

```
1.0.0IDAXXXXXXO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7
TuserID19959248596551
```

- 计算 SHA1 得到签名：

```
D7606F1741DDCF90757DA924EDCF152A200AC7F0
```

该字符串就是最终生成的签名（40 位），不区分大小写。

Android 人脸核身

开发准备

最近更新时间：2024-04-28 17:50:51

权限检测

- **Android 6.0 以上系统**
  SDK 需要用到相机，对 Android 6.0 以上的系统会做权限的运行时检测。
- **Android 6.0 以下系统**
- 由于 Android 6.0 以下系统 Android 并没有运行时权限，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出现异常，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，建议合作方在拉起 SDK 前，帮助 SDK 做相机权限检测，提示并确认用户打开了这项权限后，再进行人脸核身，可以使整个人脸核身体验更快更好。

CPU 平台设置

目前 SDK 只支持 armeabi-v7a、armeabi-v8a 平台，为了防止在其他 CPU 平台上 SDK crash，建议在您的 App 的 build.gradle 里加上 abiFilter，如下代码所示，您也可以根据您的 App CPU 平台情况进行相应的设置和过滤：

> **注意**
> SDK 支持 armeabi-v7a 和 armeabi-v8a 两个平台，合作方可以根据自身情况设置需要的 CPU 平台。

```
defaultConfig {
    ndk {
          //设置支持的so库框架
          abiFilters 'armeabi-v7a', 'arm64-v8a'
     }
}
```

配置流程

最近更新时间：2024-07-03 18:06:31

注意事项

- 人脸核身 SDK（WbCloudFaceLiveSdk）最低支持到 Android API 16: Android 4.1.0（ICS），请在构建项目时注意。
- 人脸核身 SDK 将以 AAR 文件的形式提供，默认黑色皮肤，无需格外设置。
- 人脸核身 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。
- 需要为人脸核身 SDK 添加依赖，方式如下：
  将提供的 AAR 文件加入到 App 工程的 libs 文件夹下，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-v7
 compile 'com.android.support:appcompat-v7:23.0.1'
 //1. 云刷脸SDK,
//aar的名称，例如：WbCloudFaceLiveSdk-v6.0.0-1234567.aar，填入'WbCloudFaceLiveSdk-v6.0.0-1234567'
 compile(name: 'aar的名称', ext: 'aar')
//2. 云normal SDK,
//aar的名称，例如：WbCloudNormal-v5.1.10-123456789.aar，填入 'WbCloudNormal-v5.1.10-123456789.aar'
 compile(name: 'aar的名称', ext: 'aar')
}
    }
```

混淆配置

人脸核身产品的混淆规则如下：

人脸核身 SDK 的混淆规则

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

```
#####################云刷脸混淆规则    faceverify-BEGIN#########################
####
#不混淆内部类
-keepattributes InnerClasses

#不混淆jni调用类
-keepclasseswithmembers class *{
    native <methods>;
}

##################### faceverify-BEGIN ##########################
-ignorewarnings
-keep public class com.tencent.ytcommon.**{*;}
-keep class com.tencent.turingfd.sdk.mfa.TNative$aa { public *; }
-keep class com.tencent.turingfd.sdk.mfa.TNative$aa$bb { public *;}
-keep class com.tencent.turingcam.** {*;}
-keep class com.tencent.turingfd.** {*;}

-keep public class com.tencent.youtu.ytagreflectlivecheck.jni.**{*;}
-keep public class com.tencent.youtu.ytagreflectlivecheck.YTAGReflectLiveCheckInterface{
```

```
    public <methods>;
}
-keep public class com.tencent.youtu.ytposedetect.jni.**{*;}
-keep public class com.tencent.youtu.ytposedetect.data.**{*;}
-keep public class com.tencent.youtu.liveness.YTFaceTracker{*;}
-keep public class com.tencent.youtu.liveness.YTFaceTracker$*{*;}

-keep public class com.tencent.cloud.huiyansdkface.facelight.net.*$*{
    *;
}
-keep public class com.tencent.cloud.huiyansdkface.facelight.net.**{
    *;
}
-keep public class com.tencent.cloud.huiyansdkface.facelight.config.cdn.WbUiTips{
    *;
}

#================数据上报混淆规则 start==========================
#实体类
-keep class com.tencent.cloud.huiyansdkface.analytics.EventSender{
    *;
}
-keep class com.tencent.cloud.huiyansdkface.analytics.EventSender$*{
    *;
}
-keep class com.tencent.cloud.huiyansdkface.analytics.WBSAEvent{
     *;
}
-keep class com.tencent.cloud.huiyansdkface.analytics.WBSAParam{
     *;
}
#================数据上报混淆规则 end==========================

#######################faceverify-END##############################

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

-keep public class com.tencent.cloud.huiyansdkface.normal.net.*$*{
    *;
}
-keep public class com.tencent.cloud.huiyansdkface.normal.net.*{
    *;
}
#bugly
-keep class com.tencent.bugly.idasc.**{
    *;
}
#wehttp混淆规则
-dontwarn com.tencent.cloud.huiyansdkface.okio.**
-keep class com.tencent.cloud.huiyansdkface.okio.**{
    *;
}
-dontwarn com.tencent.cloud.huiyansdkface.okhttp3.OkHttpClient$Builder

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

接口调用

最近更新时间：2025-06-06 17:12:52

SDK 接口调用方法

SDK 代码调用的入口为 `WbCloudFaceVerifySdk` 这个类。

```
public class WbCloudFaceVerifySdk {

/**
* 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 WbCloudFaceVerifySdk getInstance(){
//     ...
}
/**
* 在使用 SDK 前先初始化，传入需要的数据 data
* 由 WbCloudFaceVerifyLoginListener 返回是否登录 SDK 成功
* 关于传入数据 data 见后面的说明
*/
public void initAdvSdk(Context context,Bundle data,
WbCloudFaceVerifyLoginListener loginListerner){
//     ...
}
/**
* 登录成功后，调用此函数拉起 sdk 页面。
* 由 startWbFaceVerifySdk 返回刷脸结果。
*/
public void startWbFaceVerifySdk(Context ctx,
WbCloudFaceVerifyResultListener listener) {     // ...
}
/**
 * 释放资源，防止内存泄漏。收到刷脸结果后即可调用
 */
public void release() {
}
```

- NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。
- `WbCloudFaceVerifySdk.initAvdSdk()` 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以下内容（参数详情请参见 接口参数说明）：

```
//这些都是 WbCloudFaceVerifySdk.InputData 对象里的字段，是需要传入的数据信息
String faceId;   //此次刷脸用户标识，合作方需要向人脸识别后台拉取获得，详见获取 faceId 接口
String agreementNo;   //订单号
String openApiAppId;   //参考获取 appId 指引在人脸核身控制台内申请
String openApiAppVersion;   //openapi  Version
String openApiNonce; //32位随机字符串
String openApiUserId; //user id
String openApiSign; //签名信息

//刷脸类别：分级活体 FaceVerifyStatus.Mode.GRADE
FaceVerifyStatus.Mode verifyMode;
String keyLicence;    //在人脸核身控制台内申请
```

> ⚠ 注意
> 以上参数被封装在 WbCloudFaceVerifySdk.InputData 对象中（它是一个 Serializable 对象）。

接入示例

详情请参见 Android 光线活体 + 人脸比对 接入示例 。

接口参数说明

InputData 对象说明

InputData 是用来给 SDK 传递一些必须参数所需要使用的对象（WbCloudFaceVerifySdk.initSdk() 的第二个参数），合作方需要传入 SDK 需要的一些数据以便启动刷脸 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（字节）	是否必填
faceId	刷脸 ID 号，获取方式请参见 合作方上传身份信息	String	-	是
agreementNo	订单号，合作方订单的唯一标识	String	32	是
openApiAppId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	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	User Id，每个用户唯一的标识	String	32	是
openApiSign	获取方式请参考 生成 SDK 接口调用步骤使用签名	String	40	是
verifyMode	刷脸类型：分级模式 FaceVerifyStatus.Mode.GRADE	FaceVerifyStatus.Mode	-	是
keyLicence	在 人脸核身控制台 内申请的SDKlicense	String	以实际申请为准	是

个性化参数设置（可选）

`WbCloudFaceVerifySdk.initSdk()` 里 `Bundle data` ，除了必须要传的 InputData 对象之外，还可以由合作方为其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 支持境外调用

SDK 默认不支持境外调用。如果合作方需支持境外调用，可以通过该字段进行设置。设置代码如下：

```
# 优先使用境外域名，默认false，不使用
# 请注意：如果合作方可以分辨国内或者境外ip，建议境外ip开启这个配置，国内ip访问不进行配置
data.putBoolean(WbCloudFaceContant.IS_ABROAD, true);
```

选择 SDK 提示语言类型

合作方可以选择 SDK 显示提示语的语言类型，默认简体中文，可以选择繁体中文，英语，日语，韩语，泰语和印尼语。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(WbCloudFaceContant.INPUT_DATA, inputData);
  //简体中文  WbCloudFaceContant.LANGUAGE_ZH_CN
  //繁体中文  WbCloudFaceContant.LANGUAGE_ZH_HK
  //英语  WbCloudFaceContant.LANGUAGE_EN
  //印尼语  WbCloudFaceContant.LANGUAGE_ID
  //日语  WbCloudFaceContant.LANGUAGE_JA
  //韩语  WbCloudFaceContant.LANGUAGE_KO
  //泰语 WbCloudFaceContant.LANGUAGE_TH
  //默认设置为简体中文，此处设置为英文
  data.putString(WbCloudFaceContant.LANGUAGE,WbCloudFaceContant.LANGUAGE_EN);
```

特别注意，在设置了国际化语言的情况下，SDK 不会播放语音，自定义退出框也无效。同时合作方设定的自定义提示语也需要合作方传入相应语种的提示语。

SDK 样式选择

合作方可以选择 SDK 样式。目前 SDK 有黑色模式和白色模式两种，默认显示白色模式。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(WbCloudFaceContant.INPUT_DATA, inputData);
  //对 sdk 样式进行设置，默认为白色模式
  //此处设置为白色模式
  data.putString(WbCloudFaceContant.COLOR_MODE, WbCloudFaceContant.WHITE);
```

SDK 还支持自定义皮肤，支持定制刷脸过程中各个组件的色值，因涉及可设置元素较多，此处可以参考接入 demo。使用自定义皮肤时需要设置：

```
# 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((WbCloudFaceContant.INPUT_DATA, inputData);
  //对 sdk 样式进行设置，默认为白色模式
  //此处设置为自定义模式
  data.putString(WbCloudFaceContant.COLOR_MODE, WbCloudFaceContant.CUSTOM);
```

可配置的颜色值如下：

```
# 在App的colors.xml中设置：
   <!--供客户定制颜色值，可修改仅自己需要修改的颜色，其余复制粘贴demo示例即可-->
    <!--请注意，根据客户定制的需求，有些色值除了需要在用户app的colors里加上以下所有配置之外，还需要同步增加mipmaps和drawable的
xml文件-->
    <!--详情见下面说明，所有的定制res都以wbcf_custom开头，用户可以参考demo使用-->

    <!--授权页面还有一张背景图需要设置,请在mipmap文件夹里替换-->
    <!--授权页面titlebar背景颜色-->
    <color name="wbcf_custom_auth_title_bar">#ffffff</color>
    <!--授权页面titlebar返回键颜色-->
    <color name="wbcf_custom_auth_back_tint">#ffffff</color>
    <!--授权页面背景颜色-->
    <color name="wbcf_custom_auth_bg">#ffffff</color>
    <!--授权页面标题颜色-->
    <color name="wbcf_custom_auth_title">#1d2232</color>
    <!--授权页面文字颜色-->
    <color name="wbcf_custom_auth_text">#a5a7ad</color>
    <!--授权页面协议名称颜色-->
    <color name="wbcf_custom_auth_name_text">#1d2232</color>
    <!--授权页面按钮文字颜色-->
    <color name="wbcf_custom_auth_btn_text">#80ffffff</color>
    <!--授权页面被选中按钮背景色，需要配合drawable/wbcf_custom_auth_btn_checked.xml使用-->
    <color name="wbcf_custom_auth_btn_checked_bg">#1e2642</color>
    <!--授权页面未选中按钮背景色，需要配合drawable/wbcf_custom_auth_btn_unchecked.xml使用-->
    <color name="wbcf_custom_auth_btn_unchecked_bg">#9ca1ae</color>

    <!--识别页面背景色-->
    <color name="wbcf_custom_verify_bg">#ffffff</color>
    <!--识别页面初始化圆框-->
    <color name="wbcf_custom_initial_border">#33ffffff</color>
    <!--识别页面有脸时圆框颜色-->
    <color name="wbcf_custom_border">#ffb155</color>
    <!--识别页面有脸时提示文字颜色-->
    <color name="wbcf_custom_tips_text">#1d2232</color>
    <!--识别页面人脸不符合条件时圆框颜色-->
    <color name="wbcf_custom_border_error">#fa5a5a</color>
    <!--识别页面人脸不符合条件时提示文字颜色-->
    <color name="wbcf_custom_tips_text_error">#fa5a5a</color>
```

```
    <!--识别页面用户自定义提示文字颜色-->
    <color name="wbcf_custom_customer_tip_text">#5d646d</color>
    <!--识别页面长提示文字颜色-->
    <color name="wbcf_custom_long_tip_text">#777A84</color>
    <!--识别页面长提示背景颜色,需要配合drawable/wbcf_custom_long_tip_bg.xml使用-->
    <color name="wbcf_custom_long_tip_bg">#fafafa</color>

    <!--弹窗背景颜色-->
    <color name="wbcf_custom_dialog_bg">#ffffff</color>
    <!--弹窗标题文字颜色-->
    <color name="wbcf_custom_dialog_title_text">#363C62</color>
    <!--弹窗内容文字颜色-->
    <color name="wbcf_custom_dialog_text">#363C62</color>
    <!--弹窗右边按钮文字颜色-->
    <color name="wbcf_custom_dialog_right_text">#363C62</color>
    <!--弹窗左边按钮文字颜色-->
    <color name="wbcf_custom_dialog_left_text">#363C62</color>
```

合作方个性化提示定制（国际化语言模式下不支持）

合作方可以设置定制的提示语，通过配置传给 SDK。自定义提示语分为短提示（不长于17个字符）和长提示（不长于70个字符）。如果不设置，默认无。如果超过字数限制，将会被截断显示。
自定义短提示分为验证阶段提示和上传阶段提示，可以分开设置。位置可以选择位于预览圆框的上方或者下方。默认自定义短提示位于预览圆框下方。若设置自定义短提示位于预览圆框上方的话，SDK 过程提示将自动调整到圆框下方。
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(WbCloudFaceContant.INPUT_DATA, inputData);
  //定制合作方个性化提示语
  //此处将设置人脸采集时的个性化提示语
  data.putString(WbCloudFaceContant.CUSTOMER_TIPS_LIVE, “扫描人脸后与您身份证进行对比”);
  //此处将设置上传人脸时的个性化提示语
  data.putString(WbCloudFaceContant. CUSTOMER_TIPS_UPLOAD, “已提交审核，请等待结果”);
  //设置合作方定制提示语的位置，默认为识别框的下方
  //识别框的下方： WbCloudFaceContant.CUSTOMER_TIPS_LOC_BOTTOM
  //识别框的上方：WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP
  data.putInt(WbCloudFaceContant.CUSTOMER_TIPS_LOC,WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP);
```

自定义长提示位于整个验证页面的下方，存在于整个刷脸流程中，不能设置位置和时机。长提示的显示除了设置代码外，还需邮件申请开通，具体 联系小助手 。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(WbCloudFaceContant.INPUT_DATA, inputData);
//整个识别过程中合作方定制长提示语（不超过70个字符）
  data.putString(WbCloudFaceContant.CUSTOMER_LONG_TIP, customerLongTip);
```

合作方还可以自定义 SDK 退出二次确认弹窗的文字内容，包括提示标题，提示内容，确认键文案和取消键文案。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(WbCloudFaceContant.INPUT_DATA, inputData);
//退出确认弹窗定制提示标题
  data.putString(WbCloudFaceContant.DIALOG_TITLE, dialogTitle);
  //退出确认弹窗定制提示内容
  data.putString(WbCloudFaceContant.DIALOG_TEXT, dialogText);
  //退出确认弹窗定制确认键文案
  data.putString(WbCloudFaceContant.DIALOG_YES, dialogYes);
   //退出确认弹窗定制取消键文案
```

```
    data.putString(WbCloudFaceContant.DIALOG_NO, dialogNo);
```

比对类型选择

SDK 提供权威数据源比对、自带比对源比对，合作方可以选择传入参数控制对比类型。

- **权威库网纹图片比对类型**
  合作方必须要先在获取 faceId 的接口里送入用户的姓名与身份证号信息，得到相应的 faceId 后，再送入 SDK，供 SDK 与权威库网纹图片进行比对。不需要额外设置对比类型。
- **自带比对源比对类型**
  合作方在获取 faceId 的接口里送入用户提供对比源数据，获取 faceId 后送入 SDK 进行对比。合作方可以上传两类照片，一类是网纹照，一类是高清照；不需要额外设置对比类型。

是否录制视频存证

SDK 默认不录制视频存证。如果合作方需要视频存证，可以通过该字段进行设置。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(WbCloudFaceContant.INPUT_DATA, inputData);
  //设置是否录制视频进行存证，默认不录制存证。
  //此处设置为录制存证
  data.putBoolean(WbCloudFaceContant.VIDEO_UPLOAD, true);
```

是否对录制视频进行检查

> **说明**
> 如果在 “是否录制视频存证” 步骤的设置为录制存证，则目前该字段有效；否则，设置为不录制存证，也不会对视频进行检查，设置无效。

- 在 SDK 使用过程中，发现视频录制在性能不太好的手机上可能会报错，导致刷脸中断，影响用户体验。
- 为了减少录制视频导致的人脸核身中断问题，SDK 默认设置对录制的视频不作检测。
- 如果合作方对人脸核身安全有更加严格的要求，可以考虑打开这一选项。**但打开这个字段可能导致某些低性能手机上用户人脸核身不能进行，请慎重考虑。**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(WbCloudFaceContant.INPUT_DATA, inputData);
  //设置是否对录制的视频进行检测，默认不检测
  //此处设置为检测
  data.putBoolean(WbCloudFaceContant.VIDEO_CHECK, true);
```

是否播放语音提示（国际化语言模式下不支持）

SDK 默认不打开语音提示。合作方可以根据自身需求选择是否开启，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);
  //设置是否打开语音提示，默认关闭
  //此处设置为打开
  data.putBoolean(WbCloudFaceContant. PLAY_VOICE, true);
```

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(WbCloudFaceContant.INPUT_DATA, inputData);
  //个性化参数设置，此处均设置为与默认相反
   //默认设置为简体中文，此处设置为英文
  data.putString(WbCloudFaceContant.LANGUAGE,WbCloudFaceContant.LANGUAGE_EN);
```

```
//sdk样式设置，默认为白色
//此处设置为白色
data.putString(WbCloudFaceContant.COLOR_MODE, WbCloudFaceContant.WHITE);
//定制合作方个性化提示语
//此处将设置人脸采集时的个性化提示语
data.putString(WbCloudFaceContant.CUSTOMER_TIPS_LIVE, “扫描人脸后与您身份证进行对比”);
//此处将设置上传人脸时的个性化提示语
data.putString(WbCloudFaceContant.CUSTOMER_TIPS_UPLOAD, “已提交审核，请等待结果”);
//设置合作方定制提示语的位置，默认为识别框的下方
//识别框的下方： WbCloudFaceContant.CUSTOMER_TIPS_LOC_BOTTOM
//识别框的上方：WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP
//此处设置为识别框的上方
data.putInt(WbCloudFaceContant.CUSTOMER_TIPS_LOC,WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP);
//设置选择的比对类型  默认为权威数据源对比
//此处设置权威数据源比对
data.putString(WbCloudFaceContant.COMPARE_TYPE, WbCloudFaceContant.ID_CARD);
//是否需要录制上传视频 默认不需要，此处设置为不需要
data.putBoolean(WbCloudFaceContant.VIDEO_UPLOAD, false);
//是否对录制视频进行检查，默认不检查，此处设置为不检查
data.putBoolean(WbCloudFaceContant.VIDEO_CHECK, false);
//设置是否打开语音提示，默认关闭，此处设置为关闭
data.putBoolean(WbCloudFaceContant. PLAY_VOICE, false);
```

核身结果返回

最近更新时间：2025-01-07 10:21:42

Android SDK 结果分别由以下两个回调返回：
如果您只需要获取刷脸结果，不需要拉取视频和照片，您可以信任前端 SDK 核身结果，返回中的 isSuccess 字段 True 代表人脸核身对比成功；false 代表人脸核身失败。核身 SDK 和核身后台服务之间通信采用加密方式，可以有效防止结果篡改。

```
/**
  * 登录回调接口  返回登录 sdk 是否成功
  */
public interface WbCloudFaceVerifyLoginListener {
        void onLoginSuccess();
        void onLoginFailed(WbFaceError error);
    }

/**
  * 刷脸结果回调接口
  */
public interface WbCloudFaceVerifyResultListener {
        void onFinish(WbFaceVerifyResult result);
```

WbFaceVerifyResult 对象说明

WbFaceVerifyResult 是 SDK 用来给合作方传递身份识别结果的对象，在 WbCloudFaceVerifyResultListener 回调中作为参数返回给合作方 App。
WbFaceVerifyResult 对象的各个字段意义如下表所示：

字段名	类型	字段含义	说明
isSuccess	boolean	人脸核身是否成功	True 代表人脸核身对比成功 false 代表人脸核身失败，具体的失败原因请参考 WbFaceError 对象说明
sign	String	签名	供 App 校验人脸核身结果的安全性
liveRate	String	活体检测分数	-
similarity	String	人脸比对分数	“仅活体检测” 类型不提供此分数
RiskInfo	自定义对象	后台返回的刷脸风险信息	详见备注说明。 请注意：部分情况下此字段不返回，请不要做强校验
WbFaceError	自定义对象	人脸核身错误	人脸核身成功时为 null

RiskInfo 对象说明

RiskInfo 是 SDK 用来给合作方传递刷脸风险信息的对象，在 WbCloudFaceVerifyLoginListener 回调的 WbFaceVerifyResult 对象中作为参数返回给合作方 App。
其中 RiskInfo 对象的各个字段意义如下表所示：

字段名	类型	字段含义	说明
deviceInfoLevel	String	设备风险等级	设备风险级别，如果命中风险则显示
deviceInfoTag	String	设备风险标签	设备风险等级描述
riskInfoLevel	String	身份风险等级	如果命中身份风险则显示风险等级
riskInfoTag	String	身份风险标签	身份风险等级描述

详情见 增强版人脸核身错误码 风险标签说明。

WbFaceError 对象说明

WbFaceError 是 SDK 用来给合作方传递人脸核身错误信息的对象，在 WbCloudFaceVerifyLoginListener 回调和 WbFaceVerifyResult 对象中作为参数返回给合作方 App。WbFaceError 对象的各个字段意义如下表所示，各个字段的内容取值详情请参见 错误码 。

字段名	类型	字段含义	说明
domain	String	错误发生的阶段	只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，后端无法查询到刷脸结果
code	String	错误码	-
desc	String	错误描述	如有需求，可以展示给用户
reason	String	错误信息内容	错误的详细实际原因，主要用于定位问题

Android 错误码

最近更新时间：2021-11-16 14:58:12

SDK 在登录以及返回人脸服务结果时，如果发生错误或者识别失败会返回 WBFaceError 对象，其中各个字段的内容如下：

WBFaceErrorDomainParams

错误码	说明	原因
11000	传入参数为空	传入的 xx 为空
11001	传入的 keyLicence 不可用	传入的 keyLicence 不可用
11002	报文加解密失败	报文加解密失败

WBFaceErrorDomainLoginNetwork

错误码	说明	原因
21100	网络异常	登录时网络异常（请求未到达后台）
21200	网络异常	登录时后台返回参数有误（请求已到达后台）

WBFaceErrorDomainLoginServer

错误码	说明	原因
其他错误码	透传后台错误码	例如签名问题等

WBFaceErrorDomainGetInfoNetwork

错误码	说明	原因
31100	网络异常	获取活体类型/光线资源，网络异常（请求未到达后台）
31200	网络异常	获取活体类型/光线资源，后台返回参数有误（请求到达后台）

WBFaceErrorDomainNativeProcess

错误码	说明	原因
41000	用户取消	回到后台/单击 home/左上角/上传时左上角取消
41001	无法获取唇语数据	获取数字活体的数字有问题
41002	权限异常，未获取权限	相机
41003	相机运行中出错	-
41004	视频录制中出错	不能存/启动失败/结束失败
41005	请勿晃动人脸,保持姿势	未获取到最佳图片
41006	视频大小不满足要求	视频大小不满足要求
41007	超时	预检测/动作活体
41008	检测中人脸移出框外	活体/数字/反光
41009	光线活体本地错误	-
41010	风险控制超出次数	用户重试太多次
41011	没有检测到读数声音	数字活体过程中没有发声
41012	初始化模型失败，请重试	初始化算法模型失败

41013	初始化 SDK 异常	WbCloudFaceVerifySdk 未被初始化
41014	简单模式本地加密失败	编码转换异常/加解密编码失败

WBFaceErrorDomainCompareNetwork

错误码	说明	原因
51100	网络异常	对比时，网络异常（请求未到达后台）
51200	网络异常	对比时，后台返回参数有误（请求已到达后台）

WBFaceErrorDomainCompareServer

错误码	说明	原因
其他错误码	透传后台错误码	-

接入示例

最近更新时间：2024-07-03 18:06:31

权威库网纹图片比对、自带对比源对比接入示例：

```
# 在 MainActivity 中单击某个按钮的代码逻辑：
//先填好数据
Bundle data = new Bundle();
WbCloudFaceVerifySdk.InputData inputData = new WbCloudFaceVerifySdk.InputData(
                faceId,
                agreementNo,
                openApiAppId,
                openApiAppVersion,
                openApiNonce,
                userId,
                userSign,
                verifyMode,
                keyLicence);
  data.putSerializable(WbCloudFaceContant.INPUT_DATA, inputData);

//个性化参数设置,可以不设置，不设置则为默认选项。
//默认设置为简体中文，此处设置为英文
data.putString(WbCloudFaceContant.LANGUAGE,WbCloudFaceContant.LANGUAGE_EN);
//sdk样式设置，默认为白色
//此处设置为白色
data.putString(WbCloudFaceContant.COLOR_MODE, WbCloudFaceContant.WHITE);
//定制合作方个性化提示语，默认不设置
//此处将设置人脸采集时的个性化提示语data.putString(WbCloudFaceContant.CUSTOMER_TIPS_LIVE, “扫描人脸后与您身份证进行对
比”);
//此处将设置上传人脸时的个性化提示语
data.putString(WbCloudFaceContant. CUSTOMER_TIPS_UPLOAD, “已提交审核，请等待结果”);
//设置合作方定制提示语的位置，默认为识别框的下方
//识别框的下方： WbCloudFaceContant.CUSTOMER_TIPS_LOC_BOTTOM
//识别框的上方：WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP
//此处设置为识别框的上方
data.putInt(WbCloudFaceContant.CUSTOMER_TIPS_LOC,WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP);
//设置选择的比对类型  默认为权威库网纹图片比对
//此处设置权威数据源对比
data.putString(WbCloudFaceContant.COMPARE_TYPE, WbCloudFaceContant.ID_CARD);
//是否需要录制上传视频 默认不需要，此处设置为不需要
data.putBoolean(WbCloudFaceContant.VIDEO_UPLOAD, false);
//是否对录制视频进行检查，默认不检查，此处设置为不检查
data.putBoolean(WbCloudFaceContant.VIDEO_CHECK, false);
//设置是否打开语音提示，默认关闭，此处设置为关闭
data.putBoolean(WbCloudFaceContant. PLAY_VOICE, false);

//初始化 SDK，得到是否登录 SDK 成功的结果，由 WbCloudFaceVerifyLoginListener  返回登录结果
//【特别注意】建议对拉起人脸识别按钮做防止重复点击的操作
//避免用户快速点击导致二次登录，二次拉起刷脸等操作引起问题
WbCloudFaceVerifySdk.getInstance().
    initAdvSdk(DemoActivity .this, data, new WbCloudFaceVerifyLoginListener() {
        @Override
        public void onLoginSuccess () {
            //登录成功，拉起 sdk 页面，由 FaceVerifyResultListener   返回刷脸结果
            WbCloudFaceVerifySdk.getInstance().startWbFaceVerifySdk(DemoActivity.this, new
WbCloudFaceVerifyResultListener() {
                @Override
                public void onFinish(WbFaceVerifyResult result) {
                    if (result != null) {
                        if (result.isSuccess()) {
                            Log.d(TAG, "刷脸成功！");
```

```
                        } else {
                            Log.d(TAG, "刷脸失败！");
                        }
                    }
                    //刷脸结束后，及时释放资源
                    WbCloudFaceVerifySdk.getInstance().release();
                }
            });
        }
        @Override
        public void onLoginFailed (WbFaceError error){
            Log.d(TAG, "登录失败！");

            //刷脸结束后，及时释放资源
            WbCloudFaceVerifySdk.getInstance().release();
        }
    });
}
```

iOS 人脸核身
配置流程

最近更新时间：2025-06-06 17:12:52

使用 Cocoapod 集成

SDK 最低支持到 iOS11.0，请在构建项目时候注意，目前仅支持 Xcode11.0 及更高版本编译。如需特别支持 iOS8.0 版本，请联系技术支持。
以下为接入配置的步骤：
1. 将 TencentCloudHuiyanSDKFace_framework 文件夹拷贝到自己项目的 podfile 文件所在的同一目录。
2. 在 podfile 使用如下配置：

> ⚠ 注意
> target 后面内容根据自己项目配置，demo 可参考 SDK，请到控制台自助接入列表页获取密码。

```
target 'WBCloudReflectionFaceVerify-Demo' do
pod 'TencentCloudHuiyanSDKFace_framework', :path=> './TencentCloudHuiyanSDKFace_framework'
end
```

3. 使用 pod install 命令。
4. SDK 需要使用相机权限，请在 info.plist 中添加：

```
Privacy - Camera Usage Description
```

直接引用 Framework

SDK 最低支持到 iOS11.0，请在构建项目时候注意。
以下为接入配置的步骤：
1. 引用以下资源文件到项目：

```
TencentCloudHuiyanSDKFace.framework
TuringShieldCamRisk.framework
YTFaceTrackerLiveness.framework
YTCommonLiveness.framework
YTPoseDetector.framework
YTFaceAlignmentTinyLiveness.framework
YTFaceDetectorLiveness.framework
tnnliveness.framework
YTFaceLiveReflect.framework
TencentCloudHuiyanSDKFace.bundle
face-tracker-003.bundle
YTCv.framework
YtSDKKitFrameworkTool.framework
```

2. SDK 依赖以下系统框架，需要在**BuildPhases>Link Binary With Libraries**中添加，可以参考 Demo，具体依赖的系统库如下：

```
UIKit.framewrok
AVFoundation.framework
CoreVideo.framework
Security.framework
SystemConfiguration.framework
CoreMedia.framework
CoreTelephony.framework
ImageIO.framework
VideoToolbox.framework
Accelerate.framework
webkit.framework
libc++.tbd
```

```
libz.tbd
```

3. SDK 需要使用相机权限，请在 info.plist 中添加：

```
Privacy - Camera Usage Description
```

4. 需要在BuildSettings>Other Linker Flags中设置：`-ObjC`

接口调用

最近更新时间：2025-01-15 17:14:53

SDK 接口调用方法

SDK 的功能通过 WBFaceVerifyCustomerService 这个类的方法进行调用，**其中 SDK 中使用的 nonce，sign 等重要信息，需要合作方从自己后台拉取，并且两者不能缓存，使用后即失效**，详细接口说明如下，其他的操作请参考 Demo 中的登录接口的参数说明：

版本号及宏定义说明

```
#import <UIKit/UIKit.h>
#ifndef WBFaceVerifyConst_h
#define WBFaceVerifyConst_h
#define WBCloudReflectionFaceVerifyVersion

UIKIT_EXTERN NSString *const WBCloudFaceVerifySDKVersion;

/**
 SDK使用的主题风格

 - WBFaceVerifyThemeDarkness: 暗黑色系主题
 - WBFaceVerifyThemeLightness: 明亮色系主题
 - WBFaceVerifyThemeOrange: 橙色主题
 - WBFaceVerifyThemeCustom: 自定义主题，通过修改 bundle 中的 custom.json 实现自定义

 */
typedef NS_ENUM(NSInteger, WBFaceVerifyTheme) {
    WBFaceVerifyThemeDarkness = 0,
    WBFaceVerifyThemeLightness,
    WBFaceVerifyThemeOrange,
    WBFaceVerifyThemeCustom,
};

typedef NS_ENUM(NSInteger, WBFaceVerifyLanguage) {
    WBFaceVerifyLanguage_ZH_CN = 0, //简体中文
    WBFaceVerifyLanguage_ZH_HK,     //繁体中文
    WBFaceVerifyLanguage_EN,        //英语
    WBFaceVerifyLanguage_ID,        //印尼语
    WBFaceVerifyLanguage_JA,        //日语
    WBFaceVerifyLanguage_KO,        //韩语
    WBFaceVerifyLanguage_TH         //泰语
};

typedef NS_ENUM(NSInteger, WBFaceCustomTipsLoc) {
    WBFaceCustomTipsLoc_Bottom = 0, //提示语在下
    WBFaceCustomTipsLoc_Top,
};
#endif /* WBFaceVerifyConst_h */
```

入口方法说明

NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。
faceID +活体检测+人脸比对服务

```
/*
增强级SDK核身入口，注意传入的 faceId 不能为空，且必须为增强 faceId，否则会报 failure

@param userid 用户唯一标识，由合作方自行定义
@param nonce  满足接入要求的32位随机数
@param sign 获取方式请参考 [生成 SDK 接口调用步骤使用签名](https://cloud.tencent.com/document/product/1007/63358)
```

```
@param appid 业务流程唯一标识，即 wbappid，可参考 获取 WBappid 指引在人脸核身控制台内申请
@param orderNo 每次人脸身份认证请求的唯一订单号：建议为32位字符串（不超过32位）
@param apiVersion 后台 api 接口版本号（不是 SDK 的版本号），默认请填写@"1.0.0"
@param licence 在人脸核身控制台内申请（该 licence 同 App 当前使用的 bundle id 绑定）
@param faceId 合作方必须要先获取*增强级*faceId，再送入sdk，不允许为空，比对or活体检测服务由 sdkConfig.useAdvanceCompare 设置
@param sdkConfig SDK 基础配置项目
@param success 服务登录成功回调，登录成功以后开始进行活体和检测服务
@param failure 服务登录失败回调，具体参考错误码文档
*/
-(void)initAdvanceSDKWithUserId:(NSString *)userid
                          nonce:(NSString *)nonce
                           sign:(NSString *)sign
                          appid:(NSString *)appid
                        orderNo:(NSString *)orderNo
                     apiVersion:(NSString *)apiVersion
                        licence:(NSString *)licence
                         faceId:(nonnull NSString *)faceId
                      sdkConfig:(WBFaceVerifySDKConfig *)sdkConfig
                        success:(void (^)())success
                        failure:(void (^)(WBFaceError * _Nonnull))failure;
/**
 以上一次的登录结果拉起刷脸页面，必须先登录再拉起刷脸页面

 @return 拉起是否成功
 */
- (BOOL)startWbFaceVerifySdk;
```

个性化参数设置

SDK 登录接口 initSDK 方法中需要传入 WBFaceVerifySDKConfig 字段，通过该对象可以配置 SDK 中其他基础配置：包括设置主题风格，资源路径等，仅活体 or 比对服务，通过 useAdvanceCompare 属性区分，务必根据自己的业务进行设置，具体参考头文件。

```
/**
 人脸识别 SDK 基础配置类
 */
@interface WBFaceVerifySDKConfig : NSObject

#pragma mark - common
/**
 sdk中拉起人脸活体识别界面中使用UIWindow时的windowLevel配置，默认配置是1 + UIWindowLevelNormal

 如果接入放app中有其他自定义UIWindow， 为了防止界面覆盖，可以酌情设置该参数
 */
@property (nonatomic, assign) NSUInteger windowLevel;

/**
 是否境外用户
 默认：NO
 请注意：如果合作方可以分辨国内或者境外ip，建议境外ip开启这个配置，国内ip访问不进行配置
 */
@property (nonatomic, assign) BOOL isAbroad;

/**
 人脸识别服务是否进行通过录像，从而进行视频存证

 default: NO
 */
@property (nonatomic, assign) BOOL recordVideo;

/**
 是否由 SDK 内部处理 sdk 网络请求的 cookie
```

```
 默认值：YES
 */
@property (nonatomic, assign) BOOL manualCookie;

/**
 人脸识别页面中的主题风格，需要配合不同资源包使用：
 WBFaceVerifyThemeDarkness - 暗灰主题
 WBFaceVerifyThemeLightness - 明亮主题（默认）
 */
@property (nonatomic, assign) WBFaceVerifyTheme theme;
/**
 多语言配置
 默认中文，当使用其他语言时，强制静音
 */
@property (nonatomic, assign) WBFaceVerifyLanguage language;
/**
 是否静音
 默认值：YES
 */
@property (nonatomic, assign) BOOL mute;
/*
 送入自定义提示文案的位置
 默认：WBFaceCustomTipsLoc_Bottom
 */
@property (nonatomic, assign) WBFaceCustomTipsLoc tipsLoc;

/*
 检测过程中展示的文案
 默认为空
 */
@property (nonatomic, copy) NSString *customTipsInDetect;

/*
 上传过程中展示的文案
 默认为空
 */
@property (nonatomic, copy) NSString *customTipsInUpload;

/*
 底部提示文案，长度不超过70字
 */
@property (nonatomic, copy) NSString *bottomCustomTips;
/*
 退出二次确认UI配置
 */
@property (nonatomic, copy) NSString *exitAlertTitle; //标题
@property (nonatomic, copy) NSString *exitAlertMessage; //内容
@property (nonatomic, copy) NSString *exitAlertRight; //右边按钮
@property (nonatomic, copy) NSString *exitAlertLeft; //左边按钮
/*
 如果有使用苹果分屏模式（UIWindowScene），打开此开关
 Xcode11新建工程有使用 Scene，可以参考资料自行调整
 默认为 NO
 */
@property (nonatomic, assign) BOOL useWindowSecene;

/**
 资源路径，不包含bundle本身（仅当需要自己下发资源时配置，本地资源无需配置）
 ！！！重要：此目录下必须包含face-tracker-v001.bundle和WBCloudReflectionFaceVerify.bundle两个文件，否则无法拉起SDK
 */
@property (nonatomic, copy) NSString *bundlePath;
```

```
/**
 是否采用增强比对服务，仅增强接口生效，仅活体服务设置为 NO
 默认为 NO，注意：如果需要使用完整活体+比对服务请设置为 YES
 */
@property (nonatomic, assign) BOOL useAdvanceCompare;
#pragma mark - simple //非标特有字段，标准模式无需设置
/**
 是否返回录制的视频

 default: NO
 */
@property (nonatomic, assign) BOOL returnVideo;

/**
 返回视频加密的公钥，如果不配置则不加密

 需要recordVideo returnVideo同时为YES，才返回加密的视频内容
 */
@property (nonatomic, copy) NSString *publicKey;

/**
 AES 加密需要用到的 IV
 */
@property (nonatomic, copy) NSString *aesIV;
/**
 默认 sdk 配置
 */
+(instancetype)sdkConfig;

@end
```

自定义皮肤设置

可以通过修改 JSON 文件，自定义刷脸的部分 UI。首先把 Theme 设置为 WBFaceVerifyThemeCustom，再修改 JSON 文件，路径：`WBCloudReflectionFaceVerify_framework/Resource/WBCloudReflectionFaceVerify.bundle/custom.json`。

```
{
        "name": "custom",
    "statusBarStyle" :   "light",  /** 状态栏颜色 */
    "navbarTintColor":   "0xffffff", /** 导航栏色调 */
    "baseNavBarColor" :  "0x0", /** 导航栏默认颜色 */
    "authNavBarColor" :  "0x22252A", /** 授权详情页导航栏颜色 */

    "imageBgColor" :     "0x23262b", /** 刷脸页背景色 */
    "faceNormalColor": "0x33FFFFFF", /** 默认刷脸框颜色 */
    "faceSatisfyColor": "0x5065FF", /** 人脸框识别成功颜色 */
    "faceErrorColor": "0xFF6034", /** 人脸框识别失败颜色 */

    "guideLabelColor" :  "0xffffff", /** 提示语默认颜色 */
    "guideLabelErrorColor" : "0xff5240", /** 识别错误时提示语颜色 */
    "customTipsColor":   "0xFFFFFF", /** 自定义提示语颜色 */
    "bottomTipsBgColor": "0x0DFFFFFF", /** 底部提示背景色 */
    "bottomTipsTextColor": "0x80FFFFFF", /** 底部提示文案颜色 */

    "backButtonImage":"backbutton@dark", /** 刷脸页面返回按钮图片 */
    "authBackButtonImage":"backbutton@dark", /** 授权页面返回按钮图片 */
    "authBodyImage": "wbcf_auth_face@dark", /** 授权页人脸框图片 */
    "authCheckboxUnSelectImage":"wbcf_checkbox_unselect", /** 授权页勾选框未勾选图片 */
    "authCheckboxSelectImage":"wbcf_checkbox_select", /** 授权页勾选框勾选图片 */
    "authCheckboxTipsColor": "0xFFFFFF", /** 授权页勾选提示语颜色 */
    "authCheckboxLinkColor": "0x6F80FF", /** 授权页详情链接提示颜色 */
```

```
    "authAgreeButtonNormalColor": "0xB7C0FF", /** 授权页同意按钮默认颜色 */
    "authAgreeButtonHighlightColor": "0x5065FF", /** 授权页同意按钮高亮颜色 */
    "authAgreeTextNormalColor":"0xFFFFFF", /** 授权页同意按钮文案默认颜色 */
    "authAgreeTextHighlightColor":"0xFFFFFF", /** 授权页同意按钮文案高亮颜色 */

    "alertTitleColor":"0x000000", /** 弹框主题颜色 */
    "alertMessageColor":"0x000000", /** 弹框详细信息颜色 */
    "alertLeftBtnColor":"0x5065FF", /** 弹框左边按钮文案颜色 */
    "alertRightBtnColor":"0x5065FF", /**弹框右边按钮文案颜色 */
    "alertBackgroundColor":"0xFFFFFF", /** 弹框背景色 */
}
```

注意

1. 需要替换返回按钮图片时，需要修改对应 navbarTintColor 到按钮颜色。
2. 有透明色时，透明度放在前面，例如 80FFFFFF，FFFFFF 代表白色，80代表透明度。
3. 需要替换图片的目录在 JSON 文件同级目录 common 文件夹下，替换时注意保证相同尺寸大小，否则显示会异常。

接入流程与说明

最近更新时间：2023-08-25 14:33:51

整体工作流程

人脸识别主要流程包括以下步骤：

1. 登录人脸识别 SDK，即 initAdvanceSDKWithXXX 方法
2. 登录结果返回，在第一步登录成功以后，再调用 startWbFaceVerifySdk 方法拉起人脸识别页面，如果登录失败，会通过 failure block 回调返回一个 WBFaceError 对象。
3. 人脸识别的具体过程，当中各种失败情况以及用户主动退出，直接跳到第5步。
4. 人脸识别前端 SDK 检测结果上传后台进行活体检测以及人脸比对等服务。
5. 结果返回,通过 delegate 回调方法或者注册 NSNotification 的方式获取活体检测/人脸比对的结果，结果封装在 WBFaceVerifyResult。

登录接口说明

iOS 人脸识别接口大部分参数说明请参考头文件注释。

> **注意**
> userid、nonce、sign、orderNo 等参数建议通过接入方后台透传给前端，前端无需在本地生成。

方法功能一：

如果使用活体检测服务+人脸比对服务（身份证的网纹照片进行对比）方式，则需要传入 faceID，其中 faceID 需要合作方后台调用 合作方后台上传身份信息 生成 faceID，其中 userid、nonce、sign、orderNo 要与生成 sign 时，使用的相同。

方法功能二：

如果使用活体检测服务+人脸比对服务（合作方提供的比对源图片进行比对），则需要传入 faceID，其中 faceID 需要合作方后台调用 合作方后台上传身份信息 生成 faceID，其中 userid、nonce、sign、orderNo 要与生成 sign 时，使用的相同，在生成 faceID 时传入比对源图片。

识别界面拉起方式说明

默认情况 SDK 会创建一个全屏幕大小的 UIWindow，覆盖在当前屏幕上方，并且默认 windowLevel=UIWindowLevelNormal+1。

核身结果返回

最近更新时间：2025-01-07 10:21:42

iOS SDK 中，人脸识别结果返回给接入方有两种方式，两种结果回调都在 Main Thread 完成：
如果您只需要获取刷脸结果，不需要拉取视频和照片，您可以信任前端 SDK 核身结果，返回中的 isSuccess 字段 YES 代表人脸核身对比成功；NO 代表人脸核身失败。核身 SDK 和核身后台服务之间通信采用加密方式，可以有效防止结果篡改。

1. 通过实现 WBFaceVerifyCustomerServiceDelegate 的 WBFaceVerifyCustomerServiceDelegate 方法，通过该方法返回 WBFaceVerifyResult 对象。
2. 无需实现 delegate 方法，通过注册 WBFaceVerifyCustomerServiceDidFinishedNotification 通知，在接受到该通知时，进行结果处理。

WBFaceVerifyCustomerServiceDidFinishedNotification 通知中，通过获取该通知的 userInfo，获取字典中 key 为 faceVerifyResult 对应的 value 对象就是 WBFaceVerifyResult。

WBFaceVerifyResult 说明

字段名	类型	字段含义	说明
isSuccess	BOOL	人脸核身是否成功	YES 代表认证成功，NO 代表认证失败，具体原因请参见 iOS 错误码
sign	NSString	签名	供 App 校验人脸核身结果的安全性
liveRate	NSString	活体检测分数	-
similarity	NSString	人脸比对分数	“仅活体检测” 类型不提供此分数
WbFaceError	自定义对象	人脸核身错误	人脸核身成功时为 nil

WBFaceError 对象说明

字段名	类型	字段含义	说明
domain	NSString	错误发生的阶段	只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，后端无法查询到刷脸结果
code	NSString	错误码	-
desc	NSString	错误描述	如有需求，可以展示给用户
reason	NSString	错误信息内容	错误的详细实际原因，主要用于定位问题
RiskInfo	自定义对象	后台返回的刷脸风险信息	详见备注说明。请注意：部分情况下此字段不返回，请不要做强校验

RiskInfo 对象说明

RiskInfo 是 SDK 用来给合作方传递刷脸风险信息的对象，在结果回调的 WBFaceVerifyResult 对象中作为参数返回给合作方 App。
其中 RiskInfo 对象的各个字段意义如下表所示：

字段名	类型	字段含义	说明
deviceInfoLevel	String	设备风险等级	设备风险级别，如果命中风险则显示
deviceInfoTag	String	设备风险标签	设备风险等级描述
riskInfoLevel	String	身份风险等级	如果命中身份风险则显示风险等级
riskInfoTag	String	身份风险标签	身份风险等级描述

详情见 增强版人脸核身服务错误码 。

```
@interface WBFaceSimpleModeResult : NSObject

/**
 结果对应的订单号
 */
```

```
@property (nonatomic, copy, readonly) NSString *orderNo;

/**
 接下来用于人脸对比的安全性参数
 */
@property (nonatomic, copy, readonly) NSString *encryptAESKey;

/**
 视频编码
 */
@property (nonatomic, copy, readonly) NSString *userVideoString;

/**
 使用传入 publickey 加密过的 AES
 */
@property (nonatomic, copy, readonly) NSString *videoEncryptAESKey;
/**
 用于后面进行人脸比对的数据参数
 */
@property (nonatomic, copy, readonly) NSString *identifyStr;

@end

/*
 增强级结果，具体参数含义参考后台返回字段，结果为透传
 */
@interface WBFaceRiskInfo : NSObject
@property (nonatomic, copy, readonly) NSString *deviceInfoLevel;
@property (nonatomic, copy, readonly) NSString *deviceInfoTag;
@property (nonatomic, copy, readonly) NSString *riskInfoLevel;
@property (nonatomic, copy, readonly) NSString *riskInfoTag;
@end

/**
 人脸服务返回结果对象
 */
@interface WBFaceVerifyResult : NSObject
/**
 人脸比对结果是否通过：

 YES：表示人脸服务通过
 NO：表示人脸服务不通过
 */
@property (nonatomic, assign, readonly) BOOL isSuccess;
/**
 结果对应的订单号
 */
@property (nonatomic, copy, readonly) NSString *orderNo;
/**
 isSuccess == YES 时，sign 有值，可以去后台拉取本次人脸服务的照片，视频存证
 isSuccess == NO  时，sign 无意义
 */
@property (nonatomic, copy, readonly) NSString * sign;
/**
 活体检测服务得分

 isSuccess == YES 时，liveRate 有值：
        1. liveRate 可能是 @"分数为空"，这种情况具体咨询合作方
        2. float类型的字符串，请调用 [liveRate floatValue] 获取具体分数
 isSuccess == NO  时，liveRate 无意义
 */
@property (nonatomic, copy, readonly) NSString * liveRate;
```

```
/**
 人脸比对服务得分

 isSuccess == YES 时, similarity 有值:
         1. similarity 可能是 @"分数为空", 这种情况具体咨询合作方
         2. float类型的字符串, 请调用 [similarity floatValue] 获取具体分数
 isSuccess == NO  时, similarity 无意义
 */
@property (nonatomic, copy, readonly) NSString * similarity;

/**
 人脸比对图片之一

 isSuccess == YES 时，该属性是上送比对图片之一UIImage的base64编码字符串(图片格式是jpg)

 isSuccess == NO  时，如果是 SDK 返回的错误码，该属性为nil，如果是后端返回的错误，该属性是上送比对图片之一UIImage的base64编码
字符串(图片格式是jpg)
 */
@property (nonatomic, copy, readonly) NSString * userImageString;

/**
 isSuccess == YES 时候, error 无意义
 isSuccess == NO  时, error中存储的具体错误信息,参考 WBFaceError.h
 */
@property (nonatomic, strong, readonly) WBFaceError * error;

#pragma mark - 非标专用返回参数

@property (nonatomic, strong, readonly) WBFaceSimpleModeResult *simpleModeResult;

#pragma mark - 增强SDK专用参数
@property (nonatomic, strong, readonly) WBFaceRiskInfo *riskInfo;

#pragma mark -

-(NSString *)description;
@end
```

iOS 错误码

最近更新时间：2024-10-11 15:23:41

SDK 在登录以及返回人脸服务结果时，如果发生错误或者识别失败会返回 WBFaceError 对象，该对象的结构如下,并且在判断具体错误时，需要先根据 domain 字段判断错误发生在 sdk 服务运行中的位置，然后根据 code 判断具体的错误：

```
NS_ASSUME_NONNULL_BEGIN
/*
 错误 domain 划分成两类：

 出现在登录时，通过调用 startXXXX 方法的 failure block 进行回调返回：
 WBFaceErrorDomainInputParams、WBFaceErrorDomainLoginNetwork、WBFaceErrorDomainLoginServer


 人脸服务结果返回(封装在 WBFaceVerifyResult 中)：
 WBFaceErrorDomainGetInfo, WBFaceErrorDomainNativeProcess, WBFaceErrorDomainCompareNetwork,
WBFaceErrorDomainCompareServer
 */

/* 登录时传入参数有误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainInputParams;
/* 登录时网络请求错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainLoginNetwork;
/* 登录时后台拒绝登录，具体参考后台 word 版本错误码，这里直接透传 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainLoginServer;
/* 拉取有效信息时候网络错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainGetInfo;
/* native 本地在活体检测中，某些原因导致错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainNativeProcess;
/* 上送后台比对时,网络错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainCompareNetwork;
/* 后台比对完成,返回比对结果的错误原因*/
UIKIT_EXTERN NSString *const WBFaceErrorDomainCompareServer;

@interface WBFaceError: NSObject
/**
 错误发生的地方，具体的发生地方由上面定义的 WBFaceErrorDomainXXXX 指定
 */
@property (nonatomic, readonly, copy) NSString *domain;
/**
 每个domain中有相应的错误代码，具体的错误代码见
 */
@property (nonatomic, readonly, assign) NSInteger code; //错误代码
@property (nonatomic, readonly, copy) NSString *desc; //获取本地化描述
@property (nonatomic, readonly, copy) NSString *reason; // 错误出现的真实原因
@property (nonatomic, readonly, copy) NSDictionary * _Nullable otherInfo;// 预留接口
@end
```

若 Domain 为：WBFaceErrorDomainInputParams

错误码	描述	详细实际原因
12000	传入参数为空	传入的参数为空
12001	传入的 keyLicence 不可用	传入的 keyLicence 不可用
12002	身份证格式不正确	身份证格式不正确
12003	使用自带对比源，传入参数错误，非 base64	传入的 srcPhotoString 不是 base64
12004	使用自带对比源，传入参数错误，超过1MB	传入的 srcPhotoString 超过1MB

12005	SDK 资源引入版本不匹配	没有引入资源包或者引入的资源包版本与当前 SDK 版本不匹配
12006	订单号不能为0或者超过32位	-
12007	nonce 字符串位数不为32位	-

Domain 为：WBFaceErrorDomainLoginNetwork

错误码	描述	详细实际原因
22100	网络异常	登录时网络异常（请求未到达后台）
22200	网络异常	登录时后台返回参数有误（请求已到达后台）

Domain 为：WBFaceErrorDomainLoginServer

错误码	描述	详细实际原因
其他错误码	透传后台错误码	例如签名问题等

Domain 为：WBFaceErrorDomainGetInfo

错误码	描述	详细实际原因
32100	网络异常	获取活体类型或光线阈值，网络异常(请求未到达后台)
32200	网络异常	获取活体类型或光线阈值，后台返回参数有误（请求已到达后台）

Domain 为：WBFaceErrorDomainNativeProcess

错误码	描述	详细实际原因
42000	用户取消	回到后台或单击 home 或左上角或上传时左上角取消
42001	网络环境不满足认证需求	无网络或2G 网络
42002	权限异常，未获取权限	相机或麦克风或 read phone 或 external 或 storage
42003	相机运行中出错	-
42004	视频录制中出错	不能存或启动失败或结束失败
42005	请勿晃动人脸，保持姿势	未获取到最佳图片
42006	视频大小不满足要求	视频大小不满足要求
42007	超时	预检测/动作活体
42008	检测中人脸移出框外	活体或反光
42009	光线活体本地错误	-
42010	风险控制超出次数	用户重试太多次
42011	没有检测到读数声音	活体过程中没有发声
42012	模型初始化失败	-
42015	报文解密失败	请求报文解密失败

Domain 为：WBFaceErrorDomainCompareNetwork

错误码	描述	详细实际原因
52100	网络异常	对比时，网络异常（请求未到达后台）
52200	网络异常	对比时，后台返回参数有误（请求已到达后台）

Domain 为：WBFaceErrorDomainCompareServer

错误码	描述	详细实际原因
其他错误码	透传后台错误码	-

Harmony NEXT 人脸核身
开发准备

最近更新时间：2024-07-29 17:21:41

权限

SDK 需要用到相机权限和网络权限，如下示例：

```
"requestPermissions": [
      {
        "name": "ohos.permission.INTERNET",
      },
      {
        "name": "ohos.permission.GET_NETWORK_INFO"
      },
      {
        "name": "ohos.permission.CAMERA",
        "reason": "face verify",
        "usedScene": {
          "abilities": [
            "EntryAbility",
          ],
          "when": "inuse"
        }
      }
    ]
```

CPU 平台设置

当前 Harmony NEXT 手机都是64位手机，目前 SDK 只支持 armeabi-v8a 平台。

配置流程

最近更新时间：2025-06-06 17:12:52

> **注意：**
> 以下文章出现 “刷脸” 一词均可以表示为 “人脸核身”。

接入配置

SDK（WbCloudFaceLiveSdk-xx.har, 具体版本号以 SDK 交付件为准）最低支持到 5.0.0(12)，请在构建项目时注意。
刷脸 SDK 将以 HAR 文件的形式提供。
将提供的 HAR 文件添加到工程的 libs 目录下，并且在 **oh-package.json5** 中添加下面的配置：

```
"dependencies": {
"wbcloudfaceverifysdk": "file:../libs/ WbCloudFaceVerifySdk.har"
}
```

接口调用

最近更新时间：2025-01-15 17:14:53

本章将展示 SDK 核心接口的调用方法。具体的代码示例参考交付 Demo(demo/module/src/main/ets/pages/HuiYanModePage.ets)

> ⚠ **注意：**
> 以下文章出现 “刷脸” 一词均可以表示为 “人脸核身”。

SDK 接口调用方法

SDK 代码调用的入口为：**WbCloudFaceVerifySdk** 这个类。

```
export class WbCloudFaceVerifySdk {

  /**
   * 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 getInstance(): WbCloudFaceVerifySdk {
    …
  }

  /**
   * 在使用SDK前先初始化，传入需要的数据WbFaceVerifyConfig
   * 由 WbCloudFaceVerifyLoginCallback返回是否登录SDK成功
   * 关于传入WbFaceVerifyConfig见后面的说明
   */
  public initAdvSdk(context: Context, config: WbFaceVerifyConfig, loginCallback:
WbCloudFaceVerifyLoginCallback) {
    …
  }

  /**
   * 登录成功后，调用此函数拉起sdk页面。
   * 由 WbCloudFaceVerifyResultCallback返回刷脸结果。
   */
  public startWbFaceVerifySdk(context: Context, resultCallback: WbCloudFaceVerifyResultCallback) {
    …
  }

  /**
   * 拿到刷脸结果后，释放资源， 防止内存泄漏
   * 【注意】请务必在拿到刷脸结果后释放资源，否则可能会发生丢失回调的情况！
   */
  public release() {
    …
  }
}
```

WbCloudFaceVerifySdk.initSdk() 的第二个参数 config: WbFaceVerifyConfig 用来传递 sdk 初始化必需数据和 sdk 配置项参数。

```
export class WbFaceVerifyConfig {
    //sdk初始化必需数据
  public inputData: InputData;
  //是否开启日志，默认不打开sdk日志，
  //【注意】上线请务必关闭sdk日志！
  public isEnableLog: boolean = false;
}
```

必须传递的参数包括（参考要求详见本页接口参数说明的描述）：

```
//这些都是InputData对象里的字段，是需要传入的数据信息
export class InputData {
  //此次刷脸用户标识，合作方需要向人脸识别后台拉取获得，详见获取faceId接口
  public faceId: string;
//订单号
  public orderNo: string;
//APP_ID
  public appId: string;
  //openapi  Version
public version: string;
//32位随机字符串
  public nonce: string;
  //User id
  public userId: string;
  //签名信息
  public sign: string;
  ////在人脸核身控制台内申请
  public licence: string;
}
```

关于接口调用的示例可参考 接入示例 。

接口参数说明

InputData 对象说明

InputData 是用来给 SDK 传递必须参数所需要使用的对象，合作方需要往里塞入 SDK 需要的一些数据以便启动刷脸 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（字节）	是否必填
faceId	刷脸 id 号，由合作方向人脸识别后台拉取获得	String	-	是
orderNo	订单号，合作方订单的唯一标识	String	32	是
appId	业务流程唯一标识，即 wbappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
version	接口版本号，默认填1.0.0	String	20	是
nonce	32 位随机字符串，每次请求需要的一次性 nonce	String	32	是
userId	User Id，每个用户唯一的标识	String	30	是
sign	获取方式请参考 生成 SDK 接口调用步骤使用签名	String	40	是
licence	在 人脸核身控制台 内申请的 SDKlicense	String	以实际申请为准	是

个性化参数设置（可选）

WbCloudFaceVerifySdk.initPlusSdk() 里 WbFaceVerifyConfig，除了必须要传的 InputData 对象之外，还可以由合作方为其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。
合作方可以选择 SDK 样式。目前 SDK **有黑色模式和白色模式两种，默认显示白色模式**。设置代码如下：

```
//对 sdk 样式进行设置，默认为白色模式
//此处设置为白色模式
wbFaceVerifyConfig.themeMode = ThemeMode.Light
SDK 还支持自定义皮肤，支持定制刷脸过程中各个组件的色值。使用自定义皮肤时需要设置：
wbFaceVerifyConfig.themeMode = ThemeMode.Custom
并在App的resources/rawfile目录中新建wbcf_custom.json主题配置文件：
可配置的颜色值参考 json 文件如下：
{
```

```
"name": "custom",
"statusBarStyle": "light",
"navbarTintColor": "#ffffff",
"baseNavBarColor": "#0",
"authNavBarColor": "#22252A",
"imageBgColor": "#22252A",
"faceNormalColor": "#33FFFFFF",
"faceSatisfyColor": "#5065FF",
"faceErrorColor": "#EB4140",
"guideLabelColor": "#ffffff",
"guideLabelErrorColor": "#EB4140",
"customTipsColor": "#FFFFFF",
"bottomTipsBgColor": "#0DFFFFFF",
"bottomTipsTextColor": "#80FFFFFF",
"backButtonImage": "app.media.wbcf_back_white",
"authBackButtonImage": "app.media.wbcf_back_white",
"authBodyImage_651": "app.media.wbcf_protocal_img",
"authBodyImage_will": "app.media.wbwf_auth_head_image",
"authCheckboxHighlightColor": "#5065FF",
"authCheckboxTipsColor": "#FFFFFF",
"authCheckboxLinkColor": "#6F80FF",
"authAgreeButtonNormalColor": "#B7C0FF",
"authAgreeButtonHighlightColor": "#5065FF",
"authAgreeTextNormalColor": "#FFFFFF",
"authAgreeTextHighlightColor": "#FFFFFF",
"authTipBackgroundColor_651": "#292C31",
"authTipTitileTextColor_651": "#ADB1C6",
"authTipDetailTextColor_651": "#80FFFFFF",
"alertTitleColor": "#000000",
"alertMessageColor": "#000000",
"alertLeftBtnColor": "#5065FF",
"alertRightBtnColor": "#5065FF",
"alertBackgroundColor": "#FFFFFF"
}
```

合作方个性化提示定制（国际化语言模式下不支持）

合作方可以设置定制的提示语，通过配置传给 SDK。自定义提示语分为短提示（不长于17个字符）和长提示（不长于70个字符）。如果不设置，默认无。如果超过字数限制，将会被截断显示。 自定义短提示分为验证阶段提示和上传阶段提示，可以分开设置。位置可以选择位于预览圆框的上方或者下方。默认自定义短提示位于预览圆框下方。若设置自定义短提示位于预览圆框上方的话，SDK 过程提示将自动调整到圆框下方。 设置代码如下：

```
let uiConfig: WbUiConfig = new WbUiConfig();
//设置合作方定制提示语的位置，默认为识别框的下方
//识别框的下方： CustomTipsLocation.Bottom
//识别框的上方： CustomTipsLocation.Top
uiConfig.tipsLocation = CustomTipsLocation.Bottom
//此处将设置人脸采集时的个性化提示语
uiConfig.customTipsInDetect = "扫描人脸后与您身份证进行对比"
//此处将设置上传人脸时的个性化提示语
uiConfig.customTipsInUpload = "已提交审核，请等待结果"
uiConfig.bottomCustomTips = cache.bottomCustomTips
wbFaceVerifyConfig.uiConfig = uiConfig
```

自定义长提示位于整个验证页面的下方，存在于整个刷脸流程中，不能设置位置和时机。长提示的显示除了设置代码外，还需邮件申请开通，详情请联系我们 。设置代码如下：

```
uiConfig.bottomCustomTips = customerLongTip
```

核身接口返回

最近更新时间：2025-01-07 10:21:42

SDK 结果分别由以下两个回调返回：
如果您只需要获取刷脸结果，不需要拉取视频和照片，您可以信任前端 SDK 核身结果，返回中的 isSuccess 字段 True 代表人脸核身对比成功；false 代表人脸核身失败。核身 SDK 和核身后台服务之间通信采用加密方式，可以有效防止结果篡改。

```
/**
  * 登录回调接口  返回登录 sdk 是否成功
  */
export interface WbCloudFaceVerifyLoginCallback {
  onLoginSuccess: () => void;
  onLoginFail: (error: WbFaceError) => void;
}
/**
  * 刷脸结果回调接口
  */
export interface WbCloudFaceVerifyResultCallback {
  onFinish: (result: WbFaceVerifyResult) => void;
}
```

WbFaceVerifyResult 对象说明

WbFaceVerifyResult 是 SDK 用来给合作方传递身份识别结果的对象，在 WbCloudFaceVerifyResultListener 回调中作为参数返回给合作方 App。
WbFaceVerifyResult 对象的各个字段意义如下表所示：

字段名	类型	字段含义	说明
isSuccess	boolean	人脸核身是否成功	True 代表人脸核身对比成功；false 代表人脸核身失败，具体的失败原因请参考 WbFaceError 对象说明
sign	String	签名	供 App 校验人脸核身结果的安全性
liveRate	String	活体检测分数	-
similarity	String	人脸比对分数	“仅活体检测” 类型不提供此分数
WbFaceError	自定义对象	人脸核身错误	人脸核身成功时为 null

WbFaceError 对象说明

WbFaceError 是 SDK 用来给合作方传递人脸核身错误信息的对象，在 WbCloudFaceVerifyLoginListener 回调和 WbFaceVerifyResult 对象中作为参数返回给合作方 App。WbFaceError 对象的各个字段意义如下表所示，各个字段的内容取值详情请参见 增强版人脸核身服务错误码 。

字段名	类型	字段含义	说明
domain	String	错误发生的阶段	只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，后端无法查询到刷脸结果
code	String	错误码	-
desc	String	错误描述	如有需求，可以展示给用户
reason	String	错误信息内容	错误的详细实际原因，主要用于定位问题

错误码描述

最近更新时间：2024-07-29 17:21:41

> **注意：**
> 以下文章出现 “刷脸” 一词均可以表示为 “人脸核身”。

SDK 在登录以及返回人脸服务结果时，如果发生错误或者识别失败会返回 WBFaceError 对象，该对象的字段结构意义见 接口参数说明，其中各个字段的内容如下：

WBFaceErrorDomainParams

Code（错误码）	Description（描述）	Reason（详细实际原因）
13000	传入参数为空	传入的 xx 为空
13001	传入的 keyLicence 不可用	传入的 keyLicence 不可用
13002	报文加解密失败	报文加解密失败

WBFaceErrorDomainLoginNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
23100	网络异常	登录时网络异常（请求未到达后台）
23200	网络异常	登录时后台返回参数有误（请求到达后台）

WBFaceErrorDomainLoginServer

Code（错误码）	Description（描述）	Reason（详细实际原因）
其他错误码	透传后台错误码	例如签名问题等等

WBFaceErrorDomainGetInfoNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
33100	网络异常	获取活体类型/光线资源，网络异常（请求未到达后台）
33200	网络异常	获取活体类型/光线资源，后台返回参数有误（请求到达后台）

WBFaceErrorDomainNativeProcess

Code（错误码）	Description（描述）	Reason（详细实际原因）
43000	用户取消	回到后台/单击 home /左上角/上传时左上角取消
43001	无法获取唇语数据	获取数字活体的数字有问题
43002	权限异常，未获取权限	相机
43003	相机运行中出错	–
43004	视频录制中出错	不能存/启动失败/结束失败
43005	请勿晃动人脸,保持姿势	未获取到最佳图片
43006	视频大小不满足要求	视频大小不满足要求
43007	超时	预检测/动作活体
43008	检测中人脸移出框外	活体/数字/反光
43009	光线活体本地错误	–
43010	风险控制超出次数	用户重试太多次

43011	没有检测到读数声音	数字活体过程中没有发声
43012	初始化模型失败，请重试	初始化算法模型失败
43013	初始化sdk异常	WbCloudFaceVerifySdk 未被初始化
43014	简单模式本地加密失败	编码转换异常/加解密编码失败

WBFaceErrorDomainCompareNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
53100	网络异常	对比时，网络异常（请求未到达后台）
53200	网络异常	对比时，后台返回参数有误（请求到达后台）

WBFaceErrorDomainCompareServer

Code（错误码）	Description（描述）	Reason（详细实际原因）
其他错误码	透传后台错误码	–

接入示例

最近更新时间：2024-07-29 19:13:31

权威库网纹图片比对、自带对比源对比接入示例：

```
# 在 DemoPage 中单击某个按钮的代码逻辑：
//先填好数据
let inputData = new InputData(
      orderNo,
      this.appId,
      '1.0.0',
      this.nonce,
      this.userId,
      sign,
      this.licence,
      faceId
    )

    //设置必需参数
    let wbFaceVerifyConfig = new WbFaceVerifyConfig(inputData);
    //是否打开sdk日志开关
    wbFaceVerifyConfig.isEnableLog = true;

    WbCloudFaceVerifySdk.getInstance().initAdvSdk(getContext(), wbFaceVerifyConfig, {
      onLoginSuccess: () => {
        DemoLog.i(this.TAG, `onLoginSuccess:`)
        WbCloudFaceVerifySdk.getInstance().startWbFaceVerifySdk(getContext(), {
          onFinish: (_result: WbFaceVerifyResult) => {
            DemoLog.i(this.TAG, `WbCloudFaceVerifySdk onFinish`)
            //todo 处理刷脸结果
        ……
            //处理完后释放sdk
            //【特别注意】请在拿到sdk结果后对sdk进行释放，不要在页面结束时释放，避免未能获取刷脸回调结果的情况
            WbCloudFaceVerifySdk.getInstance().release();
          }
        })
      },
      onLoginFail: (error: WbFaceError) => {
        DemoLog.e(this.TAG, `onLoginFailed:JSON.stringify(error)`)
        //todo 处理登录错误逻辑
    ……
        //【特别注意】请在拿到sdk结果后对sdk进行释放，不要在页面结束时释放，避免未能获取刷脸回调结果的情况
        WbCloudFaceVerifySdk.getInstance().release();
      }
    })
```

人脸核身结果查询
查询核身结果

最近更新时间：2025-01-15 17:14:53

当您的用户完成核身认证后，如果您需要拉取人脸核身的视频和图片用于存证等其他需要，您可以调用查询核身结果接口来获取。
重要提示：
1. 您需要在前端完成刷脸的回调后，再来调用查询核身结果接口获取刷脸视频和照片。
2. 人脸核身完成后的相关业务数据请尽快拉取，超过人脸核身服务必须最短缓存时间的业务数据将完全清理。

> ⚠ 注意
> 当您的 App 接入的是我们的基础版或增强版 SDK 服务时，SDK 回调中只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过查询核身结果接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，查询核身结果接口无法查询到刷脸结果。

步骤如下：

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，获取方式请参见 获取 SIGN ticket 。
- 合作方为人脸核身服务生成签名，需要具有以下参数：

参数	说明	来源
appId	业务流程唯一标识，即WBappid	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，字母/数字组成的字符串，本次人脸核身合作伙伴上送的订单号，唯一标识	合作方自行分配，不要带有特殊字符
version	默认值：1.0.0	-
api ticket	合作伙伴服务端缓存的 ticket，注意是 SIGN 类型	获取方式请参见 获取 SIGN ticket
nonce	32位随机字符串，由字母和数字组成	合作方自行生成，不要带有特殊字符

基本步骤

1. 生成一个32位的随机字符串 nonce（由字母和数字组成，登录时也要用到）。
2. 将 appId、orderNo、version、api ticket、nonce 共5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 。

人脸核身结果查询接口(升级)

请求

- **请求 URL：**https://kyc1.qcloud.com/api/v2/base/queryfacerecord?orderNo=xxx

> ⚠ 注意
> 为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。

- **请求方法：**POST
- **报文格式：**Content-Type: application/json
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填

appId	人脸核身控制台申请的 WBappid	String	8	是
version	版本号，默认值：1.0.0	String	20	是
nonce	随机数	String	32	是
orderNo	订单号，字母/数字组成的字符串,是您需要查询结果的人脸核身订单号	String	32	是
sign	签名值，使用本页第一步生成的签名	String	40	是
getFile	是否需要获取人脸识别的视频和文件，值为1则返回视频和照片、值为2则返回照片、值为3则返回视频；其他则不返回	String	1	否
queryVersion	查询接口版本号(传1.0则返回 sdk 版本号和 trtc 标识)	String	8	否

响应

- **响应参数：**

参数	类型	说明
code	String	0：表示身份验证成功且认证为同一人
msg	String	返回结果描述
bizSeqNo	String	业务流水号
transactionTime	String	请求接口的时间
orderNo	String	订单编号
liveRate	String	活体检测得分
riskInfo	object	后台返回的刷脸风险信息
similarity	String	人脸比对得分
occurredTime	String	进行刷脸的时间
photo	Base 64 string	人脸核身时的照片，base64 位编码
video	Base 64 string	人脸核身时的视频，base64 位编码
sdkVersion	String	人脸核身时的 sdk 版本号
trtcFlag	String	Trtc 渠道刷脸则标识"Y"
appId	String	腾讯云控制台申请的 appid

- **响应示例：**


```
{"code":"0",
"msg":"请求成功",
"bizSeqNo":"22041520001184442415491408594474",
"result":{
  "orderNo":"testReflect1650008613761",
  "liveRate":"99",
  "riskInfo":{
  "deviceInfoLevel":"1",
  "deviceInfoTag":"",
  "riskInfoLevel":"",
  "riskInfoTag":""
```

```
  },
  "similarity":"97.0",
  "occurredTime":"20220415154341",
  "appId":"IDAXXXXX",
  "photo":"*",
  "sdkVersion":"v4.5.2.1",
  "video":"*",
  "bizSeqNo":"2204152000118444241549140859447 4"
  },
"transactionTime":"20220415154914"
}
```


code 非 0 时，有时不返回图片和视频。

注意事项

- 照片和视频信息作为存证，合作伙伴可以通过此接口拉取视频等文件，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，默认不返回视频和照片信息。
- 由于照片和视频信息有可能超过 1M，考虑传输的影响，建议合作伙伴在使用时注意，建议获取比对结果用于后续流程处理和存证使用分开调用。避免网络传输带来的影响。
- 如果合作方是完成人脸核身流程后马上去拉取视频/照片，建议在查询接口加上一个查询机制：判断图片是否存在，轮询3次，每次2s。
- 照片和视频均为 base64 位编码，其中照片解码后格式一般为 JPG 和 PNG。视频格式解码后一般为 MP4。
- 服务端验证结果接口返回66660011无此查询结果的可能原因：

  1.1 66660017 验证次数超限后被风控，风控后的订单为无效，查询结果为无此查询结果。

  1.2 用户中途退出刷脸，没有完成人脸验证的流程，没有比对，查询结果为无此查询结果。

  1.3 66660018 操作超时，请退出重试 无此 ID 的用户身份信息，H5faceid 5分钟有效期，失效后无法进入刷脸流程，查询结果为无此查询结果。

  1.4 66660016 视频格式或大小不合法 文件或视频不合法，无法进行比对，查询结果为无此查询结果。

  1.5 400604 上传的视频非实时录制,被时间戳校验拦截，查询结果为无此查询结果。

  1.6 查询超过3天的订单返回无此查询结果。
- 响应参数请勿做强校验，后续可能会有扩展。

RiskInfo 对象说明

RiskInfo 是用来给合作方传递刷脸风险信息的对象。其中 RiskInfo 对象的各个字段意义如下表所示：

字段名	类型	字段含义	说明
deviceInfoLevel	String	设备风险等级	设备风险级别，如果命中风险则显示
deviceInfoTag	String	设备风险标签	设备风险等级描述
riskInfoLevel	String	身份风险等级	如果命中身份风险则显示风险等级
riskInfoTag	String	身份风险标签	身份风险等级描述

详情见 增强版人脸核身服务错误码 风险标签说明。

核身结果-多张照片返回

最近更新时间：2024-06-17 17:09:41

人脸认证多张照片查询接口：获取人脸认证结果多张照片的接口。

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 获取 SIGN ticket 。
- **合作方为人脸验证服务生成签名，需要具有以下参数：**

参数	说明	来源
appId	业务流程唯一标识，即WBappid	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，字母/数字组成的字符串，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，不要带有特殊字符
version	默认值：1.0.0	-
api ticket	合作伙伴服务端缓存的 ticket，注意是 SIGN 类型	获取方式见 获取 SIGN ticket
nonce	32位随机字符串，字母和数字	合作方自行生成，不要带有特殊字符

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 appId、orderNo、version 连同 api ticket、nonce 共5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 。

人脸认证多张照片查询接口

请求

请求URL：https://kyc1.qcloud.com/api/v2/base/queryphotoinfo?orderNo=xxx

> ⚠ **注意**
> 为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。

请求方法：POST
HTTP 请求 header：

参数名	是否必选	类型	说明
Content-Type	是	String	application/json

请求参数：

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
version	版本号，默认值：1.0.0	String	20	是
nonce	随机数	String	32	是

orderNo	订单号，字母/数字组成的字符串，合作方订单的唯一标识	String	32	是
sign	签名值，使用本页第一步生成的签名	String	40	是

响应

返回参数说明：

参数名	类型	说明
code	int	0：成功 非0：失败
msg	String	请求结果描述
bizSeqNo	String	请求业务流水号
orderNo	String	订单编号
occurredTime	String	刷脸时间（yyyyMMddHHmmss）
photoList	List	Base64 图像列表（返回1-3张照片），若照片不存在，则返回 null

返回示例：

```
{
"code": 0,
"msg": "请求成功",
"bizSeqNo": "业务流水号",
"result": {
"orderNo": "AAAAAA001",
"occurredTime": "20180907142653",
"photoList": ["第一个base64photo字符串", "第二个base64photo字符串", "第三个base64photo字符串"]
}
}
```

登录鉴权
获取 Access Token

最近更新时间：2025-01-07 10:21:42

注意事项

- 所有场景默认采用 UTF-8 编码。
- **Access Token 必须缓存在磁盘，并定时刷新，且不能并发刷新，建议每20分钟请求新的 Access Token，获取之后立即使用最新的 Access Token。旧的只有一分钟的并存期。**
- **如果未按照上述做定时刷新，可能导致鉴权不通过，影响人脸服务正常调用。**
- **每次用户登录时必须重新获取 NONCE ticket。**

请求

- **请求 URL：** `https://kyc1.qcloud.com/api/oauth2/access_token`
- **请求方法：** `GET`
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
secret	WBappid 对应的密钥，申请 WBappid 时得到，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	64	是
grant_type	授权类型，默认值为：client_credential（必须小写）	String	20	是
version	版本号，默认值为：1.0.0	String	20	是

- **请求示例**

```
https://kyc1.qcloud.com/api/oauth2/access_token?
appId=xxx&secret=xxx&grant_type=client_credential&version=1.0.0
```

响应

响应示例：

```
{
"code":"0",
"msg":" 请求成功 ",
"transactionTime":"20151022043831",
"access_token":"accessToken_string",
"expire_time":"20151022043831",
"expire_in":"7200"
}
```

注意

1. code 不为0则表示获取失败，可以根据 code 和 msg 字段进行定位和调试，code 详情请参见 增强版人脸核身服务错误码 。
2. expire_in 为 access_token 的最大生存时间，单位秒，合作伙伴在 判定有效期时以此为准。
3. expire_time 为 access_token 失效的绝对时间，由于各服务器时间差异，不能使用作为有效期的判定依据，只展示使用。
4. 修改 secret 之后，该appId 生成的 access_token 和 ticket 都失效。

获取 SIGN ticket

最近更新时间：2025-02-11 15:52:52

注意事项

- **前置条件：**请合作方确保 Access Token 已经正常获取，获取方式见 Access Token 获取 。
- **用途：**SIGN ticket 是合作方**后台服务端业务请求**生成签名鉴权参数之一，用于后台查询验证结果、调用其他业务服务等。
- API ticket 的 SIGN 类型，必须缓存在磁盘，并定时刷新，刷新的机制如下：
- **由于 API ticket 的生命周期依赖于 Access Token，为了简单方便，建议 API ticket 的刷新机制与 Access Token 定时机制原理一致，建议按照每20分钟和 Access Token 绑定定时刷新，且不能并发刷新。**
- **获取新的之后立即使用最新的，旧的有一分钟的并存期。**
- **如果未按照上述做定时刷新，可能导致鉴权不通过，影响人脸服务正常调用。**

请求

- **请求 URL：** `https://kyc1.qcloud.com/api/oauth2/api_ticket`
- **请求方法：** `GET`
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
access_token	请根据 获取 Access Token 指引进行获取	String	64	是
type	ticket 类型，默认值：SIGN（必须大写）	String	20	是
version	版本号，默认值：1.0.0	String	20	是

- **请求示例：**

```
https://kyc1.qcloud.com/api/oauth2/api_ticket?appId=xxx&access_token=xxx&type=SIGN&version=1.0.0
```

响应

响应示例：

```
{
  "code":"0",
  "msg":" 请求成功 ",
  "transactionTime":"20151022044027",
  "tickets":[
{
  "value":"ticket_string",
  "expire_in":"3600",
  "expire_time":"20151022044027"}
]
}
```

注意

1. code 不为 0 则表示获取失败，可以根据 code 和 msg 字段进行定位和调试。code 详情请参见 增强版人脸核身服务错误码 。
2. expire_in 为 SIGN ticket 的最大生存时间，单位秒，合作伙伴在 判定有效期时以此为准。
3. expire_time 为 SIGN ticket 失效的绝对时间，由于各服务器时间差异，不能使用作为有效期的判定依据，只展示使用。
4. access_token 失效时，该 access_token 生成的 ticket 都失效。

5. tickets 只有一个。

获取 NONCE ticket

最近更新时间：2025-01-15 17:14:53

注意事项

- **前置条件：**确保 Access Token 已经正常获取，获取方式见 Access Token 获取 。
- **用途：** NONCE ticket 是合作方 前端包含 App 和 H5 等 生成签名鉴权参数之一，启动 H5 或 SDK 人脸核身。
- API ticket 的 NONCE 类型，其有效期为120秒，且一次性有效, 即每次启动 SDK 刷脸都要重新请求 NONCE ticket。

请求

- **请求 URL：** `https://kyc1.qcloud.com/api/oauth2/api_ticket`
- **请求方法：** `GET`
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
access_token	请根据 Access Token 获取 指引进行获取	String	64	是
type	ticket 类型，默认值：NONCE（必须大写）	String	20	是
version	版本号	String	20	是
user_id	当前使用用户的唯一标识，需合作伙伴自行定义注意：合作伙伴必须保证 user_id 的全局唯一性，不要带有特殊字符	String	32	是

- **请求示例：**

```
https://kyc1.qcloud.com/api/oauth2/api_ticket?
appId=xxx&access_token=xxx&type=NONCE&version=1.0.0&user_id=xxx
```

响应

响应示例：

```
{
  "code":"0",
  "msg":" 请求成功 ",
  "transactionTime":"20151022044027",
  "tickets":[
{
  "value":"ticket_string",
  "expire_in":"120",
  "expire_time":"20151022044027"}
]
}
```

注意

1. code 不为 0 则表示获取失败，可以根据 code 和 msg 字段定位和调试。code 详情请参见 增强版人脸核身服务错误码 。
2. expire_in 为 NONCE ticket 的最大生存时间，单位秒，合作伙伴在**判定有效期时以此为准**。
3. expire_time 为 NONCE ticket 失效的绝对时间，由于各服务器时间差异，不能使用作为有效期的判定依据，只展示使用。
4. access_token 失效时，该 access_token 生成的 ticket 都失效。

签名算法说明

最近更新时间：2022-07-05 17:37:29

Java 签名算法

以下是生成签名算法，合作伙伴可以直接使用。

```
public static String sign(List<String> values, String ticket) { //values传ticket外的其他参数
if (values == null) {
throw new NullPointerException("values is null");
}
values.removeAll(Collections.singleton(null));// remove null
values.add(ticket); java.util.Collections.sort(values);
StringBuilder sb = new StringBuilder();
for (String s : values) { sb.append(s);
}
return Hashing.sha1().hashString(sb,
Charsets.UTF_8).toString().toUpperCase();
}
```

> **注意**
> Hashing 使用的是 com.google.common.hash.Hashing，版本是 guava-18.0。

调用方法

获取 token 和 ticket 之后再用 ticket 对数据进行签名，相关代码如下所示：

```
String accessToken = client.getAccessToken();//http 请求
String ticket = client.getTicket(accessToken);//http 请求
String sign = SignUtils.sign(data, ticket);
```

微信小程序
微信小程序接入指引

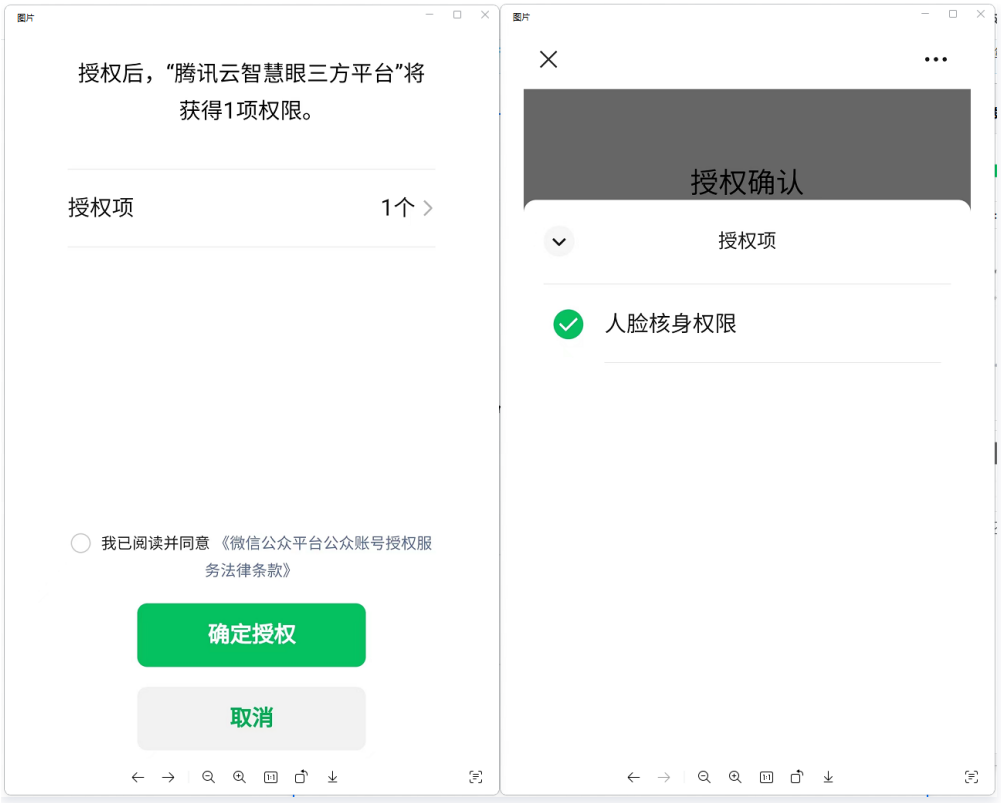
最近更新时间：2025-03-14 15:33:12

小程序在微信使用场景中越来越广泛，腾讯云人脸核身针对小程序提供嵌入式 SDK，开发人员可以将相应的 SDK 添加到小程序开发工具中，从而直接调用小程序 SDK 中提供的 OCR 识别、活体检测和 1：1人脸比对服务。
基于微信小程序原生体验，人脸核身微信小程序接入渠道提供一闪活体检测模式，要求符合以下行业要求且具备相关资质的客户才能申请，接入前请确认公众号的主体和类目是否符合以下范围，并准备好对应的资质文件，资质审核需要5 – 7个工作日，请预留好上线时间。所需对应的资质文件请参见 微信小程序资质文件列表 。

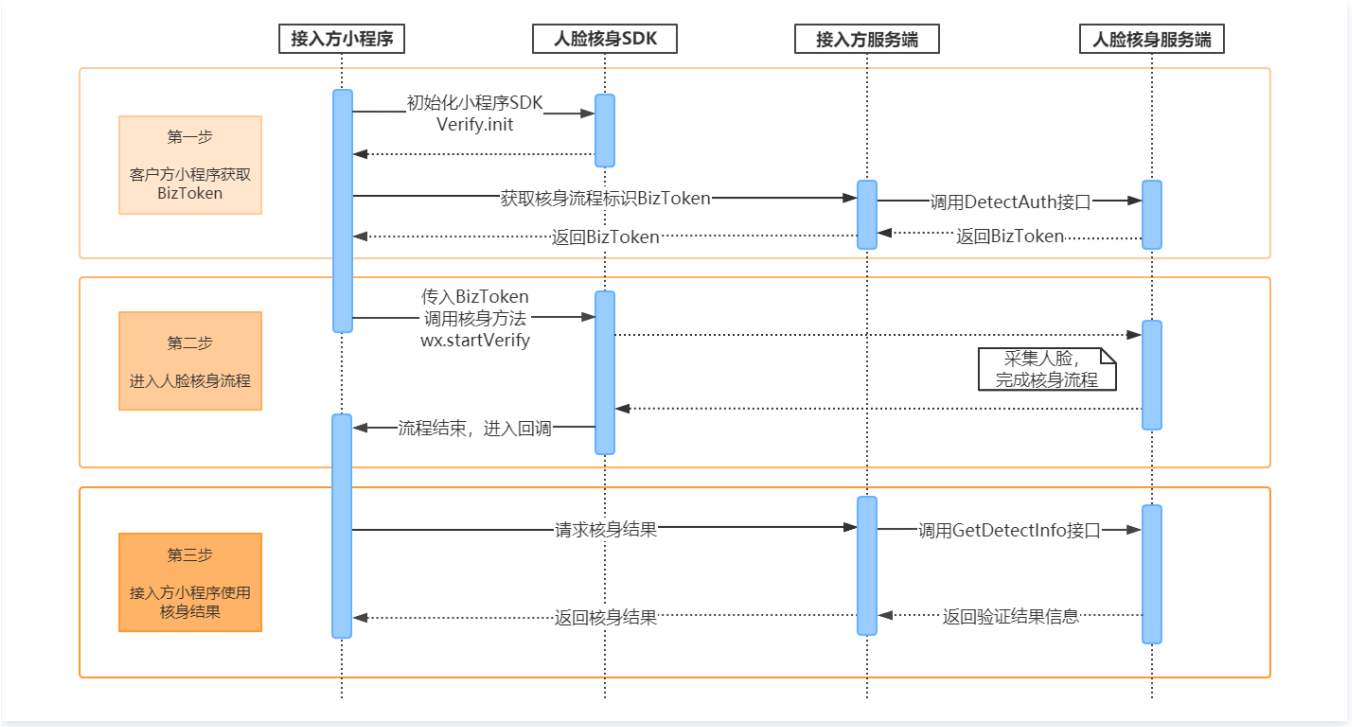
- 政务：机构或事业单位。
- 金融：银行、保险、信托、基金、证券/期货、持牌消费金融、非金融机构自营小额贷款、汽车金融/金融租赁。
- 医疗：公立医疗机构、互联网医院、三级私立医疗机构。
- 运营商：基础电信运营商、虚拟运营商（含转售移动通信）。
- 教育：学历教育（学校）、公立学校、私立学校。
- 出行与交通：网约车（快车/出租车/专车/其他网约车）、航空、公交/地铁、水运、骑车、火车/高铁/动车、长途汽车、租车、高速服务。
- 生活服务：生活缴费。
- 旅游：酒店。
- 商业服务：公证。
- 社交：直播。
- 物流：收件/派件、货物运输。

说明
本模式需要先扫码授权：打开二维码 ，小程序管理员扫码后，单击自定义权限，只勾选人脸核身权限，将该权限授权给慧眼第三方平台。

授权指引



接入流程



1. 登录 人脸核身控制台，单击**自助接入>创建业务流程**，完成**微信H5/小程序服务**的业务流程创建，获取 RuleID 和小程序 SDK 下载链接。
2. 下载小程序 SDK，并在小程序代码中引入，调用 init 方法进行初始化。
3. 接入方服务端调用实名核身鉴权 DetectAuth 接口，获取到核身流程标识(BizToken)。
4. 客户后端将 BizToken 返回给客户小程序端，然后小程序调用核身方法 startVerify 进入核身流程。
5. 用户完成人脸核身后，会以回调函数形式返回 BizToken ，接入方小程序将 BizToken 传给接入方服务端，接入方服务端即可凭借 BizToken 参数调用获取实名核身结果信息增强版 GetDetectInfoEnhanced 接口去获取本次核身的详细信息，最后将核身结果返回给接入方小程序。

SDK 接入

开发准备

下载 SDK

登录 人脸核身控制台 下载小程序 SDK，并在小程序代码中引入，调用 init 方法进行初始化。

安装 SDK

将小程序 SDK 文件夹放在小程序根目录下，使用 require 函数引入。

```
const Verify = require('/verify_mpsdk/main');
```

调试 SDK

请在微信开发者工具中使用手机“预览”模式进行调试，请勿使用“真机调试”。

卸载 SDK

卸载时删除 `verify_mpsdk` 文件夹，移除相应 require 代码即可。

快速入门

1. 将 verify_mpsdk 文件夹放到小程序项目根目录。
2. 初始化慧眼实名核身 SDK。
   在 App.js 的 onLaunch() 中加入相应代码，在 App.json 文件里添加活体验证页面 `verify_mpsdk/index/index` 。

```
//app.js
App({
 onLaunch: function () {
     // 初始化慧眼实名核身组件
     const Verify = require('/verify_mpsdk/main');
     Verify.init();
 }
})
// app.json
{
 "pages":[
     "verify_mpsdk/index/index"
 ]
}
```

3. 调用 SDK 功能函数 wx.startVerify()。
   在需要实名认证的地方调用 wx.startVerify() 进入实名认证页面，认证完成会触发对应的回调函数。

```
// 单击某个按钮时，触发该函数
gotoVerify: function () {
 let BizToken = getBizToken();// 该函数为客户自定义函数，去客户后端调用 DetectAuth 接口获取 BizToken
 // 调用实名核身功能
 wx.startVerify({
     data: {
         token: BizToken // BizToken
     },
     success: (res) => { // 验证成功后触发
         // res 包含验证成功的token，这里需要加500ms延时，防止iOS下不执行后面的逻辑
         setTimeout(() => {
             // 验证成功后，拿到token后的逻辑处理，具体以客户自身逻辑为准
         }, 500);
     },
     fail: (err) => {  // 验证失败时触发
         // err 包含错误码，错误信息，弹窗提示错误
         setTimeout(() => {
             wx.showModal({
                 title: "提示",
                 content: err.ErrorMsg,
```

```
                showCancel: false
            })
        }, 500);
    }
 });
}
```

4. 添加域名服务器白名单。

   您需要在小程序上线前进入：微信公众号管理平台 > 管理 > 开发管理 > 开发设置 > 服务器域名将以下域名添加至白名单，小程序前端接口请求有域名白名单限制，未添加白名单的域名只能在调试模式下运行。

```
// request 合法域名、uploadFile 合法域名、downloadFile 合法域名这三种都要添加
faceid.qq.com、faceid.qcloud.com
// socket合法域名 （v1.0.20及以上版本需要添加以下socket域名）
wss://faceid.qq.com
// v1.0.17及以上版本身份校验环节如需NFC方式读取证件，需要添加以下socket域名
wss://idcloudread.eidlink.com
```

基本 API 描述

- `Verify.init(options)`：初始化插件。
  - `options`：Object required 初始化的参数。
- `wx.startVerify(options)`：进入实名认证页面。
  - `options`：Object required 初始化的参数。
  - `options.data.token`：String required 客户后端调用 DetectAuth 接口获取的 BizToken。
  - `options.success`：Function(res) required 验证成功的回调。res 包含验证成功的 token。
  - `options.fail`：Function(err) required 验证失败的回调。err 包含错误码、错误信息。

操作步骤

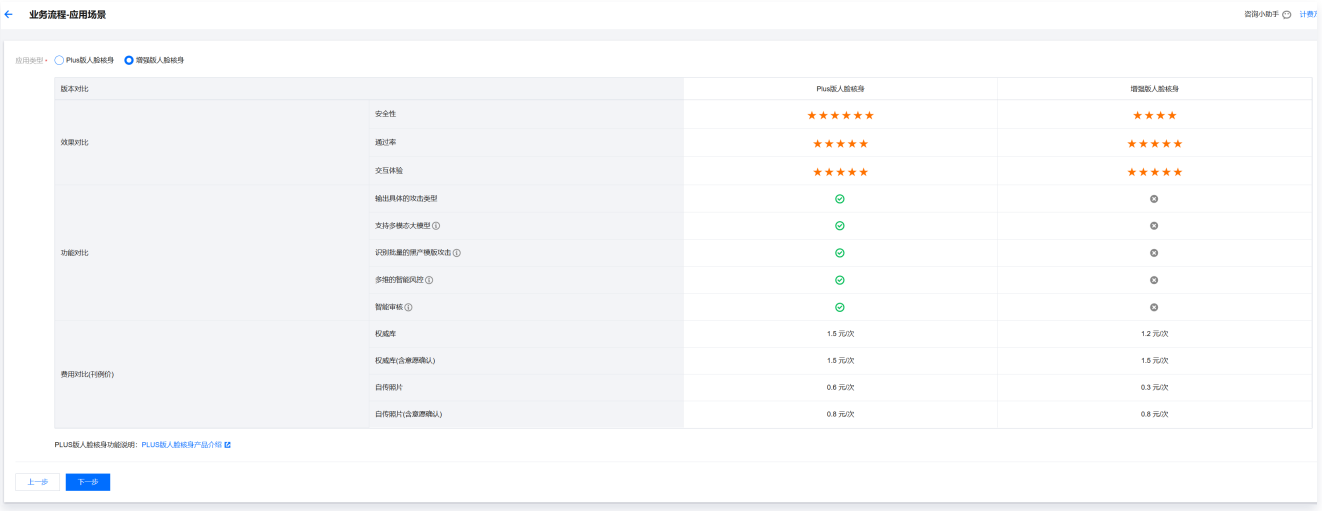
步骤1：创建 RuleId

RuleId 用于调用配置的业务流程，创建步骤如下：

1. 登录腾讯云人脸核身控制台，在 自助接入 页面，单击**创建业务流程**。

2. 选择应用场景：**微信小程序**，并根据需求填写相关信息，填写完成后单击**下一步**。**微信小程序**需要上传对应的资质文件，我们会在3 – 5个工作日完成审核。

3. 选择应用类型：选择**增强版人脸核身**，然后点击**下一步**。

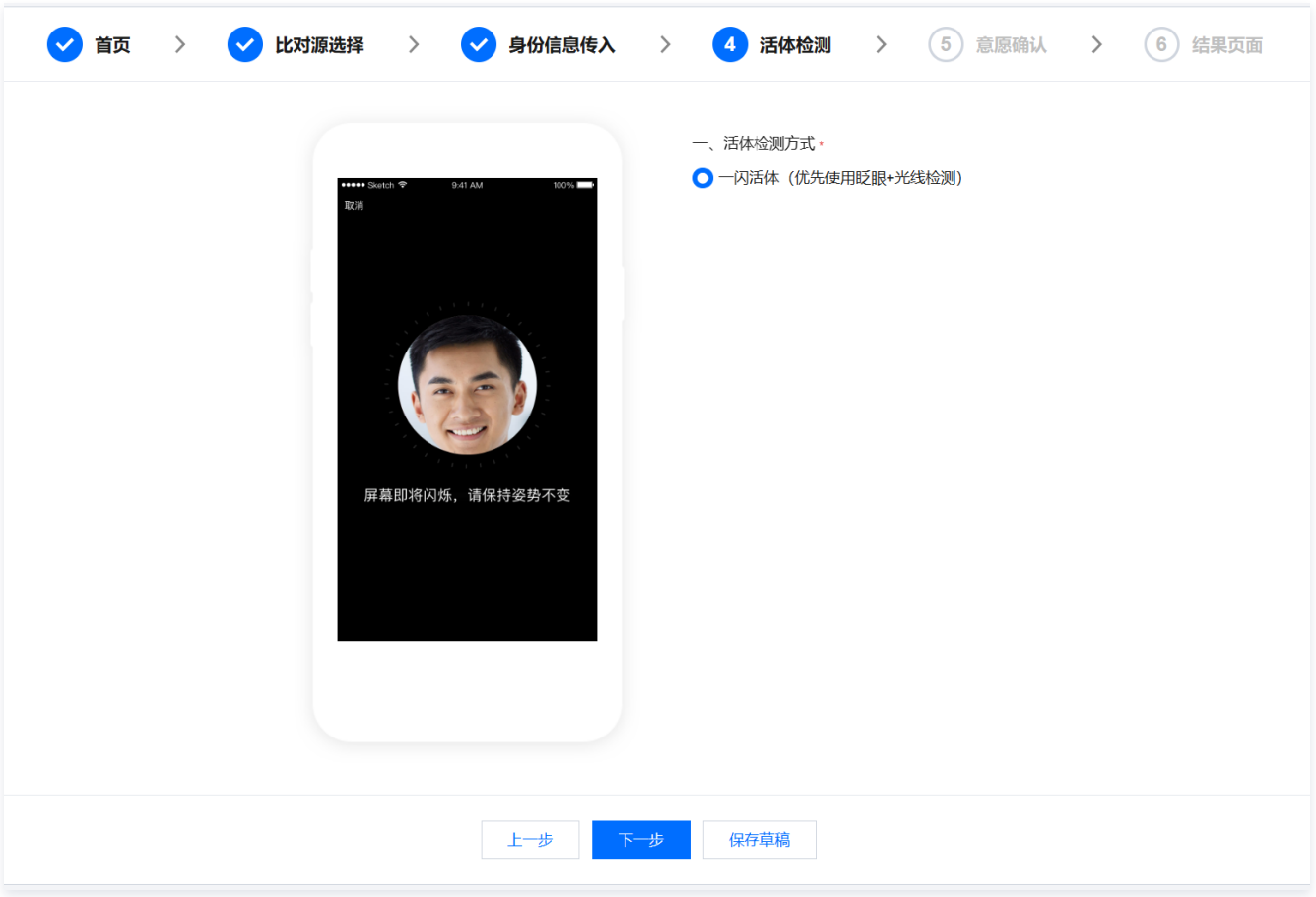


4. 进入接入配置，根据您业务的实际场景填写页面标题、业务名称、业务描述信息、业务联系方式后单击下一步。

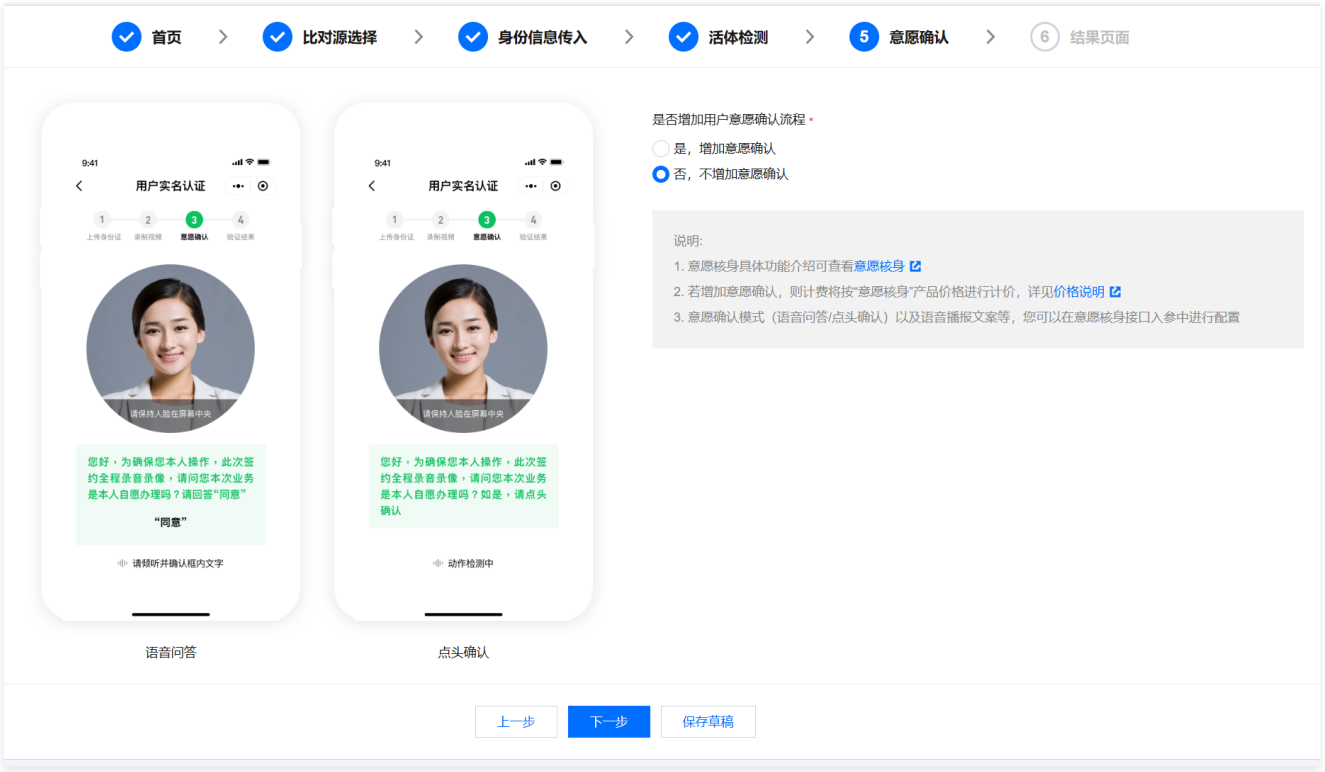
5. 选择人脸比对库源。人脸核身支持两种方式：**跟权威库比对**和**跟上传照片比对**。
   - 其中**跟权威库比对**的收费价格为：增强版人脸核身（权威库）- 微信的价格。
   - **跟上传照片比对**的收费价格为：增强版人脸核身（自传照片）- 微信的价格。OCR 不再单独收费。

6. 配置身份证 OCR 功能，如果不需要则勾选“**不需要用户在验证时上传**”，然后单击**下一步**。

7. 配置活体检测方式，勾选后单击下一步。



8. 配置意愿页面，根据您的业务情况勾选：是否增加用户意愿确认流程，然后单击**下一步**。请注意，增强版人脸核身默认是不增加意愿确认流程的，如配置了增加意愿确认，则将按**意愿核身**价格计费，详情请参见 价格说明 。



9. 配置结果页的文案描述，然后单击**下一步**。

10. 业务信息填写完成后，确认您的配置信息和计费信息，然后单击**确认并提交审核**。

11. 流程审核通过后，系统会自动创建流程并分配业务 ID（RuleId）

步骤2：获取 BizToken

完成 RuleId 创建后，需获取 BizToken，用于调用您配置的人脸核身验证的流程。
调用 实名核身鉴权接口，传入 步骤1 生成的 RuleId，和认证结束后重定向的回调链接地址 RedirectUrl，得到核验流程唯一密钥（BizToken）和用于发起核身流程的 URL。

- 在线调试

> **注意：**
> BizToken 是仅一次核身流程的标识，有效时间为7,200秒；用户完成核身后，开发者可用该标识获取验证结果信息，结果信息仅保留3天。
> 如果输入参数 RedirectUrl 为空，则用户完成核验后默认跳转腾讯云人脸核身产品介绍页。

步骤3： 回调小程序SDK

请参考该文档的 快速入门 章节， 回调到小程序 SDK 后，用户进入核身流程，完成身份证拍摄识别、活体检测等操作完成身份核验。

步骤4： 查询核验结果信息

用户完成人脸核身后，页面会跳转到 wx.startVerify 的回调地址（该回调地址是业务方自定义的绝对路径地址）之后，您在服务端即可凭借 BizToken 参数调用 获取实名核身结果信息 接口去获取本次核身的详细信息。获取到用户验证过程数据，包括文本信息、识别分数、照片和视频。也可以通过访问 腾讯云人脸核身控制台 查看服务调用情况。

> **说明：**
> wx.startVerify 的回调地址请参考：该文档 快速入门 章节的第3小节示例中对success和fail 回调的处理方法。

> **注意：**
> - 为了保证用户的隐私，结果信息人脸核身侧仅保留3天，请您及时拉取。
> - 在开发、产品使用、费用、合同等问题有任何疑问，欢迎您 联系我们 进行询问。

微信小程序 uni-app 接入流程

最近更新时间：2025-04-25 17:43:22

接入准备

授权指引及接入准备参见 微信小程序接入指引 。
小程序前端接口请求有域名白名单限制，如果不添加只能再调试模式下运行，上线前需要将如下两个域名在小程序后台添加服务器域名。

```
https://faceid.qq.com;
https://faceid.qcloud.com;
```

uni-app接入

步骤一：注册并创建 uni-app 开发环境

uni-app 开发接入具体参照 uni官网 。

步骤二：下载 SDK

控制台下载最新版本的 SDK。

步骤三：并配置verify_mpsdk

本 SDK 仅支持 uni 微信小程序端。

1. 将 verify-mpsdk 文件夹拷贝到项目根目录的 wxcomponents 文件夹下。
2. 创建一个空页面调用组件。

   pages.json 注册组件地址，如pages/verify/index 。

```
{
  "pages": [
    ...
    {
      "path": "pages/verify/index",
      "style": {
        "navigationBarTitleText": "uni-app",
        "usingComponents": {
              "verify-mp": "/wxcomponents/verify_mpsdk/indexCom"
          }
      }
    }
  ]
}
```

在注册的地址生成小程序sdk页，调用verify-mp组件。

> **注意：**
> 该页面为空内容组件调用页，不需要其他逻辑代码。

```
<template>
    <verify-mp></verify-mp>
</template>

<script>
export default {
  data() {
    return {   }
  },
  onLoad() {},
  methods: {},
}
```

```
</script>
```

3. **初始化。**

- **方法一：可以在 App.vue 中全局初始化。**

```
export default {
  onLaunch: function() {
    const verify = require('/wxcomponents/verify_mpsdk/main');
    verify.init();
  },
};
```

- **方法二：在需要调用到的页面方法之前初识化即可。**

4. **调用 startVerify。**

> ⚠ **注意：**
> startVerify 调用需要在 init 初始化之后。

```
// 业务发起调用页面
  wx.startVerify({
   data: {
     token: '', // 必要参数，BizToken
     startPath: 'pages/verify/index' // 必要参数，配置了verify核身组件的页面地址
   },
   success: (res) => { // 验证成功后触发
     // res 包含验证成功的token，这里需要加500ms延时，防止iOS下不执行后面的逻辑
     // 验证成功后，拿到token后的逻辑处理，具体以客户自身逻辑为准
   },
   fail: (err) => {  // 验证失败时触发
     // err 包含错误码，错误信息，弹窗提示错误
   }
  });
```

基本 API 描述

- Verify.init(options)：初始化插件。
  - options：Object required 初始化的参数。
- wx.startVerify(options)：进入实名认证页面。
  - options：Object required 初始化的参数。
  - options.data.startPath：组件初始化页面，必填。
  - options.data.token：String required 客户后端调用 DetectAuth 接口获取的 BizToken。
  - options.success：Function(res) required 验证成功的回调。res 包含验证成功的 token。
  - options.fail：Function(err) required 验证失败的回调。err 包含错误码、错误信息。

获取实名核身结果信息

用户完成人脸核身后，页面会跳转到 RedirectUrl 上，地址中会带上此次验证流程使用的 BizToken，您在服务端即可凭借 BizToken 参数调用 获取实名核身结果信息 接口去获取本次核身的详细信息。获取到用户验证过程数据，包括文本信息、识别分数、照片和视频。也可以通过访问 腾讯云人脸核身控制台 查看服务调用情况。

卸载 SDK

卸载时删除 verify_mpsdk 文件夹，移除相应代码即可。

示例 demo

示例 demo 下载 。

微信小程序资质文件列表

最近更新时间：2025-03-25 10:14:32

微信小程序是基于微信原生的体验，客户体验好，但有明确的主体行业限制。
接入前请确认小程序的主体和类目是否符合以下范围，并准备好对应的资质文件，资质审核需要5 – 7个工作日，请预留好上线时间。

微信小程序的主体类目

微信小程序支持行业类目及资质文件要求如下：

小程序一级类目	小程序二级类目	小程序三级类目	使用人脸核验接口所需资质
快递与邮政	寄件/收件	/	《快递业务经营许可证》
物流服务	货物运输	/	《道路运输经营许可证》（经营范围需含网络货运）
教育	学历教育（学校）	/	（2选1）： 1. 公立学校：由教育行政部门出具的审批设立证明 或《事业单位法人证书》 2. 私立学校：《民办学校办学许可证》与《民办非企业单位登记证书》
医疗	公立医疗机构	/	《医疗机构执业许可证》与《事业单位法人证书》
	互联网医院	/	仅支持公立医疗机构互联网医院（2选1）： 1. 卫生健康部门的《设置医疗机构批准书》 2.《医疗机构执业许可证》（范围均需含“互联网诊疗”或名称含“互联网医院”等相关内容）
医疗服务	三级私立医疗机构	/	仅支持三级以上私立医疗机构，提供《医疗机构执业许可证》、《营业执照》及《医院等级证书》
政务民生	所有二级类目	/	仅支持政府/事业单位，提供《组织机构代码证》或《统一社会信用代码证》
金融业	银行	/	（2选1）： 1.《金融许可证》 2.《金融机构许可证》
	保险	/	（8选1）： 1.《保险公司法人许可证》 2.《经营保险业务许可证》 3.《保险营销服务许可证》 4.《保险中介许可证》 5.《经营保险经纪业务许可证》 6.《经营保险公估业务许可证》 7.《经营保险资产管理业务许可证》 8.《保险兼业代理业务许可证》
	信托	/	（2选1）： 1.《金融许可证》 2.《金融机构许可证》
	基金	公募基金	（4选1）： 1.《经营证券期货业务许可证》且业务范围必须包含“基金” 2.《基金托管业务许可证》 3.《基金销售业务资格证书》 4.《基金管理资格证书》
	证券/期货	/	《经营证券期货业务许可证》
	消费金融	/	银监会核准开业的审批文件与《金融许可证》与《营业执照》

	非金融机构自营小额贷款	/	仅支持省金融办监管的网络小贷主体，同时提供： 1.《小额贷款公司经营许可证》或《小额贷款机构经营许可证》 2. 申请主体资质承诺函
	汽车金融	/	仅支持汽车金融主体，同时提供： 1.《营业执照》（公司名称包含“汽车金融”；营业范围包含“汽车金融”业务） 2.《金融许可证》或银保监会及其派出机构颁发的开业核准批复文件
交通服务	打车(网约车)	快车/出租车/专车/其他网约车	自营性网约车：提供《网络预约出租汽车经营许可证》 网约车平台：提供与网约车公司的合作协议以及合作网约车公司的《网络预约出租汽车经营许可证》
	航空	/	航司：提供《公共航空运输企业经营许可证》 机场：提供《民用机场使用许可证》或《运输机场使用许可证》
	地铁	/	提供地铁公司《营业执照》
	水运	/	船企：提供《水路运输许可证》 港口：提供《港口经营许可证》
	骑车	/	仅支持共享单车，提供共享单车公司《营业执照》
	火车/高铁/动车	/	仅支持铁路局/公司官方，提供铁路局/公司《营业执照》
	公交	公交公司	提供公交公司《营业执照》
	长途汽车	/	（2选1）： 1.《道路运输经营许可证》（经营范围需含客运） 2. 官方指定联网售票平台（授权或协议或公开可查询文件）
	租车	/	运营公司提供《备案证明》与对应公司《营业执照》，且营业执照中包含汽车租赁业务
	高速服务	/	仅支持 ETC 发行业务，（2选1）： 1. 事业单位主体，需提供《事业单位法人证书》 2. 官方指定的发行单位（一发单位），需提供“官方授权或协议，或公开可查询的文件”
生活服务	生活缴费	/	（供电类）提供《电力业务许可证》与《营业执照》，且《营业执照》经营范围含供电。（燃气类）提供《燃气经营许可证》与《营业执照》，且《营业执照》且经营范围含供气。（供水类）提供《卫生许可证》与《营业执照》。
IT 科技	基础电信运营商	/	（2选1）： 1. 基础电信运营商：提供《基础电信业务经营许可证》 2. 运营商分/子公司：提供营业执照（含相关业务范围）
	转售移动通信	/	仅支持虚拟运营商，提供《增值电信业务许可证》（业务种类需含通过转售方式提供移动通信业务）
旅游服务	住宿服务	/	仅支持酒店，提供《酒店业特种行业经营许可证》
商业服务	公证	/	仅支持公证处，提供《公证处执业许可证》
社交	直播	/	（2选1）： 1.《信息网络传播视听节目许可证》 2.《网络文化经营许可证》（经营范围含网络表演）

人脸核身对账指引

最近更新时间：2025-01-15 17:14:53

本文将为您描述使用腾讯云慧眼人脸核身服务中的微信渠道产品时（如使用人脸核身微信小程序、微信原生 H5、微信普通 H5 的人脸核身 SaaS 服务），如何与慧眼人脸核身进行对账操作。

对账工作主要分为2部分：

1. 收集使用慧眼业务时的明细信息。
2. 根据明细信息计算出账单。

下文将为您提供两种方式进行收集明细和计算账单。

方式1

在业务流程中保存明细

用户完成核验后，接入方调用 DetectAuth 接口生成的 `BizToken` 是一次核身流程的流水号。在业务完成后，接入方需要调用 GetDetectInfoEnhanced 接口来获取 `DetectInfo`，`DetectInfo` 就是描述本次核身的详细信息，包括核身结果，收费详情，照片视频信息等。**一个 BizToken 对应且只对应一个 DetectInfo。**

收集完整明细信息的关键点在于：

1. 标记 `BizToken`，确保不遗漏任何一个 `BizToken`。
   在调用 DetectAuth 接口获取 `BizToken` 后，必须将其存储起来，并标记该 `BizToken` 是否已经拉取到 `DetectInfo`。
2. 建立机制，确保每一个 `BizToken` 都能拉取到 `DetectInfo`。
   一般情况下慧眼侧是建议接入方在自己的回调接口中（DetectAuth 接口传入的 `RedirectUrl` 参数）调用 GetDetectInfoEnhanced 接口 去获取 `DetectInfo`。但由于用户侧核身场景环境比较复杂，并不能保证每一次的核身流程都能回调到 `RedirectUrl`。所以，接入方需要建立一个机制，来检查出未拉取 `DetectInfo` 的 `BizToken`，并重新拉取一遍。
3. 确保拉取到完成状态时的 `DetectInfo`。
   在人脸核身业务中，每个 `BizToken` 都有自己的状态，当 `BizToken` 到达完成状态时，所有的接口均不允许调用，其对应的 `DetectInfo` 也不能再更新。也就是说，若 `BizToken` 未达到完成状态的情况下，用户有可能使用此 `BizToken` 进行重试。未达到完成状态的 `BizToken` 都须重新拉取 `DetectInfo` 来确保拉取到的 `DetectInfo` 是完成状态时的。
   满足以下条件中的任意一个，即可认为 `BizToken` 处在完成状态：
   - GetDetectInfoEnhanced 接口返回的 `Response.Text.ErrCode` 为0。
   - 距离 Token 生成时间已超过2小时。

> **注意**
> 慧眼人脸核身支持配置最大可重试次数，该次数仅影响活体一比一的次数，并不影响 OCR、短信等接口的调用。

计算账单

`DetectInfo` 中与计费相关的信息全部都位于 `Response.Text.LivenessDetail` 数组中，该数组存放的是该 `BizToken` 进行的活体一比一的次数与每次活体一比一的详情。**遍历该数组，统计 `IsNeedCharge` 为 `true` 的个数即是此 BizToken 的收费次数。**

> **注意**
> `Response.Text.ErrCode` 是用于判断本次核验是否通过的字段，不用于判断计费与否。

为方便对账，建议将 `LivenessDetail` 数组中的每一项均保存下来。

```
let totalCount = 0; // 活体一比一的总记录数
let chargeCount = 0; // 活体一比一的收费记录数
let GetDetectInfoResponse = GetDetectInfoEnhanced(xxx, xxx, ...); // 调用GetDetectInfoEnhanced接口获取DetectInfo
for (let detail of GetDetectInfoResponse.Text.LivenessDetail){
    totalCount = totalCount + 1;
    if (detail.IsNeedCharge == true) {
        chargeCount = chargeCount + 1; // 当IsNeedCharge为true时，收费记录数加1
    }
    saveChargeDetailToDB(detail); // 保存至数据库，以便日后查看
}
```

方式2

使用 GetWeChatBillDetails 接口每日拉取

为了方便接入方更快捷的获取明细信息，慧眼人脸核身提供了一个接口（GetWeChatBillDetails）用于拉取每日的明细信息。该接口可以拉取子账号 + RuleId 维度的某一天的所有 token 数据。
具体使用方式如下：
1. 建立定时任务，每日早上6点后开始执行。**早于6点调用 GetWeChatBillDetails 接口会报参数异常。**
2. 调用 GetWeChatBillDetails 接口，拉取前一天所有的验证明细信息。

```
    let WeChatBillDetails = []; // 当日全量的账单详情
    let cursor = 0; // 游标位，拉取第一页时传0；剩下页面根据上一次请求回包的结果传入。
    let hasNextPage = true; // 是否还有下一页的标志位
    while (hasNextPage) { // 如果还有下一页，则继续拉取
        const result = GetWeChatBillDetails(Date /* 传入某一天日期*/, RuleId /* 传入RuleId*/, cursor); // 调用
GetWeChatBillDetails接口，并获取返回。
        WeChatBillDetails = WeChatBillDetails.concat(result.WeChatBillDetails); // 将拉取到的数据合并到全量的数组中
        cursor = result.NextCursor; // 将回包中返回的下一页的游标赋值给游标位，准备下一次拉取
        hasNextPage = result.HasNextPage; // 将是否还有下一页的标志位重新赋值，用于决定是否进行下一次拉取
    }
    // 此时WeChatBillDetails中就是某一天全量的账单信息
```

计算账单

统计 WeChatBillDetail 数组中的 ChargeCount 总数，即为当日计费总数。同时记录 ChargeDetail 内容，可用于校验：ChargeCount 值 = ChargeDetail 中 IsNeedCharge 为 true 的条目总数 = ErrorCode 为收费错误码的条目总数。

```
let totalCount = 0; // 活体一比一的总记录数
let chargeCount = 0; // 活体一比一的收费记录数
for (let WeChatBillDetail of WeChatBillDetails) { // 遍历WeChatBillDetails数组
    let ChargeDetails = WeChatBillDetail.ChargeDetails; // 取计费详情属性
    totalCount += ChargeDetails.length; // 总记录数为计费详情的总条目数
    chargeCount += WeChatBillDetail.ChargeCount; // 收费记录数为ChargeCount的总和
    for (let ChargeDetail of ChargeDetails) {
        saveChargeDetailToDB(ChargeDetail); // 遍历ChargeDetails数组，保存至数据库，以便日后查看
    }
}
```

常见问题

1. **什么时候进行计费上报?**
   计费上报时机是以**一比一请求时间为准**，即 `Response.Text.LivenessDetail[].ReqTime` 、GetWeChatBillDetails 接口中的 `ChargeDetail.ReqTime` 。例如，BizToken 是在7月1日的23:59:59生成的，但是进行比对的时机是在7月2日的0点，那一比一的请求时间也会落在7月2日的0点，同时这次调用也会上报至7月2日的账单中。
2. **LivenessDetail 中有些字段为 null 是什么原因?**
   当本次核身没有进行一比一或者其他原因造成一比一失败时，有可能导致 ReqTime、Seq、Idcard、Name、Sim、IsNeedCharge、Comparestatus、Comparemsg、CompareLibType 字段是个空值，此时本条明细是不会进行计费的。可以理解为，只有 IsNeedCharge 字段为 true 的时候是计费的，该字段为 false、null 都不计费。

附录

GetWeChatBillDetails 接口介绍

输入参数

参数名	是否必填	类型	描述
Date	Y	Date	拉取的日期（YYYY-MM-DD）。最大可追溯到365天前。当天6点后才能拉取前一天的数据。
RuleId	Y	String	业务对应的 ruleid
Cursor	Y	Integer	游标。用于分页，取第一页时传0，取后续页面时，传入本接口响应中的 NextCursor 字段的值。

输出参数

参数名	类型	描述
HasNextPage	Boolean	是否还有下一页。该字段为 true 时，需要将 NextCursor 的值作为入参继续调用本接口。
NextCursor	Integer	下一页的游标。用于分页。
WeChatBillDetails	WeChatBillDetail 数组	数据，明细如下表。

WeChatBillDetail

属性名	类型	说明
BizToken	String	token。
ChargeCount	Integer	本 token 收费次数。
ChargeDetails	ChargeDetail数组	本 token 计费详情，明细如下表。

ChargeDetail

属性名	类型	说明
ReqTime	String	一比一请求时间戳，13位。
Seq	String	一比一请求的唯一标记。
Idcard	String	一比一时使用的、脱敏后的身份证号。
Name	String	一比一时使用的、脱敏后的姓名。
Sim	String	一比一的相似度。0-100，保留2位小数。
IsNeedCharge	Boolean	本次详情是否收费。
ChargeType	String	收费类型，比对、核身、混合部署。
ErrorCode	String	本次活体一比一最终结果。
ErrorMessage	String	本次活体一比一最终结果描述。

H5（微信公众号）
微信原生 H5（微信公众号）

最近更新时间：2025-01-15 16:06:12

本文档以接入微信原生 H5 的操作流程为例，描述通过微信 HTML5 方式接入人脸核身的完整操作流程。

接入说明

1. 主体行业要求
   微信原生 H5 有明确的主体行业要求，在控制台申请 RuleId 时需上传对应的行业资质。所以在接入微信原生 H5 前，请确认公众号的主体和类目是否符合范围，并准备好对应的资质文件，详情可查阅 微信原生H5（微信公众号）资质文件列表 。
2. 审核时间
   微信原生 H5 有主体行业限制，资质审核需要3 - 5个工作日，请预留好上线时间。
3. 活体检测模式
   微信原生 H5 支持一闪活体检测（眨眼 + 光线）、随机双动作检测。
4. 公众号管理员授权
   微信原生 H5 需要公众号管理员授权，打开二维码，将该权限授权给人脸核身第三方平台。
5. 认证页面流程
6. 微信原生 H5 用户进行人脸核身流程：

前提条件

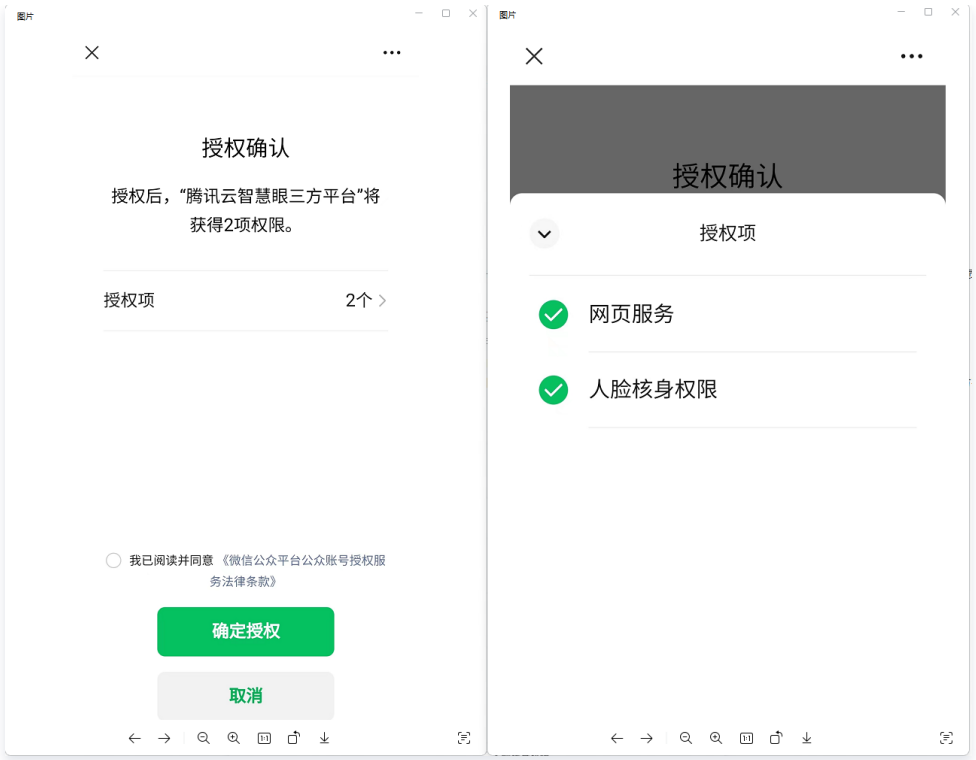
1. 已注册腾讯云账号，并完成实名认证。
2. 已申请通过腾讯云人脸核身。

如果还未完成以上操作，可参考 流程指引 完成操作。

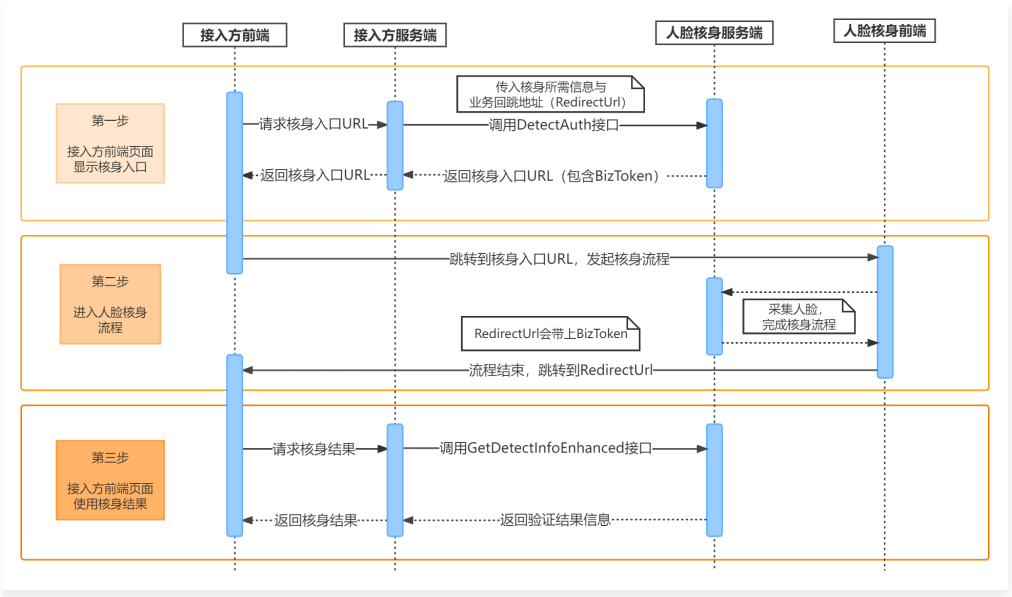
准备事项

结合您的实际业务场景，确认以下事项：

1. **确认接入方式**：选择微信原生 H5 模式，微信原生 H5 的接入方式有行业限制，且资质文件中主体与需要接入公众号主体一致，详细行业类目和资质材料请查阅 微信原生H5（微信公众号）资质文件列表 。
2. **确认人脸比对库源**：是需要人脸核身跟权威库比对还是跟上传照片比对，两个详细价格可参阅 计费概述 。
3. 登录官网控制台 创建 API 密钥（SecretId 和 SecretKey）。
4. **确认微信公众号已权限授权给人脸核身**：微信原生 H5 模式需要公众号管理员授权，步骤如下：
   打开二维码，公众号管理员扫码后，确定授权。



接入时序图



操作步骤

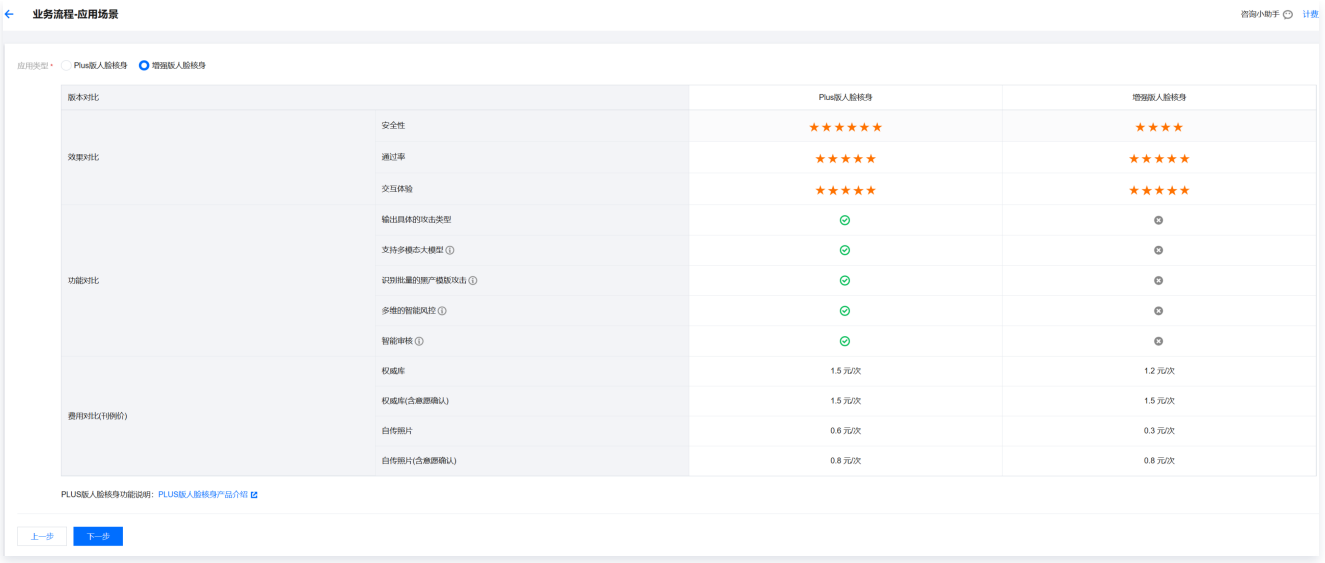
步骤1：创建 RuleId

RuleId 用于调用配置的业务流程，创建步骤如下：

1. 登录腾讯云人脸核身控制台，在 自助接入 页面，单击**创建业务流程**。

2. 选择应用场景：**微信原生 H5**，并根据需求填写相关信息，填写完成后单击**下一步**。**微信原生 H5** 需要上传对应的 资质文件，我们会在3 – 5个工作日完成审核。

3. 选择应用类型：选择**增强版人脸核身**，然后单击**下一步**。

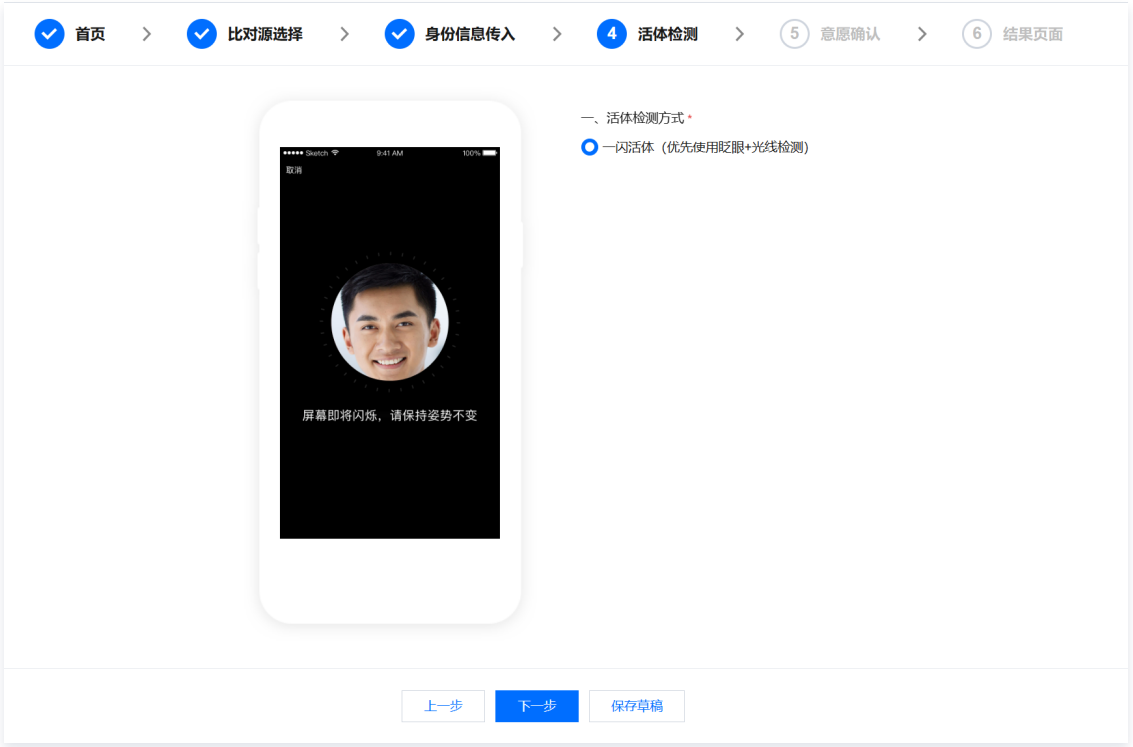


4. 进入接入配置，根据您业务的实际场景填写页面标题、业务名称、业务描述信息、业务联系方式后单击**下一步**。

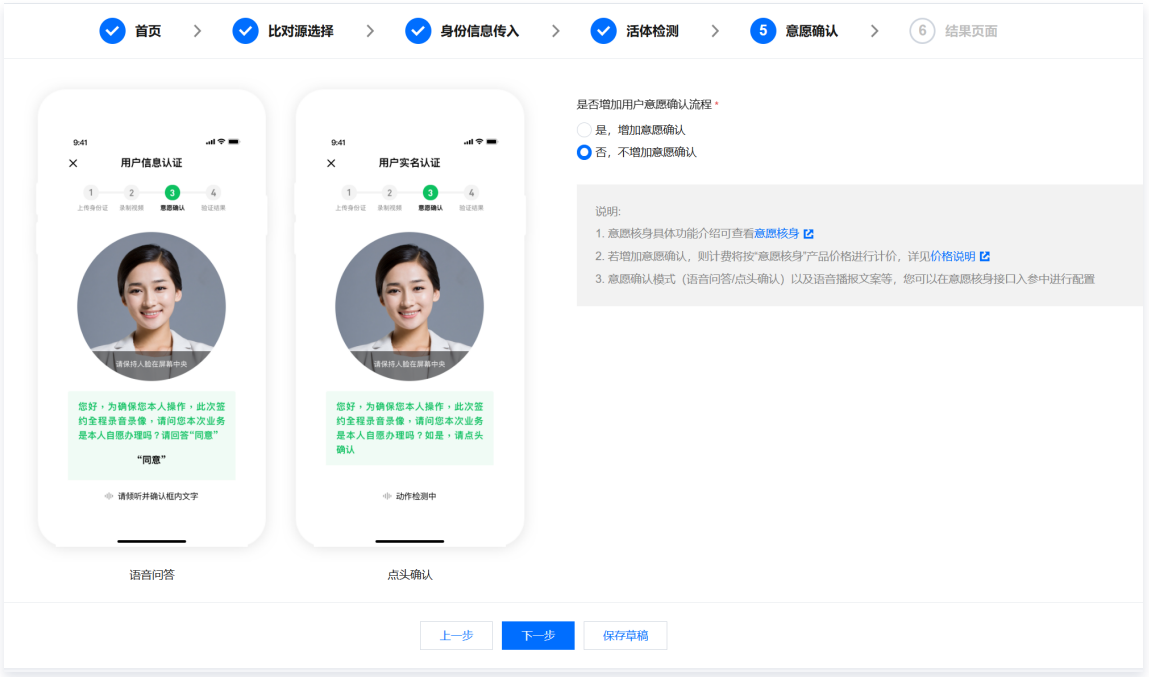
5. 选择人脸比对库源。人脸核身支持两种方式：**跟权威库比对**和**跟上传照片比对**。
   - 其中**跟权威库比对**的收费价格为：增强版人脸核身（权威库）- 微信的价格。
   - **跟上传照片比对**的收费价格为：增强版人脸核身（自传照片）- 微信的价格。OCR 不再单独收费。

6. 配置身份证 OCR 功能，如果不需要则勾选“**不需要用户在验证时上传**”，然后单击**下一步**。

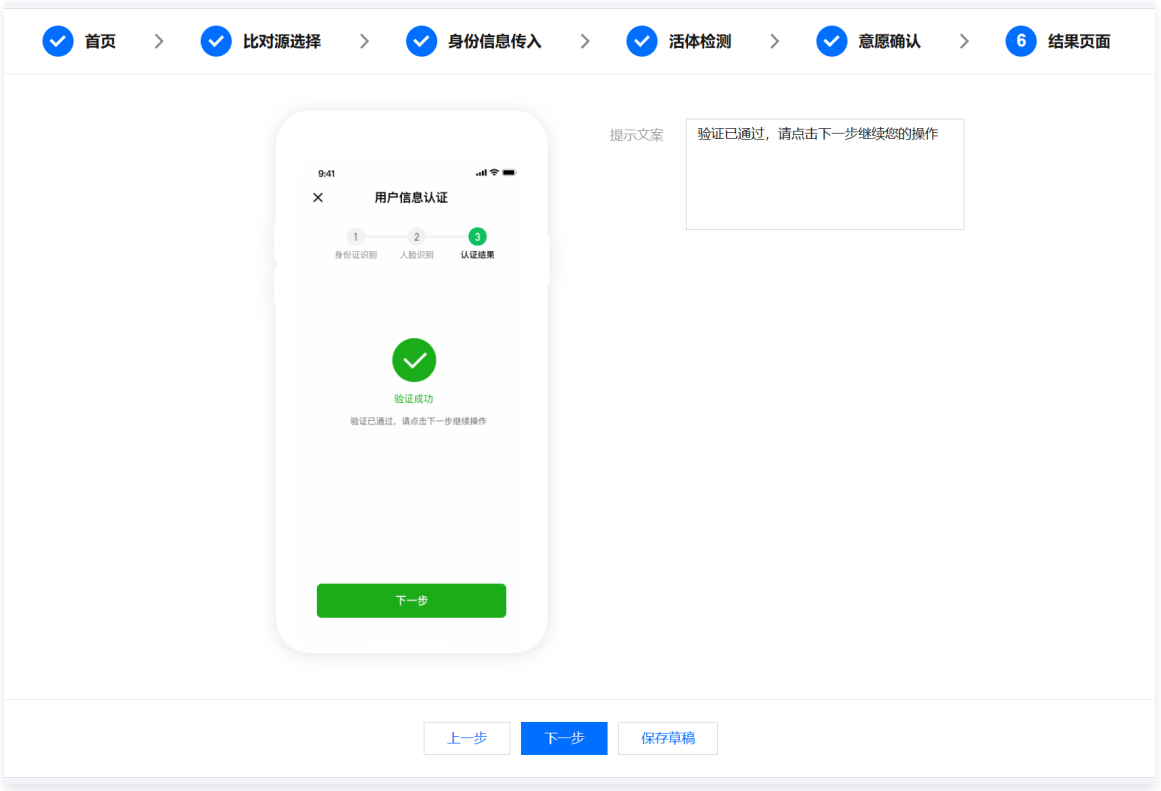
7. 配置活体检测方式，勾选后单击**下一步**。



8. 配置意愿页面，根据您的业务情况勾选：是否增加用户意愿确认流程，然后单击**下一步**。请注意，增强版人脸核身默认是不增加意愿确认流程的，如配置了增加意愿确认，则将按**意愿核身**价格计费，详情请参见 价格说明 。



9. 配置结果页的文案描述，然后单击**下一步**。



10. 业务信息填写完成后，确认您的配置信息和计费信息，然后单击**确认并提交审核**。

11. 流程审核通过后，系统会自动创建流程并分配业务 ID（RuleId）。

步骤2：获取 BizToken

完成 RuleId 创建后，需获取 BizToken，用于调用您配置的人脸核身验证的流程。
调用 实名核身鉴权接口，传入 步骤1 生成的 RuleId，和认证结束后重定向的回调链接地址 RedirectUrl，得到核验流程唯一密钥（BizToken）和用于发起核身流程的 URL。

- 在线调试

> **注意：**
> BizToken 是仅一次核身流程的标识，有效时间为7,200秒；用户完成核身后，开发者可用该标识获取验证结果信息，结果信息仅保留3天。
> 如果输入参数 RedirectUrl 为空，则用户完成核验后默认跳转腾讯云人脸核身产品介绍页。

步骤3： 跳转人脸核身 URL 完成核验

您在前端通过地址跳转方式重定向至 步骤2 中获取的核身入口 URL，用户进入核身流程，完成身份证拍摄识别、录制视频等操作完成身份核验。

步骤4： 查询核验结果信息

用户完成人脸核身后，页面会跳转到 RedirectUrl 上，地址中会带上此次验证流程使用的 BizToken，您在服务端即可凭借 BizToken 参数调用 获取实名核身结果信息 接口去获取本次核身的详细信息。获取到用户验证过程数据，包括文本信息、识别分数、照片和视频。也可以通过访问 腾讯云人脸核身控制台 查看服务调用情况。

> **注意：**
> 为了保证用户的隐私，结果信息人脸核身侧仅保留3天，请您及时拉取。
> 在开发、产品使用、费用、合同等问题有任何疑问，欢迎您 点此链接 扫描二维码添加腾讯云人脸核身小助手进行询问。

微信原生 H5（微信公众号）资质文件列表

最近更新时间：2024-07-11 09:58:41

微信原生 H5 浮层模式是基于微信原生的体验，客户体验好，但有明确的主体行业限制。
接入前请确认公众号的主体和类目是否符合以下范围，并准备好对应的资质文件，资质审核需要5 - 7个工作日，请预留好上线时间。

微信原生 H5 的主体类目

微信公众号支持行业类目及资质文件要求如下：

一级类目	二级类目	使用人脸核验接口所需资质
教育	学历教育（学校）	公立学校：由教育行政部门出具的审批设立证明或《事业单位法人证书》。
医疗	公立医疗机构	《医疗机构执业许可证》与《事业单位法人证书》。
	互联网医院	仅支持公立医疗机构互联网医院（2选1）： 1. 卫生健康部门的《设置医疗机构批准书》。 2. 《医疗机构执业许可证》（范围均需含“互联网诊疗”或名称含“互联网医院”等相关内容）。
政务民生	所有二级类目	仅支持政府/事业单位，提供《组织机构代码证》或《统一社会信用代码证》。

人脸核身对账指引

最近更新时间：2025-01-15 17:14:53

本文将为您描述使用腾讯云慧眼人脸核身服务中的微信渠道产品时（如使用人脸核身微信小程序、微信原生 H5、微信普通 H5 的人脸核身 SaaS 服务），如何与慧眼人脸核身进行对账操作。

对账工作主要分为2部分：

1. 收集使用慧眼业务时的明细信息。
2. 根据明细信息计算出账单。

下文将为您提供两种方式进行收集明细和计算账单。

方式1

在业务流程中保存明细

用户完成核验后，接入方调用 DetectAuth 接口生成的 `BizToken` 是一次核身流程的流水号。在业务完成后，接入方需要调用 GetDetectInfoEnhanced 接口来获取 `DetectInfo`，`DetectInfo` 就是描述本次核身的详细信息，包括核身结果，收费详情，照片视频信息等。**一个 BizToken 对应且只对应一个 DetectInfo。**

收集完整明细信息的关键点在于：

1. 标记 `BizToken`，确保不遗漏任何一个 `BizToken`。
   在调用 DetectAuth 接口获取 `BizToken` 后，必须将其存储起来，并标记该 `BizToken` 是否已经拉取到 `DetectInfo`。
2. 建立机制，确保每一个 `BizToken` 都能拉取到 `DetectInfo`。
   一般情况下慧眼侧是建议接入方在自己的回调接口中（DetectAuth 接口传入的 `RedirectUrl` 参数）调用 GetDetectInfoEnhanced 接口 去获取 `DetectInfo`。但由于用户侧核身场景环境比较复杂，并不能保证每一次的核身流程都能回调到 `RedirectUrl`。所以，接入方需要建立一个机制，来检查出未拉取 `DetectInfo` 的 `BizToken`，并重新拉取一遍。
3. 确保拉取到完成状态时的 `DetectInfo`。
   在人脸核身业务中，每个 `BizToken` 都有自己的状态，当 `BizToken` 到达完成状态时，所有的接口均不允许调用，其对应的 `DetectInfo` 也不能再更新。也就是说，若 `BizToken` 未达到完成状态的情况下，用户有可能使用此 `BizToken` 进行重试。未达到完成状态的 `BizToken` 都须重新拉取 `DetectInfo` 来确保拉取到的 `DetectInfo` 是完成状态时的。
   满足以下条件中的任意一个，即可认为 `BizToken` 处在完成状态：
   - GetDetectInfoEnhanced 接口返回的 `Response.Text.ErrCode` 为0。
   - 距离 Token 生成时间已超过2小时。

> ⚠ **注意**
> 慧眼人脸核身支持配置最大可重试次数，该次数仅影响活体一比一的次数，并不影响 OCR、短信等接口的调用。

计算账单

`DetectInfo` 中与计费相关的信息全部都位于 `Response.Text.LivenessDetail` 数组中，该数组存放的是该 `BizToken` 进行的活体一比一的次数与每次活体一比一的详情。**遍历该数组，统计 `IsNeedCharge` 为 `true` 的个数即是此 BizToken 的收费次数。**

> ⚠ **注意**
> `Response.Text.ErrCode` 是用于判断本次核验是否通过的字段，不用于判断计费与否。

为方便对账，建议将 `LivenessDetail` 数组中的每一项均保存下来。

```
let totalCount = 0; // 活体一比一的总记录数
let chargeCount = 0; // 活体一比一的收费记录数
let GetDetectInfoResponse = GetDetectInfoEnhanced(xxx, xxx, ...); // 调用GetDetectInfoEnhanced接口获取DetectInfo
for (let detail of GetDetectInfoResponse.Text.LivenessDetail){
    totalCount = totalCount + 1;
    if (detail.IsNeedCharge == true) {
        chargeCount = chargeCount + 1; // 当IsNeedCharge为true时，收费记录数加1
    }
    saveChargeDetailToDB(detail); // 保存至数据库，以便日后查看
}
```

方式2

使用 GetWeChatBillDetails 接口每日拉取

为了方便接入方更快捷的获取明细信息，慧眼人脸核身提供了一个接口（GetWeChatBillDetails）用于拉取每日的明细信息。该接口可以拉取子账号 + RuleId 维度的某一天的所有 token 数据。
具体使用方式如下：
1. 建立定时任务，每日早上6点后开始执行。**早于6点调用 GetWeChatBillDetails 接口会报参数异常。**
2. 调用 GetWeChatBillDetails 接口，拉取前一天所有的验证明细信息。

```
    let WeChatBillDetails = []; // 当日全量的账单详情
    let cursor = 0; // 游标位，拉取第一页时传0；剩下页面根据上一次请求回包的结果传入。
    let hasNextPage = true; // 是否还有下一页的标志位
    while (hasNextPage) { // 如果还有下一页，则继续拉取
        const result = GetWeChatBillDetails(Date /* 传入某一天日期*/, RuleId /* 传入RuleId*/, cursor); // 调用
GetWeChatBillDetails接口，并获取返回。
        WeChatBillDetails = WeChatBillDetails.concat(result.WeChatBillDetails); // 将拉取到的数据合并到全量的数组中
        cursor = result.NextCursor; // 将回包中返回的下一页的游标赋值给游标位，准备下一次拉取
        hasNextPage = result.HasNextPage; // 将是否还有下一页的标志位重新赋值，用于决定是否进行下一次拉取
    }
    // 此时WeChatBillDetails中就是某一天全量的账单信息
```

计算账单

统计 WeChatBillDetail 数组中的 ChargeCount 总数，即为当日计费总数。同时记录 ChargeDetail 内容，可用于校验：ChargeCount 值 = ChargeDetail 中 IsNeedCharge 为 true 的条目总数 = ErrorCode 为收费错误码的条目总数。

```
let totalCount = 0; // 活体一比一的总记录数
let chargeCount = 0; // 活体一比一的收费记录数
for (let WeChatBillDetail of WeChatBillDetails) { // 遍历WeChatBillDetails数组
    let ChargeDetails = WeChatBillDetail.ChargeDetails; // 取计费详情属性
    totalCount += ChargeDetails.length; // 总记录数为计费详情的总条目数
    chargeCount += WeChatBillDetail.ChargeCount; // 收费记录数为ChargeCount的总和
    for (let ChargeDetail of ChargeDetails) {
        saveChargeDetailToDB(ChargeDetail); // 遍历ChargeDetails数组，保存至数据库，以便日后查看
    }
}
```

常见问题

1. 什么时候进行计费上报?
计费上报时机是以**一比一请求时间**为准，即 `Response.Text.LivenessDetail[].ReqTime` 、GetWeChatBillDetails 接口中的 `ChargeDetail.ReqTime` 。例如，BizToken 是在7月1日的23:59:59生成的，但是进行比对的时机是在7月2日的0点，那一比一的请求时间也会落在7月2日的0点，同时这次调用也会上报至7月2日的账单中。
2. LivenessDetail 中有些字段为 null 是什么原因?
当本次核身没有进行一比一或者其他原因造成一比一失败时，有可能导致 ReqTime、Seq、Idcard、Name、Sim、IsNeedCharge、Comparestatus、Comparemsg、CompareLibType 字段是个空值，此时本条明细是不会进行计费的。可以理解为，只有 IsNeedCharge 字段为 true 的时候是计费的，该字段为 false、null 都不计费。

附录

GetWeChatBillDetails 接口介绍

输入参数

参数名	是否必填	类型	描述
Date	Y	Date	拉取的日期（YYYY-MM-DD）。最大可追溯到365天前。当天6点后才能拉取前一天的数据。
RuleId	Y	String	业务对应的 ruleid
Cursor	Y	Integer	游标。用于分页，取第一页时传0，取后续页面时，传入本接口响应中得 NextCursor 字段的值。

输出参数

参数名	类型	描述
HasNextPage	Boolean	是否还有下一页。该字段为 true 时，需要将 NextCursor 的值作为入参继续调用本接口。
NextCursor	Integer	下一页的游标。用于分页。
WeChatBillDetails	WeChatBillDetail 数组	数据，明细如下表

WeChatBillDetail

属性名	类型	说明
BizToken	String	token。
ChargeCount	Integer	本 token 收费次数。
ChargeDetails	ChargeDetail数组	本 token 计费详情，明细如下表。

ChargeDetail

属性名	类型	说明
ReqTime	String	一比一请求时间戳，13位。
Seq	String	一比一请求的唯一标记。
Idcard	String	一比一时使用的、脱敏后的身份证号。
Name	String	一比一时使用的、脱敏后的姓名。
Sim	String	一比一的相似度。0-100，保留2位小数。
IsNeedCharge	Boolean	本次详情是否收费。
ChargeType	String	收费类型，比对、核身、混合部署。
ErrorCode	String	本次活体一比一最终结果。
ErrorMessage	String	本次活体一比一最终结果描述。

H5（微信浏览器）
微信浮层 H5（微信浏览器）

最近更新时间：2025-01-15 16:06:12

本文档以接入微信浮层 H5 的操作流程为例，描述通过微信 HTML5 方式接入人脸核身的完整操作流程。

> **说明：**
> 若需同时在微信场景及非微信场景（如企业微信、支付宝、移动 H5 等）使用，可根据此文档完成接入后 联系我们 配置支持。

接入说明

1. 主体行业要求
   无行业类目没有限制。
2. 兼容性要求
   浮层 H5 能够兼容大部分用户的微信浏览器环境，仅有少部分浏览器不支持（低于5%），当检测到浏览器不兼容时会自动切换为普通 H5 的模式，以保证人脸核身的正常进行。

手机平台	应用端	兼容性要求
iOS	微信浏览器	iOS 系统版本 14.3+，微信版本 6.5+
	企微微信浏览器	iOS 系统版本 14.3+，企微版本4.0.8+
Android	微信浏览器	都支持
	企微微信浏览器	都支持

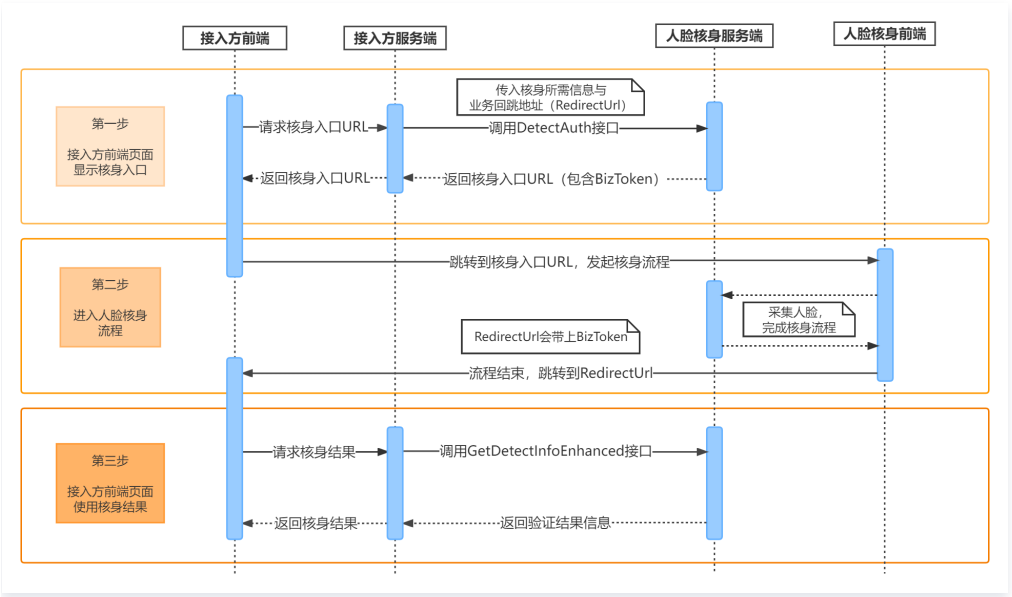
3. 活体检测模式
   微信浮层 H5 支持一闪活体检测（眨眼+光线）、随机双动作检测。
4. 认证页面流程

前提条件

1. 已注册腾讯云账号，并完成实名认证。
2. 已申请通过腾讯云人脸核身。

如果还未完成以上操作，可参考 流程指引 完成操作。

接入时序图

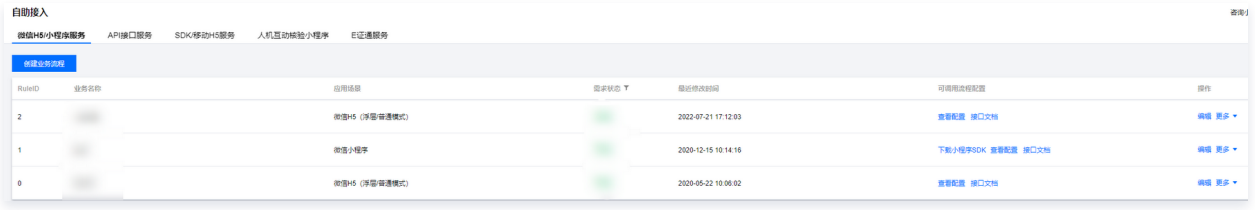


操作步骤

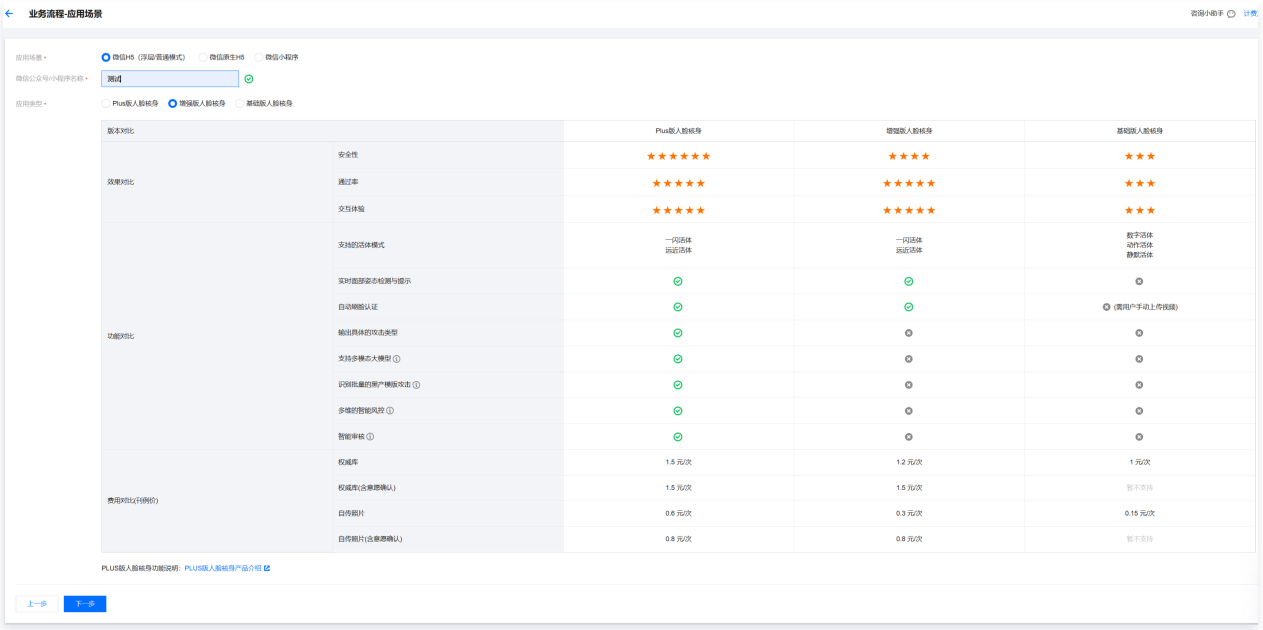
步骤1：创建 RuleId

RuleId 用于调用配置的业务流程，创建步骤如下：

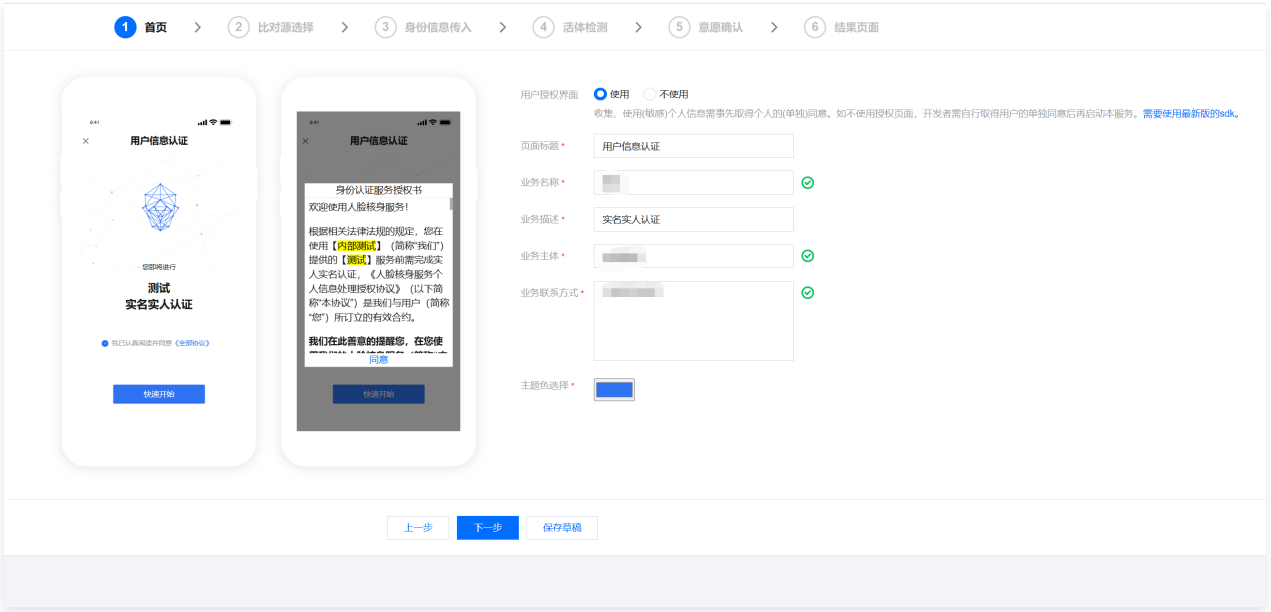
1. 登录腾讯云人脸核身控制台，在 **自主接入** 页面，单击**创建业务流程**。



2. 选择应用场景选择：**微信 H5（浮层/普通模式）**，应用类型选择：**增强版人脸核身**，然后单击**下一步**。

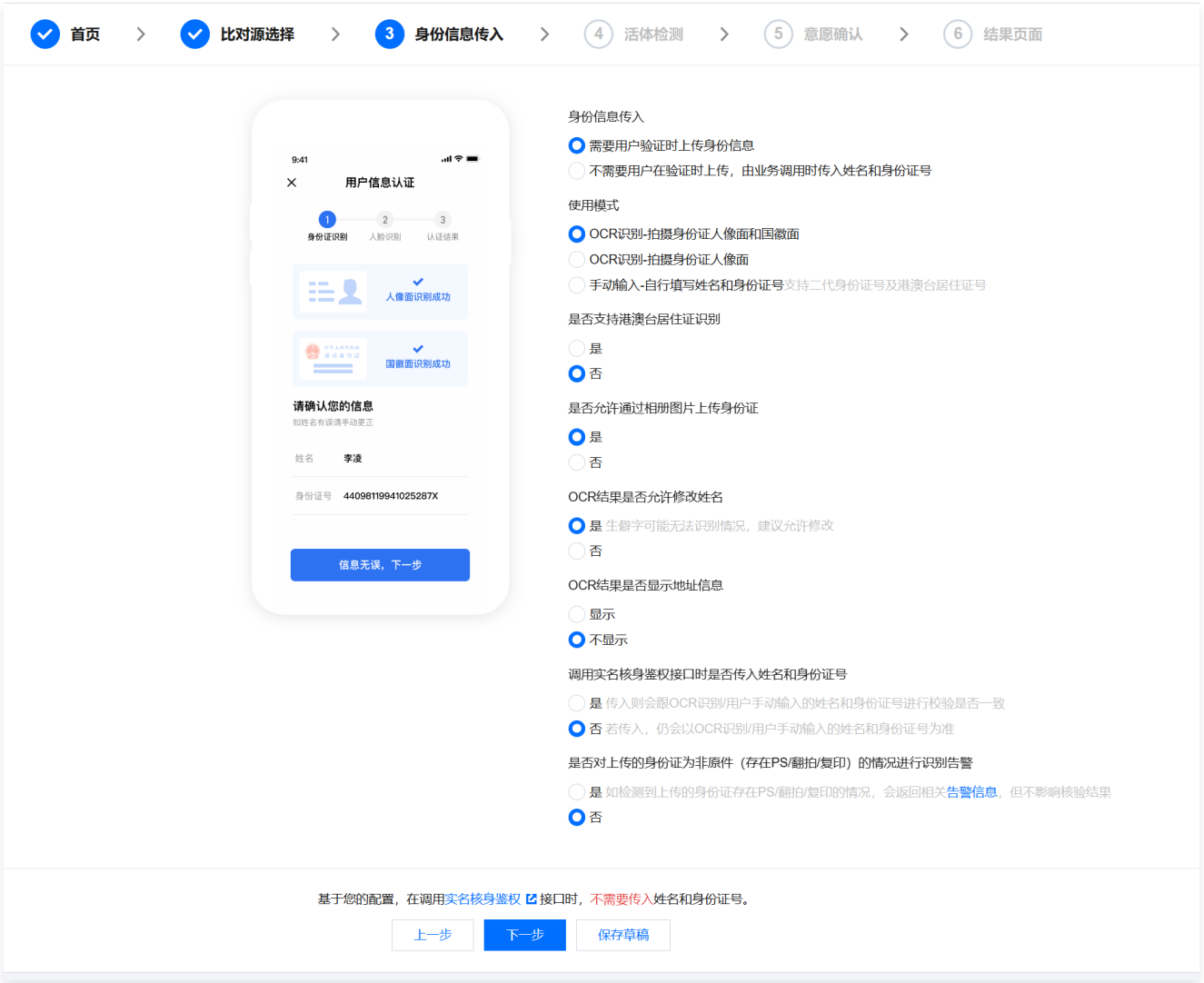


3. 进入接入配置，根据您业务的实际场景填写页面标题、业务名称、业务描述信息后单击**下一步**。



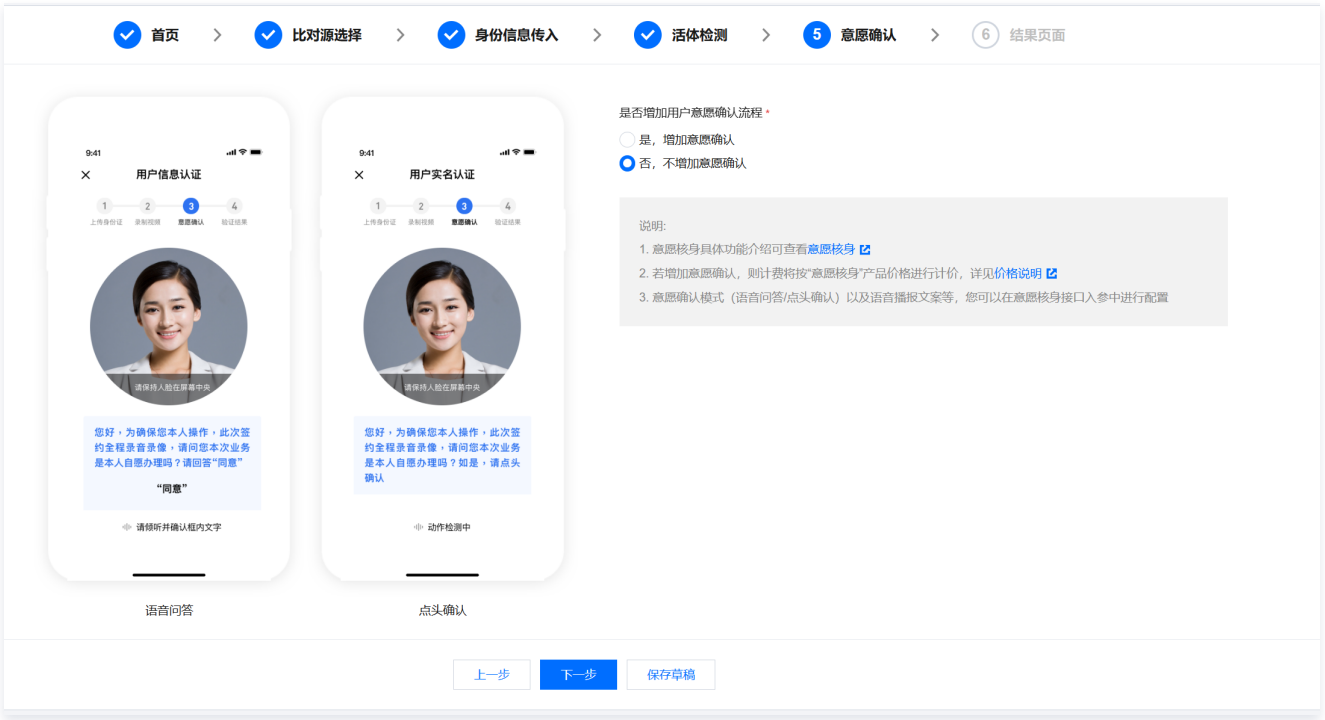
4. 选择人脸比对库源。人脸核身支持两种方式：**跟权威库比对**和**跟上传照片比对**。
   - 其中**跟权威库比对**收费价格为：增强版人脸核身（权威库）– 浮层 H5 的价格。
   - **跟上传照片比对**收费的价格为：增强版人脸核身（自传照片）– 浮层 H5 的价格。OCR 不再单独收费。

5. 配置身份证 OCR 功能，如果不需要则勾选“**不需要用户在验证时上传**”，然后单击**下一步**。

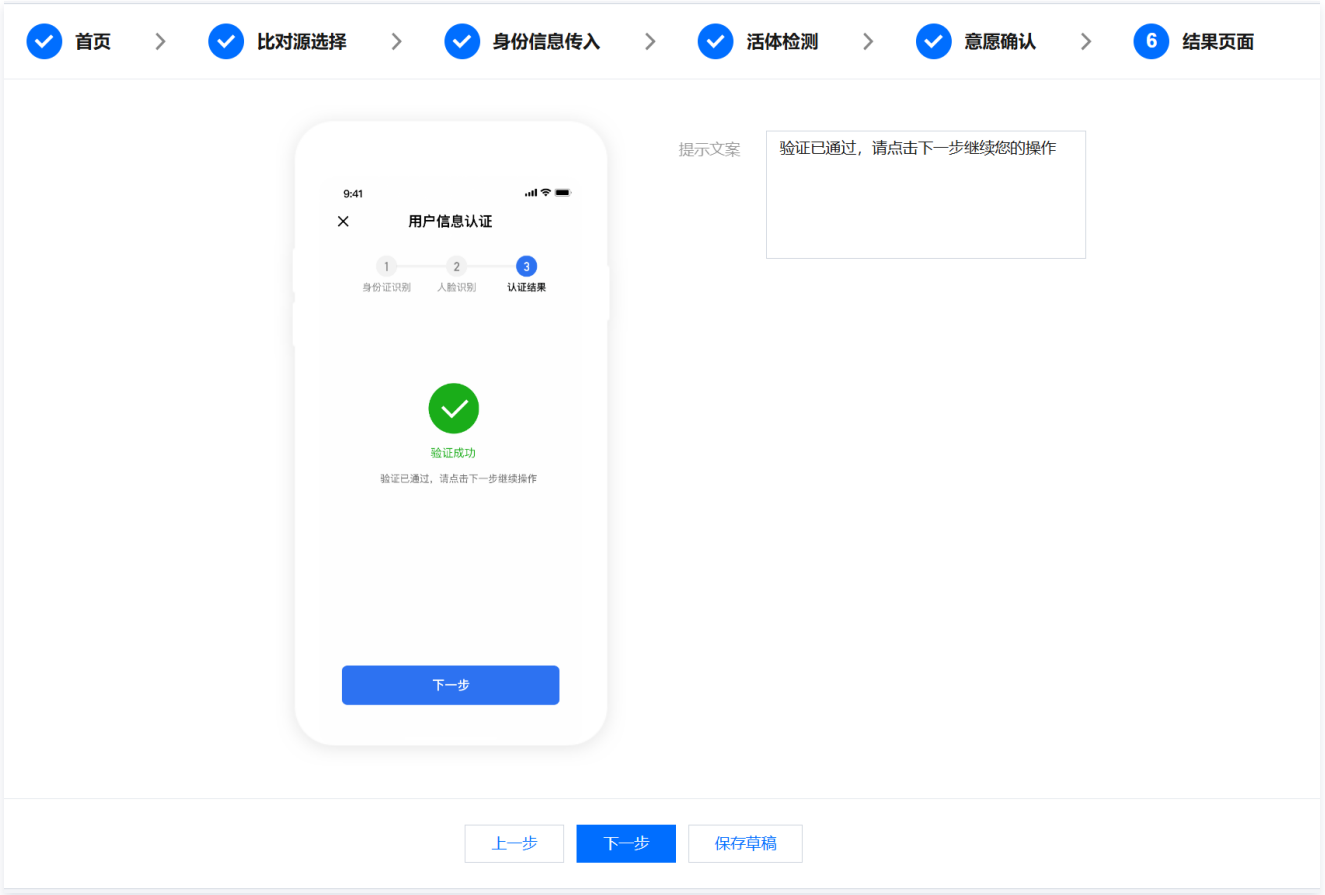


6. **配置活体检测方式，勾选后单击下一步。**

7. **配置意愿页面，根据您的业务情况勾选：是否增加用户意愿确认流程，然后单击下一步。请注意，增强版人脸核身默认是不增加意愿确认流程的，如配置了增加意愿确认，则将按意愿核身价格计费，详情请参见 价格说明 。**



8. 配置结果页的文案描述，然后单击**下一步**。



9. 业务信息填写完成后，确认您的配置信息然后单击**确认并提交审核**。

步骤2：获取 BizToken

- 完成 RuleId 创建后，需获取 BizToken，用于调用您配置的人脸核身验证的流程。
- 调用 实名核身鉴权，传入 步骤1 生成的 RuleId，和认证结束后重定向的回调链接地址 RedirectUrl，得到核验流程唯一密钥（BizToken）和用于发起核身流程的 URL。
- 在线调试。

> **注意：**
> - BizToken 是仅一次核身流程的标识，有效时间为7,200秒；用户完成核身后，开发者可用该标识获取验证结果信息，结果信息仅保留3天。
> - 如果输入参数 RedirectUrl 为空，则用户完成核验后默认跳转腾讯云人脸核身产品介绍页。

步骤3： 跳转人脸核身 URL 完成核验

您在前端通过地址跳转方式重定向至 步骤2 中获取的核身入口 URL，用户进入核身流程，完成身份证拍摄识别、录制视频等操作完身份核验。

步骤4： 查询核验结果信息

用户完成人脸核身后，页面会跳转到 RedirectUrl 上，地址中会带上此次验证流程使用的 BizToken，您在服务端即可凭借 BizToken 参数调用 获取实名核身结果信息 接口去获取本次核身的详细信息。获取到用户验证过程数据，包括文本信息、识别分数、照片和视频。也可以通过访问 人脸核身控制台 查看服务调用情况。

> **注意：**
> - 为了保证用户的隐私，结果信息人脸核身侧仅保留3天，请您及时拉取。
> - 在开发、产品使用、费用、合同等问题有任何疑问，欢迎您 联系我们 进行询问。

人脸核身对账指引

最近更新时间：2024-09-10 17:56:33

本文将为您描述使用腾讯云慧眼人脸核身服务中的微信渠道产品时（如使用人脸核身微信小程序、H5（微信公众号）、H5（微信浏览器）的人脸核身 SaaS 服务），如何与慧眼人脸核身进行对账操作。

对账工作主要分为2部分：

1. 收集使用慧眼业务时的明细信息。
2. 根据明细信息计算出账单。

下文将为您提供两种方式进行收集明细和计算账单。

方式1

在业务流程中保存明细

用户完成核验后，接入方调用 DetectAuth 接口生成的 `BizToken` 是一次核身流程的流水号。在业务完成后，接入方需要调用 GetDetectInfoEnhanced 接口来获取 `DetectInfo`，`DetectInfo` 就是描述本次核身的详细信息，包括核身结果，收费详情，照片视频信息等。**一个 BizToken 对应且只对应一个 DetectInfo。**

收集完整明细信息的关键点在于：

1. 标记 `BizToken`，确保不遗漏任何一个 `BizToken`。
   在调用 DetectAuth 接口获取 `BizToken` 后，必须将其存储起来，并标记该 `BizToken` 是否已经拉取到 `DetectInfo`。
2. 建立机制，确保每一个 `BizToken` 都能拉取到 `DetectInfo`。
   一般情况下慧眼侧是建议接入方在自己的回调接口中（DetectAuth 接口传入的 `RedirectUrl` 参数）调用 GetDetectInfoEnhanced 接口 去获取 `DetectInfo`。但由于用户侧核身场景环境比较复杂，并不能保证每一次的核身流程都能回调到 `RedirectUrl`。所以，接入方需要建立一个机制，来检查出未拉取 `DetectInfo` 的 `BizToken`，并重新拉取一遍。
3. 确保拉取到完成状态时的 `DetectInfo`。
   在人脸核身业务中，每个 `BizToken` 都有自己的状态，当 `BizToken` 到达完成状态时，所有的接口均不允许调用，其对应的 `DetectInfo` 也不能再更新。也就是说，若 `BizToken` 未达到完成状态的情况下，用户有可能使用此 `BizToken` 进行重试。未达到完成状态的 `BizToken` 都须重新拉取 `DetectInfo` 来确保拉取到的 `DetectInfo` 是完成状态时的。
   满足以下条件中的任意一个，即可认为 `BizToken` 处在完成状态：
   - GetDetectInfoEnhanced 接口返回的 `Response.Text.ErrCode` 为0。
   - 距离 Token 生成时间已超过2小时。

> **注意**
> 慧眼人脸核身支持配置最大可重试次数，该次数仅影响活体一比一的次数，并不影响 OCR、短信等接口的调用。

计算账单

`DetectInfo` 中与计费相关的信息全部都位于 `Response.Text.LivenessDetail` 数组中，该数组存放的是该 `BizToken` 进行的活体一比一的次数与每次活体一比一的详情。**遍历该数组，统计 `IsNeedCharge` 为 `true` 的个数即是此 BizToken 的收费次数。**

> **注意**
> `Response.Text.ErrCode` 是用于判断本次核验是否通过的字段，不用于判断计费与否。

为方便对账，建议将 `LivenessDetail` 数组中的每一项均保存下来。

```
let totalCount = 0; // 活体一比一的总记录数
let chargeCount = 0; // 活体一比一的收费记录数
let GetDetectInfoResponse = GetDetectInfoEnhanced(xxx, xxx, ...); // 调用GetDetectInfoEnhanced接口获取DetectInfo
for (let detail of GetDetectInfoResponse.Text.LivenessDetail){
    totalCount = totalCount + 1;
    if (detail.IsNeedCharge == true) {
        chargeCount = chargeCount + 1; // 当IsNeedCharge为true时，收费记录数加1
    }
    saveChargeDetailToDB(detail); // 保存至数据库，以便日后查看
}
```

方式2

使用 GetWeChatBillDetails 接口每日拉取

为了方便接入方更快捷的获取明细信息，慧眼人脸核身提供了一个接口（GetWeChatBillDetails）用于拉取每日的明细信息。该接口可以拉取子账号 + RuleId 维度的某一天的所有 token 数据。
具体使用方式如下：
1. 建立定时任务，每日早上6点后开始执行。**早于6点调用 GetWeChatBillDetails 接口会报参数异常。**
2. 调用 GetWeChatBillDetails 接口，拉取前一天所有的验证明细信息。

```
    let WeChatBillDetails = []; // 当日全量的账单详情
    let cursor = 0; // 游标位，拉取第一页时传0；剩下页面根据上一次请求回包的结果传入。
    let hasNextPage = true; // 是否还有下一页的标志位
    while (hasNextPage) { // 如果还有下一页，则继续拉取
        const result = GetWeChatBillDetails(Date /* 传入某一天日期*/, RuleId /* 传入RuleId*/, cursor); // 调用
GetWeChatBillDetails接口，并获取返回。
        WeChatBillDetails = WeChatBillDetails.concat(result.WeChatBillDetails); // 将拉取到的数据合并到全量的数组中
        cursor = result.NextCursor; // 将回包中返回的下一页的游标赋值给游标位，准备下一次拉取
        hasNextPage = result.HasNextPage; // 将是否还有下一页的标志位重新赋值，用于决定是否进行下一次拉取
    }
    // 此时WeChatBillDetails中就是某一天全量的账单信息
```

计算账单

统计 WeChatBillDetail 数组中的 ChargeCount 总数，即为当日计费总数。同时记录 ChargeDetail 内容，可用于校验：ChargeCount 值 = ChargeDetail 中 IsNeedCharge 为 true 的条目总数 = ErrorCode 为收费错误码的条目总数。

```
let totalCount = 0; // 活体一比一的总记录数
let chargeCount = 0; // 活体一比一的收费记录数
for (let WeChatBillDetail of WeChatBillDetails) { // 遍历WeChatBillDetails数组
    let ChargeDetails = WeChatBillDetail.ChargeDetails; // 取计费详情属性
    totalCount += ChargeDetails.length; // 总记录数为计费详情的总条目数
    chargeCount += WeChatBillDetail.ChargeCount; // 收费记录数为ChargeCount的总和
    for (let ChargeDetail of ChargeDetails) {
        saveChargeDetailToDB(ChargeDetail); // 遍历ChargeDetails数组，保存至数据库，以便日后查看
    }
}
```

常见问题

1. **什么时候进行计费上报?**
   计费上报时机是以一比一请求时间为准，即 `Response.Text.LivenessDetail[].ReqTime` 、GetWeChatBillDetails 接口中的 `ChargeDetail.ReqTime` 。例如，BizToken 是在7月1日的23:59:59生成的，但是进行比对的时机是在7月2日的0点，那一比一的请求时间也会落在7月2日的0点，同时这次调用也会上报至7月2日的账单中。
2. **LivenessDetail 中有些字段为 null 是什么原因?**
   当本次核身没有进行一比一或者其他原因造成一比一失败时，有可能导致 ReqTime、Seq、Idcard、Name、Sim、IsNeedCharge、Comparestatus、Comparemsg、CompareLibType 字段是个空值，此时本条明细是不会进行计费的。可以理解为，只有 IsNeedCharge 字段为 true 的时候是计费的，该字段为 false、null 都不计费。

附录

GetWeChatBillDetails 接口介绍

输入参数

参数名	是否必填	类型	描述
Date	Y	Date	拉取的日期（YYYY-MM-DD）。最大可追溯到365天前。当天6点后才能拉取前一天的数据。
RuleId	Y	String	业务对应的 ruleid。
Cursor	Y	Integer	游标。用于分页，取第一页时传0，取后续页面时，传入本接口响应中得 NextCursor 字段的值。

输出参数

参数名	类型	描述
HasNextPage	Boolean	是否还有下一页。该字段为 true 时，需要将 NextCursor 的值作为入参继续调用本接口。
NextCursor	Integer	下一页的游标。用于分页。
WeChatBillDetails	WeChatBillDetail 数组	数据，明细如下表。

WeChatBillDetail

属性名	类型	说明
BizToken	String	token。
ChargeCount	Integer	本 token 收费次数。
ChargeDetails	ChargeDetail数组	本 token 计费详情，明细如下表。

ChargeDetail

属性名	类型	说明
ReqTime	String	一比一请求时间戳，13位。
Seq	String	一比一请求的唯一标记。
Idcard	String	一比一时使用的、脱敏后的身份证号。
Name	String	一比一时使用的、脱敏后的姓名。
Sim	String	一比一的相似度。0-100，保留2位小数。
IsNeedCharge	Boolean	本次详情是否收费。
ChargeType	String	收费类型，比对、核身、混合部署。
ErrorCode	String	本次活体一比一最终结果。
ErrorMessage	String	本次活体一比一最终结果描述。

H5（移动端浏览器）
H5 人脸核身介绍

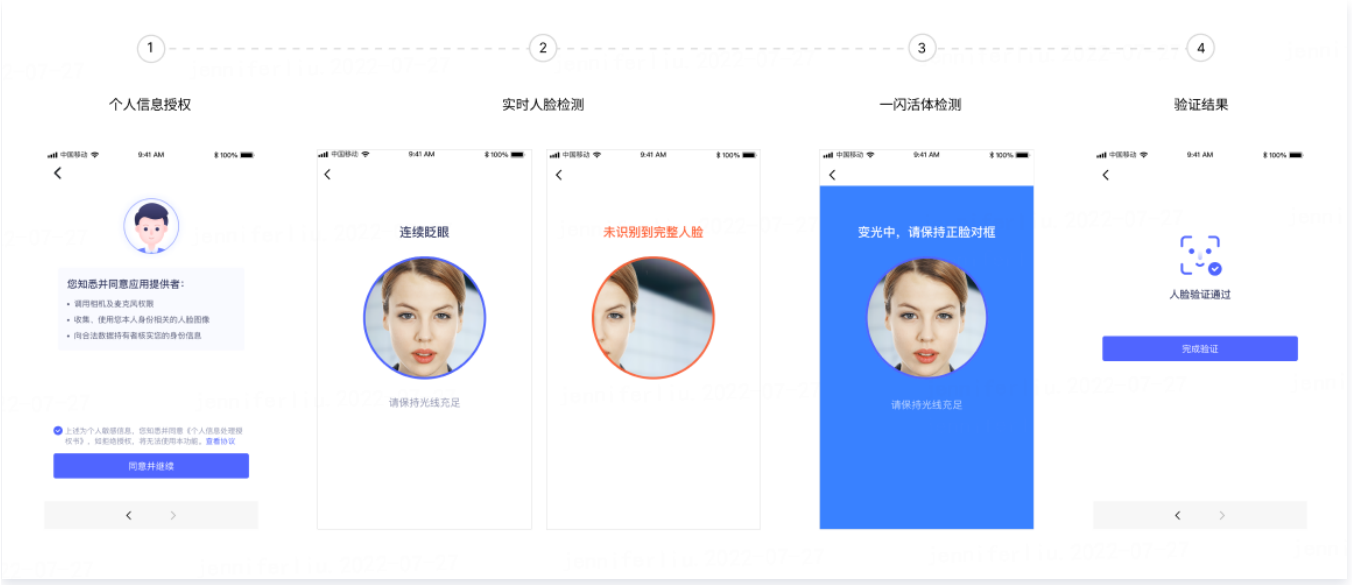
最近更新时间：2025-06-16 17:02:32

方案介绍

移动 H5 人脸核身全新升级，支持人脸实时检测和一闪活体模式，特点如下：

- 支持人脸实时检测，用户交互体验更好。
- 支持动作 + 一闪活体，活体安全性更高。
- 优先实时检测模式，如遇到不兼容的情况，则会平滑切换为视频录制模式。
- 支持多场景接入，包括 App 调用 H5、浏览器等。

> **注意**
> 接入前务必注意：请参见 移动 H5 人脸核身接入步骤 进行接入，如是 App 调用，务必按照 兼容性配置指引 进行兼容性适配，否则将影响正常使用。



兼容性说明

因网页端实时音视频技术是在2017年初次提出，对浏览器和手机系统存在兼容性要求，经过大规模测试，腾讯云 H5 实时检测人脸核身的兼容性情况如下：

手机平台	应用端	应用端说明	兼容性要求	机型支持率
iOS	App	在客户自有 App 中使用，适用于在客户 App 中触达用户的业务场景	iOS 系统版本14.3+，需按照 兼容性配置指引 做适配	90%
	浏览器	在手机浏览器使用，适用于通过短信或其他方式发送链接触达用户的业务场景	iOS 系统版本11.1.2+，支持 Safari 浏览器，浏览器版本11+	100%
Android	App	在客户自有 App 中使用，适用于在客户 App 中触达用户的业务场景	Android 系统版本7+，需按照 兼容性配置指引 做适配	95%
	自带浏览器	在手机系统自带浏览器使用，适用于通过短信或其他方式发送链接触达用户的业务场景	Android 系统版本7+ 华为、OPPO、VIVO、魅族、荣耀、三星等自带浏览器兼容性较好（支持率80%） 小米自带浏览器兼容性一般（支持率30%）	
	QQ 浏览器	在 QQ 浏览器使用，适用于通过短信或其他方式发送链接触达用户的业务场景	Android 系统版本7+，支持 QQ 浏览器，不支持 UC、百度浏览器	100%

- 采用达到版本要求的具有一定市场占有率的主流手机型号，测试机型共计829款。

- Android 手机测试品牌名单（按拼音排序）：HTC、OPPO、OPPO_REALME、VIVO、VIVO_IQOO、锤子-坚果、黑鲨、华硕、华为、荣耀、金立、联想、魅族、魅族-魅蓝、努比亚、努比亚-红魔、三星-GALAXY、小米、小米-红米、一加、中兴。

H5 人脸核身接入步骤

最近更新时间：2023-06-08 11:19:05

注意

接入前务必注意：请按照以下接入步骤进行接入，否则将影响人脸核身的正常使用。

序号	接入步骤	描述
1	App 调用 H5 兼容性配置	如是 App 调用，请务必按照兼容性配置指引进行 iOS 及 Android 手机的兼容性适配，否则将影响正常使用。
2	合作方后台上传身份信息	合作方后台上传人脸核身信息，包括 appId、orderNo、name、idNo、userID、liveInterType。
3	启动 H5 人脸核身	合作方上传 h5faceId 以及 sign，后台校验 sign 通过之后重定向到 H5 人脸核身。
4	人脸核身结果返回及跳转	刷脸完成后，认证结果页会回调启动 H5 人脸核身入参中指定的回调 URL 地址，使用回调参数中的 code 判断是否核身通过。
5	人脸核身结果查询	当用户完成核身认证后，如果合作方需要拉取人脸核身的视频用于存证等其他需要，可以调用查询核身结果接口来获取。

人脸核身 App 调用 H5 兼容性配置指引

最近更新时间：2024-09-24 15:04:51

请按照下文**兼容性配置**指引进行 iOS 及 Android 手机的兼容性适配。

iOS 接入

iPhone 的兼容性适配，需在配置里加上摄像头和麦克风的使用权限。App 的 info.plist 中加入：

```
.NSMicrophoneUsageDescription
.NSCameraUsageDescription
```

使用 WKWebView 时，需要通过 WKWebViewConfiguration 配置允许使用相机：

```
WKWebViewConfiguration *config = [[WKWebViewConfiguration alloc] init];
config.allowsInlineMediaPlayback = YES;
```

Android 接入

由于 Android 机器碎片化严重，用系统 WebView 调起系统摄像头完成视频录制可能存在很多兼容性问题，如部分机器出现调不起摄像头、调起摄像头无法录制视频等。因此整理了接入指引。H5 刷脸包括 trtc 和录制模式，合作方需要对这两种模式都做兼容性配置。
请合作方**务必**按照如下步骤顺序，实现兼容性处理：

1. 引入工具类
   将 WBH5FaceVerifySDK.java 文件拷贝到项目中。该文件下载地址：人脸核身控制台（请到控制台自助接入列表页获取密码）。
2. 申请权限
   - 在 Manifest.xml 文件中增加申请以下权限

     ```
     <uses-permission android:name="android.permission.CAMERA" />
     <uses-permission android:name="android.permission.INTERNET" />
     ```

   - 动态申请权限
     - 如果第三方编译的 targetSdkVersion >= 23，则需要动态申请权限。
     - 如果第三方编译的 targetSdkVersion < 23，则不需要动态申请权限。
     - 权限代码申请处理逻辑，demo 仅供参考，合作方可根据自身业务需求自行处理。
     - 一定要在动态权限申请成功后，才能去调用 enterOldModeFaceVerify() 录制模式或enterTrtcFaceVerify() trtc 模式体验 h5 刷脸功能。
3. 设置 WebSettings
   调用 WebView.loadUrl(String url) 前一行添加如下代码设置 WebSettings。

   ```
   /**
   * 对 WebSettings 进行设置：添加 ua 字段上送kyc/h5face;kyc/2.0
   */

   webSetting.setUserAgentString(ua + ";kyc/h5face;kyc/2.0");//设置ua包含;kyc/h5face;kyc/2.0
   ```

4. 重写 WebChromeClient
   调用 WebView.loadUrl(String url) 前，WebView 必须调用 setWebChromeClient(WebChromeClient webChromeClient)，并重写 WebChromeClient 的如下5个函数：

   ```
   /**
        *TRTC 刷脸模式配置，这里负责处理来自H5页面发出的相机权限申请
   先申请终端的相机权限，再授权h5的trtc刷脸请求
        * @param request 来自H5页面的权限请求
        */
       @Override
       public void onPermissionRequest(PermissionRequest request) {
           if
   (request!=null&&request.getOrigin()!=null&&WBH5FaceVerifySDK.getInstance().isTencentH5FaceVerify(request.get
   ```

```
Origin().toString())){  //判断是腾讯h5刷脸的域名
            Log.d(TAG,"onPermissionRequest 收到腾讯h5刷脸页面的相机授权");
            this.request=request;
            if (activity!=null){
//申请终端的相机权限，trtc模式一定需要申请相机权限。申请权限的代码demo仅供参考，合作方可根据自身业务定制
                activity.requestCameraPermission(true,false);
            }
        }
    }

    /**
     * 终端相机权限申请成功后，拉起TRTC刷脸模式进行实时刷脸验证
     */
    public void enterTrtcFaceVerify(){
if (Build.VERSION.SDK_INT>Build.VERSION_CODES.LOLLIPOP){  // android sdk 21以上
            if (request!=null&&request.getOrigin()!=null){
               if (WBH5FaceVerifySDK.getInstance().isTencentH5FaceVerify(request.getOrigin().toString())){
//判断是腾讯h5刷脸的域名，如果合作方对授权域名无限制的话，这个if条件判断可以去掉，直接进行授权即可。
                    //授权
                    request.grant(request.getResources());
                    request.getOrigin();
                }
            }else {
                if (request==null){
                    Log.d(TAG,"enterTrtcFaceVerify request==null");
                    if (webView!=null&&webView.canGoBack()){
                        webView.goBack();
                    }
                }
      }
        }
     }

     // For Android >= 4.1  录制模式中，点击h5页面的【开始录制】按钮后触发的系统方法
    public void openFileChooser(ValueCallback<Uri> uploadMsg, String acceptType, String capture) {
if (WBH5FaceVerifySDK.getInstance().isTencentH5FaceVerify(null,null,acceptType)){ //判断是腾讯h5刷脸的域名
            this.uploadMsg=uploadMsg;
            this.acceptType=acceptType;
WBH5FaceVerifySDK.getInstance().setmUploadMessage(uploadMsg);
            if (activity!=null){
                //申请系统相机权限
                activity.requestCameraPermission(false,true);
            }
        }
     }

    // For Lollipop 5.0+ Devices  录制模式中，点击h5页面的【开始录制】按钮后触发的系统方法
    @TargetApi(21)
    @Override
    public boolean onShowFileChooser(WebView webView, ValueCallback<Uri[]> filePathCallback,
FileChooserParams fileChooserParams) {
        if (WBH5FaceVerifySDK.getInstance().isTencentH5FaceVerify(webView,fileChooserParams,null)){ //判断是
腾讯h5刷脸的域名
            this.webView=webView;
            this.filePathCallback=filePathCallback;
            this.fileChooserParams=fileChooserParams;
WBH5FaceVerifySDK.getInstance().setmUploadCallbackAboveL(filePathCallback);
            if (activity!=null){
 //申请系统相机权限
                activity.requestCameraPermission(false,false);
            }
        }
        return true;
```

```
 }

    //录制模式中，拉起系统相机进行录制视频
    public boolean enterOldModeFaceVerify(boolean belowApi21){
        if (belowApi21){ // For Android < 5.0
            if (WBH5FaceVerifySDK.getInstance().recordVideoForApiBelow21(uploadMsg, acceptType, activity)) {
                return true;
            }else{
// todo 合作方如果其他的h5页面处理，则再次补充其他页面逻辑
}
        }else { // For Android >= 5.0
            if (WBH5FaceVerifySDK.getInstance().recordVideoForApi21(webView, filePathCallback,
activity,fileChooserParams)) {
                return true;
            } else{
// todo 合作方如果其他的h5页面处理，则再次补充其他页面逻辑
}

        }
        return false;
    }
```

注意

- 如果第三方已重写以上函数，只要将如上述所示的函数体内容添加至第三方的对应函数体首行即可。
- 如果第三方没有重写以上函数，则直接按上述所示重写。
- WebView 不要使用 layerType 属性，否则导致刷脸界面白屏。

5. 重写 Activity
   WebView 所属的 Activity 必须重写如下函数：

注意

- 如果第三方 WebView 所属的 Activity 已重写以下函数，则将如下所示的函数体内容添加至第三方的对应函数体首行即可。
- 如果第三方 WebView 所属的 Activity 没有重写以下函数，则直接按以下代码重写。

```
@Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        Log.d(TAG, "onActivityResult --------"+requestCode);
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == VIDEO_REQUEST) {//录制模式中，调用系统相机录制完视频后再回到当前app的回调
            if (WBH5FaceVerifySDK.getInstance().receiveH5FaceVerifyResult(requestCode, resultCode, data)) {
                return;
            }
        }else if (requestCode==PERMISSION_QUEST_TRTC_CAMERA_VERIFY){ //trtc模式中，申请相机权限时，从系统设置页面跳
转回当前app页面的处理。由于权限申请逻辑demo仅供参考，合作方自己处理即可。
            requestCameraPermission(true,belowApi21);
        }else if (requestCode==PERMISSION_QUEST_OLD_CAMERA_VERIFY){//录制模式中，申请权限时，从系统设置页面跳转回当
前app页面的处理。由于权限申请逻辑demo仅供参考，合作方自己处理即可。
            requestCameraPermission(false,belowApi21);
        }
    }
```

6. 权限拒绝的处理
   在录制模式中，当用户点击了**开始录制**后，如果没有授权相机和录制权限，会弹框让用户授权。如果用户拒绝授权的话，一定要调用下面的方法：

```
WBH5FaceVerifySDK.getInstance().resetReceive();//https://www.teachcourse.cn/2224.html
```

可参看 demo 的调用和备注，demo 下载地址：人脸核身控制台（请到控制台自助接入列表页获取密码）。

Harmony Next 接入

demo下载地址：人脸核身控制台（请到控制台自助接入列表页获取密码）。

1. 申请权限

a.在 model.json5文件中增加申请以下权限

```
  "requestPermissions": [
      {
        "name": "ohos.permission.INTERNET",
      },
      {
        "name" : "ohos.permission.CAMERA",
        "reason": "$string:app_name",
        "usedScene": {
          "abilities": [
            "EntryAbility"
          ],
  "when":"inuse"
        }
      }
    ]
```

2. WebviewController 的设置

调用 Web 前添加如下代码给 WebviewController 设置 ua。

```
@State controller: webview.WebviewController = new webview.WebviewController();
```

3. 创建 Web 组件

```
Web({
        src: 'https://kyc.qcloud.com/s/web/h5/#/entry', //设置加载的实时模式刷脸h5页面的url地址
        controller: this.controller  //给web设置controller
      })
        .onControllerAttached(() => {
          this.controller.setCustomUserAgent(this.controller.getCustomUserAgent() + ';kyc/h5face;hoskyc/2.0')
//设置ua
        })
        .fileAccess(true)//web的配置，不能少
        .javaScriptAccess(true)//web的配置，不能少
```

4. web 组件配置实时模式

```
onPermissionRequest((event) => { // 终端页面收到h5刷脸实时模式的请求
          if (event) {
            this.checkPermission().then(() => { //1. 终端申请相机权限成功
              event.request.grant(event.request.getAccessibleResource()) // 2. 授权h5页面
            }).catch((error: BusinessError) => { // 终端申请相机权限失败
              console.error(TAG, "申请权限异常" + error.message)
              event.request.deny() //2.告诉h5页面没有权限
            })
          }
        })
// 注意：checkersmission()是申请相机的方法，接入方可以根据自身业务需求发挥，demo的这个方法仅供参考。
```

5. web 组件配置传统录制模式

```
.onShowFileSelector((event) => { //终端页面收到h5刷脸传统录制模式的请求
          if (event) {
```

```
        let result = picCamera().then(result => { //1.打开系统相机
          let str: string[] = [result.resultUri]
          event.result.handleFileList(str) // 2. 把录制的视频传给h5页面
        }).catch((error: BusinessError) => { //打开系统相机异常
          console.error(TAG, "打开相机异常" + error.message)
        });
      }
      return true;
    })
```

Uniapp 接入（ Android ）

1. 注册并创建 uni-app 开发环境。
   uni-app 开发接　具体参照 uni 官　。
2. 下载 demo 并根据指引配置插件。
   demo下 载地址：人脸核身控制台（请到控制台自助接入列表页获取密码）。
   - 在 uni-app 工程 nativeplugins 目录下，放置 only android 插件以及插件的配置文件。

   - 在 uni-app 页面中调用插件方法，实现 H5 刷脸功能。

```
const h5FaceVerifyPlugin  = uni.requireNativePlugin('DC-WBH5FaceVerifyService');
  export default {
          methods: {
                  enterH5FaceVerify() {
                          let url="https://kyc1.qcloud.com/s/web/h5/#/entry";//拉起h5刷脸的url
                          let thirdurl="https://www.qq.com/";//h5刷脸完成后要跳转的接入方的url，这个接入方填写自
己的url
                          h5FaceVerifyPlugin.startH5FaceVerify({h5faceurl:url,
                          h5thirdurl:thirdurl},result => {
                                  console.log(result,"H5刷脸后跳转到thirdurl所在h5页面的回调");
                                  h5FaceVerifyPlugin.destroyH5Activity(null);//调用关闭插件的webView.
                                  //uniapp todo 接入方自己的逻辑

                          },result=>{
                                  //这里是终端接受h5页面的消息回调。uniapp与h5页面两者通信可通过这个回调作为中间桥接
实现。
                                  //注意：约定h5页面和webView通信通过JavaScriptInterface接口和JavaScript进行交
互。
                                  //在H5页面中使用window.tencentApi.postMessage的方式来调用这个方法，参数为
String类型。
                                  //如果是jsonobject需要转String
                                  console.log(result,"自定义回调");
                                  //uniapp todo 接入方自己的逻辑
                          });
                          console.log("click=======意愿性刷脸====>startH5FaceVerify");
                  }
```

```
            }
        }
```

> **注意**
> 调用 destroyH5Activity() 可主动关闭插件。

合作方后台上传身份信息

最近更新时间：2025-01-03 16:04:42

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 获取 SIGN ticket 。
- SIGN 类型的 ticket，其有效期为60分钟，且可多次使用。
- 合作方为人脸核身服务生成签名，需要具有以下参数：

参数	说明	来源
appId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
userId	用户唯一标识，同一个用户的 userId 请保持一致，不同用户请不要使用同一个 userId，我们会根据 userId 来做登录态校验	合作方自行分配，需跟用户绑定，32位以内，不包含特殊字符
version	参数值为：1.0.0	-
ticket	合作伙伴服务端获取的 ticket，注意是 SIGN 类型	获取方式请参见 获取 SIGN ticket
nonce	必须是32位随机数	合作方自行生成

> **注意**
> 参与签名的数据需要与使用该签名的 SDK 中的请求参数保持一致。

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 nonce、userId、appId 连同 ticket、version 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**
> Java 签名算法可参考 签名算法说明 。

参考示例

- 请求参数：

参数	参数值
appId	IDAXXXXX
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
ticket	XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

- 字典排序后的参数为：

```
[1.0.0, IDAXXXXX, XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS,
kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551]
```

- 拼接后的字符串为：

```
1.0.0IDAXXXXXXO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7
TuserID19959248596551
```

- **计算 SHA1 得到签名：**
  该字符串就是最终生成的签名（40位），不区分大小写。

```
D7606F1741DDCF90757DA924EDCF152A200AC7F0
```

合作方后台上传身份信息

请求

- **请求 URL：** https://kyc1.qcloud.com/api/server/getAdvFaceId?orderNo=xxx

> **注意**
> **为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。**

- **请求方法：** POST
- **报文格式：** Content-Type: application/json
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
orderNo	订单号，字母/数字组成的字符串，由合作方上传，每次唯一，不能超过32位	String	不能超过 32 位	是
name	姓名	String	-	使用权威源比对时：姓名+证件号必须输入 使用自带源比对时：姓名+证件号可不输入
idNo	证件号码	String	-	使用权威源比对时：姓名+证件号必须输入 使用自带源比对时：姓名+证件号可不输入
userId	用户 ID，用户的唯一标识（不能带有特殊字符），需要跟生成签名的 userId 保持一致。同一个用户的 userId 请保持一致，不同用户请不要使用同一个userId，我们会根据 userId 来做登录态校验	String	不能超过 32 位	是
sourcePhotoStr	比对源照片，注意：原始图片不能超过500k，且必须为 JPG 或 PNG、BMP 格式。参数有值：使用合作伙伴提供的比对源照片进行比对，必须注意是正脸可信照片，照片质量由合作方保证参数为空：根据身份证号 + 姓名使用权威数据源比对	BASE64String	1048576	否，非必填请使用标准的图片转base64方法，base64编码不可包含换行符，不需要加前缀
sourcePhotoType	比对源照片类型参数值为1 时是：水纹正脸照参数值为 2 时是：高清正脸照重要提示：照片上无水波纹的为高清照，请勿传错，否则影响比对准确率。如有疑问，请联系腾讯云技术支持线下确认	String	1	否，提供比对源照片需要传
liveInterType	活体交互模式参数值为1时，表示仅使用实时检测模式，不兼容的情况下回调错误码3005参数值非1或不入参，表示优先使用实时检测模式，如遇不兼容情况，自动降级为视频录制模式	String	1	否
version	默认参数值为：1.0.0	String	20	是
sign	签名：使用上面生成的签名	String	40	是

nonce	随机数	String	32	是

水纹照示例

响应

响应参数：

参数	类型	说明
code	String	0：成功 非0：失败 详情请参见 SaaS服务错误码
msg	String	请求结果描述
bizSeqNo	String	请求业务流水号
orderNo	String	订单编号
faceId	String	此次刷脸用户标识
transactionTime	String	接口请求的时间
optimalDomain	String	启动 H5 人脸核身步骤中调用 login 接口使用的域名

响应示例：


```
{
"code":"0",
"msg":"请求成功",
"bizSeqNo":"21062120001184438418322908010297",
"result":{
        "bizSeqNo":"21062120001184438418322908010297",
        "transactionTime":"20210621183229",
        "orderNo":"16170918856091743257691658 50",
        "faceId":"tx0375fa5243984381ea7b7013f13795",
        "optimalDomain":"kyc1.qcloud.com",
        "success":false
        },
"transactionTime":"20210621183229"
}
```


> **说明：**
> success：false 无意义，合作伙伴可忽略，无需处理。

> faceId 有效期为5分钟，每次进行人脸核身都需要重新获取。

启动 H5 人脸核身

最近更新时间：2024-06-17 17:09:41

前置条件

合作方如果使用 App 内调起 H5 人脸核身，需要 App 平台的 webkit/blink 等组件支持调用摄像头录视频，方可正常使用人脸核身功能。

说明

1. App 内调用 H5 人脸核身或者短信链接调用 H5 人脸核身等场景，若当前设备无法正常调用摄像头进行视频录制，前端将返回特定的错误码（如下）告知合作方，合作方拿到错误码后可选择用备用方案核身处理。
2. 在 App、微信公众号、浏览器中调用H5实时检测人脸核身，需要用户允许使用摄像头权限（授权操作形式包括弹窗、动作菜单、应用设置等，具体形式视手机型号而异）。如用户误操作导致拒绝授权，需要退出人脸核身并重新进入和允许授权。部分浏览器会缓存用户之前的授权操作，故如果退出人脸核身并重新进入时仍然出现无摄像头权限的情况，可以尝试清除浏览器缓存。

返回码	返回信息	处理措施
3001	该浏览器不支持视频录制	请使用其它验证方案
3002	登录态异常，cookie 参数缺失	重新进入
3003	人脸核身中途中断	重新进入
3004	无摄像头权限	重新进入并授权摄像头
3005	该浏览器不支持实时检测模式	请使用其它浏览器
300101	报文包体问题	重新进入

摄像头授权操作示例：

形式1：弹窗（popup）

形式2：动作菜单（action sheet）

形式3：应用设置（setting）

生成签名

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 。合作方根据本次人脸核身的如下参数生成签名,需要签名的参数信息如下：

> **说明**
> 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	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，本次人脸核身合作伙伴上传的订单号，字母/数字组成的字符串，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（与接口中使用的 userId 保持一致）
version	参数值为：1.0.0	-
faceId	getAdvFaceId接口返回的唯一标识	-
ticket	合作伙伴服务端获取的 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（与接口中的随机数保持一致）

基本步骤

1. 生成一个32位的随机字符串（字母和数字）nonce（接口请求时也要用到）。
2. 将 appId、userId、orderNo、version、faceId 连同 ticket、nonce 共7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）。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	appId001
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
faceId	bwiwe1457895464
orderNo	aabc1457895464
ticket	zxc9QfxIti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

- **字典排序后的参数为：**

```
[1.0.0, aabc1457895464, appId001, bwiwe1457895464, kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551,
zxc9QfxIti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS]
```

- **拼接后的字符串为：**

```
1.0.0aabc1457895464appId001bwiwe1457895464kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7TuserID19959248596551zxc9QfxIti9iTV
gHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS
```

- **计算 SHA1 得到签名：**

```
4E9DFABF938BF37BDB7A7DC25CCA1233D12D986B  该字串就是最终生成的签名（40位），不区分大小写。
```

启动 H5 人脸核身

合作方上传 faceId 以及 sign，后台校验 sign 通过之后重定向到 H5 人脸核身。

> **注意**
> 为了保证服务的高可用，全面消除单点风险，我们启用了多域名服务。启动 H5 人脸核身需要使用 合作方后台上传身份信息 接口返回内容中 optimalDomain 字段的域名。

请求 URL：

```
https://{optimalDomain}/api/web/login
```

optimalDomain 使用 合作方后台上传身份信息 接口返回的 optimalDomain 域名，如果 optimalDomain 字段返回为空或者取不到，默认使用域名 `kyc1.qcloud.com`。
该跳转 url 不能直接暴露在前端 html 页面的 `<a>` 标签中。某些浏览器会预加载 `<a>` 标签中的 url，导致用户点击访问该 url 时，因 url 已经被预加载访问过，于是签名失效，页面报错“签名不合法”。
请求方法：GET
请求参数：

参数	说明	类型	长度	是否必填
appId	参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	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，由合作方上传，字母/数字组成的字符串,每次唯一，此信息为本次人脸核身上传的信息，不能超过32位	String	32	是
faceId	getAdvFaceId接口返回的唯一标识	String	32	是
url	H5 人脸核身完成后回调的第三方 URL，**需要第三方提供完整 URL 且做 URL Encode**。完整 URL Encode 示例： • 原 URL：`https://cloud.tencent.com` • Encode 后：`https%3a%2f%2fcloud.tencent.com`	String	-	是
resultType	是否显示结果页面，参数值为“1”时直接跳转到 url 回调地址，null 或其他值跳转提供的结果页面	String	-	否，非必填
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
from	browser：表示在浏览器启动刷脸；App：表示在 App 里启动刷脸，默认值为 App	String	-	是
redirectType	跳转模式，参数值为“1”时，刷脸页面使用 replace 方式跳转，不在浏览器 history 中留下记录；不传或其他值则正常跳转	String	-	否，非必填

请求示例：

```
https:// kyc1.qcloud.com/api/web/login?appId=appId001
&version=1.0.0
&nonce=4bu6a5nv9t678m2t9je5819q46y9hf93
&orderNo=16170918856091743257691658 5
&faceId=wb04f10695c3651ce155fea7070b74c9
&url=https%3a%2f%2fcloud.tencent.com
&from=browser
&userId=23333333333333
&sign=5DD4184F4FB26B7B9F6DC3D7D2AB3319E5F7415F
&redirectType=1
```

H5 人脸核身结果跳转

最近更新时间：2024-04-12 15:35:32

认证结束后，认证结果页会通过前端跳转启动 H5 人脸核身入参中指定的回调 URL 地址。并带有 code、orderNo、h5faceId 等参数。
您可以根据我们刷脸完成后的回调请求参数中的 code 判断是否核身通过，code 0 表示人脸核身成功，其他错误码 表示失败。如果您需要二次验证结果准确性，您可以通过 服务端获取验证结果 获取最终认证结果。服务端获取验证结果能够返回人脸核身的结果以及相应的视频和图片用于您存证等其他需要。（请注意：务必在收到完成刷脸的回调后，再来调用服务端获取验证结果。）
参数：

参数	说明	类型	长度（字节）
code	人脸核身结果的返回码，0 表示人脸核身成功，其他错误码表示失败。	字符串	-
orderNo	订单号，本次人脸核身上送的订单号。	字符串	32
h5faceId	本次请求返回的唯一标识，此信息为本次人脸核身上传的信息。	字符串	32
newSignature	对 URL 参数 App ID、orderNo 和 SIGN ticket、code 的签名。	字符串	40
liveRate	本次人脸核身的活体检测分数。	字符串	40

注意

- 若 H5 浏览器不支持调用摄像头录制视频，则直接返回前端错误码 code（详见 启动 H5 人脸核身 的前置条件）。
- 在 H5 实时检测模式中，若人脸核身过程中发生中断，例如 App 被切换到后台等情况，前端会返回特定的错误码3003（人脸核身中途中断）。

人脸核身结果查询
查询核身结果

最近更新时间：2024-06-17 17:09:41

当您的用户完成核身认证后，如果您需要拉取人脸核身的视频和图片用于存证等其他需要，您可以调用查询核身结果接口来获取。
重要提示：
1. 您需要在前端完成刷脸的回调后，再来调用查询核身结果接口获取刷脸视频和照片。
2. 人脸核身完成后的相关业务数据请尽快拉取，超过人脸核身服务必须最短缓存时间的业务数据将完全清理。

> **注意**
> 当您的 App 接入的是我们的基础版或增强版 SDK 服务时，SDK 回调中只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过查询核身结果接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，查询核身结果接口无法查询到刷脸结果。

步骤如下：

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，获取方式请参见 获取 SIGN ticket 。
- 合作方为人脸核身服务生成签名，需要具有以下参数：

参数	说明	来源
appId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，字母/数字组成的字符串，本次人脸核身合作伙伴上送的订单号，唯一标识	合作方自行分配，不要带有特殊字符
version	默认值：1.0.0	-
ticket	合作伙伴服务端缓存的 ticket，注意是 SIGN 类型	获取方式请参见 获取 SIGN ticket
nonce	32位随机字符串，由字母和数字组成	合作方自行生成，不要带有特殊字符

基本步骤

1. 生成一个32位的随机字符串 nonce（由字母和数字组成，登录时也要用到）。
2. 将 appId、orderNo、version、ticket、nonce 共5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 。

人脸核身结果查询接口(升级)

请求

- **请求 URL：** `https://kyc1.qcloud.com/api/v2/base/queryfacerecord?orderNo=xxx`

> **注意**
> 为方便查询耗时，该请求url后面请拼接 orderNo 订单号参数。

- **请求方法：**POST
- **报文格式：**Content-Type: application/json
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填

appId	腾讯云控制台申请的 appid	String	8	是
version	版本号，默认值：1.0.0	String	20	是
nonce	随机数	String	32	是
orderNo	订单号，字母/数字组成的字符串,是您需要查询结果的人脸核身订单号	String	32	是
sign	签名值，使用本页第一步生成的签名	String	40	是
getFile	是否需要获取人脸识别的视频和文件，值为1则返回视频和照片、值为2则返回照片、值为3则返回视频；其他则不返回	String	1	否
queryVersion	查询接口版本号(传1.0则返回 sdk 版本号和 trtc 标识)	String	8	否

响应

- **响应参数：**

参数	类型	说明
code	String	0：表示身份验证成功且认证为同一人
msg	String	返回结果描述
bizSeqNo	String	业务流水号
transactionTime	String	请求接口的时间
orderNo	String	订单编号
liveRate	String	活体检测得分
similarity	String	人脸比对得分
occurredTime	String	进行刷脸的时间
photo	Base 64 string	人脸核身时的照片，base64 位编码
video	Base 64 string	人脸核身时的视频，base64 位编码
sdkVersion	String	人脸核身时的 sdk 版本号
trtcFlag	String	Trtc 渠道刷脸则标识"Y"
appId	String	腾讯云控制台申请的 appid

- **响应示例：**


```
{"code":"0",
 "msg":"请求成功",
 "bizSeqNo":"2203292000118445321117401579089 4",
 "result":{
       "orderNo":"161709188560917432576916 5852",
       "liveRate":"99",
       "similarity":"88.01",
       "occurredTime":"20220329104717",
       "appId":"IDAXXXXX",
       "photo":"******",
       "video":"******",
       "bizSeqNo":"2203292000118445321117401579089 4",
       "sdkVersion":"1.12.12",
```

```
        "trtcFlag":"Y"},
  "transactionTime":"20220329111740"
  }
```

code 非 0 时，有时不返回图片和视频。

注意事项

- 照片和视频信息作为存证，合作伙伴可以通过此接口拉取视频等文件，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，默认不返回视频和照片信息。
- 由于照片和视频信息有可能超过 1M，考虑传输的影响，建议合作伙伴在使用时注意，建议获取比对结果用于后续流程处理和存证使用分开调用。避免网络传输带来的影响。
- 如果合作方是完成人脸核身流程后马上去拉取视频/照片，建议在查询接口加上一个查询机制：判断图片是否存在，轮询3次，每次2s。
- 照片和视频均为 base64 位编码，其中照片解码后格式一般为 JPG 和 PNG。视频格式解码后一般为 MP4。
- 服务端验证结果接口返回66660011无此查询结果的可能原因：

  1.1 66660017 验证次数超限后被风控，风控后的订单为无效，查询结果为无此查询结果。

  1.2 用户中途退出刷脸，没有完成人脸验证的流程，没有比对，查询结果为无此查询结果。

  1.3 66660018 操作超时，请退出重试 无此 ID 的用户身份信息，H5faceid 5分钟有效期，失效后无法进入刷脸流程，查询结果为无此查询结果。

  1.4 66660016 视频格式或大小不合法 文件或视频不合法，无法进行比对，查询结果为无此查询结果。

  1.5 400604 上传的视频非实时录制,被时间戳校验拦截，查询结果为无此查询结果。

  1.6 查询超过3天的订单返回无此查询结果。
- 响应参数请勿做强校验，后续可能会有扩展。

服务端响应码

详情请参见 SaaS 服务错误码 。

人脸认证多张照片查询接口

最近更新时间：2024-06-17 17:09:41

人脸认证多张照片查询接口：获取人脸认证结果多张照片的接口。

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 。
- **合作方为人脸验证服务生成签名，需要具有以下参数：**

参数	说明	来源
appId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，字母/数字组成的字符串，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，不要带有特殊字符
version	默认值：1.0.0	-
ticket	合作伙伴服务端缓存的 ticket，注意是 SIGN 类型	获取方式见 SIGN ticket 获取
nonce	32位随机字符串，字母和数字	合作方自行生成，不要带有特殊字符

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 appId、orderNo、version 连同 ticket、nonce 共5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 。

人脸认证多张照片查询接口

请求

请求URL：https://kyc1.qcloud.com/api/v2/base/queryphotoinfo?orderNo=xxx

> **注意**
> 为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。

请求方法：POST
HTTP 请求 header：

参数名	是否必选	类型	说明
Content-Type	是	String	application/json

请求参数：

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
version	版本号，默认值：1.0.0	String	20	是
nonce	随机数	String	32	是

orderNo	订单号，字母/数字组成的字符串，合作方订单的唯一标识	String	32	是
sign	签名值，使用本页第一步生成的签名	String	40	是

响应

返回参数说明：

参数名	类型	说明
code	int	0：成功 非0：失败
msg	String	请求结果描述
bizSeqNo	String	请求业务流水号
orderNo	String	订单编号
occurredTime	String	刷脸时间（yyyyMMddHHmmss）
photoList	List	Base64 图像列表（返回1 – 3张照片），若照片不存在，则返回 null

返回示例：

```
{
"code": 0,
"msg": "请求成功",
"bizSeqNo": "业务流水号",
"result": {
"orderNo": "AAAAAA001",
"occurredTime": "20180907142653",
"photoList": ["第一个base64photo字符串", "第二个base64photo字符串", "第三个base64photo字符串"]
}
}
```

说明

success：false 无意义，合作伙伴可忽略，无需处理。

基础版人脸核身
App SDK
合作方后台上传身份信息

最近更新时间：2025-01-07 10:21:42

> **注意**
> 如果因自身业务需要对人脸核身的影像文件进行存储或其他用途，请合作方务必自行保存订单号，通过订单号拉取人脸核身的影像文件是唯一方式。

生成签名

准备步骤

- 下载 SDK，请到控制台自助接入列表页获取密码。
- uni 插件接入： https://ext.dcloud.net.cn/plugin?id=10378
- Normal 插件地址：https://ext.dcloud.net.cn/plugin?id=1491
- **前置条件：请合作方确保 SIGN ticket 已经正常获取，获取方式请参见 获取 SIGN ticket。**
- **SIGN 类型的 ticket，其有效期为60分钟，且可多次使用。**
- **合作方为人脸核身服务生成签名，需要具有下表中的参数：**

参数	说明	来源
appId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
userId	用户唯一标识，同一个用户的 userId 请保持一致，不同用户请不要使用同一个userId，我们会根据 userId 来做防重复点击优化以及登录态校验	合作方自行分配，需跟用户绑定，32位以内，不包含特殊字符
version	参数值为：1.0.0	-
ticket	合作伙伴服务端获取的 ticket，注意是 SIGN 类型	获取方式请参见 获取 SIGN ticket
nonce	必须是32位随机数	合作方自行生成

> **注意**
> 参与签名的数据需要与使用该签名的 SDK 中的请求参数保持一致。

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 appId、userId、version 连同 ticket、nonce 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**
> Java 签名算法可参考 签名算法说明 。

参考示例

- **请求参数：**

参数	参数值
appId	IDAXXXXX
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T

version	1.0.0
ticket	XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

- **字典排序后的参数为：**

```
[1.0.0, IDAXXXXX, XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS,
kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551]
```

- **拼接后的字符串为：**

```
1.0.0IDAXXXXXXO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7
TuserID19959248596551
```

- **计算 SHA1 得到签名：**
  该字符串就是最终生成的签名（40位），不区分大小写。

```
D7606F1741DDCF90757DA924EDCF152A200AC7F0
```

合作方后台上传身份信息

请求

- **请求 URL：** https://kyc1.qcloud.com/api/server/getfaceid?orderNo=xxx

> ⚠ **注意**
> **为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。**

- **请求方法：** POST
- **报文格式：** Content-Type: application/json
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
orderNo	订单号，字母/数字组成的字符串，由合作方上传，每次唯一，不能超过32位	String	不能超过32位	是
name	姓名	String	-	使用权威源比对时：姓名+证件号必须输入 使用自带源比对时：姓名+证件号可不输入
idNo	证件号码	String	-	使用权威源比对时：姓名+证件号必须输入 使用自带源比对时：姓名+证件号可不输入
userId	用户 ID，用户的唯一标识（不能带有特殊字符），需要跟生成签名的 userId 保持一致。同一个用户的 userId 请保持一致，不同用户请不要使用同一个userId，我们会根据 userId 来做防重复点击优化以及登录态校验	String	不能超过32位	是
sourcePhotoStr	比对源照片，注意：原始图片不能超过500k，且必须为 JPG 或 PNG、BMP 格式。 参数有值：使用合作伙伴提供的比对源照片进行比对，必须注意是正脸可信照片，照片质量由合作方保证 参数为空：根据身份证号 + 姓名使用权威数据源比对	BASE64 String	1048576	否，非必填请使用标准的图片转base64方法，base64编码不可包含换行符，不需要加前缀 **注意：**只有您需要使用自带比对源比对时上传，使用权威比对源比对时请不要上传该字段

sourcePhotoType	比对源照片类型 参数值为1 时是：水纹正脸照 参数值为 2 时是：高清正脸照 重要提示：照片上无水波纹的为高清照，请勿传错，否则影响比对准确率。 如有疑问，请联系腾讯云技术支持线下确认	String	1	否，提供比对源照片需要传 **注意：**只有您需要使用自带比对源比对时上传，使用权威比对源比对时请不要上传该字段
version	默认参数值为：1.0.0	String	20	是
sign	签名：使用上面生成的签名	String	40	是
nonce	随机数	String	32	是

水纹照示例

响应

响应参数：

参数	类型	说明
code	String	0：成功 非0：失败 详情请参见 基础版人脸核身服务错误码
msg	String	请求结果描述
bizSeqNo	String	请求业务流水号
orderNo	String	订单编号
faceId	String	此次刷脸用户标识，调 SDK 时传入

响应示例：

```
{
    "code": "0",
    "msg": "请求成功",
    "bizSeqNo": "业务流水号",
    "result": {
        "bizSeqNo": "业务流水号",
        "transactionTime": "20201019110305",
        "orderNo": "合作方订单号",
        "faceId": "175177e03bc53d57222418e18c731488",
        "success": false
    },
    "transactionTime": "20201019110305"
}
```

说明
success：false 无意义，合作伙伴可忽略，无需处理。
faceId 有效期为5分钟，每次进行人脸核身都需要重新获取。

生成 SDK 接口调用步骤使用签名

最近更新时间：2024-12-13 11:08:32

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 。
- NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。
- 合作方为人脸核身服务生成签名，需要具有以下参数：

参数名	说明	来源
appId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
userId	用户唯一标识	合作方自行分配（和 SDK 里面定义的 userId 保持一致）
version	参数值为：1.0.0	-
ticket	合作伙伴服务端获取的 ticket，注意是 NONCE 类型	获取方式见 NONCE ticket 获取
nonce	必须是32位随机数	合作方自行生成（和 SDK 里面定义的随机数保持一致）

> **注意**
> 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。

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 appId、userId、version 连同 ticket、nonce 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	IDAXXXXX
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
ticket	XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

- 字典排序后的参数为：

```
[1.0.0, IDAXXXXX, XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS ,
kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551]
```

- 拼接后的字符串为：

```
1.0.0IDAXXXXXXO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7
TuserID19959248596551
```

- 计算 SHA1 得到签名：

```
D7606F1741DDCF90757DA924EDCF152A200AC7F0
```

该字符串就是最终生成的签名（40 位），不区分大小写。

Android 人脸核身
开发准备

最近更新时间：2024-04-28 17:50:51

权限检测

- **Android 6.0 以上系统**
  SDK 需要用到相机，对 Android 6.0 以上的系统会做权限的运行时检测。
- **Android 6.0 以下系统**
- 由于 Android 6.0 以下系统 Android 并没有运行时权限，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出现异常，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，建议合作方在拉起 SDK 前，帮助 SDK 做相机权限检测，提示并确认用户打开了这项权限后，再进行人脸核身，可以使整个人脸核身体验更快更好。

CPU 平台设置

目前 SDK 只支持 armeabi-v7a，armeabi-v8a 平台，为了防止在其他 CPU 平台上 SDK crash，建议在您的 App 的 build.gradle 里加上 abiFilter，如下代码所示，您也可以根据您的 App CPU 平台情况进行相应的设置和过滤：

> ⚠ **注意**
> SDK 支持 armeabi-v7a 和 armeabi-v8a 两个平台，合作方可以根据自身情况设置需要的 CPU 平台。

```
defaultConfig {
    ndk {
          //设置支持的so库框架
          abiFilters 'armeabi-v7a', 'arm64-v8a'
     }
}
```

配置流程

最近更新时间：2024-06-05 14:33:31

接入配置

注意事项

- 人脸核身 SDK（WbCloudFaceLiveSdk）最低支持到 Android API 16: Android 4.1.0（ICS），请在构建项目时注意。
- 人脸核身 SDK 将以 AAR 文件的形式提供，默认黑色皮肤，无需格外设置。
- 人脸核身 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。
- 需要为人脸核身 SDK 添加依赖，方式如下：
  将提供的 AAR 文件加入到 App 工程的 libs 文件夹下，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-v7
 compile 'com.android.support:appcompat-v7:23.0.1'
 //1. 云刷脸SDK，
//aar的名称，例如：WbCloudFaceLiveSdk-v6.0.0-1234567.aar，填入'WbCloudFaceLiveSdk-v6.0.0-1234567'
 compile(name: 'aar的名称', ext: 'aar')
//2. 云normal SDK，
//aar的名称，例如：WbCloudNormal-v5.1.10-123456789.aar，填入 'WbCloudNormal-v5.1.10-123456789.aar'
 compile(name: 'aar的名称', ext: 'aar')
}
    }
```

混淆配置

云刷脸产品的混淆规则如下：

云刷脸 SDK 的混淆规则

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 kyc-cloud-face-consumer-proguard-rules.pro 拷贝到主工程根目录下，然后通过 -include kyc-cloud-face-consumer-proguard-rules.pro 加入到您的混淆文件中。

```
#####################云刷脸混淆规则    faceverify-BEGIN#########################
####
#不混淆内部类
-keepattributes InnerClasses

#不混淆jni调用类
-keepclasseswithmembers class *{
    native <methods>;
}

##################### faceverify-BEGIN ##########################
-ignorewarnings
-keep public class com.tencent.ytcommon.**{*;}
-keep class com.tencent.turingfd.sdk.mfa.TNative$aa { public *; }
-keep class com.tencent.turingfd.sdk.mfa.TNative$aa$bb { public *;}
-keep class com.tencent.turingcam.** {*;}
-keep class com.tencent.turingfd.** {*;}

-keep public class com.tencent.youtu.ytagreflectlivecheck.jni.**{*;}
```

```
-keep public class com.tencent.youtu.ytagreflectlivecheck.YTAGReflectLiveCheckInterface{
    public <methods>;
}
-keep public class com.tencent.youtu.ytposedetect.jni.**{*;}
-keep public class com.tencent.youtu.ytposedetect.data.**{*;}
-keep public class com.tencent.youtu.liveness.YTFaceTracker{*;}
-keep public class com.tencent.youtu.liveness.YTFaceTracker$*{*;}

-keep public class com.tencent.cloud.huiyansdkface.facelight.net.*$*{
    *;
}
-keep public class com.tencent.cloud.huiyansdkface.facelight.net.**{
    *;
}
-keep public class com.tencent.cloud.huiyansdkface.facelight.config.cdn.WbUiTips{
    *;
}

#================数据上报混淆规则 start==========================
#实体类
-keep class com.tencent.cloud.huiyansdkface.analytics.EventSender{
    *;
}
-keep class com.tencent.cloud.huiyansdkface.analytics.EventSender$*{
    *;
}
-keep class com.tencent.cloud.huiyansdkface.analytics.WBSAEvent{
     *;
}
-keep class com.tencent.cloud.huiyansdkface.analytics.WBSAParam{
     *;
}
#================数据上报混淆规则 end==========================

#######################faceverify-END##############################

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

-keep public class com.tencent.cloud.huiyansdkface.normal.net.*$*{
    *;
}
-keep public class com.tencent.cloud.huiyansdkface.normal.net.*{
    *;
}
#bugly
-keep class com.tencent.bugly.idasc.**{
    *;
}
#wehttp混淆规则
-dontwarn com.tencent.cloud.huiyansdkface.okio.**
-keep class com.tencent.cloud.huiyansdkface.okio.**{
    *;
}
-dontwarn com.tencent.cloud.huiyansdkface.okhttp3.OkHttpClient$Builder

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

接口调用

最近更新时间：2025-06-06 17:12:52

SDK 接口调用方法

SDK 代码调用的入口为 `WbCloudFaceVerifySdk` 这个类。

```
public class WbCloudFaceVerifySdk {

/**
* 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 WbCloudFaceVerifySdk getInstance(){
//    ...
}
/**
* 在使用 SDK 前先初始化，传入需要的数据 data
* 由 WbCloudFaceVerifyLoginListener 返回是否登录 SDK 成功
* 关于传入数据 data 见后面的说明
*/
public void initSdk(Context context,Bundle data,
WbCloudFaceVerifyLoginListener loginListerner){
//    ...
}
    /**
     * 登录成功后，调用此函数拉起 sdk 页面。
     * 由 startWbFaceVerifySdk 返回刷脸结果。
     */
public void startWbFaceVerifySdk(Context ctx,
WbCloudFaceVerifyResultListener listener) {    // ...
}
/**
 * 释放资源，防止内存泄漏。收到刷脸结果后即可调用
 */
public void release() {
}
```

- NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。
- `WbCloudFaceVerifySdk.initSdk()` 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以下内容（参数详情请参见 接口参数说明）：

```
//这些都是 WbCloudFaceVerifySdk.InputData 对象里的字段，是需要传入的数据信息
String faceId;  //此次刷脸用户标识，合作方需要向人脸识别后台拉取获得，详见获取 faceId 接口
String agreementNo;  //订单号
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

//刷脸类别：分级活体 FaceVerifyStatus.Mode.GRADE
FaceVerifyStatus.Mode verifyMode;
String keyLicence;   //在人脸核身控制台内申请
```

> **注意**
> 以上参数被封装在 WbCloudFaceVerifySdk.InputData 对象中（它是一个 Serializable 对象）。

接入示例

详情请参见 Android 光线活体 + 人脸比对 接入示例 。

接口参数说明

InputData 对象说明

NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。
InputData 是用来给 SDK 传递一些必须参数所需要使用的对象（WbCloudFaceVerifySdk.initSdk() 的第二个参数），合作方需要传入 SDK 需要的一些数据以便启动刷脸 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（字节）	是否必填
faceId	刷脸 ID 号，获取方式请参见 合作方后台上传身份信息	String	-	是
agreementNo	订单号，合作方订单的唯一标识	String	32	是
openApiAppId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
openApiAppVersion	接口版本号，默认填：1.0.0	String	20	是
openApiNonce	与服务端生成签名的随机数保持一致	String	32	是
openApiUserId	User Id，每个用户唯一的标识	String	32	是
openApiSign	获取方式请参考 生成 SDK 接口调用步骤使用签名	String	40	是
verifyMode	刷脸类型：分级模式 FaceVerifyStatus.Mode.GRADE	FaceVerifyStatus.Mode	-	是
keyLicence	在人脸核身控制台申请的 SDKlicense	String	以实际申请为准	是

个性化参数设置（可选）

`WbCloudFaceVerifySdk.initSdk()` 里 `Bundle data`，除了必须要传的 InputData 对象之外，还可以由合作方为其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 支持境外调用

SDK 默认不支持境外调用。如果合作方需支持境外调用，可以通过该字段进行设置。设置代码如下：

```
# 优先使用境外域名，默认false，不使用
# 请注意：如果合作方可以分辨国内或者境外ip，建议境外ip开启这个配置，国内ip访问不进行配置
data.putBoolean(WbCloudFaceContant.IS_ABROAD, true);
```

选择 SDK 提示语言类型

合作方可以选择 SDK 显示提示语的语言类型，默认简体中文，可以选择繁体中文，英语，日语，韩语，泰语和印尼语。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(WbCloudFaceContant.INPUT_DATA, inputData);
  //简体中文  WbCloudFaceContant.LANGUAGE_ZH_CN
  //繁体中文  WbCloudFaceContant.LANGUAGE_ZH_HK
  //英语  WbCloudFaceContant.LANGUAGE_EN
  //印尼语  WbCloudFaceContant.LANGUAGE_ID
  //日语  WbCloudFaceContant.LANGUAGE_JA
  //韩语  WbCloudFaceContant.LANGUAGE_KO
   //泰语 WbCloudFaceContant.LANGUAGE_TH
//默认设置为简体中文，此处设置为英文
data.putString(WbCloudFaceContant.LANGUAGE,WbCloudFaceContant.LANGUAGE_EN);
```

特别注意，在设置了国际化语言的情况下，SDK 不会播放语音，自定义退出框也无效。同时合作方设定的自定义提示语也需要合作方传入相应语种的提示语。

SDK 样式选择

合作方可以选择 SDK 样式。目前 SDK 有黑色模式和白色模式两种，默认显示白色模式。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(WbCloudFaceContant.INPUT_DATA, inputData);
  //对 sdk 样式进行设置，默认为白色模式
  //此处设置为白色模式
  data.putString(WbCloudFaceContant.COLOR_MODE, WbCloudFaceContant.WHITE);
```

SDK 还支持自定义皮肤，支持定制刷脸过程中各个组件的色值，因涉及可设置元素较多，此处可以参考接入 demo。使用自定义皮肤时需要设置：

```
# 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(WbCloudFaceContant.INPUT_DATA, inputData);
  //对 sdk 样式进行设置，默认为白色模式
  //此处设置为自定义模式
  data.putString(WbCloudFaceContant.COLOR_MODE, WbCloudFaceContant.CUSTOM);
```

可配置的颜色值如下：

```
# 在App的colors.xml中设置：
   <!--供客户定制颜色值，可修改仅自己需要修改的颜色，其余复制粘贴demo示例即可-->
    <!--请注意，根据客户定制的需求，有些色值除了需要在用户app的colors里加上以下所有配置之外，还需要同步增加mipmap-xhdpi和
drawble的xml文件-->
    <!--详情见下面说明，所有的定制res都以wbcf_custom开头，用户可以参考demo使用-->

    <!--授权页面还有一张背景图需要设置,请在mipmap文件夹里替换-->
    <!--授权页面titlebar背景颜色-->
    <color name="wbcf_custom_auth_title_bar">#ffffff</color>
    <!--授权页面titlebar返回键颜色-->
    <color name="wbcf_custom_auth_back_tint">#ffffff</color>
    <!--授权页面背景颜色-->
    <color name="wbcf_custom_auth_bg">#ffffff</color>
    <!--授权页面标题颜色-->
    <color name="wbcf_custom_auth_title">#1d2232</color>
    <!--授权页面文字颜色-->
    <color name="wbcf_custom_auth_text">#a5a7ad</color>
    <!--授权页面协议名称颜色-->
    <color name="wbcf_custom_auth_name_text">#1d2232</color>
    <!--授权页面按钮文字颜色-->
    <color name="wbcf_custom_auth_btn_text">#80ffffff</color>
    <!--授权页面被选中按钮背景色，需要配合drawable/wbcf_custom_auth_btn_checked.xml使用-->
    <color name="wbcf_custom_auth_btn_checked_bg">#1e2642</color>
    <!--授权页面未选中按钮背景色，需要配合drawable/wbcf_custom_auth_btn_unchecked.xml使用-->
    <color name="wbcf_custom_auth_btn_unchecked_bg">#9ca1ae</color>

    <!--识别页面背景色-->
    <color name="wbcf_custom_verify_bg">#ffffff</color>
    <!--识别页面初始化圆框-->
    <color name="wbcf_custom_initial_border">#33ffffff</color>
    <!--识别页面有脸时圆框颜色-->
    <color name="wbcf_custom_border">#ffb155</color>
    <!--识别页面有脸时提示文字颜色-->
    <color name="wbcf_custom_tips_text">#1d2232</color>
    <!--识别页面人脸不符合条件时圆框颜色-->
    <color name="wbcf_custom_border_error">#fa5a5a</color>
    <!--识别页面人脸不符合条件时提示文字颜色-->
    <color name="wbcf_custom_tips_text_error">#fa5a5a</color>
```

```
    <!--识别页面用户自定义提示文字颜色-->
    <color name="wbcf_custom_customer_tip_text">#5d646d</color>
    <!--识别页面长提示文字颜色-->
    <color name="wbcf_custom_long_tip_text">#777A84</color>
    <!--识别页面长提示背景颜色,需要配合drawable/wbcf_custom_long_tip_bg.xml使用-->
    <color name="wbcf_custom_long_tip_bg">#fafafa</color>

    <!--弹窗背景颜色-->
    <color name="wbcf_custom_dialog_bg">#ffffff</color>
    <!--弹窗标题文字颜色-->
    <color name="wbcf_custom_dialog_title_text">#363C62</color>
    <!--弹窗内容文字颜色-->
    <color name="wbcf_custom_dialog_text">#363C62</color>
    <!--弹窗右边按钮文字颜色-->
    <color name="wbcf_custom_dialog_right_text">#363C62</color>
    <!--弹窗左边按钮文字颜色-->
    <color name="wbcf_custom_dialog_left_text">#363C62</color>
```

合作方个性化提示定制（国际化语言模式下不支持）

合作方可以设置定制的提示语，通过配置传给 SDK。自定义提示语分为短提示（不长于17个字符）和长提示（不长于70个字符）。如果不设置，默认无。如果超过字数限制，将会被截断显示。
自定义短提示分为验证阶段提示和上传阶段提示，可以分开设置。位置可以选择位于预览圆框的上方或者下方。默认自定义短提示位于预览圆框下方。若设置自定义短提示位于预览圆框上方的话，sdk过程提示将自动调整到圆框下方。
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(WbCloudFaceContant.INPUT_DATA, inputData);
  //定制合作方个性化提示语
  //此处将设置人脸采集时的个性化提示语
  data.putString(WbCloudFaceContant.CUSTOMER_TIPS_LIVE, “扫描人脸后与您身份证进行对比”);
//此处将设置上传人脸时的个性化提示语
data.putString(WbCloudFaceContant. CUSTOMER_TIPS_UPLOAD, “已提交审核，请等待结果”);
//设置合作方定制提示语的位置，默认为识别框的下方
 //识别框的下方： WbCloudFaceContant.CUSTOMER_TIPS_LOC_BOTTOM
 //识别框的上方：WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP
data.putInt(WbCloudFaceContant.CUSTOMER_TIPS_LOC,WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP);
```

自定义长提示位于整个验证页面的下方，存在于整个刷脸流程中，不能设置位置和时机。
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(WbCloudFaceContant.INPUT_DATA, inputData);
//整个识别过程中合作方定制长提示语（不超过70个字符）
  data.putString(WbCloudFaceContant.CUSTOMER_LONG_TIP, customerLongTip);
```

合作方还可以自定义 SDK 退出二次确认弹窗的文字内容，包括提示标题，提示内容，确认键文案和取消键文案。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(WbCloudFaceContant.INPUT_DATA, inputData);
//退出确认弹窗定制提示标题
  data.putString(WbCloudFaceContant.DIALOG_TITLE, dialogTitle);
  //退出确认弹窗定制提示内容
  data.putString(WbCloudFaceContant.DIALOG_TEXT, dialogText);
  //退出确认弹窗定制确认键文案
  data.putString(WbCloudFaceContant.DIALOG_YES, dialogYes);
   //退出确认弹窗定制取消键文案
```

```
    data.putString(WbCloudFaceContant.DIALOG_NO, dialogNo);
```

比对类型选择

SDK 提供权威数据源比对、自带比对源比对，合作方可以选择传入参数控制对比类型。

- **权威库网纹图片比对类型**
  合作方必须要先在获取 faceId 的接口里送入用户的姓名与身份证号信息，得到相应的 faceId 后，再送入 SDK，供 SDK 与权威库网纹图片进行比对。不需要额外设置对比类型。
- **自带比对源比对类型**
  合作方在获取 faceId 的接口里送入用户提供对比源数据，获取 faceId 后送入 SDK 进行对比。合作方可以上传两类照片，一类是网纹照，一类是高清照；不需要额外设置对比类型。

是否录制视频存证

SDK 默认不录制视频存证。如果合作方需要视频存证，可以通过该字段进行设置。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(WbCloudFaceContant.INPUT_DATA, inputData);
  //设置是否录制视频进行存证，默认不录制存证。
  //此处设置为录制存证
  data.putBoolean(WbCloudFaceContant.VIDEO_UPLOAD, true);
```

是否对录制视频进行检查

> **说明**
> 如果在“是否录制视频存证”步骤的设置为录制存证，则目前该字段有效；否则，设置为不录制存证，也不会对视频进行检查，设置无效。

- 在 SDK 使用过程中，发现视频录制在性能不太好的手机上可能会报错，导致刷脸中断，影响用户体验。
- 为了减少录制视频导致的人脸核身中断问题，SDK 默认设置对录制的视频不作检测。
- 如果合作方对人脸核身安全有更加严格的要求，可以考虑打开这一选项。**但打开这个字段可能导致某些低性能手机上用户人脸核身不能进行，请慎重考虑。**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(WbCloudFaceContant.INPUT_DATA, inputData);
  //设置是否对录制的视频进行检测，默认不检测
  //此处设置为检测
  data.putBoolean(WbCloudFaceContant.VIDEO_CHECK, true);
```

是否播放语音提示（国际化语言模式下不支持）

SDK 默认不打开语音提示。合作方可以根据自身需求选择是否开启，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);
  //设置是否打开语音提示，默认关闭
  //此处设置为打开
  data.putBoolean(WbCloudFaceContant.PLAY_VOICE, true);
```

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(WbCloudFaceContant.INPUT_DATA, inputData);
  //个性化参数设置，此处均设置为与默认相反
   //默认设置为简体中文，此处设置为英文
data.putString(WbCloudFaceContant.LANGUAGE,WbCloudFaceContant.LANGUAGE_EN);
```

```
//sdk样式设置，默认为白色
//此处设置为白色
 data.putString(WbCloudFaceContant.COLOR_MODE, WbCloudFaceContant.WHITE);
  //定制合作方个性化提示语
  //此处将设置人脸采集时的个性化提示语
data.putString(WbCloudFaceContant.CUSTOMER_TIPS_LIVE, “扫描人脸后与您身份证进行对比”);
//此处将设置上传人脸时的个性化提示语
data.putString(WbCloudFaceContant. CUSTOMER_TIPS_UPLOAD, “已提交审核，请等待结果”);
//设置合作方定制提示语的位置，默认为识别框的下方
 //识别框的下方： WbCloudFaceContant.CUSTOMER_TIPS_LOC_BOTTOM
 //识别框的上方：WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP
//此处设置为识别框的上方
data.putInt(WbCloudFaceContant.CUSTOMER_TIPS_LOC,WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP);
//设置选择的比对类型  默认为权威数据源对比
//此处设置权威数据源比对
  data.putString(WbCloudFaceContant.COMPARE_TYPE, WbCloudFaceContant.ID_CARD);
//是否需要录制上传视频 默认不需要，此处设置为不需要
 data.putBoolean(WbCloudFaceContant.VIDEO_UPLOAD, false);
  //是否对录制视频进行检查,默认不检查，此处设置为不检查
  data.putBoolean(WbCloudFaceContant.VIDEO_CHECK, false);
  //设置是否打开语音提示，默认关闭，此处设置为关闭
  data.putBoolean(WbCloudFaceContant. PLAY_VOICE, false);
```

核身结果返回

最近更新时间：2024-12-13 11:08:32

Android SDK 结果分别由以下两个回调返回：
如果您只需要获取刷脸结果，不需要拉取视频和照片，您可以信任前端 SDK 核身结果，返回中的 isSuccess 字段 True 代表人脸核身对比成功；false 代表人脸核身失败。核身 SDK 和核身后台服务之间通信采用加密方式，可以有效防止结果篡改。

```
/**
  * 登录回调接口  返回登录 sdk 是否成功
  */
public interface WbCloudFaceVerifyLoginListener {
        void onLoginSuccess();
        void onLoginFailed(WbFaceError error);
    }

/**
  * 刷脸结果回调接口
  */
public interface WbCloudFaceVerifyResultListener {
        void onFinish(WbFaceVerifyResult result);
      }
```

WbFaceVerifyResult 对象说明

WbFaceVerifyResult 是 SDK 用来给合作方传递身份识别结果的对象，在 WbCloudFaceVerifyResultListener 回调中作为参数返回给合作方 App。
WbFaceVerifyResult 对象的各个字段意义如下表所示：

字段名	类型	字段含义	说明
isSuccess	boolean	人脸核身是否成功	True 代表人脸核身对比成功 false 代表人脸核身失败，具体的失败原因请参考 WbFaceError 对象说明
sign	String	签名	供 App 校验人脸核身结果的安全性
liveRate	String	活体检测分数	-
similarity	String	人脸比对分数	“仅活体检测” 类型不提供此分数
WbFaceError	自定义对象	人脸核身错误	人脸核身成功时为 null

WbFaceError 对象说明

WbFaceError 是 SDK 用来给合作方传递人脸核身错误信息的对象，在 WbCloudFaceVerifyLoginListener 回调和 WbFaceVerifyResult 对象中作为参数返回给合作方 App。WbFaceError 对象的各个字段意义如下表所示，各个字段的内容取值详情请参见 错误码 。

字段名	类型	字段含义	说明
domain	String	错误发生的阶段	只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，后端无法查询到刷脸结果
code	String	错误码	-
desc	String	错误描述	如有需求，可以展示给用户
reason	String	错误信息内容	错误的详细实际原因，主要用于定位问题

人脸 Android 错误码

最近更新时间：2021-04-16 11:11:45

SDK 在登录以及返回人脸服务结果时，如果发生错误或识别失败会返回 WBFaceError 对象，其中各个字段的内容如下：

WBFaceErrorDomainParams

错误码	说明	原因
11000	传入参数为空	传入的 xx 为空
11001	传入的 keyLicence 不可用	传入的 keyLicence
11002	报文加解密失败	报文加解密失败

WBFaceErrorDomainLoginNetwork

错误码	说明	原因
21100	网络异常	登录时网络异常（请求未到达后台）
21200	网络异常	登录时后台返回参数有误（请求已到达后台）

WBFaceErrorDomainLoginServer

错误码	说明	原因
其他错误码	透传后台错误码	例如签名问题等

WBFaceErrorDomainGetInfoNetwork

错误码	说明	原因
31100	网络异常	获取活体类型/光线资源，网络异常（请求未到达后台）
31200	网络异常	获取活体类型/光线资源，后台返回参数有误（请求到达后台）

WBFaceErrorDomainNativeProcess

错误码	说明	原因
41000	用户取消	回到后台/单击 home/左上角/上传时左上角取消
41001	无法获取唇语数据	获取数字活体的数字有问题
41002	权限异常，未获取权限	相机
41003	相机运行中出错	-
41004	视频录制中出错	不能存/启动失败/结束失败
41005	请勿晃动人脸,保持姿势	未获取到最佳图片
41006	视频大小不满足要求	视频大小不满足要求
41007	超时	预检测/动作活体
41008	检测中人脸移出框外	活体/数字/反光
41009	光线活体本地错误	-
41010	风险控制超出次数	用户重试太多次
41011	没有检测到读数声音	数字活体过程中没有发声
41012	初始化模型失败，请重试	初始化算法模型失败

41013	初始化 SDK 异常	WbCloudFaceVerifySdk 未被初始化
41014	简单模式本地加密失败	编码转换异常/加解密编码失败

WBFaceErrorDomainCompareNetwork

错误码	说明	原因
51100	网络异常	对比时，网络异常（请求未到达后台）
51200	网络异常	对比时，后台返回参数有误（请求已到达后台）

WBFaceErrorDomainCompareServer

错误码	说明	原因
其他错误码	透传后台错误码	-

接入示例

最近更新时间：2024-12-13 11:08:32

权威库网纹图片比对、自带对比源对比接入示例：

```
# 在 MainActivity 中单击某个按钮的代码逻辑：
//先填好数据
Bundle data = new Bundle();
WbCloudFaceVerifySdk.InputData inputData = new WbCloudFaceVerifySdk.InputData(
                faceId,
                agreementNo,
                openApiAppId,
                openApiAppVersion,
                openApiNonce,
                userId,
                userSign,
                verifyMode,
                keyLicence);
  data.putSerializable(WbCloudFaceContant.INPUT_DATA, inputData);

  //个性化参数设置，可以不设置，不设置则为默认选项。
  //默认设置为简体中文，此处设置为英文
  data.putString(WbCloudFaceContant.LANGUAGE,WbCloudFaceContant.LANGUAGE_EN);
//sdk样式设置，默认为白色
//此处设置为白色
  data.putString(WbCloudFaceContant.COLOR_MODE, WbCloudFaceContant.WHITE);
//定制合作方个性化提示语，默认不设置
//此处将设置人脸采集时的个性化提示语data.putString(WbCloudFaceContant.CUSTOMER_TIPS_LIVE, “扫描人脸后与您身份证进行对比”);
//此处将设置上传人脸时的个性化提示语
data.putString(WbCloudFaceContant. CUSTOMER_TIPS_UPLOAD, “已提交审核，请等待结果”);
//设置合作方定制提示语的位置，默认为识别框的下方
//识别框的下方： WbCloudFaceContant.CUSTOMER_TIPS_LOC_BOTTOM
//识别框的上方：WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP
//此处设置为识别框的上方
data.putInt(WbCloudFaceContant.CUSTOMER_TIPS_LOC,WbCloudFaceContant.CUSTOMER_TIPS_LOC_TOP);
//设置选择的比对类型  默认为权威库网纹图片比对
//此处设置权威数据源对比
data.putString(WbCloudFaceContant.COMPARE_TYPE, WbCloudFaceContant.ID_CARD);
//是否需要录制上传视频 默认不需要，此处设置为不需要
data.putBoolean(WbCloudFaceContant.VIDEO_UPLOAD, false);
//是否对录制视频进行检查,默认不检查，此处设置为不检查
data.putBoolean(WbCloudFaceContant.VIDEO_CHECK, false);
//设置是否打开语音提示，默认关闭，此处设置为关闭
data.putBoolean(WbCloudFaceContant.PLAY_VOICE, false);

//初始化 SDK，得到是否登录 SDK 成功的结果,由 WbCloudFaceVerifyLoginListener  返回登录结果
//【特别注意】建议对拉起人脸识别按钮做防止重复点击的操作
//避免用户快速点击导致二次登录，二次拉起刷脸等操作引起问题
WbCloudFaceVerifySdk.getInstance().initSdk(DemoActivity.this, data, new WbCloudFaceVerifyLoginListener() {
        @Override
        public void onLoginSuccess() {
                //登录成功，拉起 sdk 页面,由 FaceVerifyResultListener   返回刷脸结果
                WbCloudFaceVerifySdk.getInstance().startWbFaceVerifySdk(DemoActivity.this, new
WbCloudFaceVerifyResultListener() {
                        @Override
                        public void onFinish(WbFaceVerifyResult result) {
                                if (result != null) {
                                        if (result.isSuccess()) {
                                                Log.d(TAG, "刷脸成功！");
                                        } else {
```

```
                                        Log.d(TAG, "刷脸失败！");
                                    }
                                }
                                //刷脸结束后，及时释放资源
                                WbCloudFaceVerifySdk.getInstance().release();
                            }
                        });
                }
                @Override
                public void onLoginFailed(WbFaceError error) {
                        Log.d(TAG, "登录失败！");
                        //刷脸结束后，及时释放资源
                        WbCloudFaceVerifySdk.getInstance().release();
                }
        });
```

iOS 人脸核身
配置流程

最近更新时间：2025-06-06 17:12:52

使用 Cocoapod 集成

SDK 最低支持到 iOS11.0，请在构建项目时候注意，目前仅支持 Xcode11.0 及更高版本编译。如需特别支持 iOS8.0 版本，请联系技术支持。
以下为接入配置的步骤：
1. 将 TencentCloudHuiyanSDKFace_framework 文件夹拷贝到自己项目的 podfile 文件所在的同一目录。
2. 在 podfile 使用如下配置：

> ⚠ 注意
> target 后面内容根据自己项目配置。demo可参考 SDK，请到控制台自助接入列表页获取密码。

```
target 'WBCloudReflectionFaceVerify-Demo' do
pod 'TencentCloudHuiyanSDKFace_framework', :path=> './TencentCloudHuiyanSDKFace_framework'
end
```

3. 使用 pod install 命令。
4. SDK 需要使用相机权限，请在 info.plist 中添加：

```
Privacy - Camera Usage Description
```

直接引用 Framework

SDK 最低支持到 iOS11.0，请在构建项目时候注意。
以下为接入配置的步骤：
1. 引用以下资源文件到项目：

```
TencentCloudHuiyanSDKFace.framework
YTCommonLiveness.framework
YTFaceTrackerLiveness.framework
YTFaceAlignmentTinyLiveness.framework
YTPoseDetector.frameworks
YTFaceDetectorLiveness.framework
YTFaceLiveReflect.framework
tnnliveness.framework
TuringShieldCamRisk.framework
TencentCloudHuiyanSDKFace.bundle
face-tracker-003.bundle
YTCv.framework
YtSDKKitFrameworkTool.framework
```

2. SDK 依赖以下系统框架，需要在**BuildPhases>Link Binary With Libraries**中添加，可以参考 Demo，具体依赖的系统库如下：

```
UIKit.framewrok
AVFoundation.framework
CoreVideo.framework
Security.framework
SystemConfiguration.framework
CoreMedia.framework
CoreTelephony.framework
ImageIO.framework
VideoToolbox.framework
Accelerate.framework
webkit.framework
libc++.tbd
```

```
libz.tbd
```

3. SDK 需要使用相机权限，请在 info.plist 中添加：

```
Privacy - Camera Usage Description
```

4. 需要在BuildSettings>Other Linker Flags中设置：`-ObjC`

接口调用

最近更新时间：2025-01-15 17:14:53

SDK 接口调用方法

SDK 的功能通过 WBFaceVerifyCustomerService 这个类的方法进行调用，**其中 SDK 中使用的 nonce、sign 等重要信息，需要合作方从自己后台拉取，并且两者不能缓存，使用后即失效**，详细接口说明如下，其他的操作请参考 Demo 中的登录接口的参数说明：

版本号及宏定义说明

```
#import <UIKit/UIKit.h>
#ifndef WBFaceVerifyConst_h
#define WBFaceVerifyConst_h
#define WBCloudReflectionFaceVerifyVersion

UIKIT_EXTERN NSString *const WBCloudFaceVerifySDKVersion;

/**
 SDK使用的主题风格

 - WBFaceVerifyThemeDarkness: 暗黑色系主题
 - WBFaceVerifyThemeLightness: 明亮色系主题
 - WBFaceVerifyThemeOrange: 橙色主题
 - WBFaceVerifyThemeCustom: 自定义主题，通过修改bundle中的custom.json实现自定义

 */
typedef NS_ENUM(NSInteger, WBFaceVerifyTheme) {
    WBFaceVerifyThemeDarkness = 0,
    WBFaceVerifyThemeLightness,
    WBFaceVerifyThemeOrange,
    WBFaceVerifyThemeCustom,
};

typedef NS_ENUM(NSInteger, WBFaceVerifyLanguage) {
    WBFaceVerifyLanguage_ZH_CN = 0, //简体中文
    WBFaceVerifyLanguage_ZH_HK,     //繁体中文
    WBFaceVerifyLanguage_EN,        //英语
    WBFaceVerifyLanguage_ID,        //印尼语
    WBFaceVerifyLanguage_JA,        //日语
    WBFaceVerifyLanguage_KO,        //韩语
    WBFaceVerifyLanguage_TH         //泰语
};

typedef NS_ENUM(NSInteger, WBFaceCustomTipsLoc) {
    WBFaceCustomTipsLoc_Bottom = 0, //提示语在下
    WBFaceCustomTipsLoc_Top,
};
#endif /* WBFaceVerifyConst_h */
```

入口方法说明

NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。faceID +活体检测+人脸比对服务（身份证的网纹照片进行对比）

```
/*
初始化云刷脸 sdk，仅做参数初始化与登录，不拉起刷脸页面
登录有时效性，建议在登录完成后 success 回调中拉起刷脸页面！！
登录过程为异步操作，多次登录以最后一次收到的结果为准！！

此 SDK 接口中
合作方后台开发需要通过后台接口获取 sign，然后根据自带比对源接口，通过后台接口获取 faceId!!!!(native 端无需传入自带比对源图)
```

```
注意，请使用 dispatch_async(dispatch_get_main_queue(), ^{  }); 异步调用 SDK 的入口方法

@param userid 用户唯一标识，由合作方自行定义
@param nonce  满足接入要求的32位随机数
@param sign 获取方式请参考 [生成 SDK 接口调用步骤使用签名](https://cloud.tencent.com/document/product/1007/63359)
@param appid 业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请
@param orderNo 每次人脸身份认证请求的唯一订单号：建议为32位字符串(不超过32位)
@param apiVersion 后台api接口版本号(不是 SDK 的版本号),默认请填写@"1.0.0"
@param licence 在人脸核身控制台内申请(该 licence 同 app 当前使用的 bundle id 绑定)
@param faceId 合作方必须要先在获取 faceId 的接口里送入用户自带比对源图片信息，得到相应的 faceId 后，再送入 sdk，若为空，则开启仅活体检测模式!!!!
@param sdkConfig SDK基础配置项目
@param success 服务登录成功回调,登录成功以后开始进行活体和检测服务
@param failure 服务登录失败回调,具体参考错误码文档
*/
-(void)initSDKWithUserId:(NSString *)userid
                   nonce:(NSString *)nonce
                    sign:(NSString *)sign
                   appid:(NSString *)appid
                 orderNo:(NSString *)orderNo
              apiVersion:(NSString *)apiVersion
                 licence:(NSString *)licence
                  faceId:(NSString *)faceId
               sdkConfig:(WBFaceVerifySDKConfig *)sdkConfig
                 success:(void (^)())success
                 failure:(void (^)(WBFaceError * _Nonnull))failure;

/**
 以上一次的登录结果拉起刷脸页面，必须先登录再拉起刷脸页面

 @return 拉起是否成功
 */
- (BOOL)startWbFaceVerifySdk;
```

个性化参数设置

SDK 登录接口 initSDK 方法中需要传入 WBFaceVerifySDKConfig 字段，通过该对象可以配置 SDK 中其他基础配置：包括设置主题风格，资源路径等，具体参考头文件。

```
/**
 人脸识别 SDK 基础配置类
 */
@interface WBFaceVerifySDKConfig : NSObject

#pragma mark - common
/**
 sdk中拉起人脸活体识别界面中使用UIWindow时的windowLevel配置,默认配置是1 + UIWindowLevelNormal

 如果接入方app中有其他自定义UIWindow，为了防止界面覆盖,可以酌情设置该参数
 */
@property (nonatomic, assign) NSUInteger windowLevel;

/**
 是否境外用户
 默认：NO
 请注意：如果合作方可以分辨国内或者境外ip，建议境外ip开启这个配置，国内ip访问不进行配置
 */
@property (nonatomic, assign) BOOL isAbroad;

/**
 人脸识别服务是否进行通过录像，从而进行视频存证
```

```
 default: NO
 */
@property (nonatomic, assign) BOOL recordVideo;

/**
 是否由 SDK 内部处理 SDK 网络请求的 cookie

 默认值：YES
 */
@property (nonatomic, assign) BOOL manualCookie;

/**
 人脸识别页面中的主题风格，需要配合不同资源包使用：
 WBFaceVerifyThemeDarkness - 暗灰主题
 WBFaceVerifyThemeLightness - 明亮主题（默认）
 */
@property (nonatomic, assign) WBFaceVerifyTheme theme;

/**
 多语言配置
 默认中文，当使用其他语言时，强制静音
 */
@property (nonatomic, assign) WBFaceVerifyLanguage language;

/**
 是否静音
 默认值：YES
 */
@property (nonatomic, assign) BOOL mute;

/*
 送入自定义提示文案的位置
 默认：WBFaceCustomTipsLoc_Bottom
 */
@property (nonatomic, assign) WBFaceCustomTipsLoc tipsLoc;

/*
 检测过程中展示的文案
 默认为空
 */
@property (nonatomic, copy) NSString *customTipsInDetect;

/*
 上传过程中展示的文案
 默认为空
 */
@property (nonatomic, copy) NSString *customTipsInUpload;

/*
 底部提示文案，长度不超过70字
 */
@property (nonatomic, copy) NSString *bottomCustomTips;
/*
 退出二次确认UI配置
 */
@property (nonatomic, copy) NSString *exitAlertTitle; //标题
@property (nonatomic, copy) NSString *exitAlertMessage; //内容
@property (nonatomic, copy) NSString *exitAlertRight; //右边按钮
@property (nonatomic, copy) NSString *exitAlertLeft; //左边按钮

/*
 如果有使用苹果分屏模式（UIWindowScene），打开此开关
```

```
 Xcode11新建工程有使用Scene，可以参考资料自行调整
 默认为NO
 */
@property (nonatomic, assign) BOOL useWindowSecene;

#pragma mark - simple //非标特有字段，标准模式无需设置
/**
 是否返回录制的视频

 default: NO
 */
@property (nonatomic, assign) BOOL returnVideo;

/**
 返回视频加密的公钥，如果不配置则不加密

 需要recordVideo returnVideo同时为YES，才返回加密的视频内容
 */
@property (nonatomic, copy) NSString *publicKey;

/**
 AES加密需要用到的IV
 */
@property (nonatomic, copy) NSString *aesIV;
/**
 默认sdk配置
 */
+(instancetype)sdkConfig;

@end
```

自定义皮肤设置

可以通过修改 JSON 文件，自定义刷脸的部分 UI。首先把 Theme 设置为 WBFaceVerifyThemeCustom，再修改 JSON 文件，路径：`WBCloudReflectionFaceVerify_framework/Resource/WBCloudReflectionFaceVerify.bundle/custom.json`。

```
{
        "name": "custom",
    "statusBarStyle" :   "light",  /** 状态栏颜色 */
    "navbarTintColor":   "0xffffff", /** 导航栏色调 */
    "baseNavBarColor" :  "0x0", /** 导航栏默认颜色 */
    "authNavBarColor" :  "0x22252A", /** 授权详情页导航栏颜色 */

    "imageBgColor" :     "0x23262b", /** 刷脸页背景色 */
    "faceNormalColor": "0x33FFFFFF", /** 默认刷脸框颜色 */
    "faceSatisfyColor": "0x5065FF", /** 人脸框识别成功颜色 */
    "faceErrorColor": "0xFF6034", /** 人脸框识别失败颜色 */

    "guideLabelColor" :  "0xffffff", /** 提示语默认颜色 */
    "guideLabelErrorColor" : "0xff5240", /** 识别错误时提示语颜色 */
    "customTipsColor":   "0xFFFFFF", /** 自定义提示语颜色 */
    "bottomTipsBgColor": "0x0DFFFFFF", /** 底部提示背景色 */
    "bottomTipsTextColor": "0x80FFFFFF", /** 底部提示文案颜色 */

    "backButtonImage":"backbutton@dark", /** 刷脸页面返回按钮图片 */
    "authBackButtonImage":"backbutton@dark", /** 授权页面返回按钮图片 */
    "authBodyImage": "wbcf_auth_face@dark", /** 授权页人脸框图片 */
    "authCheckboxUnSelectImage":"wbcf_checkbox_unselect", /** 授权页勾选框未勾选图片 */
    "authCheckboxSelectImage":"wbcf_checkbox_select", /** 授权页勾选框勾选图片 */
    "authCheckboxTipsColor": "0xFFFFFF", /** 授权页勾选提示语颜色 */
    "authCheckboxLinkColor": "0x6F80FF", /** 授权页详情链接提示颜色 */
```

```
    "authAgreeButtonNormalColor": "0xB7C0FF", /** 授权页同意按钮默认颜色 */
    "authAgreeButtonHighlightColor": "0x5065FF", /** 授权页同意按钮高亮颜色 */
    "authAgreeTextNormalColor":"0xFFFFFF", /** 授权页同意按钮文案默认颜色 */
    "authAgreeTextHighlightColor":"0xFFFFFF", /** 授权页同意按钮文案高亮颜色 */

    "alertTitleColor":"0x000000", /** 弹框主题颜色 */
    "alertMessageColor":"0x000000", /** 弹框详细信息颜色 */
    "alertLeftBtnColor":"0x5065FF", /** 弹框左边按钮文案颜色 */
    "alertRightBtnColor":"0x5065FF", /**弹框右边按钮文案颜色 */
    "alertBackgroundColor":"0xFFFFFF", /** 弹框背景色 */

}
```

注意

1. 需要替换返回按钮图片时，需要修改对应 navbarTintColor 到按钮颜色。
2. 有透明色时，透明度放在前面，例如 80FFFFFF，FFFFFF 代表白色，80代表透明度。
3. 需要替换图片的目录在 JSON 文件同级目录 common 文件夹下，替换时注意保证相同尺寸大小，否则显示会异常。

接入流程与说明

最近更新时间：2024-08-15 15:26:21

整体工作流程

人脸识别主要流程包括以下步骤：

1. 登录人脸识别 SDK，即 initSDKWithXXX 方法。
2. 登录结果返回，在第一步登录成功以后，再调用 startWbFaceVerifySdk 方法拉起人脸识别页面，如果登录失败，会通过 failure block 回调返回一个 WBFaceError 对象。
3. 人脸识别的具体过程，当中各种失败情况以及用户主动退出，直接跳到第5步。
4. 人脸识别前端 SDK 检测结果上传后台进行活体检测以及人脸比对等服务。
5. 结果返回，通过 delegate 回调方法或者注册 NSNotification 的方式获取活体检测/人脸比对的结果，结果封装在 WBFaceVerifyResult。

登录接口说明

iOS 人脸识别接口大部分参数说明请参考头文件注释。

> **注意**
> userid、nonce、sign、orderNo 等参数建议通过接入方后台透传给前端，前端无需在本地生成。

方法功能一：

如果使用活体检测服务+人脸比对服务（身份证的网纹照片进行对比）方式，则需要传入 faceID，其中 faceID 需要合作方后台调用 合作方后台上传身份信息 生成 faceID，其中 userid、nonce、sign、orderNo 要与生成 sign 时，使用的相同。

方法功能二：

如果使用活体检测服务+人脸比对服务（合作方提供的比对源图片进行比对），则需要传入 faceID，其中 faceID 需要合作方后台调用 合作方后台上传身份信息 生成 faceID，其中 userid、nonce、sign、orderNo 要与生成 sign 时，使用的相同，在生成 faceID 时传入比对源图片。

识别界面拉起方式说明

默认情况 SDK 会创建一个全屏幕大小的 UIWindow，覆盖在当前屏幕上方，并且默认 windowLevel=UIWindowLevelNormal+1。

核身结果返回

最近更新时间：2022-11-08 15:22:11

IOS SDK 中，人脸识别结果返回给接入方有两种方式，两种结果回调都在 Main Thread 完成：
如果您只需要获取刷脸结果，不需要拉取视频和照片，您可以信任前端 SDK 核身结果，返回中的 isSuccess 字段 YES 代表人脸核身对比成功；NO 代表人脸核身失败。核身 SDK 和核身后台服务之间通信采用加密方式，可以有效防止结果篡改。

1. 通过实现 WBFaceVerifyCustomerServiceDelegate 的 WBFaceVerifyCustomerServiceDelegate 方法，通过该方法返回 WBFaceVerifyResult 对象。
2. 无需实现 delegate 方法，通过注册 WBFaceVerifyCustomerServiceDidFinishedNotification 通知，在接受到该通知时，进行结果处理。

WBFaceVerifyCustomerServiceDidFinishedNotification 通知中，通过获取该通知的 userInfo，获取字典中 key 为 faceVerifyResult 对应的 value 对象就是 WBFaceVerifyResult。

WBFaceVerifyResult 说明：

字段名	类型	字段含义	说明
isSuccess	BOOL	人脸核身是否成功	YES 代表认证成功，NO 代表认证失败，具体原因参考 人脸核身 IOS 错误码
sign	NSString	签名	供 App 校验人脸核身结果的安全性
liveRate	NSString	活体检测分数	-
similarity	NSString	人脸比对分数	“仅活体检测” 类型不提供此分数
WbFaceError	自定义对象	人脸核身错误	人脸核身成功时为 nil

WBFaceError 对象说明

字段名	类型	字段含义	说明
domain	NSString	错误发生的阶段	只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，后端无法查询到刷脸结果
code	NSString	错误码	-
desc	NSString	错误描述	如有需求，可以展示给用户
reason	NSString	错误信息内容	错误的详细实际原因，主要用于定位问题

```
@interface WBFaceSimpleModeResult : NSObject

/**
 结果对应的订单号
 */
@property (nonatomic, copy, readonly) NSString *orderNo;

/**
 接下来用于人脸对比的安全性参数
 */
@property (nonatomic, copy, readonly) NSString *encryptAESKey;

/**
 视频编码
 */
@property (nonatomic, copy, readonly) NSString *userVideoString;

/**
 使用传入publickey加密过的AES
 */
@property (nonatomic, copy, readonly) NSString *videoEncryptAESKey;
/**
```

```
 用于后面进行人脸比对的数据参数
 */
@property (nonatomic, copy, readonly) NSString *identifyStr;

@end
/**
 人脸服务返回结果对象
 */
@interface WBFaceVerifyResult : NSObject
/**
 人脸比对结果是否通过：

 YES: 表示人脸服务通过
 NO:  表示人脸服务不通过
 */
@property (nonatomic, assign, readonly) BOOL isSuccess;
/**
 结果对应的订单号
 */
@property (nonatomic, copy, readonly) NSString *orderNo;
/**
 isSuccess == YES 时，sign有值，可以去后台拉取本次人脸服务的照片，视频存证
 isSuccess == NO  时，sign 无意义
 */
@property (nonatomic, copy, readonly) NSString * sign;
/**
 活体检测服务得分

 isSuccess == YES 时，liveRate 有值：
        1. liveRate 可能是 @"分数为空"，这种情况具体咨询合作方
        2. float类型的字符串，请调用 [liveRate floatValue] 获取具体分数
 isSuccess == NO  时，liveRate 无意义
 */
@property (nonatomic, copy, readonly) NSString * liveRate;

/**
 人脸比对服务得分

 isSuccess == YES 时，similarity 有值：
         1. similarity 可能是 @"分数为空"，这种情况具体咨询合作方
         2. float类型的字符串，请调用 [similarity floatValue] 获取具体分数
 isSuccess == NO  时，similarity 无意义
 */
@property (nonatomic, copy, readonly) NSString * similarity;

/**
 人脸比对图片之一

 isSuccess == YES 时，该属性是上送比对图片之一UIImage的base64编码字符串(图片格式是jpg)

 isSuccess == NO  时，如果是 SDK 返回的错误码，该属性为nil，如果是后端返回的错误，该属性是上送比对图片之一UIImage的base64编码
字符串(图片格式是jpg)
 */
@property (nonatomic, copy, readonly) NSString * userImageString;

/**
 isSuccess == YES 时候，error 无意义
 isSuccess == NO  时，error中存储的具体错误信息，参考 WBFaceError.h
 */
@property (nonatomic, strong, readonly) WBFaceError * error;

#pragma mark - 非标专用返回参数
```

```
@property (nonatomic, strong, readonly) WBFaceSimpleModeResult *simpleModeResult;

#pragma mark -

-(NSString *)description;
@end
```

人脸 iOS 错误码

最近更新时间：2024-10-11 15:23:41

SDK 在登录以及返回人脸服务结果时，如果发生错误或者识别失败会返回 WBFaceError 对象，该对象的结构如下,并且在判断具体错误时，需要先根据 domain 字段判断错误发生在 sdk 服务运行中的位置，然后根据 code 判断具体的错误：

```
NS_ASSUME_NONNULL_BEGIN
/*
 错误 domain 划分成两类：

 出现在登录时，通过调用 startXXXX 方法的 failure block 进行回调返回：
 WBFaceErrorDomainInputParams, WBFaceErrorDomainLoginNetwork, WBFaceErrorDomainLoginServer


 人脸服务结果返回(封装在 WBFaceVerifyResult 中)：
 WBFaceErrorDomainGetInfo, WBFaceErrorDomainNativeProcess, WBFaceErrorDomainCompareNetwork,
WBFaceErrorDomainCompareServer
 */

/* 登录时传入参数有误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainInputParams;
/* 登录时网络请求错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainLoginNetwork;
/* 登录时后台拒绝登录，具体参考后台 word 版本错误码，这里直接透传 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainLoginServer;
/* 拉取有效信息时候网络错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainGetInfo;
/* native 本地在活体检测中，某些原因导致错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainNativeProcess;
/* 上送后台比对时,网络错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainCompareNetwork;
/* 后台比对完成,返回比对结果的错误原因*/
UIKIT_EXTERN NSString *const WBFaceErrorDomainCompareServer;

@interface WBFaceError: NSObject
/**
 错误发生的地方，具体的发生地方由上面定义的 WBFaceErrorDomainXXXX 指定
 */
@property (nonatomic, readonly, copy) NSString *domain;
/**
 每个domain中有相应的错误代码，具体的错误代码见
 */
@property (nonatomic, readonly, assign) NSInteger code; //错误代码
@property (nonatomic, readonly, copy) NSString *desc; //获取本地化描述
@property (nonatomic, readonly, copy) NSString *reason; // 错误出现的真实原因
@property (nonatomic, readonly, copy) NSDictionary * _Nullable otherInfo;// 预留接口
@end
```

若 Domain 为：WBFaceErrorDomainInputParams

错误码	描述	详细实际原因
12000	传入参数为空	传入的参数为空
12001	传入的 keyLicence 不可用	传入的 keyLicence 不可用
12002	身份证格式不正确	身份证格式不正确
12003	使用自带对比源，传入参数错误，非 base64	传入的 srcPhotoString 不是 base64
12004	使用自带对比源，传入参数错误，超过1MB	传入的 srcPhotoString 超过1MB

12005	SDK 资源引入版本不匹配	没有引入资源包或者引入的资源包版本与当前 SDK 版本不匹配
12006	订单号不能为0或者超过32位	-
12007	nonce 字符串位数不为32位	-
12008	SDK 正在提供服务	连续调用了 SDK 的启动方法

Domain 为：WBFaceErrorDomainLoginNetwork

错误码	描述	详细实际原因
22100	网络异常	登录时网络异常（请求未到达后台）
22200	网络异常	登录时后台返回参数有误（请求已到达后台）

Domain 为：WBFaceErrorDomainLoginServer

错误码	描述	详细实际原因
其他错误码	透传后台错误码	例如签名问题等

Domain 为：WBFaceErrorDomainGetInfo

错误码	描述	详细实际原因
32100	网络异常	获取活体类型或光线阈值，网络异常(请求未到达后台)
32200	网络异常	获取活体类型或光线阈值，后台返回参数有误（请求已到达后台）

Domain 为：WBFaceErrorDomainNativeProcess

错误码	描述	详细实际原因
42000	用户取消	回到后台或单击 home 或左上角或上传时左上角取消
42001	网络环境不满足认证需求	无网络或2G 网络
42002	权限异常，未获取权限	相机或麦克风或 read phone 或 external 或 storage
42003	相机运行中出错	-
42004	视频录制中出错	不能存或启动失败或结束失败
42005	请勿晃动人脸，保持姿势	未获取到最佳图片
42006	视频大小不满足要求	视频大小不满足要求
42007	超时	预检测/动作活体
42008	检测中人脸移出框外	活体或反光
42009	光线活体本地错误	-
42010	风险控制超出次数	用户重试太多次
42011	没有检测到读数声音	活体过程中没有发声
42012	模型初始化失败	-
42015	报文解密失败	请求报文解密失败

Domain 为：WBFaceErrorDomainCompareNetwork

错误码	描述	详细实际原因
52100	网络异常	对比时，网络异常（请求未到达后台）

52200	网络异常	对比时，后台返回参数有误（请求已到达后台）

Domain 为：WBFaceErrorDomainCompareServer

错误码	描述	详细实际原因
其他错误码	透传后台错误码	-

Harmony NEXT人脸核身
开发准备

最近更新时间：2024-08-23 15:16:21

权限

SDK 需要用到相机权限和网络权限，如下示例：

```
"requestPermissions": [
      {
        "name": "ohos.permission.INTERNET",
      },
      {
        "name": "ohos.permission.GET_NETWORK_INFO"
      },
      {
        "name": "ohos.permission.CAMERA",
        "reason": "face verify",
        "usedScene": {
          "abilities": [
            "EntryAbility",
          ],
          "when": "inuse"
        }
      }
    ]
```

CPU 平台设置

当前 Harmony NEXT 手机都是64位手机，目前 SDK 只支持 armeabi-v8a 平台。

配置流程

最近更新时间：2025-06-06 17:12:52

> **注意：**
> 以下文章出现 “刷脸” 一词均可以表示为 “人脸核身”。

接入配置

SDK（WbCloudFaceLiveSdk-xx.har, 具体版本号以 SDK 交付件为准）最低支持到 5.0.0(12)，请在构建项目时注意。
刷脸 SDK 将以 HAR 文件的形式提供。
将提供的 HAR 文件添加到工程的libs目录下，并且在 oh-package.json5 中添加下面的配置：

```
"dependencies": {
"wbcloudfaceverifysdk": "file:../libs/ WbCloudFaceVerifySdk.har"
}
```

接口调用

最近更新时间：2025-01-15 17:14:53

本章将展示 SDK 核心接口的调用方法。具体的代码示例参考交付 Demo(demo/module/src/main/ets/pages/HuiYanModePage.ets)。

> ⚠ **注意：**
> 以下文章出现 “刷脸” 一词均可以表示为 “人脸核身”。

SDK 接口调用方法

SDK 代码调用的入口为：**WbCloudFaceVerifySdk** 这个类。

```
export class WbCloudFaceVerifySdk {

  /**
   * 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 getInstance(): WbCloudFaceVerifySdk {
    …
  }

  /**
   * 在使用SDK前先初始化，传入需要的数据WbFaceVerifyConfig
   * 由 WbCloudFaceVerifyLoginCallback返回是否登录SDK成功
   * 关于传入WbFaceVerifyConfig见后面的说明
   */
  public initSdk(context: Context, config: WbFaceVerifyConfig, loginCallback: WbCloudFaceVerifyLoginCallback) {
    …
  }

  /**
   * 登录成功后，调用此函数拉起sdk页面。
   * 由 WbCloudFaceVerifyResultCallback返回刷脸结果。
   */
  public startWbFaceVerifySdk(context: Context, resultCallback: WbCloudFaceVerifyResultCallback) {
    …
  }

  /**
   * 拿到刷脸结果后，释放资源， 防止内存泄漏
   * 【注意】请务必在拿到刷脸结果后释放资源，否则可能会发生丢失回调的情况!
   */
  public release() {
    …
  }
}
```

WbCloudFaceVerifySdk.initSdk() 的第二个参数 config: WbFaceVerifyConfig 用来传递 sdk 初始化必需数据和 sdk 配置项参数。

```
export class WbFaceVerifyConfig {
    //sdk初始化必需数据
  public inputData: InputData;
  //是否开启日志，默认不打开sdk日志，
  //【注意】上线请务必关闭sdk日志!
  public isEnableLog: boolean = false;
}
```

必须传递的参数包括（参考要求详见本页接口参数说明的描述）：

```
//这些都是InputData对象里的字段，是需要传入的数据信息
export class InputData {
  //此次刷脸用户标识，合作方需要向人脸识别后台拉取获得，详见获取faceId接口
  public faceId: string;
//订单号
  public orderNo: string;
//APP_ID
  public appId: string;
  //openapi  Version
public version: string;
//32位随机字符串
  public nonce: string;
  //User id
  public userId: string;
  //签名信息
  public sign: string;
  ////在人脸核身控制台内申请
  public licence: string;
}
```

关于接口调用的示例可参考 接入示例

接口参数说明

InputData 对象说明
InputData 是用来给 SDK 传递必须参数所需要使用的对象，合作方需要往里塞入SDK 需要的一些数据以便启动刷脸 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（字节）	是否必填
faceId	刷脸id号，由合作方向人脸识别后台拉取获得	String	-	是
orderNo	订单号，合作方订单的唯一标识	String	32	是
appId	业务流程唯一标识，即wbappid，可参考 获取WBappid 指引在人脸核身控制台内申请	String	8	是
version	接口版本号，默认填1.0.0	String	20	是
nonce	32 位随机字符串，每次请求需要的一次性 nonce	String	32	是
userId	User Id，每个用户唯一的标识	String	30	是
sign	获取方式请参考 生成 SDK 接口调用步骤使用签名	String	40	是
licence	在 人脸核身控制台 内申请的 SDKlicense	String	以实际申请为准	是

个性化参数设置（可选）

WbCloudFaceVerifySdk.initPlusSdk() 里 WbFaceVerifyConfig，除了必须要传的 InputData 对象之外，还可以由合作方为其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。
合作方可以选择 SDK 样式。目前 SDK **有黑色模式和白色模式两种，默认显示白色模式**。设置代码如下：

```
//对 sdk 样式进行设置，默认为白色模式
//此处设置为白色模式
wbFaceVerifyConfig.themeMode = ThemeMode.Light
SDK 还支持自定义皮肤，支持定制刷脸过程中各个组件的色值。使用自定义皮肤时需要设置：
wbFaceVerifyConfig.themeMode = ThemeMode.Custom
并在App的resources/rawfile目录中新建wbcf_custom.json主题配置文件：
可配置的颜色值参考 json 文件如下：
{
"name": "custom",
"statusBarStyle": "light",
```

```
"navbarTintColor": "#ffffff",
"baseNavBarColor": "#0",
"authNavBarColor": "#22252A",
"imageBgColor": "#22252A",
"faceNormalColor": "#33FFFFFF",
"faceSatisfyColor": "#5065FF",
"faceErrorColor": "#EB4140",
"guideLabelColor": "#ffffff",
"guideLabelErrorColor": "#EB4140",
"customTipsColor": "#FFFFFF",
"bottomTipsBgColor": "#0DFFFFFF",
"bottomTipsTextColor": "#80FFFFFF",
"backButtonImage": "app.media.wbcf_back_white",
"authBackButtonImage": "app.media.wbcf_back_white",
"authBodyImage_651": "app.media.wbcf_protocal_img",
"authBodyImage_will": "app.media.wbwf_auth_head_image",
"authCheckboxHighlightColor": "#5065FF",
"authCheckboxTipsColor": "#FFFFFF",
"authCheckboxLinkColor": "#6F80FF",
"authAgreeButtonNormalColor": "#B7C0FF",
"authAgreeButtonHighlightColor": "#5065FF",
"authAgreeTextNormalColor": "#FFFFFF",
"authAgreeTextHighlightColor": "#FFFFFF",
"authTipBackgroundColor_651": "#292C31",
"authTipTitileTextColor_651": "#ADB1C6",
"authTipDetailTextColor_651": "#80FFFFFF",
"alertTitleColor": "#000000",
"alertMessageColor": "#000000",
"alertLeftBtnColor": "#5065FF",
"alertRightBtnColor": "#5065FF",
"alertBackgroundColor": "#FFFFFF"
}
```

合作方个性化提示定制（国际化语言模式下不支持）

合作方可以设置定制的提示语，通过配置传给 SDK。自定义提示语分为短提示（不长于17个字符）和长提示（不长于70个字符）。如果不设置，默认无。如果超过字数限制，将会被截断显示。 自定义短提示分为验证阶段提示和上传阶段提示，可以分开设置。位置可以选择位于预览圆框的上方或者下方。默认自定义短提示位于预览圆框下方。若设置自定义短提示位于预览圆框上方的话，SDK 过程提示将自动调整到圆框下方。 设置代码如下：

```
let uiConfig: WbUiConfig = new WbUiConfig();
//设置合作方定制提示语的位置，默认为识别框的下方
//识别框的下方： CustomTipsLocation.Bottom
//识别框的上方： CustomTipsLocation.Top
uiConfig.tipsLocation = CustomTipsLocation.Bottom
//此处将设置人脸采集时的个性化提示语
uiConfig.customTipsInDetect = "扫描人脸后与您身份证进行对比"
//此处将设置上传人脸时的个性化提示语
uiConfig.customTipsInUpload = "已提交审核，请等待结果"
uiConfig.bottomCustomTips = cache.bottomCustomTips
wbFaceVerifyConfig.uiConfig = uiConfig
```

自定义长提示位于整个验证页面的下方，存在于整个刷脸流程中，不能设置位置和时机。长提示的显示除了设置代码外，还需邮件申请开通，详情请 联系我们 。设置代码如下：

```
uiConfig.bottomCustomTips = customerLongTip
```

核身结果返回

最近更新时间：2024-07-03 18:06:31

SDK 结果分别由以下两个回调返回：
如果您只需要获取刷脸结果，不需要拉取视频和照片，您可以信任前端SDK核身结果，返回中的 isSuccess 字段 True 代表人脸核身对比成功；false 代表人脸核身失败。核身 SDK 和核身后台服务之间通信采用加密方式，可以有效防止结果篡改。

```
 /**
  * 登录回调接口  返回登录 sdk 是否成功
  */
export interface WbCloudFaceVerifyLoginCallback {
  onLoginSuccess: () => void;
  onLoginFail: (error: WbFaceError) => void;
}
/**
  * 刷脸结果回调接口
  */
export interface WbCloudFaceVerifyResultCallback {
  onFinish: (result: WbFaceVerifyResult) => void;
}
```

WbFaceVerifyResult对象说明

WbFaceVerifyResult 是 SDK 用来给合作方传递身份识别结果的对象，在 WbCloudFaceVerifyResultListener 回调中作为参数返回给合作方 App。
WbFaceVerifyResult 对象的各个字段意义如下表所示：

字段名	类型	字段含义	说明
isSuccess	boolean	人脸核身是否成功	True 代表人脸核身对比成功；false 代表人脸核身失败，具体的失败原因请参考 WbFaceError 对象说明
sign	String	签名	供 App 校验人脸核身结果的安全性
liveRate	String	活体检测分数	-
similarity	String	人脸比对分数	“仅活体检测” 类型不提供此分数
WbFaceError	自定义对象	人脸核身错误	人脸核身成功时为 null

WbFaceError 对象说明

WbFaceError 是 SDK 用来给合作方传递人脸核身错误信息的对象，在 WbCloudFaceVerifyLoginListener 回调和 WbFaceVerifyResult 对象中作为参数返回给合作方 App。WbFaceError 对象的各个字段意义如下表所示，各个字段的内容取值详情请参见 SaaS服务 错误码 。

字段名	类型	字段含义	说明
domain	String	错误发生的阶段	只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，后端无法查询到刷脸结果
code	String	错误码	-
desc	String	错误描述	如有需求，可以展示给用户
reason	String	错误信息内容	错误的详细实际原因，主要用于定位问题

错误码描述

最近更新时间：2024-07-03 18:06:31

> **注意：**
> 以下文章出现 “刷脸” 一词均可以表示为 “人脸核身”。

SDK 在登录以及返回人脸服务结果时，如果发生错误或者识别失败会返回 WBFaceError 对象，该对象的字段结构意义见 接口参数说明，其中各个字段的内容如下：

WBFaceErrorDomainParams

Code（错误码）	Description（描述）	Reason（详细实际原因）
13000	传入参数为空	传入的 xx 为空
13001	传入的 keyLicence 不可用	传入的 keyLicence 不可用
13002	报文加解密失败	报文加解密失败

WBFaceErrorDomainLoginNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
23100	网络异常	登录时网络异常（请求未到达后台）
23200	网络异常	登录时后台返回参数有误（请求到达后台）

WBFaceErrorDomainLoginServer

Code（错误码）	Description（描述）	Reason（详细实际原因）
其他错误码	透传后台错误码	例如签名问题等等

WBFaceErrorDomainGetInfoNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
33100	网络异常	获取活体类型/光线资源，网络异常（请求未到达后台）
33200	网络异常	获取活体类型/光线资源，后台返回参数有误（请求到达后台）

WBFaceErrorDomainNativeProcess

Code（错误码）	Description（描述）	Reason（详细实际原因）
43000	用户取消	回到后台/单击 home /左上角/上传时左上角取消
43001	无法获取唇语数据	获取数字活体的数字有问题
43002	权限异常，未获取权限	相机
43003	相机运行中出错	–
43004	视频录制中出错	不能存/启动失败/结束失败
43005	请勿晃动人脸,保持姿势	未获取到最佳图片
43006	视频大小不满足要求	视频大小不满足要求
43007	超时	预检测/动作活体
43008	检测中人脸移出框外	活体/数字/反光
43009	光线活体本地错误	–
43010	风险控制超出次数	用户重试太多次

43011	没有检测到读数声音	数字活体过程中没有发声
43012	初始化模型失败，请重试	初始化算法模型失败
43013	初始化 sdk 异常	WbCloudFaceVerifySdk 未被初始化
43014	简单模式本地加密失败	编码转换异常/加解密编码失败

WBFaceErrorDomainCompareNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
53100	网络异常	对比时，网络异常（请求未到达后台）
53200	网络异常	对比时，后台返回参数有误（请求到达后台）

WBFaceErrorDomainCompareServer

Code（错误码）	Description（描述）	Reason（详细实际原因）
其他错误码	透传后台错误码	–

接入示例

最近更新时间：2024-07-03 18:06:31

权威库网纹图片比对、自带对比源对比接入示例：

```
# 在DemoPage中单击某个按钮的代码逻辑：
//先填好数据
let inputData = new InputData(
      orderNo,
      this.appId,
      '1.0.0',
      this.nonce,
      this.userId,
      sign,
      this.licence,
      faceId
    )

    //设置必需参数
    let wbFaceVerifyConfig = new WbFaceVerifyConfig(inputData);
    //是否打开 sdk 日志开关
    wbFaceVerifyConfig.isEnableLog = true;

    WbCloudFaceVerifySdk.getInstance().initSdk(getContext(), wbFaceVerifyConfig, {
      onLoginSuccess: () => {
        DemoLog.i(this.TAG, `onLoginSuccess:`)
        WbCloudFaceVerifySdk.getInstance().startWbFaceVerifySdk(getContext(), {
          onFinish: (_result: WbFaceVerifyResult) => {
            DemoLog.i(this.TAG, `WbCloudFaceVerifySdk onFinish`)
            //todo 处理刷脸结果
        ……
            //处理完后释放 sdk
            //【特别注意】请在拿到 sdk 结果后对 sdk 进行释放，不要在页面结束时释放，避免未能获取刷脸回调结果的情况
            WbCloudFaceVerifySdk.getInstance().release();
          }
        })
      },
      onLoginFail: (error: WbFaceError) => {
        DemoLog.e(this.TAG, `onLoginFailed:JSON.stringify(error)`)
        //todo 处理登录错误逻辑
    ……
        //【特别注意】请在拿到 sdk 结果后对 sdk 进行释放，不要在页面结束时释放，避免未能获取刷脸回调结果的情况
        WbCloudFaceVerifySdk.getInstance().release();
      }
    })
```

人脸核身结果查询
查询核身结果

最近更新时间：2025-01-15 17:14:53

当您的用户完成核身认证后，如果您需要拉取人脸核身的视频和图片用于存证等其他需要，您可以调用查询核身结果接口来获取。
重要提示：
1. 您需要在前端完成刷脸的回调后，再来调用查询核身结果接口获取刷脸视频和照片。
2. 人脸核身完成后的相关业务数据请尽快拉取，超过人脸核身服务必须最短缓存时间的业务数据将完全清理。

> **注意**
> 当您的 App 接入的是我们的基础版或增强版 SDK 服务时，SDK 回调中只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过查询核身结果接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，查询核身结果接口无法查询到刷脸结果。

步骤如下：

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，获取方式请参见 获取 SIGN ticket 。
- 合作方为人脸核身服务生成签名，需要具有以下参数：

参数	说明	来源
appId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，字母/数字组成的字符串，本次人脸核身合作伙伴上送的订单号，唯一标识	合作方自行分配，不要带有特殊字符
version	默认值：1.0.0	-
ticket	合作伙伴服务端缓存的 ticket，注意是 SIGN 类型	获取方式请参见 获取 SIGN ticket
nonce	32位随机字符串，由字母和数字组成	合作方自行生成，不要带有特殊字符

基本步骤

1. 生成一个32位的随机字符串 nonce（由字母和数字组成，登录时也要用到）。
2. 将 appId、orderNo、version、ticket、nonce 共5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 。

人脸核身结果查询接口(升级)

请求

- **请求 URL：** `https://kyc1.qcloud.com/api/v2/base/queryfacerecord?orderNo=xxx`

> **注意**
> 为方便查询耗时，该请求url后面请拼接 orderNo 订单号参数。

- **请求方法：**POST
- **报文格式：**Content-Type: application/json
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填

appId	人脸核身控制台 申请的 WBappid	String	8	是
version	版本号，默认值：1.0.0	String	20	是
nonce	随机数	String	32	是
orderNo	订单号，字母/数字组成的字符串,是您需要查询结果的人脸核身订单号	String	32	是
sign	签名值，使用本页第一步生成的签名	String	40	是
getFile	是否需要获取人脸识别的视频和文件，值为1则返回视频和照片、值为2则返回照片、值为3则返回视频；其他则不返回	String	1	否
queryVersion	查询接口版本号(传1.0则返回 sdk 版本号和 trtc 标识)	String	8	否

响应

- **响应参数：**

参数	类型	说明
code	String	0：表示身份验证成功且认证为同一人
msg	String	返回结果描述
bizSeqNo	String	业务流水号
transactionTime	String	请求接口的时间
orderNo	String	订单编号
liveRate	String	活体检测得分
similarity	String	人脸比对得分
occurredTime	String	进行刷脸的时间
photo	Base 64 string	人脸核身时的照片，base64 位编码
video	Base 64 string	人脸核身时的视频，base64 位编码
sdkVersion	String	人脸核身时的 sdk 版本号
trtcFlag	String	Trtc 渠道刷脸则标识"Y"
appId	String	腾讯云控制台 申请的WBappid

- **响应示例：**


```
{"code":"0",
 "msg":"请求成功",
 "bizSeqNo":"22032920001184453211174015790894",
 "result":{
       "orderNo":"16170918856091743257691658 52",
       "liveRate":"99",
       "similarity":"88.01",
       "occurredTime":"20220329104717",
       "appId":"IDAXXXXX",
       "photo":"******",
       "video":"******",
       "bizSeqNo":"22032920001184453211174015790894",
       "sdkVersion":"1.12.12",
       "trtcFlag":"Y"},
 "transactionTime":"20220329111740"
 }
```


code 非 0 时，有时不返回图片和视频。

注意事项

- 照片和视频信息作为存证，合作伙伴可以通过此接口拉取视频等文件，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，默认不返回视频和照片信息。
- 由于照片和视频信息有可能超过 1M，考虑传输的影响，建议合作伙伴在使用时注意，建议获取比对结果用于后续流程处理和存证使用分开调用。避免网络传输带来的影响。
- 如果合作方是完成人脸核身流程后马上去拉取视频/照片，建议在查询接口加上一个查询机制：判断图片是否存在，轮询3次，每次2s。
- 照片和视频均为 base64 位编码，其中照片解码后格式一般为 JPG 和 PNG。视频格式解码后一般为 MP4。
- 服务端验证结果接口返回66660011无此查询结果的可能原因：

  1.1 66660017 验证次数超限后被风控，风控后的订单为无效，查询结果为无此查询结果。

  1.2 用户中途退出刷脸，没有完成人脸验证的流程，没有比对，查询结果为无此查询结果。

  1.3 66660018 操作超时，请退出重试 无此 ID 的用户身份信息，H5faceid 5分钟有效期，失效后无法进入刷脸流程，查询结果为无此查询结果。

  1.4 66660016 视频格式或大小不合法 文件或视频不合法，无法进行比对，查询结果为无此查询结果。

  1.5 400604 上传的视频非实时录制,被时间戳校验拦截，查询结果为无此查询结果。

  1.6 查询超过3天的订单返回无此查询结果。
- 响应参数请勿做强校验，后续可能会有扩展。

服务端响应码

详情请参见 SaaS 服务错误码 。

核身结果-多照片查询接口

最近更新时间：2024-06-17 17:09:41

人脸认证多张照片查询接口：获取人脸认证结果多张照片的接口。

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 。
- **合作方为人脸验证服务生成签名，需要具有以下参数：**

参数	说明	来源
appId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，字母/数字组成的字符串，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，不要带有特殊字符
version	默认值：1.0.0	-
ticket	合作伙伴服务端缓存的 ticket，注意是 SIGN 类型	获取方式见 SIGN ticket 获取
nonce	32位随机字符串，字母和数字	合作方自行生成，不要带有特殊字符

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 appId、orderNo、version 连同 ticket、nonce 共5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 。

人脸认证多张照片查询接口

请求

请求URL：https://kyc1.qcloud.com/api/v2/base/queryphotoinfo?orderNo=xxx

> **注意**
> 为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。

请求方法：POST
HTTP 请求 header：

参数名	是否必选	类型	说明
Content-Type	是	String	application/json

请求参数：

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
version	版本号，默认值：1.0.0	String	20	是
nonce	随机数	String	32	是
orderNo	订单号，字母/数字组成的字符串，合作方订单的唯一标识	String	32	是

sign	签名值，使用本页第一步生成的签名	String	40	是

响应

返回参数说明：

参数名	类型	说明
code	int	0：成功 非0：失败
msg	String	请求结果描述
bizSeqNo	String	请求业务流水号
orderNo	String	订单编号
occurredTime	String	刷脸时间（yyyyMMddHHmmss）
photoList	List	Base64 图像列表（返回1 - 3张照片），若照片不存在，则返回 null

返回示例：

```
{
"code": 0,
"msg": "请求成功",
"bizSeqNo": "业务流水号",
"result": {
"orderNo": "AAAAAA001",
"occurredTime": "20180907142653",
"photoList": ["第一个base64photo字符串", "第二个base64photo字符串", "第三个base64photo字符串"]
}
}
```

说明
success：false 无意义，合作伙伴可忽略，无需处理。

OCR SDK 接入
身份证 OCR SDK 接入
OCR 生成签名

最近更新时间：2024-06-17 17:09:41

> **注意**
> 如果因自身业务需要对 OCR 识别的影像文件进行存储或其他用途，请合作方务必自行保存订单号，通过订单号拉取 OCR 识别的影像文件是唯一方式。

准备步骤

- 下载 SDK，请到控制台自助接入列表页获取密码。
- uni 插件接入：https://ext.dcloud.net.cn/plugin?id=1233
- **前置条件：请合作方确保 NONCE ticket 已经正常获取，获取方式请参见 获取 NONCE ticket。**
- **NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket。**
- **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
appId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
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，不要带有特殊字符）
version	参数值为：1.0.0	-
ticket	合作伙伴服务端实时获取的 ticket，注意是 NONCE 类型	获取方式请参见 获取 NONCE ticket （所用的 userId 参数值需要和 SDK 里定义的 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，不要带有特殊字符）

> **说明**
> 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。

基本步骤

1. 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. appId、userId、version、ticket、nonceStr 共5个参数的值进行字典序排序。
3. 所有参数字符串拼接成一个字符串。
4. 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**
> 签名算法请参见 签名算法说明。

参考示例

- **请求参数：**

参数名	参数值
appId	IDAXXXXX
userId	userID19959248596551
nonceStr	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
ticket	XO99QfxIti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPlucaMS

- **字典排序后的参数为：**

```
[1.0.0, IDAXXXXX, XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS ,
kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551]
```

- **拼接后的字符串为：**

```
1.0.0IDAXXXXXXO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7
TuserID19959248596551
```

- **计算 SHA1 得到签名：**
  该字符串就是最终生成的签名（40 位），不区分大小写。

```
D7606F1741DDCF90757DA924EDCF152A200AC7F0
```

OCR Android SDK 接入

最近更新时间：2024-12-13 11:08:32

开发准备

权限检测

SDK 需要用到相机权限。

Android 6.0 及以上系统

SDK 运行时检测权限，需要用户授权。

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。
- 由于 SDK 的使用时间较短，为避免快速频繁开关相机可能会导致的手机抛出异常，故对 Android 6.0 以下手机没有做权限的检测。
- 为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 做以下权限检测：相机。

> **说明**
> 提示用户确认打开权限后再进行身份证 OCR 识别，可以使身份证识别体验更快更好。

CPU 平台设置

目前 SDK 支持 armeabi、armeabi-v7a、arm64-v8a，为了防止在其他 CPU 平台上 SDK Crash，建议在您的 App 的 build.gradle 里加上 abiFilter，如下所示：

```
defaultConfig {
    ndk {
          //设置支持的 so 库框架
          abiFilters 'armeabi-v7a', 'armeabi', 'arm64-v8a'
     }
}
```

配置流程

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 中添加如下配置）：

```
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')
}
```

混淆配置

云 OCR 产品的混淆规则分为两部分，分别是云 OCR SDK 的混淆规则（v2.7.X之后的版本 WbCloudOcrSdk 已经内置了混淆规则，接入方可以忽略不加载），云公共组件的混淆规则（接入方必须加载）。

- 云 OCR SDK 的混淆规则

```
######################云 ocr 混淆规则 ocr-BEGIN############################
-keepattributes InnerClasses
-keep public class com.tencent.cloud.huiyansdkocr.net.*$*{
  *;
}
-keep public class com.tencent.cloud.huiyansdkocr.net.*{
  *;
}
-keep public class com.tencent.fujikoli.recdetectdemo.Jni.ScanRecNative{*;}

######################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。

- 云公共组件的混淆规则

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

-keep public class com.tencent.cloud.huiyansdkface.normal.net.*$*{
  *;
}
-keep public class com.tencent.cloud.huiyansdkface.normal.net.*{
  *;
}
-keep public class com.tencent.cloud.huiyansdkface.normal.thread.*{
 *;
}
-keep public class com.tencent.cloud.huiyansdkface.normal.tools.*{
 *;
}
-keep public class com.tencent.cloud.huiyansdkface.normal.thread.*$*{
 *;
}

-keep public class com.tencent.cloud.huiyansdkface.normal.tools.secure.*{
 *;
}

-keep public class com.tencent.cloud.huiyansdkface.normal.tools.WLogger{
  *;
}
-keep class com.tencent.bugly.idasc.**{
  *;
}
#wehttp混淆规则
-dontwarn com.tencent.cloud.huiyansdkface.okio.**
-keep class com.tencent.cloud.huiyansdkface.okio.**{
  *;
}

# 里面 有 Java8的一些类 如 Duration
-dontwarn com.tencent.cloud.huiyansdkface.okhttp3.OkHttpClient$Builder
```

```
-keep class com.tencent.cloud.huiyansdkface.wehttp2.**{
  public <methods>;
}

-keep class com.tencent.cloud.huiyansdkface.okhttp3.**{
  public <methods>;
}

-keep interface com.tencent.cloud.huiyansdkface.wehttp2.**{
  public <methods>;
}
-keep public class com.tencent.cloud.huiyansdkface.wehttp2.WeLog$Level{
  *;
}
-keep class com.tencent.cloud.huiyansdkface.wejson.*{
  *;
}

-keep public class com.tencent.cloud.huiyansdkface.wehttp2.WeReq$ErrType{
  *;
}

-keep class * extends com.tencent.cloud.huiyansdkface.wehttp2.WeReq$WeCallback{
 public <methods>;
}

-keep class com.tencent.cloud.huiyansdkface.okio.**{
  *;
}
######################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 `kyc-cloud-normal-proguard-rules.pro` 拷贝到主工程根目录下，然后通过 `-include kyc-cloud-normal-proguard-rules.pro` 加入到您的混淆文件中。

接口调用

SDK 接口提供的是有 UI 模式：SDK 对接口进行封装并且实现了识别页面，合作方只需要调用接口，即可以快速拉起 SDK，接收结果回调。接入简单，适合快速接入。

SDK 接口调用方法

SDK 代码调用的入口为 `com.tencent.cloud.huiyansdkocr.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() {
       //    ...
       }
 /**
  * 初始化云Ocr SDK，完成登录
  * @param context         拉起SDK的context
  * @param wbocrtypemode   识别证件的模式，参数是枚举类型
  * @param data            第三方需要传入的参数
  * @param loginListener   登录SDK回调
  */
public void init(Context context,WBOCRTYPEMODE wbocrtypemode,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
     */
    Public void startActivityForOcr(Context context,IDCardScanResultListener){
    // ...
 }
/**
  * 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
 * @param resultCode 返回码， resultCode如果为0，则parcelableResult对象非空,有人像面结果或者国徽面的识别成功结果。
resultCode如果不为0，则parcelableResult对象为空，无任何识别成功的结果。
 * @param errorMsg  返回信息
 * @param result 识别结果类
 */

void onFinish(String resultCode, String resultMsg, Parcelable result);

}
```

NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。

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;  //每次OCR识别请求的唯一订单号：建议为32位字符串(不超过32位)
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
```

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 {
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
WBOCRSDKTypeContinus,//连续识别模式，识别完身份证人像面后，接着识别身份证国徽面，而不是退出相机页面
}
```

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 resultCode 返回码，识别成功返回 0，result不为空。识别失败返回非0，result为空。
 * @param resultMsg  返回信息
 * @param result 识别结果类
 */

void onFinish(String resultCode, String resultMsg, Parcelable result);
}
```

身份证识别结果类

回调 onFinish()的 EXIDCardResult 类中，该类属性如下所示：

```
    // 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;// 身份证人像面预览完整图片文件路径
    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;//身份证国徽面预览完整图片文件路径
    public String backWarning;//国徽面告警码
    public String sign;//签名
    public  String orderNo;  //每次OCR识别请求的唯一订单号：建议为32位字符串(不超过32位)
    public String ocrId;//识别的唯一标识

    //新版本新增的返回字段
    public String frontMultiWarning;//人像面多重告警码
    public String backMultiWarning;//国徽面多重告警码
    public String frontClarity;//人像面清晰度得分
    public String backClarity;//国徽面清晰度得分

    public String frontCropSrc;//身份证人像面切边照文件路径，需要调用方在调用初始化接口init()前配置
WbCloudOcrConfig.getInstance().setRetCrop(true)，frontCropSrc才有返回值;默认空。

    public String backCropSrc;//身份证国徽面切边照文件路径，需要调用方在调用初始化接口init()前配置
WbCloudOcrConfig.getInstance().setRetCrop(true)，backCropSrc才有返回值;默认空。

    public String frontCode;//人像面code,code为0，说明人像面识别成功。不为0，说明人像面识别失败。
    public String frontMsg;  //人像面msg
```

```
    public String backCode; //国徽面code,code为0，说明国徽面识别成功。不为0，说明国徽面识别失败。
    public String backMsg; //国徽面msg
```

接口参数说明

NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。
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），合作方需要传入 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	每次 OCR 识别请求的唯一订单号: 建议为32位字符串(不超过32位)	String	32	是
openApiAppId	业务流程唯一标识，即 wbappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
openApiAppVersion	接口版本号，默认填 1.0.0	String	20	是
openApiNonce	与服务端生成签名的随机数保持一致	String	32	是
openApiUserId	User 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

个性化参数设置

WbCloudOcrSdk.init() 里 Bundle data，除了必须要传的 InputData 对象之外，还可以由合作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 返回切边图

```
WbCloudOcrConfig.getInstance().setRetCrop(retCrop);//设置是否返回切边图，true返回。默认false不返回
```

OCR Android 错误码

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_ERROR_CANCELED_AUTH="200100";	用户授权时取消
IDOCR_USER_CANCEL="200101";	用户取消操作
IDOCR_RECOGNISE_TIME_OUT="200102";	识别超时
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="100105";	权限异常
IDOCR_LOGIN__ERROR="-10000";	登录错误
SERVER_FAIL="-30000";	内部服务错误
DECODE_FAIL="300101"	解码 emg 异常（包括 emg 为空）

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
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

```
# 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。

  //设置扫描识别的时间上限,默认 20 秒。用户有效设置范围是（0-60000）
  data.putLong(WbCloudOcrSDK.SCAN_TIME, 20000);
//初始化 SDK，得到是否登录 SDK 成功的结果
        WbCloudOcrSDK.getInstance().init(MainActivity.this,
WbCloudOcrSDK.WBOCRTYPEMODE.WBOCRSDKTypeContinus,data, new WbCloudOcrSDK.OcrLoginListener() {
            @Override
            public void onLoginSuccess() {  //登录成功,拉起 SDK 页面
            WbCloudOcrSDK.getInstance().startActivityForOcr(MainActivity.this,new
WbCloudOcrSDK.IDCardScanResultListener() {  //返退出 SDK 回调接口

@Override
public void onFinish(final String resultCode, final String resultMsg, Parcelable parcelableResult) {
    // 登录成功  第三方应用对ocr结果进行展示等操作
```

```
    // TODO: 2023/5/18 客户自己处理结果，下面代码仅供参考
runOnUiThread(new Runnable() {
                            @Override
                            public void run() {
                                if (parcelableResult==null){
                                    Toast.makeText(MainActivity.this,"识别结果为空，code="+resultCode+"
msg="+resultMsg,Toast.LENGTH_SHORT).show();
                                }else {
                                    Toast.makeText(MainActivity.this,"有识别结果，code="+resultCode+"
msg="+resultMsg,Toast.LENGTH_SHORT).show();
                                }

                            }
                        });


    Intent i;
    if (type.equals(WbCloudOcrSDK.WBOCRTYPEMODE.WBOCRSDKTypeBankSide)) {
        //银行卡识别，跳转到银行卡结果展示页面
        i = new Intent(MainActivity.this, BankOcrResultActivity.class);
        i.putExtra("bankcardresult", parcelableResult);
        i.putExtra("appId", appId);
        i.putExtra("envUrl", envUrl);
    } else if (type.equals(WbCloudOcrSDK.WBOCRTYPEMODE.WBOCRSDKTypeNormal) ||
            type.equals(WbCloudOcrSDK.WBOCRTYPEMODE.WBOCRSDKTypeContinus) ||
            type.equals(WbCloudOcrSDK.WBOCRTYPEMODE.WBOCRSDKTypeFrontSide) ||
            type.equals(WbCloudOcrSDK.WBOCRTYPEMODE.WBOCRSDKTypeBackSide)) {
        //身份证识别，跳转到身份证结果展示页面
        i = new Intent(MainActivity.this, ResultActivity.class);
        i.putExtra("idcardresult",parcelableResult);
    } else if (type.equals(WbCloudOcrSDK.WBOCRTYPEMODE.WBOCRSDKTypeDriverLicenseSide)) {
        //驾驶证识别，跳转到驾驶证结果展示页面
        i = new Intent(MainActivity.this, DriverLicenseOcrResultActivity.class);
        i.putExtra("driverlicenseresult",parcelableResult);
    } else {
        //行驶证识别，跳转到行驶证结果展示页面
        i = new Intent(MainActivity.this, VehicleLicenseOcrResultActivity.class);
        i.putExtra("vehiclelicenseresult",parcelableResult);
    }
    startActivity(i);
}
});
}
@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();
}
}
})
```

OCR iOS SDK 接入

最近更新时间：2024-01-10 15:19:21

配置流程

> **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. SDK 集成

SDK 支持 cocoapods 和手动两种方式集成。

1.1 SDK pod 集成

如果您的项目已经支持 cocoapods ，可以使用本方式集成 SDK，本示例使用的 cocoapods 版本号为 1.7.2。
下载的 OCR SDK 文件夹目录结构如下：

```
├── OCR_private_pod
│   ├── LICENSE
│   ├── Libs
│   └── WBOCRService.podspec
├── README
├── WBOCRDemo
│   ├── Podfile
│   ├── Podfile.lock
│   ├── Pods
│   ├── WBOCRDemo
│   ├── WBOCRDemo.xcodeproj
│   └── WBOCRDemo.xcworkspace
└── 接入指引.md
```

OCR_private_pod 文件夹下面是 OCR SDK 的 pod 仓库，这个仓库名称为 WBOCRService ，WBOCRDemo 文件夹下是示例 demo，通过 pod 方式集成，需要完成以下 3 个步骤：

1. 将 OCRprivatepod 文件夹移动到指定的位置（这个位置可以依据您项目文件的布局而定）。
2. 在 podfile 中引用 pod。
3. 执行 pod install 完成安装。

1.2 SDK 手动集成

- 下载 OCR SDK，找到 OCRprivatepod/Lib 文件夹，SDK 文件在这个文件夹下。
- 拖拽 SDK 文件到 Xcode 工程内 (请勾选Copy items if needed选项)，其中包括WBOCRService.bundle、WBOCRService.framework、YTCommon.framework、YTImageRefiner.framework 以及 tiny_opencv2.framework。

```
├── WBOCRService.bundle
├── WBOCRService.framework
├── YTCommon.framework
├── YTImageRefiner.framework
└── tiny_opencv2.framework
```

- 在Build Phases -> link with libraries 下加入如下依赖。

```
CoreTelephony.framework
CoreServices.framework
CoreMedia.framework
AssetsLibrary.framework
AVFoundation.framework
SystemConfiguration.framework
```

```
WebKit.framework
libc++.tbd
```

- Build Setting --> Linking --> Other Linker Flag 设置 增加 -ObjC 和 -lz linker flag

2. 集成过程中注意事项

cocoapods 集成时 `:path` 参数说明

使用 `:path` 参数，在指定路径下加载 pods ，这个路径是本质上是 WBOCRService.podspec 相对 podfile 的路径。在示例 WBOCRDemo 中，如下：

```
target 'WBOCRDemo' do
      pod 'WBOCRService',:path => '../OCR_private_pod'
end
```

调用接口

SDK 中需要使用 camera，需要在 Info.plist 中为 NSCameraUsageDescription 添加描述信息。

1. 概述

OCR SDK 对外提供以下两类证件的识别能力：

- 身份证识别
- 银行卡识别

SDK 提供以下接入方式：标准模式接入。传统的接入方式，SDK 完成整个识别流程，给用户返回识别结果。
在 SDK 附的 demo 中，有提供以上接入方式的接入示例，详细接入请参考 demo 工程中的示例代码。

2. SDK 头文件说明

SDK 一共对外暴露了 4 个头文件，如下所示：

```
├── Headers
│   ├── WBBankCardInfoModel.h
│   ├── WBIDCardInfoModel.h
│   ├── WBOCRConfig.h
│   ├── WBOCRService.h
```

这些头文件可以大致分为三类：

- 模型类（WBBankCardInfoModel 和 WBIDCardInfoModel ），这些模型类分别对应银行卡和身份证的识别结果字段。
- 配置类（WBOCRConfig），这个类提供了 SDK 的配置选项。
- 入口类（WBOCRService），这个类提供 SDK 的入口和回调。

2.1 模型类说明

模型类对外暴露识别结果，以身份证识别为例，WBIDCardInfoModel 类的实例返回身份证识别结果，结果中包含如下字段：

- `idcard` 公民身份号码
- `name` 姓名
- `sex` 性别
- `nation` 民族
- `address` 住址
- `birth` 出生
- `authority` 签发机关
- `validDate` 有效期限
- `frontFullImg` 国徽面截图
- `backFullImg` 人像面截图
- `orderNo` 每次OCR识别请求的唯一订单号：建议为32位字符串（不超过32位）
- `sign` 签名信息
- `warning` 识别结果警告信息
- `multiWarning` 多重告警码，人像面是 `frontMultiWarning`，国徽面对应 `backMultiWarning`
- `clarity` 图片清晰度，人像面是 `frontClarity`，国徽面对应 `backClarity`
- `frontCode` 人像面识别结果码
- `frontMsg` 人像面识别结果描述

- backCode 国徽面识别结果码
- backMsg 国徽面识别结果描述

开发者按需获取需要的信息。
其余证件识别与之类似，详情参考类头文件的字段注释。

2.2 配置类说明

WBOCRConfig 对外提供配置接口，下面的内容逐一介绍其核心接口。
WBOCRConfig 是一个单例，开发者必须通过制定的初始化方法 sharedConfig 初始化。支持的主要配置项如下：

```
/// default NO, 设置成 YES 之后, 识别结果没有告警才判定为识别成功.
@property (nonatomic, assign) BOOL checkWarnings;
/**
 * @brief 设置识别超时时间，默认是20.0s
 */
@property (nonatomic) NSTimeInterval timeoutInterval;
/**
 * @brief 是否需要录制视频。此功能目前仅适用于身份证识别，默认为NO，即不录制；设置为 YES，识别成功之后，会返回长度不超过 3s 的视频地址
 */
@property (nonatomic) BOOL needRecordVideo ;
/**
 * @brief 控制身份证识别是否返回切边图, BOOL类型，默认值为 NO
 * 当这个值设置为 YES 的时候,进行身份证人像面识别时, 切边图会在 frontCrop 字段返回; 进行身份证国徽面识别时,切边图会在 backCrop 字段返回
 * 当这个值设置为 NO  的时候,进行身份证识别的时候, frontCrop 和 backCrop 返回 nil
 */
@property (nonatomic) BOOL retCrop;
```

2.3 标准模式入口类说明

WBOCRService 是 SDK 的入口类，需要通过制定的初始化方法 sharedService 进行初始化。

```
/**
 * @brief WBOCRService 单例方法，通过调用该方法实例化 WBOCRService对象
 *
 * @return WBOCRService对象
 */
+ (nonnull instancetype) sharedService;

SDK支持身份证证件识别，提供 3 种识别模式，通过 WBOCRCardType 来定义。
/// * @brief  OCR SDK 提供证件的识别能力：身份证识别
///
/// WBOCRCardType 定义 SDK 不同的识别模式，下面描述这些模式：
/// 1. 身份证识别（识别身份证人像面和国徽面）
///     - WBOCRSDKTypeIDCardFrontSide: 身份证人像面识别模式，在 SDK 中完成人像面识别，识别完成之后，将本次识别结果返回第三方APP
///     - WBOCRSDKTypeIDCardBackSide: 身份证国徽面识别模式，在 SDK 中完成国徽面识别，识别完成之后，将本次识别结果返回第三方APP
///     - WBOCRSDKTypeIDCardContinuous:  身份证识别连拍模式，在 SDK 中完成人像面 + 国徽面识别，识别完成之后，将本次识别结果返回第三方APP

typedef NS_ENUM(NSInteger, WBOCRCardType) {
    WBOCRSDKTypeIDCardFrontSide = 1,
    WBOCRSDKTypeIDCardBackSide = 2,
    WBOCRSDKTypeIDCardContinuous = 7,
    WBOCRSDKTypeIDCardNormal NS_ENUM_DEPRECATED_IOS(9_0, 12_0,"User WBOCRSDKTypeIDCardContinuous") = 0,
};
```

SDK 接入分为两个步骤：

- init SDK，调用 init 接口完成 SDK 登录。

- startService，init 成功之后，调用这个接口开始识别。

> ⚠ 注意
> init 和 startService 接口一一对应。

init SDK 接口如下：

```
/// 标准版本 SDK 登录接口，这个接口完成 SDK 登录
/// @param sdkType      本次识别的卡证类型，详细参考 `WBOCRCardType`
/// @param appId        人脸核身控制台内申请的 WBappid
/// @param nonce        每次请求需要的一次性nonce，一次有效
/// @param userId       每个用户唯一的标识
/// @param sign         签名信息，由接入方后台提供，一次有效
/// @param orderNo      每次OCR识别请求的唯一订单号：建议为32位字符串(不超过32位)
/// @param version      openAPI接口版本号，默认为1.0.0
/// @param license      由腾讯服务分配的，和 bundle id 关联****
/// @param succeed      SDK 登录成功
/// @param failed       SDK 登录失败
- (void)initSDKWithSDKType:(WBOCRCardType)sdkType
                     appId:(nonnull NSString *)appId
                     nonce:(nonnull NSString *)nonce
                    userId:(nonnull NSString *)userId
                      sign:(nonnull NSString *)sign
                   orderNo:(nonnull NSString *)orderNo
                   version:(nonnull NSString *)version
                   license:(nonnull NSString *)license
                   succeed:(nonnull WBOCRServiceInitSucceedBlock)succeed
                    failed:(nonnull WBOCRServiceFailedBlock)failed;
startService 接口如下，通过 recognizeSucceed 接收识别成功回调，通过 failed 接收识别失败回调。
/// 标准版本 SDK 识别接口
/// @warning 调用这个接口前，需要完成 SDK 登录，即需要调用 initSDK 接口，并收到 succeed 回调
/// @param recognizeSucceed 识别成功回调
/// @param failed           识别失败回调
- (void) startOCRService:(nonnull WBOCRServiceRecognizeSuccessBlock)recognizeSucceed
                  failed:(nonnull WBOCRServiceFailedBlock)failed;
```

3. 接入示例

具体接入示例，请参考 demo 工程。

4. 识别结果处理

SDK 入口方法提供了三个回调 block ，通过这几个 block 来捕获识别结果或者异常状况。

4.1 WBOCRServiceInitSucceedBlock（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。

4.2 WBOCRServiceRecognizeSuccessBlock（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。

4.3 WBOCRServiceFailedBlock（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，具体的错误码对照表请参考 OCR iOS 错误码 文档。

```
NSError *error = [NSError errorWithDomain:@"com.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。

OCR iOS 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
300101	不合法请求(300102)	检查传入参数是否正确
300102	网络出小差啦	更换网络环境

多重告警码描述

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：清晰度 clarity 字段，多重警告码 multiWarningCode 和多重警告码描述 multiWarningMsg 字段。
- 身份证人像面识别时，在 WBIDCardInfoModel 类的头文件中，增加了两个字段：清晰度 frontClarity 字段，多重警告码 frontMultiWarning 字段。
- 身份证国徽面识别时，在 WBIDCardInfoModel 类的头文件中，增加了两个字段：清晰度 backClarity 字段，多重警告码 backMultiWarning 字段。

识别成功的时候，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。

OCR Harmony SDK 接入

最近更新时间：2024-11-15 17:21:12

申请权限

SDK 需要使用到网络和相机权限，需要在 module.json5 文件中申请 ohos.permission.INTERNET 和 ohos.permission.CAMERA 权限,权限申请参考鸿蒙官方文档和示例 demo 。

```
"requestPermissions": [
  {
    "name": "ohos.permission.INTERNET",
    "reason": "$string:reason_internet",
    "usedScene": {
      "abilities": [
        "FormAbility"
      ],
      "when": "always"
    }
  },
  {
    "name": "ohos.permission.CAMERA",
    "reason": "$string:reason_camera",
    "usedScene": {
      "abilities": [
        "FormAbility"
      ],
      "when": "inuse"
    }
  }
]
```

SDK 集成

在 oh-package.json5 中添加 SDK 的依赖。

```
{
  "name": "entry",
  "version": "1.0.0",
  "description": "Please describe the basic information.",
  "main": "",
  "author": "",
  "license": "",
  "dependencies": {
    "wb_kyc_ocr_library": "file:../entry/libs/wb_kyc_ocr_library.har"
  }
}
```

SDK调用流程

调用方法如下。
startOCRService 函数封装了 SDK 的调用，SDK 的调用流程可以分解为三个步骤：

- 合作方获取签名，SignTool 模拟合作方获取签名。
- 调用 WBOCRService.init 函数初始化 SDK
- init 成功之后，调用 WBOCRService.startService 开始识别。

> ⚠ 注意：

> init 和 start 成对调用，一次 init 一次 start。

卡片类型在 init 接口的 OCRParam.sdkType 里面设置。支持的卡片类型见 CardType 定义。

```
export enum CardType {
  idCard,
  idCardFrontSide,
  idCardBackSide,
  bankCard,
  vehicleLicenseHomePage,
  vehicleLicenseSecondaryPage,
  driverLicense,
}
```

startOCRService 的源代码如下。

```
startOCRService(type: CardType) {
  let service = WBOCRService.getInstance()
  let param = new OCRParam()
  let signTool = new SignTool()
  signTool.getSign().then((data) => {
    console.log(data);
    this.prompt = `sign: ${data}`
    param.appId = signTool.appid
    param.orderNo = signTool.randomString(12)
    param.nonce = signTool.nonce
    param.userId = signTool.userid
    param.sign = data
    param.sdkType = type;

    service.init(param, {
      onSuccess: () => {
        console.log('SDK 初始化成功 - 调用 start 接口')
        service.startService({
          onSuccess: (res: object) => {
            console.log('SDK 识别成功')
            console.log(JSON.stringify(res))
            this.prompt = JSON.stringify(res)
          },
          onFailure: (error: OCRError) => {
            console.error('SDK 识别失败！')
            console.error(JSON.stringify(error))
            this.prompt = JSON.stringify(error)
          }
        })
      },
      onFailure: (error: OCRError) => {
        console.log('SDK 初始化失败！')
        console.log(JSON.stringify(error))
        this.prompt = JSON.stringify(error)
      }
    })
  }).catch((e: BusinessError) => {
    console.error(`err:${e.code},${e.message},${e.name}`)
    this.prompt = `err:${e.code},${e.message},${e.name}`
  })
}
```

接入方 App 需要接收和处理 init 以及 start 接口的回调结果。
其中，卡片识别结果定义在如下几个类里面，不同的卡片类型，对应相应类型的结果对象。

身份证识别

身份证识别的时候，入参卡片类型可以选择如下三个：

- idCard，进入是连续识别模式，先识别人像面，再识别国徽面。
- idCardFrontSide，人像面识别。
- idCardBackSide，国徽面识别。

身份证识别的结果返回类型为 IdCardFrontModel，参数字段如下。

```
/*
 * @brief WBIDCardInfoModel类封装了身份证的正反面信息
 *        SDK会将识别结果包装成一个WBIDCardInfoModel实例，通过回调block通知第三方

 * @detail 字段含义
 - idcard        公民身份号码
 - name          姓名
 - sex           性别
 - nation        民族
 - address       住址
 - birth         出生
 - authority     签发机关
 - validDate     有效期限
 - frontFullImg  国徽面截图
 - backFullImg   人像面截图
 - warning       识别结果警告信息
 - multiWarning  多重告警码，人像面是frontMultiWarning，国徽面对应backMultiWarning
 - clarity       图片清晰度，人像面是frontClarity，国徽面对应backClarity
 */

export class IDCardModel extends KYCOCRBaseModel {
  /// 身份证人像面信息
  idcard = ''
  name = ''
  sex = ''
  nation = ''
  address = ''
  birth = ''
  /// 身份证国徽面信息
  authority = ''
  validDate = ''
  /// 人像面/国徽面识别结果截图信息
  frontFullImg = ''
  backFullImg = ''
  frontCode = ''
  frontMsg = ''
  backCode = ''
  backMsg = ''
  /// warning，人像面/国徽面识别结果对应的警告信息
  frontWarning = ''
  frontMultiWarning = ''
  backWarning = ''
  backMultiWarning = ''
  frontClarity = ''
  backClarity = ''
  /// 人像面/国徽面识别视频 URL
  frontVideoURL = ''
  backVideoURL = ''
  /// 人像面/国徽面切边图 base64 编码字符串
  frontCrop = ''
  backCrop = ''
}

export { KYCOCRBaseModel }
```

Harmony OCR 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	SDK 未登录	未登录 SDK (调用 startOCRService 之前,需要调用 init 登录)
100103	App 没有相机权限	App 没有相机权限
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
300101	不合法请求(300102)	检查传入参数是否正确
300102	网络出小差啦	更换网络环境
400100	SDK 重复进入	重复调用 SDK

OCR 验证结果

最近更新时间：2024-06-17 17:09:41

服务端验证结果

此方式适用于：
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（服务端上传 order_no 和 WBappid 查询）。

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
appId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
order_no	订单号，字母/数字组成的字符串，本次人脸核身合作伙伴上传的订单号，唯一标识	合作方自行分配，不要带有特殊字符
version	默认值：1.0.0	-
ticket	合作伙伴服务端缓存的 ticket，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，不要带有特殊字符

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、order_no、version、ticket、nonceStr 共5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 。

身份证 OCR 识别结果查询接口

请求

- **请求URL：** `https://kyc1.qcloud.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	请您 点此链接 扫描二维码添加腾讯云人脸核身小助手，进行线下对接获取	String	腾讯服务分配	是
order_no	订单号，字母/数字组成的字符串，合作方订单的唯一标识	String	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	String	1	否
nonce	随机数	String	32	是
version	版本号，默认值：1.0.0	String	20	是
sign	签名值，使用本页第一步生成的签名	String	40	是
get_native	是否需要映射身份证号籍贯信息，值为1则返回映射信息	String	1	否

- **请求示例：**

```
https://kyc1.qcloud.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
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（例：19920320）
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（例：20160725-20260725）
authority	String	backCode为 0 返回：发证机关
frontPhoto	Base64 String	人像面照片，转换后为 JPG 格式
backPhoto	Base64 String	国徽面照片，转换后为 JPG 格式
frontCrop	Base64 String	人像面切边照片，切边图在识别原图少边或者存在遮挡的情况有小概率可能会导致切图失败，该字段会返回空；如切边图为空时建议可使用原图替代
backCrop	Base64 String	国徽面切边照片，切边图在识别原图少边或者存在遮挡的情况有小概率可能会导致切图失败，该字段会返回空；如切边图为空时建议可使用原图替代
headPhoto	Base64 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 身份证 OCR 错误码
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 身份证 OCR 错误码
operateTime	String	做 OCR 的操作时间（例：2020-02-27 17:08:03）
frontMultiWarning	String	正面多重告警码，含义请参考 身份证 OCR 错误码
backMultiWarning	String	反面多重告警码，含义请参考 身份证 OCR 错误码
frontClarity	String	正面图片清晰度
backClarity	String	反面图片清晰度
nativePlace	String	籍贯信息

响应示例：


```
{
    "code":"0",
    "msg":"请求成功",
    "bizSeqNo":"2106212000118443841735580787676765",
    "result":{
          "bizSeqNo":"2106212000118443841735580787676765",
          "transactionTime":"20210621173558",
          "backCode":"0",
          "orderNo":"h1jw98k72ffe3de249qmf1723673v31v",
          "name":"xxx",
          "sex":"男",
          "nation":"汉",
          "birth":"19881001",
          "address":"xxxxxxxxxx",
          "idcard":"xxxxxxxxxxxxxxxxxx",
          "validDate":"20190128-20390128",
          "authority":"xxxxxx",
          "operateTime":"2021-06-21 17:35:27",
          "frontWarnCode":"00000000",
          "backWarnCode":"00000000",
          "frontMultiWarning":"00000000",
          "backMultiWarning":"00000000",
          "frontClarity":"64",
          "backClarity":"72",
          "nativePlace":"xxxxxxxxx",
          "success":false,
          "frontCode":"0"
},
"transactionTime":"20210621173558"
}
```


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；如果不上传参数或者参数为空，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，详情请参见 身份证 OCR 错误码 。

- success：false 无意义，合作伙伴可忽略，无需处理。

身份证 OCR 错误码

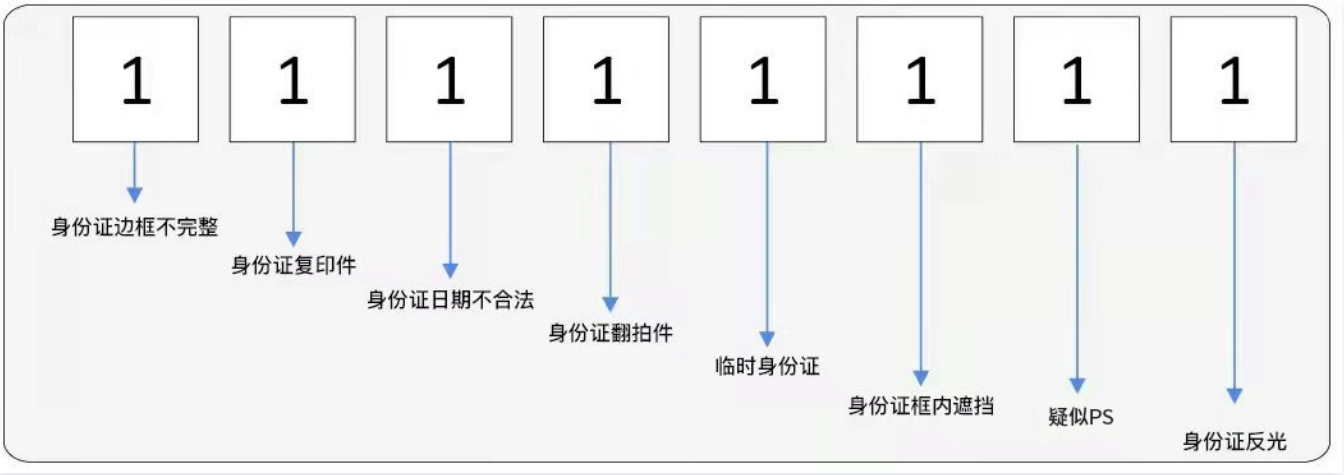
最近更新时间：2024-10-11 15:23:41

返回码	返回信息	处理措施	是否收费

66660015	视频或图片异常	视频或图片异常，请更换图片重试	否
66660020	OCR 结果查询不到	身份证识别无结果，确认订单号是否正确	否
66661001	请调整角度，保持清晰	识别错误，非身份证件或图像质量问题，需重新识别	否
66661003	身份证已失效	未能识别，不支持失效身份证	否
66661004	身份证日期不正确	未能识别	否
66661005	图像解码异常	未能识别，需重新识别	否
66661006	非身份证人像面	未能识别，需重新更换到人像面	否
66661007	非身份证国徽面	未能识别，需重新更换到国徽面	否
66661008	姓名或身份证号未能识别	姓名或身份证号识别结果为空，或有效性校验不通过，如身份证号码不满足18位、身份证号码奇偶校验不通过、姓名只有一个汉字等情况	否
66661010	格式有误	请上传图片格式	否
66661012	签发机关未能识别	签发机关未能识别	否
66661099	处理失败	处理失败	否
300101	加解密失败	退出重试	否

告警码信息

由8位字符串来表示身份证 OCR 识别的告警码，每一位代表一个告警码，“1”表示有告警，“0”表示无任何告警。



示例

- **00000000：表示无任何告警。**
- **10010100：表示同时存在3个告警码，分别是告警“身份证边框不完整、身份证翻拍件、身份证框内遮挡”，也称之为多重告警。**

> **注意**
> 告警码只是一种告警提示，不影响识别，默认处理为通过，客户可根据自身情况对告警码做相应的逻辑处理。

银行卡 OCR SDK 接入
OCR 生成签名

最近更新时间：2024-06-17 17:54:01

> **注意**
> 如果因自身业务需要对 OCR 识别的影像文件进行存储或其他用途，请合作方务必自行保存订单号，通过订单号拉取 OCR 识别的影像文件是唯一方式。

准备步骤

- 下载 SDK，请到控制台自助接入列表页获取密码。
- uni 插件接入：https://ext.dcloud.net.cn/plugin?id=1233。
- **前置条件：请合作方确保 NONCE ticket 已经正常获取，获取方式请参见 获取 NONCE ticket。**
- **NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket。**
- **合作方根据本次人脸核身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	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
userId	用户唯一标识	合作方自行分配（与 SDK 里面定义的 userId 保持一致，不要带有特殊字符）
version	参数值：1.0.0	-
ticket	合作伙伴服务端实时获取的 ticket，注意是 NONCE 类型	获取方式请参见 获取 NONCE ticket（所用的 userId 参数值需要和 SDK 里面定义的 userId 保持一致）
nonceStr	必须是32位随机数	合作方自行生成（与接口里面定义的随机数保持一致，不要带有特殊字符）

> **注意**
> 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。

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5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**
> 签名算法请参见 签名算法说明 。

参考示例

- **请求参数：**

参数	参数值
appId	IDAXXXXX
userId	userID19959248596551
nonceStr	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
ticket	XO99QfxIti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPlucaMS

- **字典排序后的参数为：**

```
[1.0.0, IDAXXXXX, XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS ,
kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551]
```

- **拼接后的字符串为：**

```
1.0.0IDAXXXXXXO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7
TuserID19959248596551
```

- **计算 SHA1 得到签名：**
  该字符串就是最终生成的签名（40位），不区分大小写。

```
D7606F1741DDCF90757DA924EDCF152A200AC7F0
```

OCR Android SDK 接入

最近更新时间：2025-01-15 17:14:53

开发准备

权限检测

SDK 需要用到相机权限

Android 6.0 及以上系统

SDK 运行时提示权限，需要用户授权。

Android 6.0 以下系统

- Android 并没有运行时权限，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 做相机权限检测，提示用户确认打开了权限后再进行银行卡 OCR 识别，可以使银行卡识别体验更快更好。

CPU 平台设置

目前 SDK 支持 armeabi、armeabi-v7a、arm64-v8a，为了防止在其他 CPU 平台上 SDK Crash，建议在您的 App 的 build.gradle 里加上 abiFilter，如下所示：

```
defaultConfig {
    ndk {
          //设置支持的 so 库框架
          abiFilters 'armeabi-v7a', 'armeabi', 'arm64-v8a'
     }
}
```

配置流程

接入配置

注意事项

- OCR SDK（WbCloudOcr）最低支持到 Android API 14: Android 4.0(ICS)，请在构建项目时注意版本是否支持。
- WbCloudOcr 将以 AAR 文件的形式提供。
- 需要为 Android SDK 添加依赖，方式如下：
  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-proRelease', ext: 'aar')
//2.云公共组件
compile(name: 'WbCloudNormal-release-v5.1.1', ext: 'aar')
}
```

混淆配置

云 OCR 产品的混淆规则分为两部分，分别是云 OCR SDK 的混淆规则(v2.7.X之后的版本 WbCloudOcrSdk 已经内置了混淆规则，接入方可以忽略不加载)，云公共组件的混淆规则（接入方必须加载）。

- 云 OCR SDK 的混淆规则

```
######################云 ocr 混淆规则 ocr-BEGIN############################
-keepattributes InnerClasses
-keep public class com.tencent.cloud.huiyansdkocr.net.*$*{
  *;
}
-keep public class com.tencent.cloud.huiyansdkocr.net.*{
  *;
}
-keep public class com.tencent.fujikoli.recdetectdemo.Jni.ScanRecNative{*;}

######################云 ocr 混淆规则 ocr-END############################
```

- 云公共组件的混淆规则

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

-keep public class com.tencent.cloud.huiyansdkface.normal.net.*$*{
  *;
}
-keep public class com.tencent.cloud.huiyansdkface.normal.net.*{
  *;
}
-keep public class com.tencent.cloud.huiyansdkface.normal.thread.*{
 *;
}
-keep public class com.tencent.cloud.huiyansdkface.normal.tools.*{
 *;
}
-keep public class com.tencent.cloud.huiyansdkface.normal.thread.*$*{
 *;
}

-keep public class com.tencent.cloud.huiyansdkface.normal.tools.secure.*{
 *;
}

-keep public class com.tencent.cloud.huiyansdkface.normal.tools.WLogger{
  *;
}
-keep class com.tencent.bugly.idasc.**{
  *;
}
#wehttp混淆规则
-dontwarn com.tencent.cloud.huiyansdkface.okio.**
-keep class com.tencent.cloud.huiyansdkface.okio.**{
  *;
}

# 里面 有 Java8的一些类 如 Duration
-dontwarn com.tencent.cloud.huiyansdkface.okhttp3.OkHttpClient$Builder
-keep class com.tencent.cloud.huiyansdkface.wehttp2.**{
  public <methods>;
}
```

```
-keep class com.tencent.cloud.huiyansdkface.okhttp3.**{
  public <methods>;
}

-keep interface com.tencent.cloud.huiyansdkface.wehttp2.**{
  public <methods>;
}
-keep public class com.tencent.cloud.huiyansdkface.wehttp2.WeLog$Level{
  *;
}
-keep class com.tencent.cloud.huiyansdkface.wejson.*{
  *;
}

-keep public class com.tencent.cloud.huiyansdkface.wehttp2.WeReq$ErrType{
  *;
}

-keep class * extends com.tencent.cloud.huiyansdkface.wehttp2.WeReq$WeCallback{
 public <methods>;
}

-keep class com.tencent.cloud.huiyansdkface.okio.**{
  *;
}
######################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 `kyc-cloud-normal-proguard-rules.pro` 拷贝到主工程根目录下，然后通过 `-include kyc-cloud-normal-proguard-rules.pro` 加入到您的混淆文件中。

接口调用

SDK 接口提供的是有 UI 模式：：SDK 对接口进行封装并且实现了识别页面，合作方只需要调用接口，即可以快速拉起 SDK，接收结果回调。接入简单，适合快速接入。

SDK 接口调用方法

SDK 代码调用的入口为 `com.tencent.cloud.huiyansdkocr.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() {
       //    ...
       }


 /**
  * 初始化云Ocr SDK，完成登录
  * @param context         拉起SDK的context
  * @param wbocrtypemode   识别证件的模式，参数是枚举类型
  * @param data            第三方需要传入的参数
  * @param loginListener   登录SDK回调
  */
public void init(Context context,WBOCRTYPEMODE wbocrtypemode,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
```

```
     */
    public void startActivityForOcr(Context context,IDCardScanResultListener){
  // ...
 }
/**
  * 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
 * @param errorCode 返回码，识别成功返回 0，result非空。识别失败返回非0，result为空。
 * @param errorMsg  返回信息
 * @param result 识别结果类
 */
void onFinish(String errorCode, String errorMsg, Parcelable result);
}
```

NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。 `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;  //每次OCR识别请求的唯一订单号：建议为32位字符串（不超过32位）
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 WbCloudFaceVerifySdk.InputData 对象中（它是一个 Serializable 对象）。

ResultOfBank 代表 SDK 返回的识别银行卡的结果，该类属性如下所示：

```
public String ocrId;//识别的唯一标识
public String bankcardNo;//识别的银行卡号
public String bankcardValidDate;//识别的银行卡的有效期
public String orderNo;//每次OCR识别请求的唯一订单号：建议为32位字符串（不超过32位）
public String warningMsg;   //识别的警告信息
public String warningCode;  //识别的警告码
public Bitmap bankcardNoPhoto;//识别的银行卡的卡号图片
public String ocrId;  //识别的唯一标识
public String bankcardNo; //银行卡号
public String bankcardValidDate; //银行卡的有效期
public String orderNo; // 每次OCR识别请求的唯一订单号：建议为32位字符串（不超过32位）
public String warningMsg;
public String warningCode;
public String bankcardNoPhoto;
public String bankcardNoPhotoSrc; //银行卡的卡号切边图的路径
public String bankcardFullPhotoSrc;//银行卡图片存放路径
public String retry;
public String sign;
//新增
public String multiWarningCode;//多重告警码
public String multiWarningMsg;//多重告警信息
public String clarity;//清晰度得分
//新版本新增字段
```

```
public String bankcardName;//银行卡名称
public String bankcardType;//银行卡类型
public String bankcardInfo;//银行卡信息
```

登录回调接口

```
/**
  * 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 返回码，识别成功返回 0
 * @param errorMsg  返回信息
 * @param result 识别结果类
 */
void onFinish(String errorCode, String errorMsg, Parcelable result);
}
```

银行卡识别结果类

ResultOfBank 类中，通过 onFinish()回调参数 parcelableResult 获得，该类属性如下所示：

```
public String ocrId;//识别的唯一标识
public String bankcardNo;//识别的银行卡号
public String bankcardValidDate;//识别的银行卡的有效期
public String orderNo;// 每次OCR识别请求的唯一订单号：建议为32位字符串（不超过32位）
public String warningMsg;    //识别的警告信息
public String warningCode;  //识别的警告码
public Bitmap bankcardNoPhoto;//识别的银行卡的卡号图片
public String ocrId;  //识别的唯一标识
 public String bankcardNo; //银行卡号
 public String bankcardValidDate; //银行卡的有效期
 public String orderNo; // 每次OCR识别请求的唯一订单号：建议为32位字符串（不超过32位）
 public String warningMsg;
 public String warningCode;
 public String bankcardNoPhoto;
 public String bankcardNoPhotoSrc; //银行卡的卡号切边图的路径
 public String bankcardFullPhotoSrc;//银行卡图片存放路径
public String retry;
 public String sign;
//新增
public String multiWarningCode;//多重告警码
public String multiWarningMsg;//多重告警信息
public String clarity;//清晰度得分
 //新版本新增字段
public String bankcardName;//银行卡名称
```

```
public String bankcardType;//银行卡类型
public String bankcardInfo;//银行卡信息
```

接口参数说明

NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。
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），合作方需要加入 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加入对应的数据。

参数	说明	类型	长度（字节）	是否必填
orderNo	每次 OCR 识别请求的唯一订单号: 建议为32位字符串(不超过32位)	String	32	是
openApiAppId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
openApiAppVersion	接口版本号，默认填1.0.0	String	20	是
openApiNonce	与服务端生成签名的随机数保持一致	String	32	是
openApiUserId	User 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

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，还可以由合作方为其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秒内若识别成功则退出扫描界面，否则一直识别，直到20秒后直接退出扫描界面。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 扫描识别证件（身份证、银行卡）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);
```

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);
  //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);
```

OCR Android 错误码

终端返回错误码

错误码	错误码描述
IDOCR_LOGIN_PARAMETER_ERROR = -20000	传入参数有误
IDOCR_USER_CANCEL = 200101	用户取消操作
IDOCR_RECOGNISE_TIME_OUT="200102"	识别超时

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 = 100105	权限异常
IDOCR_LOGIN__ERROR = -10000	登录错误
SERVER_FAIL = -30000	内部服务错误
DECODE_FAIL="300101"	解码 emg 异常(包括 emg 为空)

后台返回错误码

错误码	错误码描述
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
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

```
# 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。
  //设置扫描识别的时间上限,默认 20 秒，建议默认。用户有效设置范围（0-60000）
  data.putLong(WbCloudOcrSDK.SCAN_TIME, 20000);
//初始化 sdk，得到是否登录 sdk 成功的结果
        WbCloudOcrSDK.getInstance().init(MainActivity.this, WbCloudOcrSDK.WBOCRTYPEMODE.WBOCRSDKTypeBankSide,
data, new WbCloudOcrSDK.OcrLoginListener() {
            @Override
            public void onLoginSuccess() {  //登录成功,拉起 SDK 页面
WbCloudOcrSDK.getInstance().startActivityForOcr(MainActivity.this,new  WbCloudOcrSDK.IDCardScanResultListener()
{  //返退出 SDK 回调接口
                    @Override
                    public void onFinish(
```

```
final String resultCode, final String resultMsg, Parcelable parcelableResult
) {
        // 登录成功  第三方应用对扫描的结果进行操作
// resultCode如果为0，则parcelableResult对象不为空。
// resultCode如果不为0，则parcelableResult对象为空。
runOnUiThread(new Runnable() {
    @Override
    public void run() {
        if (!“0”.equals(resultCode)){
            Toast.makeText(MainActivity.this,"识别结果为空，code="+resultCode+"
msg="+resultMsg,Toast.LENGTH_SHORT).show();
        }else {
            Toast.makeText(MainActivity.this,"有识别结果，code="+resultCode+"
msg="+resultMsg,Toast.LENGTH_SHORT).show();
        }

    }
});


// 登录成功  第三方应用对ocr结果进行展示等操作
// TODO: 客户自己处理结果，下面代码仅供参考
Intent i;
if (type.equals(WbCloudOcrSDK.WBOCRTYPEMODE.WBOCRSDKTypeBankSide)) {
    //银行卡识别，跳转到银行卡结果展示页面
    i = new Intent(MainActivity.this, BankOcrResultActivity.class);
    i.putExtra("bankcardresult", parcelableResult);
    i.putExtra("appId", appId);
    i.putExtra("envUrl", envUrl);
}
});
}
@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();
}
}
});
```

OCR iOS SDK 接入

最近更新时间：2024-08-15 15:26:21

配置流程

> **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. SDK 集成

SDK 支持 cocoapods 和手动两种方式集成。

1.1 SDK pod 集成

如果您的项目已经支持 cocoapods ，可以使用本方式集成 SDK，本示例使用的 cocoapods 为 1.7.2。
下载的 OCR SDK 文件夹目录结构如下：

```
├── OCR_private_pod
│   ├── LICENSE
│   ├── Libs
│   └── WBOCRService.podspec
├── README
├── WBOCRDemo
│   ├── Podfile
│   ├── Podfile.lock
│   ├── Pods
│   ├── WBOCRDemo
│   ├── WBOCRDemo.xcodeproj
│   └── WBOCRDemo.xcworkspace
└── 接入指引.md
```

OCR_private_pod 文件夹下面是 OCR SDK 的 pod 仓库，这个仓库名称为 WBOCRService ，WBOCRDemo 文件夹下是示例 demo，通过 pod 方式集成，需要完成以下 3 个步骤：

1. 将 OCRprivatepod 文件夹移动到指定的位置（这个位置可以依据您项目文件的布局而定）。
2. 在 podfile 中引用 pod。
3. 执行 pod install 完成安装。

1.2 SDK 手动集成

- 下载 OCR SDK，找到 OCRprivatepod/Lib 文件夹，SDK 文件在这个文件夹下。
- 拖拽 SDK 文件到 Xcode 工程内 (请勾选Copy items if needed选项)，其中包括WBOCRService.bundle、WBOCRService.framework、YTCommon.framework、YTImageRefiner.framework 以及 tiny_opencv2.framework。

  ```
  ├── WBOCRService.bundle
  ├── WBOCRService.framework
  ├── YTCommon.framework
  ├── YTImageRefiner.framework
  └── tiny_opencv2.framework
  ```

- 在Build Phases -> link with libraries 下加入如下依赖。

  ```
  CoreTelephony.framework
  CoreServices.framework
  CoreMedia.framework
  AssetsLibrary.framework
  AVFoundation.framework
  SystemConfiguration.framework
  ```

```
WebKit.framework
libc++.tbd
```

- Build Setting --> Linking --> Other Linker Flag 设置 增加 -ObjC 和 -lz linker flag

2. 集成过程中注意事项

cocoapods 集成时 `:path` 参数说明

使用 `:path` 参数，在指定路径下加载 pods ，这个路径是本质上是 WBOCRService.podspec 相对 podfile 的路径。在示例 WBOCRDemo 中，如下：

```
target 'WBOCRDemo' do
pod 'WBOCRService',:path => '../OCR_private_pod'
end
```

接口调用

SDK 中需要使用 camera，需要在 Info.plist 中为 NSCameraUsageDescription 添加描述信息。

1. 概述

OCR SDK 对外提供以下两类证件的识别能力：

- 身份证识别
- 银行卡识别

SDK 提供以下接入方式：标准模式接入。传统的接入方式，SDK 完成整个识别流程，给用户返回识别结果。
在 SDK 附的 demo 中，有提供以上接入方式的接入示例，详细接入请参考 demo 工程中的示例代码。

2. SDK 头文件说明

SDK 一共对外暴露了 4 个头文件，如下所示：

```
├── Headers
│   ├── WBBankCardInfoModel.h
│   ├── WBIDCardInfoModel.h
│   ├── WBOCRConfig.h
│   ├── WBOCRService.h
```

这些头文件可以大致分为三类：

- 模型类（WBBankCardInfoModel 和 WBIDCardInfoModel ），这些模型类分别对应银行卡和身份证的识别结果字段。
- 配置类（WBOCRConfig），这个类提供了 SDK 的配置选项。
- 入口类（WBOCRService），这个类提供 SDK 的入口和回调。

2.1 模型类说明

模型类对外暴露识别结果，以身份证识别为例，WBIDCardInfoModel 类的实例返回身份证识别结果，结果中包含如下字段：

- `idcard` 公民身份号码
- `name` 姓名
- `sex` 性别
- `nation` 民族
- `address` 住址
- `birth` 出生
- `authority` 签发机关
- `validDate` 有效期限
- `frontFullImg` 国徽面截图
- `backFullImg` 人像面截图
- `orderNo` 每次OCR识别请求的唯一订单号：建议为32位字符串（不超过32位）
- `sign` 签名信息
- `warning` 识别结果警告信息
- `multiWarning` 多重告警码，人像面是 `frontMultiWarning`，国徽面对应 `backMultiWarning`
- `clarity` 图片清晰度，人像面是 `frontClarity`，国徽面对应 `backClarity`

开发者按需获取需要的信息。
其余证件识别与之类似，详情参考类头文件的字段注释。

2.2 配置类说明

WBOCRConfig 对外提供配置接口，下面的内容逐一介绍其核心接口。
WBOCRConfig 是一个单例，开发者必须通过制定的初始化方法 sharedConfig 初始化。支持的主要配置项如下：

```
/// default NO，设置成 YES 之后，识别结果没有告警才判定为识别成功.
@property (nonatomic, assign) BOOL checkWarnings;
/**
 * @brief 设置识别超时时间，默认是20.0s
 */
@property (nonatomic) NSTimeInterval timeoutInterval;
/**
 * @brief 是否需要录制视频。此功能目前仅适用于身份证识别，默认为NO，即不录制；设置为 YES，识别成功之后，会返回长度不超过 3s 的视频地址
 */
@property (nonatomic) BOOL needRecordVideo ;
/**
 * @brief 控制身份证识别是否返回切边图， BOOL类型，默认值为 NO
 * 当这个值设置为 YES 的时候,进行身份证人像面识别时， 切边图会在 frontCrop 字段返回; 进行身份证国徽面识别时,切边图会在 backCrop 字段返回
 * 当这个值设置为 NO  的时候,进行身份证识别的时候， frontCrop 和 backCrop 返回 nil
 */
@property (nonatomic) BOOL retCrop;
```

2.3 标准模式入口类说明

WBOCRService 是 SDK 的入口类，需要通过制定的初始化方法 sharedService 进行初始化。

```
/**
 * @brief WBOCRService 单例方法，通过调用该方法实例化 WBOCRService对象
 *
 * @return WBOCRService对象
 */
+ (nonnull instancetype) sharedService;

SDK支持身份证证件识别，提供 3 种识别模式，通过 WBOCRCardType 来定义。
/// * @brief  OCR SDK 提供证件的识别能力：身份证识别
///
/// WBOCRCardType 定义 SDK 不同的识别模式，下面描述这些模式：
/// 1. 身份证识别（识别身份证人像面和国徽面）
///     - WBOCRSDKTypeIDCardFrontSide: 身份证人像面识别模式，在 SDK 中完成人像面识别，识别完成之后，将本次识别结果返回第三方APP
///     - WBOCRSDKTypeIDCardBackSide: 身份证国徽面识别模式，在 SDK 中完成国徽面识别，识别完成之后，将本次识别结果返回第三方APP
///     - WBOCRSDKTypeIDCardContinuous:  身份证识别连拍模式，在 SDK 中完成人像面 + 国徽面识别，识别完成之后，将本次识别结果返回第三方APP

typedef NS_ENUM(NSInteger, WBOCRCardType) {
    WBOCRSDKTypeIDCardFrontSide = 1,
    WBOCRSDKTypeIDCardBackSide = 2,
    WBOCRSDKTypeIDCardContinuous = 7,
    WBOCRSDKTypeIDCardNormal NS_ENUM_DEPRECATED_IOS(9_0, 12_0,"User WBOCRSDKTypeIDCardContinuous") = 0,
};
```

SDK 接入分为两个步骤：

- init SDK，调用 init 接口完成 SDK 登录。
- startService，init 成功之后，调用这个接口开始识别。

> ⚠ 注意

> init 和 startService 接口一一对应。

init SDK 接口如下：

```
/// 标准版本 SDK 登录接口，这个接口完成 SDK 登录
/// @param sdkType      本次识别的卡证类型，详细参考 `WBOCRCardType`
/// @param appId        人脸核身控制台内申请的 WBappid
/// @param nonce        每次请求需要的一次性nonce，一次有效
/// @param userId       每个用户唯一的标识
/// @param sign         签名信息，由接入方后台提供，一次有效
/// @param orderNo      每次OCR识别请求的唯一订单号：建议为32位字符串(不超过32位)
/// @param version      openAPI接口版本号，默认为1.0.0
/// @param license      由腾讯服务分配的，和 bundle id 关联****
/// @param succeed       SDK 登录成功
/// @param failed        SDK 登录失败
- (void)initSDKWithSDKType:(WBOCRCardType)sdkType
                     appId:(nonnull NSString *)appId
                     nonce:(nonnull NSString *)nonce
                    userId:(nonnull NSString *)userId
                      sign:(nonnull NSString *)sign
                   orderNo:(nonnull NSString *)orderNo
                   version:(nonnull NSString *)version
                   license:(nonnull NSString *)license
                   succeed:(nonnull WBOCRServiceInitSucceedBlock)succeed
                    failed:(nonnull WBOCRServiceFailedBlock)failed;
startService 接口如下，通过 recognizeSucceed 接收识别成功回调，通过 failed 接收识别失败回调。
/// 标准版本 SDK 识别接口
/// @warning 调用这个接口前，需要完成 SDK 登录，即需要调用 initSDK 接口，并收到 succeed 回调
/// @param recognizeSucceed 识别成功回调
/// @param failed           识别失败回调
- (void) startOCRService:(nonnull WBOCRServiceRecognizeSuccessBlock)recognizeSucceed
                  failed:(nonnull WBOCRServiceFailedBlock)failed;
```

3. 接入示例

具体接入示例，请参考 demo 工程。

4. 识别结果处理

SDK 入口方法提供了三个回调 block ，通过这几个 block 来捕获识别结果或者异常状况。

4.1 WBOCRServiceInitSucceedBlock（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。

4.2 WBOCRServiceRecognizeSuccessBlock（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。

4.3 WBOCRServiceFailedBlock（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，具体的错误码对照表请参考 OCR iOS 错误码 文档。

```
NSError *error = [NSError errorWithDomain:@"com.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。

OCR iOS 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

多重告警码描述

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：清晰度 clarity 字段，多重警告码 multiWarningCode 和多重警告码描述 multiWarningMsg 字段。
- 身份证人像面识别时，在 WBIDCardInfoModel 类的头文件中，增加了两个字段：清晰度 frontClarity 字段，多重警告码 frontMultiWarning 字段。
- 身份证国徽面识别时，在 WBIDCardInfoModel 类的头文件中，增加了两个字段：清晰度 backClarity 字段，多重警告码 backMultiWarning 字段。

识别成功的时候，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。

OCR Harmony SDK 接入

最近更新时间：2024-12-06 17:04:02

申请权限

SDK 需要使用到网络和相机权限，需要在 module.json5 文件中申请 ohos.permission.INTERNET 和 ohos.permission.CAMERA 权限,权限申请参考鸿蒙官方文档和示例 demo。

```
"requestPermissions": [
  {
    "name": "ohos.permission.INTERNET",
    "reason": "$string:reason_internet",
    "usedScene": {
      "abilities": [
        "FormAbility"
      ],
      "when": "always"
    }
  },
  {
    "name": "ohos.permission.CAMERA",
    "reason": "$string:reason_camera",
    "usedScene": {
      "abilities": [
        "FormAbility"
      ],
      "when": "inuse"
    }
  }
]
```

SDK 集成

在 oh-package.json5 中添加 SDK 的依赖。

```
{
  "name": "entry",
  "version": "1.0.0",
  "description": "Please describe the basic information.",
  "main": "",
  "author": "",
  "license": "",
  "dependencies": {
    "wb_kyc_ocr_library": "file:../entry/libs/wb_kyc_ocr_library.har"
  }
}
```

SDK调用流程

调用方法如下。
startOCRService 函数封装了 SDK 的调用，SDK 的调用流程可以分解为三个步骤：

- 合作方获取签名，SignTool 模拟合作方获取签名。
- 调用 WBOCRService.init 函数初始化 SDK。
- init 成功之后，调用 WBOCRService.startService 开始识别。

> ⚠ 注意：

> init 和 start 成对调用，一次 init 一次 start。

卡片类型在 init 接口的 OCRParam.sdkType 里面设置。支持的卡片类型见 CardType 定义。

```
export enum CardType {
  idCard,
  idCardFrontSide,
  idCardBackSide,
  bankCard,
  vehicleLicenseHomePage,
  vehicleLicenseSecondaryPage,
  driverLicense,
}
```

startOCRService 的源代码如下。

```
startOCRService(type: CardType) {
  let service = WBOCRService.getInstance()
  let param = new OCRParam()
  let signTool = new SignTool()
  signTool.getSign().then((data) => {
    console.log(data);
    this.prompt = `sign: ${data}`
    param.appId = signTool.appid
    param.orderNo = signTool.randomString(12)
    param.nonce = signTool.nonce
    param.userId = signTool.userid
    param.sign = data
    param.sdkType = type;

    service.init(param, {
      onSuccess: () => {
        console.log('SDK 初始化成功 - 调用 start 接口')
        service.startService({
          onSuccess: (res: object) => {
            console.log('SDK 识别成功')
            console.log(JSON.stringify(res))
            this.prompt = JSON.stringify(res)
          },
          onFailure: (error: OCRError) => {
            console.error('SDK 识别失败！')
            console.error(JSON.stringify(error))
            this.prompt = JSON.stringify(error)
          }
        })
      },
      onFailure: (error: OCRError) => {
        console.log('SDK 初始化失败！')
        console.log(JSON.stringify(error))
        this.prompt = JSON.stringify(error)
      }
    })
  }).catch((e: BusinessError) => {
    console.error(`err:${e.code},${e.message},${e.name}`)
    this.prompt = `err:${e.code},${e.message},${e.name}`
  })
}
```

接入方 App 需要接收和处理 init 以及 start 接口的回调结果。
其中，卡片识别结果定义在如下几个类里面，不同的卡片类型，对应相应类型的结果对象。

银行卡识别

银行卡识别的时候，入参卡片类型选择 bankCard，返回值类型定义如下。

```
/**
 * @brief 银行卡信息
 * @detail 字段含义
 - bankcardFullImg     银行卡识别的照片
 - bankcardNoPhoto   银行卡卡号切图
 - bankcardNo          银行卡号
 - bankcardValidDate  银行卡有效期（年 / 月，没有为空）
 - warningCode         警告码
 - warningMsg          警告码描述
 - multiWarningCode    多重警告码
 - multiWarningMsg     多重警告码描述
 - clarity             图片清晰度
 - bankcardInfo 卡片信息
 - bankcardType 卡片类型
 - bankcardName 卡片名称
 */

export class BankCardModel extends KYCOCRBaseModel {
  bankcardFullImg = ''
  bankcardNoPhoto = ''
  bankcardNo = ''
  bankcardValidDate = ''
  warningCode = ''
  warningMsg = ''
  multiWarningCode = ''
  multiWarningMsg = ''
  clarity = ''
  bankcardInfo = ''
  bankcardType = ''
  bankcardName = ''
}
```

Harmony OCR 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	SDK 未登录	未登录 SDK (调用 startOCRService 之前,需要调用 init 登录)
100103	App 没有相机权限	App 没有相机权限
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
300101	不合法请求(300102)	检查传入参数是否正确
300102	网络出小差啦	更换网络环境
400100	SDK 重复进入	重复调用 SDK

OCR 验证结果

最近更新时间：2024-06-17 17:09:41

服务端验证结果

此方式用于：
合作伙伴服务端生成签名，并调用银行卡识别服务端查询结果，鉴权完成后返回结果（服务端上传 order_no 和 wbappid 查询）。

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，获取方式请参见 SIGN ticket 获取 。
- **合作方为银行卡 OCR 识别服务生成签名，需要具有以下参数：**

参数	说明	来源
appId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
order_no	订单号，字母/数字组成的字符串，本次银行卡识别合作伙伴上传的订单号，唯一标识	合作方自行分配，不要带有特殊字符
version	默认值：1.0.0	-
ticket	合作伙伴服务端缓存的 ticket，注意是 SIGN 类型	获取方式请参见 SIGN ticket 获取
nonceStr	32位随机字符串，字母和数字	合作方自行生成，不要带有特殊字符

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、order_no、version、ticket、nonceStr 共5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**
> 签名算法请参见 签名算法说明 。

银行卡 OCR 识别结果查询接口

请求

- **请求 URL：** `https://kyc1.qcloud.com/api/server/getBankCardOcrResult`
- **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	请您 点此链接 扫描二维码添加腾讯云人脸核身小助手，进行线下对接获取	String	腾讯服务分配	是
order_no	订单号，字母/数字组成的字符串，合作方订单的唯一标识	String	32	是
get_file	是否需要获取银行卡 OCR 图片文件 值为1：返回文件 其他：不返回	String	1	否
nonce	随机数	String	32	是
version	版本号，默认值：1.0.0	String	20	是
sign	签名值，使用本页第一步生成的签名	String	40	是

- **请求示例：**

```
https://kyc1.qcloud.com/api/server/getBankCardOcrResult?
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
code	String	0：银行卡识别成功
msg	String	返回结果描述
orderNo	String	订单编号
bankCardNo	String	resultCode 为0返回：银行卡号
bankCardValidDate	String	resultCode 为0返回：银行卡有效期
bankcardCropPhoto	Base64 String	银行卡切边图片
bankcardNoPhoto	Base64 String	银行卡卡号切边图片
originBankcardPhoto	Base64 String	识别原始图片
warnCode	String	银行卡告警码，在银行卡日期失效或者过期会提示 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的 银行卡 OCR 错误码
operateTime	String	做 OCR 的操作时间
multiWarnCode	String	多重告警码，含义请参考 银行卡 OCR 错误码
clarity	String	图片清晰度

响应示例：


```
{
   "code":"0",
   "msg":"请求成功",
   "bizSeqNo":"21062120001184412617492807497465",
   "result":{
        "bizSeqNo":"21062120001184412617492807497465",
        "transactionTime":"20210621174928",
        "orderNo":"bankcardPic5923ab9a3bc34488b51",
        "bankCardNo":"xxxxxxxxxxxxxxxxxx",
        "bankCardValidDate":"12/2024",
        "warnCode":"0",
        "multiWarnCode":"0",
        "operateTime":"2021-06-21 17:40:06",
        "clarity":"79",
        "success":false
},
"transactionTime":"20210621174928"
}
```


注意

- 银行卡照片信息作为存证，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；如果不上传参数或者参数为空，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。
- 对于银行卡 OCR 识别有日期失效或者过期的情况，详情请参见通用响应码列表的 银行卡 OCR 错误码 。

- success：false 无意义，合作伙伴可忽略，无需处理。

银行卡 OCR 错误码

最近更新时间：2023-10-27 14:41:13

返回码	返回信息	处理措施	是否收费

66660000/-1304	参数过长	订单号长度为不超过32位，确认订单号是否正确	否
-9011	银行卡识别失败	未能识别，需重新识别	否
-9012	银行卡图片模糊	未能识别，需确保图片对焦成功	否
-9013	不是银行卡	未能识别，未找到银行卡	否
-9014	银行卡信息不合法	未能识别，需避免图片反光或遮挡	否

告警码	返回信息
-9110	无效的日期信息
-9111	边框不完整
-9112	银行卡反光
-9113	银行卡复印件
-9114	银行卡翻拍件
66661009	银行卡已过期

注意

- 对于告警码，默认处理为通过，客户可根据情况自行对此告警码做拒绝识别处理。
- 银行卡 OCR 多重告警码：同一张银行卡图片如果识别出有多重告警，则返回的告警码信息以 [-9110,-9111,-9114] 方式显示，表示该银行卡图片同时存在 “无效的日期信息、边框不完整、银行卡翻拍件”这3个告警。

登录鉴权
获取 Access Token

最近更新时间：2025-01-07 10:21:42

注意事项

- 所有场景默认采用 UTF-8 编码。
- **Access Token 必须缓存在磁盘，并定时刷新，且不能并发刷新，建议每20分钟请求新的 Access Token，获取之后立即使用最新的 Access Token。旧的只有一分钟的并存期。**
- **如果未按照上述做定时刷新，可能导致鉴权不通过，影响人脸服务正常调用。**
- **每次用户登录时必须重新获取 NONCE ticket。**

请求

- **请求 URL：** `https://kyc1.qcloud.com/api/oauth2/access_token`
- **请求方法：** `GET`
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
secret	WBappid 对应的密钥，申请 WBappid 时得到，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	64	是
grant_type	授权类型，默认值为：client_credential（必须小写）	String	20	是
version	版本号，默认值为：1.0.0	String	20	是

- **请求示例：**

```
https://kyc1.qcloud.com/api/oauth2/access_token?
appId=xxx&secret=xxx&grant_type=client_credential&version=1.0.0
```

响应

响应参数：

参数	类型	说明
code	String	0：成功 非0：失败 详情请参见 基础版人脸核身服务错误码
msg	String	请求结果描述
transactionTime	String	调用接口的时间
access_token	String	access_token 的值
expire_time	String	access_token 失效的绝对时间
expire_in	int	access_token 的最大生存时间

响应示例：

```
{
  "code":"0","msg":"请求成功",
  "transactionTime":"20151022043831",
  "access_token":"accessToken_string",
```

```
  "expire_time":"20151022043831",
  "expire_in":"7200"
}
```


注意

- code 不为0则表示获取失败，可以根据 code 和 msg 字段进行定位和调试。code 详情请参见 基础版人脸核身服务错误码 。
- expire_in 为 access_token 的最大生存时间，单位：秒，合作伙伴在**判定有效期时以此为准**。
- expire_time 为 access_token 失效的绝对时间，由于各服务器时间差异，不能以此作为有效期的判定依据，只作为展示使用。
- 修改 secret 之后，该appId生成的 access_token 和 ticket 都失效。

获取 SIGN ticket

最近更新时间：2025-01-07 10:21:42

注意事项

- **前置条件：**请合作方确保 Access Token 已经正常获取，获取方式请参见 获取 Access Token 。
- **用途：**SIGN ticket 是合作方**后台服务端业务请求**生成签名鉴权参数之一，用于后台查询验证结果、调用其他业务服务等。
- API ticket 的 SIGN 类型，必须缓存在磁盘，并定时刷新，刷新的机制如下：
- **因为 API ticket 依赖于 Access Token 为了简单方便，建议将 API ticket 与 Access Token 绑定，每20分钟定时刷新，且不能并发刷新。**
- **获取新的之后请立即使用最新的，旧的有一分钟的并存期。**
- **如果未按照上述做定时刷新，可能导致鉴权不通过，影响人脸服务正常调用。**

请求

- **请求 URL：** `https://kyc1.qcloud.com/api/oauth2/api_ticket`
- **请求方法：**GET
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 wbappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
access_token	请根据 获取 Access Token 指引进行获取	String	90	是
type	ticket 类型，默认值：SIGN（必须大写）	String	20	是
version	版本号，默认值：1.0.0	String	20	是

- **请求示例：**

```
https://kyc1.qcloud.com/api/oauth2/api_ticket?appId=xxx&access_token=xxx&type=SIGN&version=1.0.0
```

响应

响应参数：

参数	类型	说明
code	String	0：成功 非0：失败 详情请参见 基础版人脸核身服务错误码
msg	String	请求结果描述
transactionTime	String	调用接口的时间
tickets	list	ticket 返回数组
value	String	ticket 的值
expire_time	String	ticket 失效的绝对时间
expire_in	int	ticket 的最大生存时间

响应示例：

```
{
    "code": "0",
    "msg": "请求成功",
    "transactionTime": "20151022044027",
    "tickets": [
      {
```

```
            "value": "ticket_string",
            "expire_in": "3600",
            "expire_time": "20151022044027"
        }
    ]
}
```

注意

- code 不为0则表示获取失败，可以根据 code 和 msg 字段进行定位和调试。code 详情请参见 基础版人脸核身服务错误码 。
- expire_in 为 SIGN ticket 的最大生存时间，单位：秒，合作伙伴在**判定有效期时以此为准**。
- expire_time 为 SIGN ticket 失效的绝对时间，由于各服务器时间差异，不能以此作为有效期的判定依据，只作为展示使用。
- access_token 失效时，该 access_token 生成的 ticket 都失效。
- tickets 只有一个。

获取 NONCE ticket

最近更新时间：2025-01-07 10:21:42

注意事项

- **前置条件：**请合作方确保 Access Token 已经正常获取，获取方式请参见 Access Token 获取 。
- **用途：**NONCE ticket 是合作方**前端包含 App 和 H5 等**生成签名鉴权参数之一，启动 H5 或 SDK 人脸核身。
- API ticket 的 NONCE 类型，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket。

请求

- **请求 URL：** `https://kyc1.qcloud.com/api/oauth2/api_ticket`
- **请求方法：**GET
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
access_token	请根据 Access Token 获取 指引进行获取	String	90	是
type	ticket 类型，默认值：NONCE（必须大写）	String	20	是
version	版本号	String	20	是
user_id	当前使用用户的唯一标识，需合作伙伴自行定义注意：合作伙伴必须保证 user_id 的全局唯一性，不要带有特殊字符	String	32	是

- **请求示例：**

```
https://kyc1.qcloud.com/api/oauth2/api_ticket?
appId=xxx&access_token=xxx&type=NONCE&version=1.0.0&user_id=xxx
```

响应

响应参数：

参数	类型	说明
code	String	0：成功 非0：失败 详情请参见 基础版人脸核身服务错误码
msg	String	请求结果描述
transactionTime	String	调用接口的时间
tickets	list	ticket 返回数组
value	String	ticket 的值
expire_time	String	ticket 失效的绝对时间
expire_in	int	ticket 的最大生存时间

响应示例：

```
{
    "code": "0",
    "msg": "请求成功",
    "transactionTime": "20151022044027",
    "tickets": [
```

```
        {
            "value": "ticket_string",
            "expire_in": "120",
            "expire_time": "20151022044027"
        }
    ]
}
```

注意

- code 不为0则表示获取失败，可以根据 code 和 msg 字段进行定位和调试。code 详情请参见 基础版人脸核身服务错误码 。
- expire_in 为 access_token 的最大生存时间，单位：秒，合作伙伴在**判定有效期时以此为准**。
- expire_time 为 access_token 失效的绝对时间，由于各服务器时间差异，不能以此作为有效期的判定依据，只作为展示使用。
- access_token 失效时，该 access_token 生成的 ticket 都失效。

获取 WBappid

最近更新时间：2024-01-17 10:18:51

操作场景

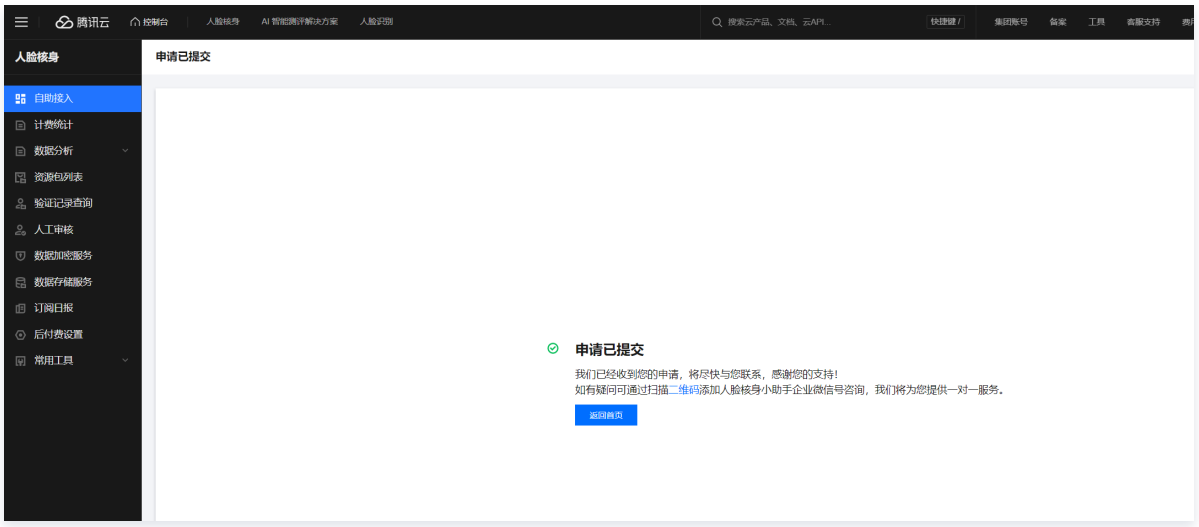
通过使用 App SDK、移动 H5、PC 端 H5 的方式接入腾讯云人脸核身时，需要获取业务流程唯一标识 WBappid，本文档将介绍如何登录控制台获取 WBappid。

操作步骤

1. 登录腾讯云慧眼 人脸核身控制台，单击左侧菜单栏**自助接入**，单击**SDK/移动 H5 服务 > 申请 WBappid**。

2. 填写机构名称，业务名称，并勾选应用类型填写相关信息。

3. 确认信息填写正确后，单击**提交申请**。



4. 审核通过，即可获取 WBappid。

签名算法说明

最近更新时间：2022-07-05 17:16:32

Java 签名算法

以下是生成签名算法，合作伙伴可以直接使用。

```
public static String sign(List<String> values, String ticket) { //values传ticket外的其他参数
    if (values == null) {
        throw new NullPointerException("values is null");
    }
    values.removeAll(Collections.singleton(null));// remove null
    values.add(ticket);
    java.util.Collections.sort(values);

    StringBuilder sb = new StringBuilder();
    for (String s : values) {
        sb.append(s);
    }
    return Hashing.sha1().hashString(sb, Charsets.UTF_8).toString().toUpperCase();
}
```

> **注意**
> Hashing 使用的是 com.google.common.hash.Hashing，版本是 guava-18.0。

调用方法

获取 token 和 ticket 后再用 ticket 对数据进行签名，相关代码如下所示：

```
String accessToken = client.getAccessToken();//http 请求
String ticket = client.getTicket(accessToken);//http 请求
String sign = SignUtils.sign(data, ticket);
```

H5（微信浏览器）
微信普通 H5 配置流程

最近更新时间：2024-11-11 11:28:12

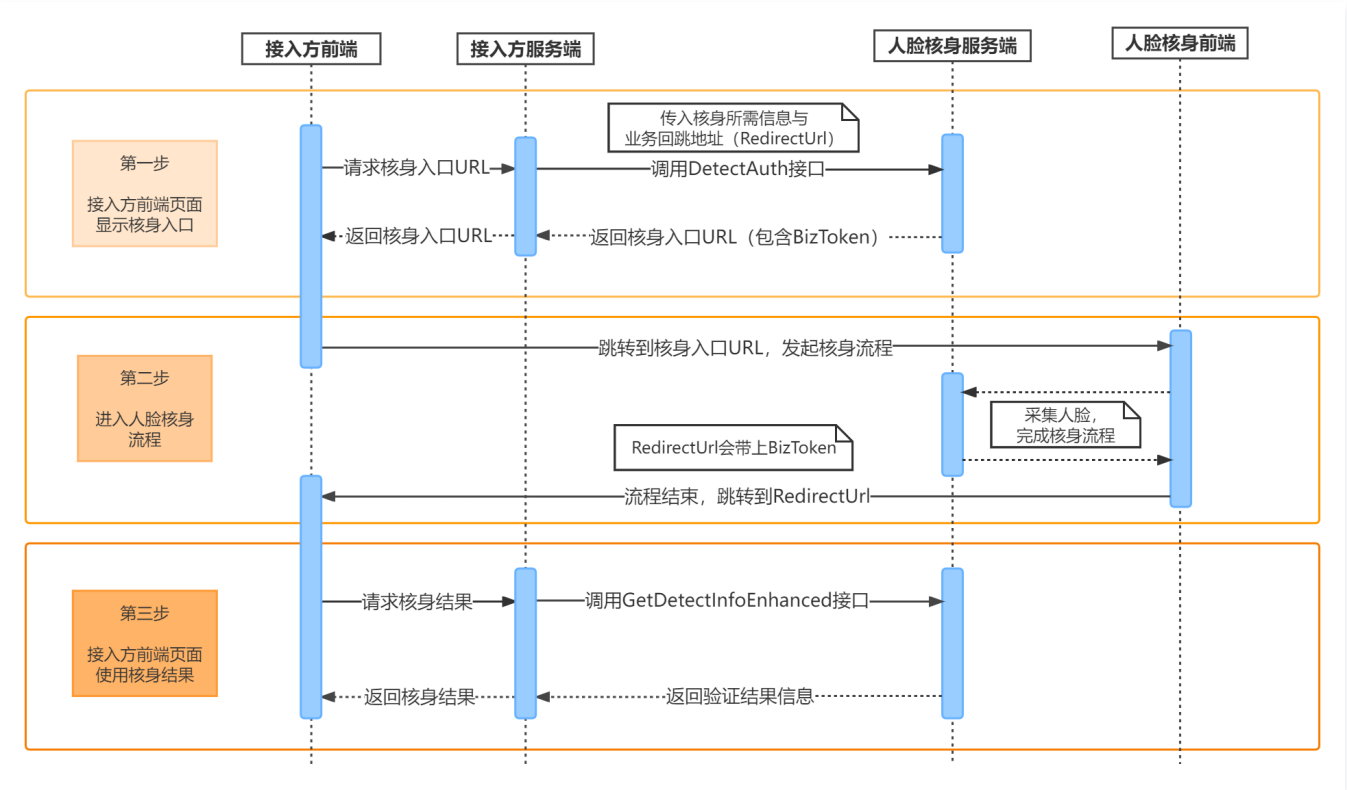
本文档以接入微信普通 H5 的操作流程为例，描述通过微信 HTML5 方式接入人脸核身的完整操作流程。

前提条件

1. 已注册腾讯云账号，并完成实名认证。
2. 已申请通过腾讯云人脸核身。

如果还未完成以上操作，可参考 流程指引 完成操作。

接入时序图

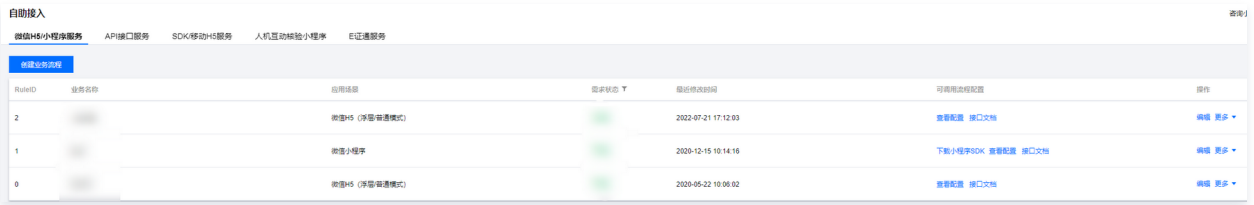


操作步骤

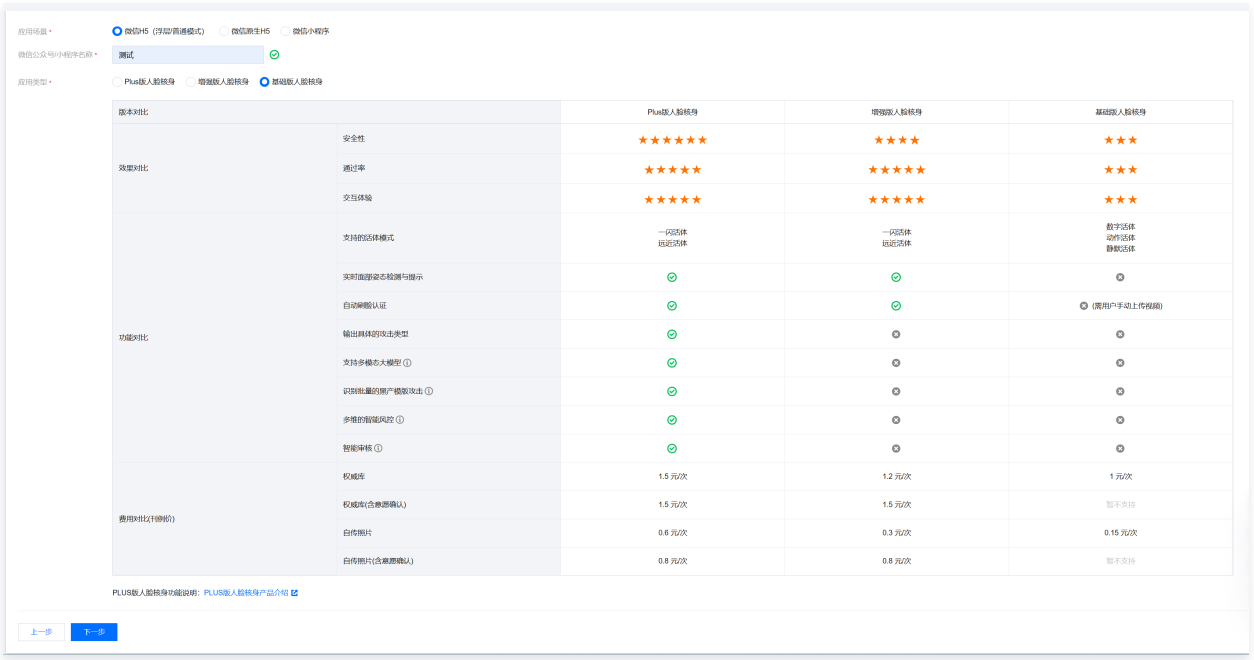
步骤1：创建 RuleId

RuleId 用于调用配置的业务流程，创建步骤如下：

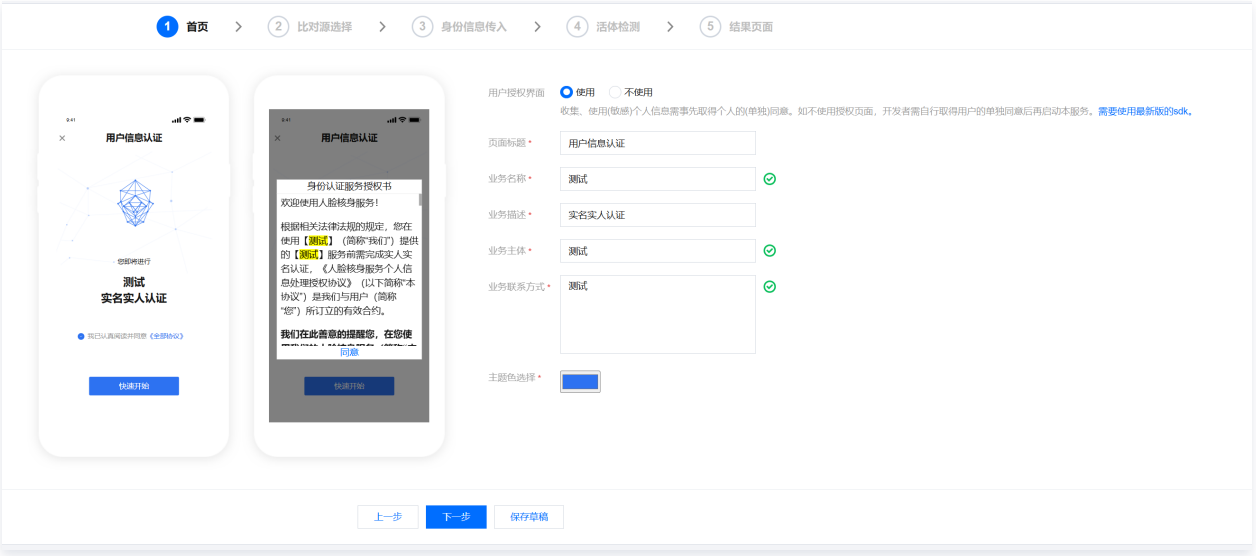
1. 登录腾讯云人脸核身控制台，在 自助接入 页面，单击**创建业务流程**。



2. 选择应用场景选择：**微信 H5（浮层/普通模式）**，应用类型选择：**基础版人脸核身**，然后单击**下一步**。

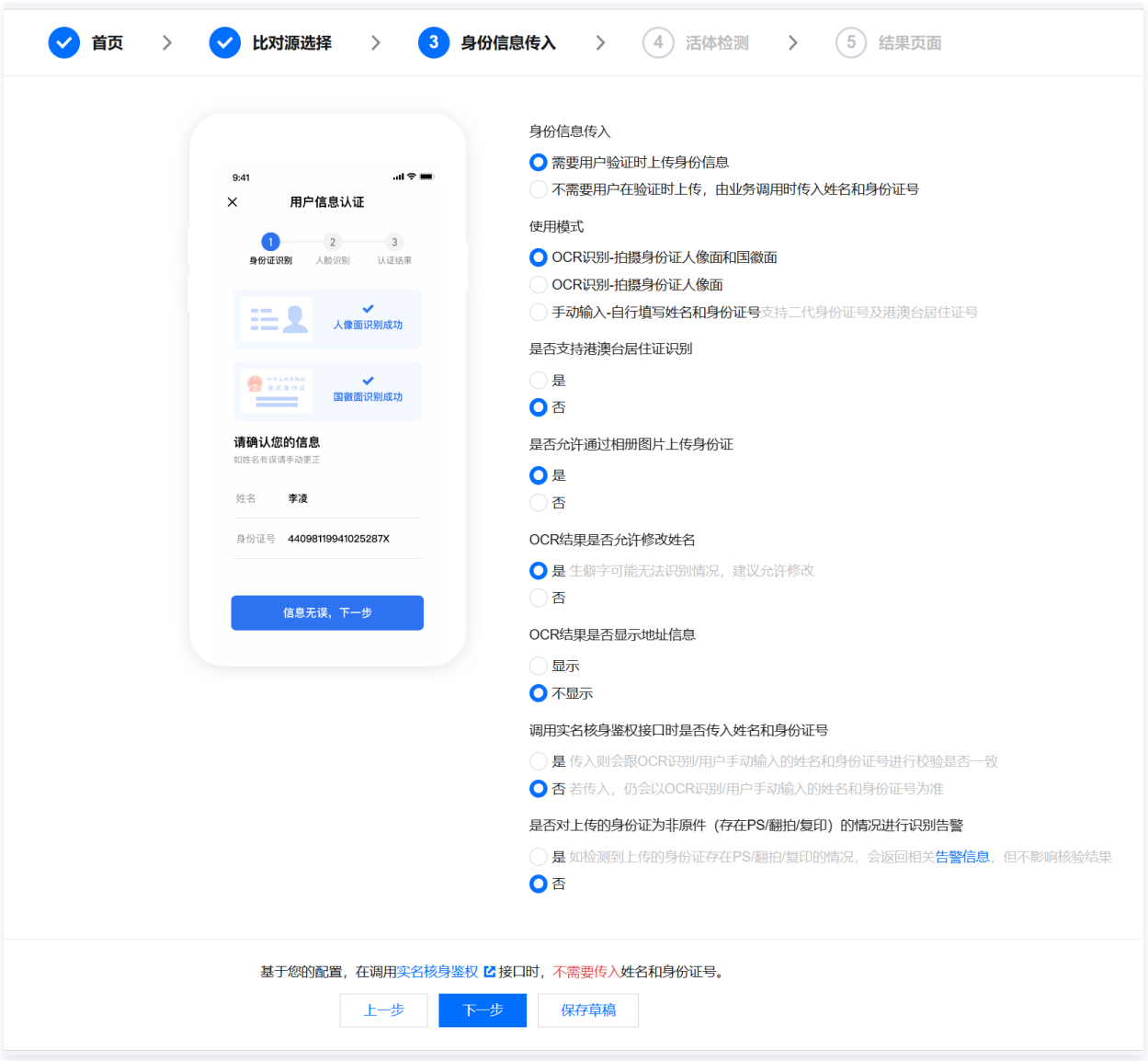


3. 进入接入配置，根据您业务的实际场景填写页面标题、业务名称、业务描述信息后单击**下一步**。

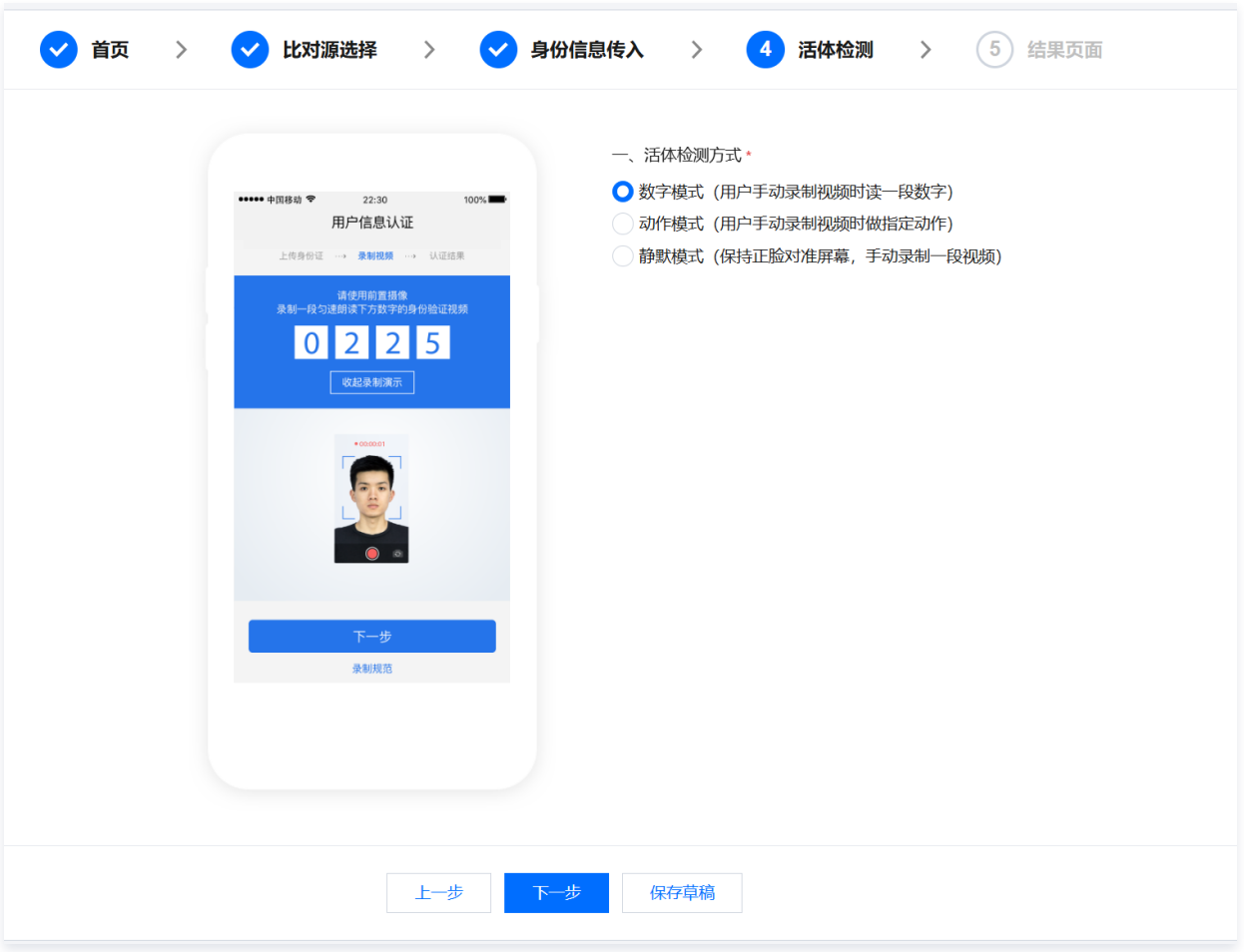


4. 选择人脸比对库源。人脸核身支持两种方式：**跟权威库比对**和**跟上传照片比对**。
   - 其中**跟权威库比对**收费价格为：基础版人脸核身（权威库）的价格.
   - **跟上传照片比对**收费的价格为：基础版人脸核身（自传照片）的价格。OCR 不再单独收费。

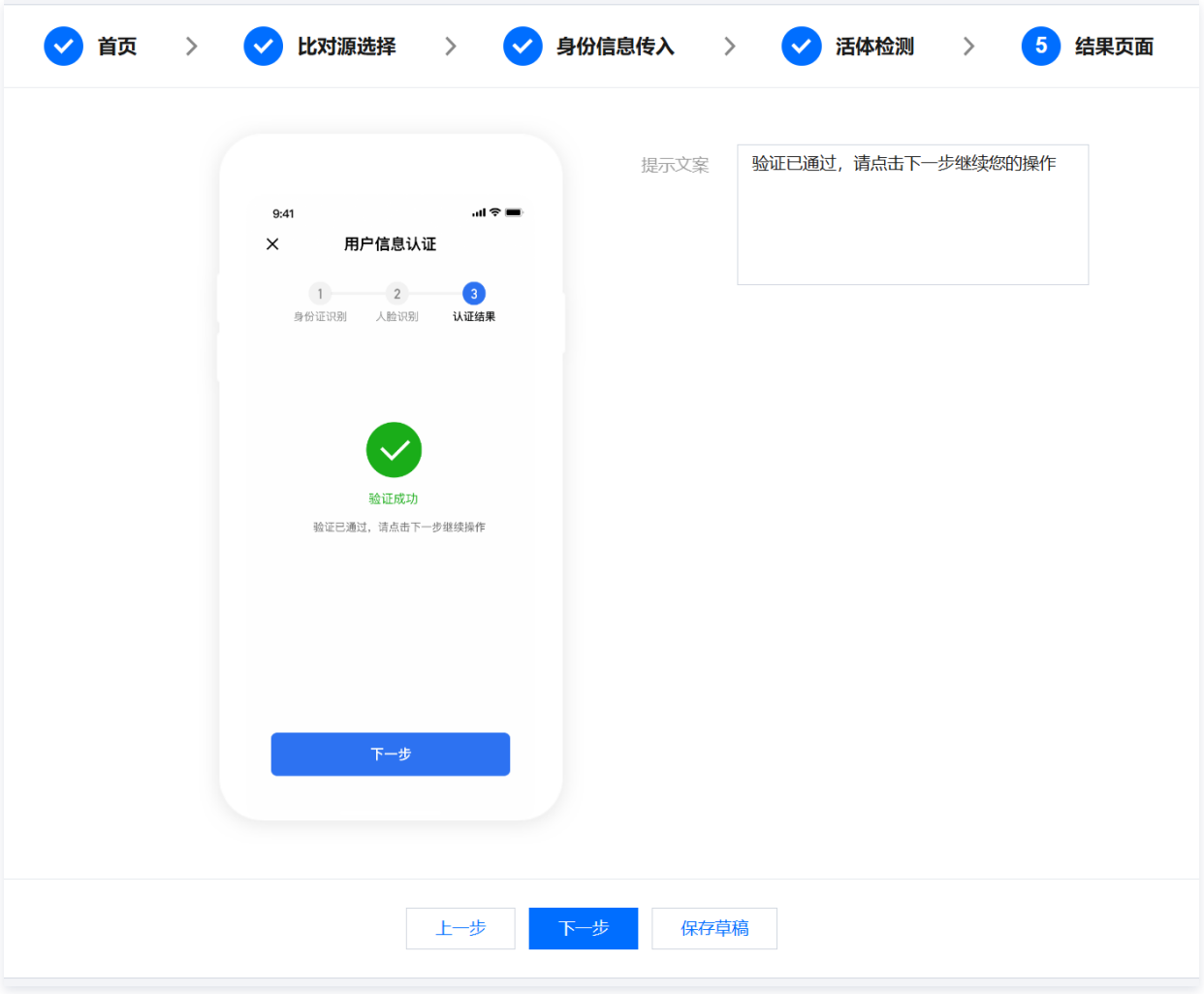
5. 配置身份证 OCR 功能，如果不需要则勾选“**不需要用户在验证时上传**”，然后单击**下一步**。



6. 配置活体检测方式，勾选所需使用的模式，然后单击**下一步**。



7. 配置结果页的文案描述，然后单击**下一步**。



8. 业务信息填写完成后，确认您的配置信息然后单击**确认并提交审核**。

步骤2：获取 BizToken

- 完成 RuleId 创建后，需获取 BizToken，用于调用您配置的人脸核身验证的流程。
- 调用 实名核身鉴权，传入 步骤1 生成的 RuleId，和认证结束后重定向的回调链接地址 RedirectUrl，得到核验流程唯一密钥（BizToken）和用于发起核身流程的 URL。
- 在线调试。

> ⚠ 注意
> - BizToken 是仅一次核身流程的标识，有效时间为7,200秒；用户完成核身后，开发者可用该标识获取验证结果信息，结果信息仅保留3天。
> - 如果输入参数 RedirectUrl 为空，则用户完成核验后默认跳转腾讯云人脸核身产品介绍页。

步骤3： 跳转人脸核身 URL 完成核验

您在前端通过地址跳转方式重定向至 步骤2 中获取的核身入口 URL，用户进入核身流程，完成身份证拍摄识别、录制视频等操作完身份核验。

步骤4： 查询核验结果信息

用户完成人脸核身后，页面会跳转到 RedirectUrl 上，地址中会带上此次验证流程使用的 BizToken，您在服务端即可凭借 BizToken 参数调用 获取实名核身结果信息 接口去获取本次核身的详细信息。获取到用户验证过程数据，包括文本信息、识别分数、照片和视频。也可以通过访问 人脸核身控制台 查看服务调用情况。

> ⚠ 注意
> - 为了保证用户的隐私，结果信息人脸核身侧仅保留3天，请您及时拉取。
> - 在开发、产品使用、费用、合同等问题有任何疑问，欢迎您 联系我们 进行询问。

H5（PC浏览器）
合作方后台上传身份信息

最近更新时间：2025-01-07 10:21:43

> **注意**
> 如果因自身业务需要对人脸核身的影像文件进行存储或其他用途，请合作方务必自行保存订单号，通过订单号拉取人脸核身的影像文件是唯一方式。

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 。
- **合作方为人脸验证服务生成签名，需要具有下表中的参数：**
- **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	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，字母/数字组成的字符串，本次人脸验证合作伙伴上传的订单号，唯一标识	合作方自行分配
name	姓名	使用自带源比对时：姓名+证件号 可不输入
idNo	证件号码	使用自带源比对时：姓名+证件号 可不输入
userId	用户唯一标识，同一个用户的 userId 请保持一致，不同用户请不要使用同一个 userId，我们会根据 userId 来做登录态校验	合作方自行分配
version	1.0.0	-
ticket	合作伙伴服务端缓存的 ticket，注意是 SIGN 类型	获取方式见 SIGN ticket 获取

基本步骤

1. 将 appId、orderNo、name、idNo、userId、version、ticket（SIGN 类型）共7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**
> Java 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	appId001
orderNo	19959248596551
name	testName
idNo	4300000000000
userId	19959248596551
version	1.0.0

ticket	duSz9ptwyW1Xn7r6gYItxz3feMdJ8Na5x7JZuoxurE7RcI5TdwCE4KT2eEeNNDoe

字典排序后的参数为：

```
[1.0.0, 19959248596551, 19959248596551, 4300000000000, appId001,
duSz9ptwyW1Xn7r6gYItxz3feMdJ8Na5x7JZuoxurE7RcI5TdwCE4KT2eEeNNDoe, testName]
```

拼接后的字符串为：

```
1.0.0199592485965511995924859655143000000000000appId001duSz9ptwyW1Xn7r6gYItxz3feMdJ8Na5x7JZuoxurE7RcI5TdwCE4KT2e
EeNNDoetestName
```

计算 SHA1 得到签名：
该字符串就是最终生成的签名（40位），不区分大小写。

```
11670F8BD6181CF71FB4656534253A0627D835CF
```

合作方后台上传身份信息

请求

请求 URL：https://kyc1.qcloud.com/api/server/h5/geth5faceid?orderNo=xxx

> **注意**
> **为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。**

请求方法：POST
报文格式： `Content-Type: application/json` **请求参数：**

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
orderNo	订单号，字母/数字组成的字符串，由合作方上传，每次唯一，不能超过32位	String	不能超过32位	是
name	姓名	String	-	使用权威源比对时：姓名+证件号 必须输入 使用自带源比对时：姓名+证件号 可不输入
idNo	证件号码	String	-	使用权威源比对时：姓名+证件号 必须输入 使用自带源比对时：姓名+证件号 可不输入
userId	用户 ID，用户的唯一标识（不能带有特殊字符），需要跟生成签名的 userId 保持一致。同一个用户的 userId 请保持一致，不同用户请不要使用同一个userId，我们会根据 userId 来做登录态校验	String	不能超过32位	是
sourcePhotoStr	比对源照片 1. 原始图片不能超过500k，且必须为 JPG 或 PNG、BMP 格式。 2. 参数有值：使用合作伙伴提供的比对源照片进行比对，照片是正脸可信照片，照片质量由合作方保证。 3. 参数为空：根据身份证号+姓名使用权威数据源比对	Base64 String	1048576	否 请使用标准的图片转base64方法，base64编码不可包含换行符，不需要加前缀 **注意：只有您需要使用自带比对源比对时上传，使用权威比对源比对时请不要上传该字段**
sourcePhotoType	比对源照片类型 参数值为1：水纹正脸照 参数值为2：高清正脸照	String	1	否，提供比对源照片需要传 **注意：只有您需要使用自带比对源比对时上传，使用权威比对源比对时请不要**

	重要提示：照片上无水波纹的为高清照，请勿传错，否则影响比对准确率。如有疑问，请联系腾讯云技术支持线下确认			上传该字段
version	默认参数值为：1.0.0	String	20	是
sign	签名：使用上面生成的签名	String	40	是

水纹照示例

响应

响应参数：

参数	类型	说明
code	String	0：成功 非0：失败 详情请参见基础版人脸核身服务错误码
msg	String	请求结果描述
bizSeqNo	String	请求业务流水号
orderNo	String	订单编号
h5faceId	String	此次刷脸用户标识
transactionTime	String	接口请求的时间
optimalDomain	String	启动 H5 人脸核身步骤中调用 login 接口使用的域名

响应示例：


```
{
    "code": "0",
    "msg": "请求成功",
    "bizSeqNo": "2106212000118443841832290801029 7",
    "result": {
        "bizSeqNo": "2106212000118443841832290801029 7",
        "transactionTime": "20210621183229",
        "orderNo": "161709188560917432576916585 0",
        "h5faceId": "wb0375fa5243984381ea7b7013f13795",
        "optimalDomain": "kyc1.qcloud.com",
        "success": false
  },
  "transactionTime": "20210621183229"
}
```

说明
success：false 无意义，合作伙伴可忽略，无需处理。
h5faceId 有效期为5分钟，每次进行人脸核身都需要重新获取。

启用 H5 人脸认证

最近更新时间：2024-06-17 17:09:41

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 。
- **NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次登录 h5 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁登录 h5 刷脸。**
- 人脸识别需要录制视频，因此要求浏览器支持视频录制功能。
  适用浏览器及版本如下：
- Firefox 浏览器29及以上版本。
- Chrome 浏览器53及以上版本。

生成签名

准备步骤

- 合作方根据本次人脸验证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	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，字母/数字组成的字符串，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，不要带有特殊字符
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（与接口中使用的 userId 一致，不要带有特殊字符）
version	参数值为：1.0.0	-
h5faceId	h5/geth5faceid 接口返回的唯一标识	-
ticket	合作伙伴服务端实时获取的 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（与接口中的随机数保持一致，不要带有特殊字符）

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 appId、userId、orderNo、version、h5faceId 连同 ticket、nonce 共7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	appId001
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
h5faceId	bwiwe1457895464
orderNo	aabc1457895464
ticket	zxc9QfxIti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPlucaMS

字典排序后的参数为：

```
[1.0.0, aabc1457895464, appId001, bwiwe1457895464, kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551,
zxc9Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS]
```

拼接后的字符串为：

```
1.0.0aabc1457895464appId001bwiwe1457895464kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7TuserID19959248596551zxc9Qfxlti9iTVgHA
jwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS
```

计算 SHA1 得到签名：
该字符串就是最终生成的签名（40位），不区分大小写。

```
4E9DFABF938BF37BDB7A7DC25CCA1233D12D986B
```

启动 H5 人脸核身

合作方后台上送 h5faceId 以及 sign，后台校验 sign 通过之后重定向到 H5 人脸核身。
为了保证服务的高可用，全面消除单点风险，我们启用了多域名服务。启动 H5 人脸核身需要使用 合作方后台上送身份信息 接口返回内容中 optimalDomain 字段域名。
请求 URL：https://{optimalDomain}/api/pc/login

> **注意**
> - optimalDomain 使用 合作方后台上送身份信息 接口返回的 optimalDomain 域名，如果 optimalDomain 字段返回为空或者取不到，默认使用域名 kyc1.qcloud.com。

- **该跳转 url 不能直接暴露在前端 html 页面的<a>标签中。某些浏览器会预加载<a>标签中的 url，导致用户点击访问该 url 时，因 url 已经被预加载访问过，于是签名失效，页面报错“签名不合法”。**

请求方法：GET
请求参数：

参数	说明	类型	长度	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	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，字母/数字组成的字符串，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，不能超过32位	String	32	是
h5faceId	h5/geth5faceid 接口返回的唯一标识	String	32	是
url	H5 人脸验证完成后回调的第三方 URL，需要第三方提供完整 URL 且做 URL Encode。完整 URL Encode 示例，原 URL：https://cloud.tencent.com，Encode 后：https%3a%2f%2fcloud.tencent.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

请求示例：

```
https://kyc1.qcloud.com/api/pc/login?appId=appId001
&version=1.0.0
&nonce=4bu6a5nv9t678m2t9je5819q46y9hf93
&orderNo=16170918856091743257691658 5
&h5faceId=wb04f10695c3651ce155fea7070b74c9
&url=https%3a%2f%2fcloud.tencent.com
&userId=23333333333333
```

```
&sign=5DD4184F4FB26B7B9F6DC3D7D2AB3319E5F7415F
```

PC 端 H5 人脸核身结果返回及跳转

最近更新时间：2025-02-21 14:39:32

认证结束后，认证结果页会回调启动 H5 人脸核身入参中指定的回调 url 地址。并带有 code、orderNo、h5faceId、newSign 参数。

如果您只需要获取核身结果，您可以根据我们刷脸完成后的回调 url 请求中的参数中的 code 判断是否核身通过，code 0 表示人脸核身成功，其他错误码表示失败。
如果您需要拉取人脸核身的视频和图片用于存证等其他需要，请参看查询核身结果。（您可以在收到完成刷脸的回调后，再来我们的服务器获取到刷脸视频和照片）
请求 URL： `https://xxx.com/xxx?code=xxxx&orderNo=xxxx&h5faceId=xxxx&newSign=xxxx`

> **注意**
> xxx.com 为启动 H5 人脸核身入参中指定的回调 url 地址。

请求方法：GET
参数：

参数	说明	类型	长度（字节）
code	人脸核身结果的返回码，0 表示人脸核身成功，其他错误码表示失败。	String	-
orderNo	订单号，由合作方上传，每次唯一，此信息为本次人脸核身上传的信息。	String	32
h5faceId	本次请求返回的唯一标识，此信息为本次人脸核身上传的信息。	String	32
newSign	对 URL 参数 App ID、orderNo 和 SIGN ticket、code 的大写签名。	String	40

人脸核身结果查询
查询核身结果

最近更新时间：2024-06-17 17:09:41

当您的用户完成核身认证后，如果您需要拉取人脸核身的视频和图片用于存证等其他需要，您可以调用查询核身结果接口来获取。
重要提示：
1. 您需要在前端完成刷脸的回调后，再来调用查询核身结果接口获取刷脸视频和照片。
2. 人脸核身完成后的相关业务数据请尽快拉取，超过人脸核身服务必须最短缓存时间的业务数据将完全清理。

> **注意**
> 当您的 App 接入的是我们的基础版或增强版 SDK 服务时，SDK 回调中只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过查询核身结果接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，查询核身结果接口无法查询到刷脸结果。

步骤如下：

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，获取方式请参见 获取 SIGN ticket 。
- 合作方为人脸核身服务生成签名，需要具有以下参数：

参数	说明	来源
appId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，字母/数字组成的字符串，本次人脸核身合作伙伴上送的订单号，唯一标识	合作方自行分配，不要带有特殊字符
version	默认值：1.0.0	-
ticket	合作伙伴服务端缓存的 ticket，注意是 SIGN 类型	获取方式请参见 获取 SIGN ticket
nonce	32位随机字符串，由字母和数字组成	合作方自行生成，不要带有特殊字符

基本步骤

1. 生成一个32位的随机字符串 nonce（由字母和数字组成，登录时也要用到）。
2. 将 appId、orderNo、version、ticket、nonce 共5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 。

人脸核身结果查询接口(升级)

请求

- **请求 URL：** `https://kyc1.qcloud.com/api/v2/base/queryfacerecord?orderNo=xxx`

> **注意**
> 为方便查询耗时，该请求url后面请拼接 orderNo 订单号参数。

- **请求方法：**POST
- **报文格式：**Content-Type: application/json
- **请求参数：**

参数	说明	类型	长度（字节）	是否必填

appId	腾讯云控制台申请的 appid	String	8	是
version	版本号，默认值：1.0.0	String	20	是
nonce	随机数	String	32	是
orderNo	订单号，字母/数字组成的字符串,是您需要查询结果的人脸核身订单号	String	32	是
sign	签名值，使用本页第一步生成的签名	String	40	是
getFile	是否需要获取人脸识别的视频和文件，值为1则返回视频和照片、值为2则返回照片、值为3则返回视频；其他则不返回	String	1	否
queryVersion	查询接口版本号(传1.0则返回 sdk 版本号和 trtc 标识)	String	8	否

响应

- **响应参数：**

参数	类型	说明
code	String	0：表示身份验证成功且认证为同一人
msg	String	返回结果描述
bizSeqNo	String	业务流水号
transactionTime	String	请求接口的时间
orderNo	String	订单编号
liveRate	String	活体检测得分
similarity	String	人脸比对得分
occurredTime	String	进行刷脸的时间
photo	Base 64 string	人脸核身时的照片，base64 位编码
video	Base 64 string	人脸核身时的视频，base64 位编码
sdkVersion	String	人脸核身时的 sdk 版本号
trtcFlag	String	Trtc 渠道刷脸则标识"Y"
appId	String	腾讯云控制台申请的 appid

- **响应示例：**


```
{"code":"0",
 "msg":"请求成功",
 "bizSeqNo":"2203292000118445321117401579089 4",
 "result":{
       "orderNo":"161709188560917432576916 5852",
       "liveRate":"99",
       "similarity":"88.01",
       "occurredTime":"20220329104717",
       "appId":"IDAXXXXX",
       "photo":"******",
       "video":"******",
       "bizSeqNo":"2203292000118445321117401579089 4",
       "sdkVersion":"1.12.12",
```

```
        "trtcFlag":"Y"},
  "transactionTime":"20220329111740"
  }
```

code 非 0 时，有时不返回图片和视频。

注意事项

- 照片和视频信息作为存证，合作伙伴可以通过此接口拉取视频等文件，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，默认不返回视频和照片信息。为确保用户操作整体流程顺利完成，部分情况下获取视频和照片会有1秒左右的延迟。
- 由于照片和视频信息有可能超过 1M，考虑传输的影响，建议合作伙伴在使用时注意，建议获取比对结果用于后续流程处理和存证使用分开调用。避免网络传输带来的影响。目前我们的查询接口是异步查询，返回的文件可能会有1/2 s的延迟。
- 如果合作方是完成人脸核身流程后马上去拉取视频/照片，建议在查询接口加上一个查询机制：判断图片是否存在，轮询3次，每次2s。
- 照片和视频均为 base64 位编码，其中照片解码后格式一般为 JPG 和 PNG。视频格式解码后一般为 MP4。
- 服务端验证结果接口返回66660011无此查询结果的可能原因：

  1.1 66660017 验证次数超限后被风控，风控后的订单为无效，查询结果为无此查询结果。

  1.2 用户中途退出刷脸，没有完成人脸验证的流程，没有比对，查询结果为无此查询结果。

  1.3 66660018 操作超时，请退出重试 无此 ID 的用户身份信息，H5faceid 5分钟有效期，失效后无法进入刷脸流程，查询结果为无此查询结果。

  1.4 66660016 视频格式或大小不合法 文件或视频不合法，无法进行比对，查询结果为无此查询结果。

  1.5 400604 上传的视频非实时录制,被时间戳校验拦截，查询结果为无此查询结果。

  1.6 查询超过3天的订单返回无此查询结果。
- 响应参数请勿做强校验，后续可能会有扩展。

核身结果-多张照片返回

最近更新时间：2024-06-17 17:09:41

人脸认证多张照片查询接口：获取人脸认证结果多张照片的接口。

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 。
- **合作方为人脸验证服务生成签名，需要具有以下参数：**

参数	说明	来源
appId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，字母/数字组成的字符串，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，不要带有特殊字符
version	默认值：1.0.0	-
ticket	合作伙伴服务端缓存的 ticket，注意是 SIGN 类型	获取方式见 SIGN ticket 获取
nonce	32位随机字符串，字母和数字	合作方自行生成，不要带有特殊字符

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 appId、orderNo、version 连同 ticket、nonce 共5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 。

人脸认证多张照片查询接口

请求

请求URL：https://kyc1.qcloud.com/api/v2/base/queryphotoinfo?orderNo=xxx

> **注意**
> 为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。

请求方法：POST
HTTP 请求 header：

参数名	是否必选	类型	说明
Content-Type	是	String	application/json

请求参数：

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
version	版本号，默认值：1.0.0	String	20	是
nonce	随机数	String	32	是
orderNo	订单号，字母/数字组成的字符串，合作方订单的唯一标识	String	32	是

sign	签名值，使用本页第一步生成的签名	String	40	是

响应

返回参数说明：

参数名	类型	说明
code	int	0：成功 非0：失败
msg	String	请求结果描述
bizSeqNo	String	请求业务流水号
orderNo	String	订单编号
occurredTime	String	刷脸时间（yyyyMMddHHmmss）
photoList	List	Base64 图像列表（返回1 – 3张照片），若照片不存在，则返回 null

返回示例：

```
{
"code": 0,
"msg": "请求成功",
"bizSeqNo": "业务流水号",
"result": {
"orderNo": "AAAAAA001",
"occurredTime": "20180907142653",
"photoList": ["第一个base64photo字符串", "第二个base64photo字符串", "第三个base64photo字符串"]
}
}
```

说明

success：false 无意义，合作伙伴可忽略，无需处理。

E证通
开始集成

最近更新时间：2024-07-31 16:54:11

1. 产品功能

E证通是腾讯云慧眼与权威机构合作研发的，基于腾讯云人脸核身技术在微信小程序上提供的权威、统一、合规、无行业类目限制的身份信息核验服务，面向互联网金融，共享经济以及其他互联网行业的应用，为应用线上线下身份真实性验证与数字身份应用提供重要参考依据。核验过程如下：

2. 集成方式

E证通面向客户提供了多种集成渠道，请根据您的需求和技术背景，选择合适的集成渠道。以下是各渠道的简要说明：

渠道	适用场景	集成方式
微信小程序	适用于使用微信原生接口开发的小程序服务进行用户身份核验	请参考《微信小程序集成指引》
uni-app（小程序）	适用于使用 uni-app 框架开发的小程序服务进行用户身份核验	请参考《uni-app（小程序）集成指引》
H5（移动端浏览器）	适用于微信环境内 H5 页面：包括微信、企业微信、微信公众号	请参考《H5（移动端浏览器）集成指引》

3. 技术支持

如果您在接入过程中遇到任何问题，请 联系我们 获取技术支持。

微信小程序
接入流程说明

最近更新时间：2025-01-20 14:54:22

1. 概述

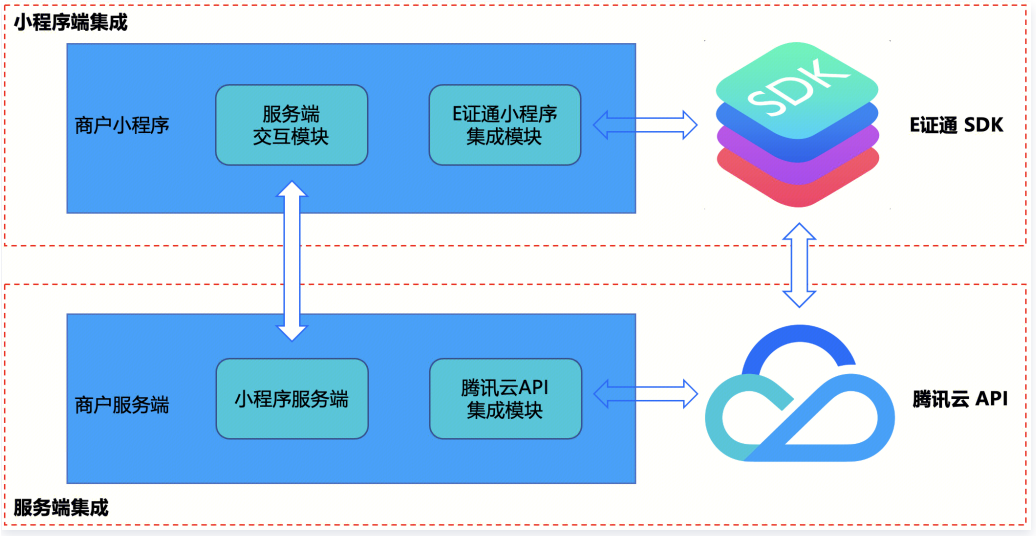
本接入流程说明旨在指导您如何将E证通微信小程序 SDK 集成到您的应用小程序中。在开始之前，请确保您已经阅读了微信小程序的相关文档并了解了所需的技术背景和配置要求。

2. 准备工作

- 注册腾讯云账号（如果您已在腾讯云注册账号，可忽略此步骤）。
- 参考 企业实名认证操作指引 完成账号企业实名认证。
- 开通腾讯云E证通服务，请参见 开通E证通 。

3. 整体架构图

下图为腾讯云E证通微信小程序的集成架构图：



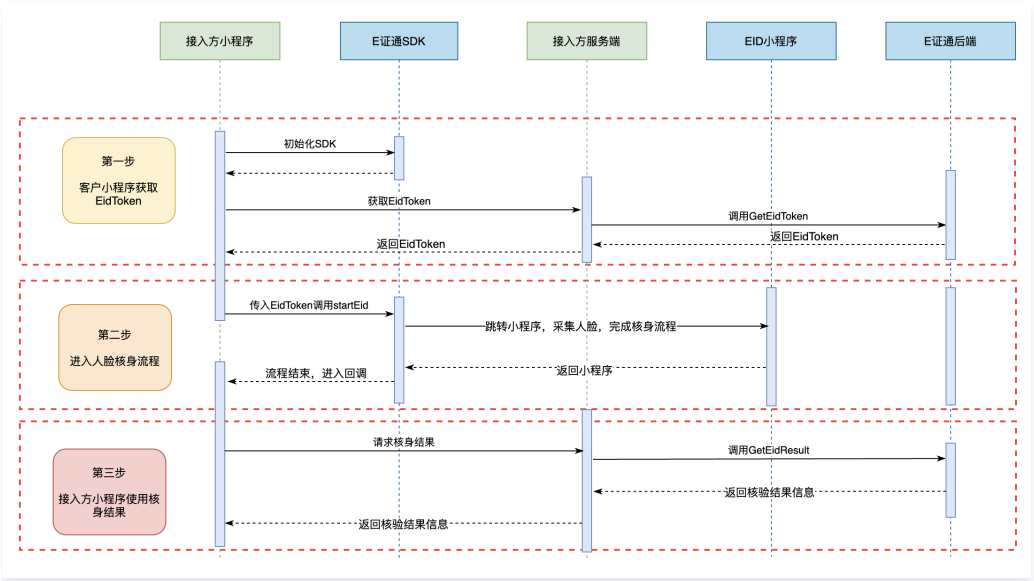
E证通微信小程序 SDK 集成包括两部分：
A. 服务端集成： 在您的服务端集成腾讯云提供的 API 接口，以获取串联整个流程的 EidToken 和获取E证通核验结果信息。
B. 小程序端集成： 将E证通微信小程序 SDK 集成到您的应用小程序中。

4. 整体交互流程

集成方只需要传入 EidToken 并启动对应的E证通微信小程序 SDK，便可实现包含 OCR 证件识别+活体检测+人脸比对全流程的用户身份认证，待微信小程序用户完成认证流程后，集成方可通过腾讯云 API 接口获取完整的身份认证结果信息。
E证通微信小程序集成相关的服务端 API 接口列表：

- 获取EidToken的API接口：GetEidToken
- 拉取E证通身份认证结果API接口：GetEidResult

下图展示了E证通 SDK、小程序端以及服务器端的整体交互逻辑：



具体的详细交互流程如下：

第一步、客户小程序获取 EidToken

1. 接入方小程序初始化E证通小程序 SDK。
2. 初始化完成后，接入方小程序发送请求到接入方服务端获取串联核身流程的 EidToken。
3. 接入方服务端与E证通服务端交互，调用 GetEidToken 接口获取核身需要的 EidToken 并返回，接口调用参考 服务端集成 。
4. 接入方服务端获取到 EidToken 后保存，并将 EidToken 返回给接入方小程序。

第二步、进入人脸核身流程

1. 接入方小程序拿到 EidToken 后调用 startEid 进入人脸核身流程。
2. E证通小程序 SDK 跳转到 EID 小程序，采集人脸，完成整个核身流程。
3. 核身流程结束后小程序进入回调，返回到接入方小程序。

第三步、接入方小程序使用核身结果

1. 接入方小程序收到回调后发送请求到接入方服务端请求核身结果。
2. 接入方服务端通过调用 GetEidResult 接口拉取核验结果信息并返回。接口调用参考服务端集成 。
3. 接入方服务端接收到E证通服务端返回的核身结果信息后保存，并告知接入方小程序核身结果。
4. 接入方小程序收到接入方服务端返回的核身结果信息后展示给用户，并进行接下来的业务流程。

5. 接入流程

5.1 申请商户ID

1. 登录 腾讯云慧眼 人脸核身控制台 ，单击自助接入 > E证通服务 > 申请商户 ID。

2. 完全填写相关信息，然后单击提交等待审核，审核通过后该商户 ID 便可用于获取E证通服务流程唯一标识 EidToken。审核时间从您提交当天开始计算，需要3个工作日，请您预留好时间并耐心等待。

> **注意：**
> 如果审核后需要您修改资料，审核预估时间会从下次提交时重新计算。

5.2 下载小程序 SDK

审核通过后，可以看到 下载小程序 SDK，**点击链接**下载即可。

5.3 如需要返回身份信息需制作 CSR，具体流程请参考指引

基于对个人隐私信息的保护，按照最小必要返回身份信息的要求，E证通服务不会返回姓名和身份证号的明文信息。如因业务需要返回，可以参考 E证通获取实名信息指引 申请加密返回。

5.4 开始接入

为了能够快速将E证通微信小程序SDK接入到您的小程序应用中，您需要分别完成以下两部分的集成工作：

- **服务端集成：** 请参考 服务端集成 完成服务端的集成工作
- **小程序集成：** 请参考 小程序集成 完成E证通小程序SDK的集成工作

6. 联系我们

如您在接入过程中遇到困难，可 联系我们 获取帮助。

服务端集成

最近更新时间：2024-07-05 16:34:31

1. 概述

本文详细介绍E证通微信小程序服务端集成流程，请您确保在阅读本文前已阅读 接入流程说明 并按照步骤执行。阅读完本文您可以轻松将您的服务端与腾讯云E证通小程序服务端连通。为了让您调用 API 更简单，腾讯云 API 3.0提供了多种编程语言的 SDK 供您选择，您可以选择您熟悉的语言进行编码，本文将使用Java 语言作为示例演示如何使用腾讯云 SDK 调用E证通微信小程序服务端接口。

2. 集成准备

2.1 获取密钥

在服务端集成之前，请您参阅控制台 云 API 密钥 页面获取密钥相关信息。

2.2 获取SDK

获取 Java SDK 请参考 腾讯云SDK README，如果需要获取其他语言的 SDK 可以阅读对应语言 SDK README 获取。

3. 接口说明

E 证通小程序服务端集成主要包含两个接口：GetEidToken 和 GetEidResult。每次调用E证通服务前，需先调用 GetEidToken 接口获取EidToken，用来串联E证通核身认证流程，在验证完成后，通过调用 GetEidResult 接口获取认证结果信息。在服务端集成接口前，您也可以通过腾讯云提供的 在线调试 功能查看每次调用的请求内容和返回结果以及自动生成 SDK 调用示例。详细的接口调用可以参考以下调用示例。

4. Example

4.1 获取EidToken

```
package com.tencent;

import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;
import com.tencentcloudapi.faceid.v20180301.FaceidClient;
import com.tencentcloudapi.faceid.v20180301.models.GetEidTokenConfig;
import com.tencentcloudapi.faceid.v20180301.models.GetEidTokenRequest;
import com.tencentcloudapi.faceid.v20180301.models.GetEidTokenResponse;

public class GetEidToken {
    private static final String SECRET_ID = ""; // TODO 您账号的腾讯云密钥
    private static final String SECRET_KEY = ""; // TODO 您账号的腾讯云密钥
    private static final String REGION = "ap-guangzhou";
    public static void main(String[] args) {
        // Step 1. 初始化客户端实例
        Credential credential = new Credential(SECRET_ID, SECRET_KEY);
        FaceidClient faceidClient = new FaceidClient(credential, REGION);

        // Step 2. 使用Tencent Cloud API SDK组装请求体，填充参数
        // 详细的接口文档请参考：https://cloud.tencent.com/document/product/1007/54089
        GetEidTokenRequest request = new GetEidTokenRequest();
        request.setMerchantId(""); // 商户ID，请在控制台查看已经申请的商户ID
        GetEidTokenConfig config = new GetEidTokenConfig();
        config.setInputType("3"); // 指定姓名和身份证号输入方式
        request.setConfig(config);

        try {
            // Step 3. 调用接口
            GetEidTokenResponse response = faceidClient.GetEidToken(request);
            System.out.println("SDK response: " + GetEidTokenResponse.toJsonString(response));
        } catch (TencentCloudSDKException e) {
            // 调用接口异常在这里处理
            System.err.println("invoke error, code: " + e.getErrorCode() + "message: " + e.getMessage());
            e.printStackTrace();
```

```
        }
    }
}
```

4.2 查询E证通核身结果

```
package com.tencent;

import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;
import com.tencentcloudapi.faceid.v20180301.FaceidClient;
import com.tencentcloudapi.faceid.v20180301.models.GetEidResultRequest;
import com.tencentcloudapi.faceid.v20180301.models.GetEidResultResponse;

public class GetEidResult {
    private static final String SECRET_ID = ""; // TODO 您账号的腾讯云密钥
    private static final String SECRET_KEY = ""; // TODO 您账号的腾讯云密钥
    private static final String REGION = "ap-guangzhou";
    public static void main(String[] args) {
        // Step 1. 初始化客户端实例
        Credential credential = new Credential(SECRET_ID, SECRET_KEY);
        FaceidClient faceidClient = new FaceidClient(credential, REGION);

        // Step 2. 使用Tencent Cloud API SDK组装请求体，填充参数
        // 详细的接口文档请参考：https://cloud.tencent.com/document/product/1007/54090
        GetEidResultRequest request = new GetEidResultRequest();
        request.setEidToken(""); // 通过GetEidResult接口获取到的EidToken
        request.setInfoType("0"); // 指定拉取的结果信息

        try {
            // Step 3. 调用接口
            GetEidResultResponse response = faceidClient.GetEidResult(request);
            System.out.println("SDK response: " + GetEidResultResponse.toJsonString(response));
        } catch (TencentCloudSDKException e) {
            // 调用接口异常在这里处理
            System.err.println("invoke error, code: " + e.getErrorCode() + "message: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

5. 错误码列表

错误码列表请参考文档 错误码列表 中服务端错误码部分。

小程序集成

最近更新时间：2025-02-13 16:42:45

1. 概述

本文为您描述如何接入E证通小程序，请您确保在阅读本文前已阅读 接入流程说明 并按照步骤执行。阅读完本文您可以轻松接入小程序SDK，调用E证通微信小程序服务。本文将使用微信小程序开发语言作为示例演示如何接入E证通微信小程序SDK。

2. 接入准备

2.1 下载小程序SDK

人脸核身控制台，单击**自助接入>E证通服务>下载小程序SDK**。

2.2 前置条件

step 1. 添加服务器域名白名单

小程序前端接口请求有域名白名单限制，未添加白名单的域名只能在调试模式下运行。接入方需要在小程序上线前将以下域名添加至服务器域名白名单：

```
// request 合法域名
eid.faceid.qq.com
```

step 2. 添加业务域名白名单

> **说明：**
> SDK V1.0.7及以上版本不需要添加业务域名白名单。
> 在小程序配置业务域名中，将下载后的**校验文件**在控制台随商户 ID 提交审核时上传，待腾讯侧审核通过后，将域名 `eid.faceid.qq.com` 添加至业务域名白名单。

3. 内嵌小程序 SDK

3.1 安装 SDK

下载完成小程序 SDK 后将E证通小程序 SDK 文件夹放在小程序根目录下。

3.2 SDK 调用步骤

step 1. 初始化 SDK

在 app.js 文件中引入初始化 SDK 的方法 initEid。 在 App.js 的 onLaunch() 中加入相应代码，在 App.json 文件里添加E证通 SDK 页面。 在 onLaunch 方法中调用 initEid。

```
//app.js
import { initEid } from './mp_ecard_sdk/main';

App({
```

```
    onLaunch() {
        initEid();
    },
});

// app.json
{
    "pages":[
        "mp_ecard_sdk/index/index",
// sdk v1.0.4及以下版本还需添加以下路径配置
        "mp_ecard_sdk/protocol/service/index",
        "mp_ecard_sdk/protocol/privacy/index",
        "mp_ecard_sdk/protocol/userAccredit/index",
        "mp_ecard_sdk/protocol/eid/index",
    ]
}
```

step 2. 调用 SDK

> **说明：**
> **接入v1.0.5及以上版本，调用 startEid 需注意：**调用 startEid 需通过按钮的**点击形式**发起调用（因v1.0.5及以上版本 sdk 更新，取消了授权页和其他按钮，使用了 wx.navigateToMiniProgram 主动拉起授权小程序）。

```
import { startEid } from './mp_ecard_sdk/main';

// 示例方法
goSDK(token) {
    startEid({
        data: {
            token,
        },
        verifyDoneCallback(res) {
            const { token, verifyDone } = res;
            console.log('收到核身完成的res:', res);
            console.log('核身的token是:', token);
            console.log('是否完成核身:', verifyDone);
        },
    });
},
```

startEid() 参数说明：

`startEid(options)`：进入实名认证页面。
`options`：Object required 接入方传入的参数。
`options.data.token`：String required 接入方小程序从接入方服务端获取 EidToken。
`options.data.needJumpPage`：是否需要中转页，默认为 false（sdk v1.0.7 及以上支持）
`options.data.enableEmbedded`：是否支持半屏打开，默认为 false（sdk v1.0.9 及以上支持）
`options.data.allowFullScreen`：半屏打开后是否支持全屏，默认为 true（sdk v1.0.9 及以上支持）
`options.verifyDoneCallback`：Function(res) required 核身完成的回调。res 包含验证成功的 token，是否完成的布尔值标志 verifyDone。请根据 res 返回的结果进行业务处理判断。

3.3 半屏接入指引（可选）

如您需要实现小程序半屏模式，可参考3.4步骤进行配置，如不需要实现半屏模式，则可直接跳到 获取 E证通的核验结果信息 。
小程序的半屏模式可以省去小程序跳转的弹窗步骤，实现如图所示的半屏页面展示，流程体验可以联系 慧眼小助手 获取：

接入准备

申请半屏调用 eID 数字身份小程序

打开小程序微信管理平台后台，选择设置，第三方设置，打开半屏小程序管理，单击添加。

输入 eID 数字身份的 AppID（联系 慧眼小助手 获取 AppID 并通过半屏申请）。

> ⚠ **注意：**

每个小程序最多可添加10个半屏打开的小程序。

SDK 接入

1. 请先完成上述步骤3.1 – 3.3的接入，并下载最新版本的 SDK（SDK 1.0.9及以上版本支持半屏接入）。
2. app.json 中添加如下代码，其中 AppID 为 `eID 数字身份` 的 AppID。

```
{  "embeddedAppIdList": ["AppID"]}
```

3. SDK 接入时传入允许半屏打开参数。

```
import { startEid } from './mp_ecard_sdk/main';

// 示例方法
goSDK(token) {
    startEid({
        data: {
            token,
            enableEmbedded: true,
            allowFullScreen: false
        },
        verifyDoneCallback(res) {
            const { token, verifyDone } = res;
            console.log('收到核身完成的res:', res);
            console.log('核身的token是:', token);
            console.log('是否完成核身:', verifyDone);
        },
    });
},
```

注意事项

1. 每个小程序最多可添加10个半屏打开的小程序。
2. 半屏接入需向E证通 `eID数字身份` 发送半屏申请，审核通过后才会生效。
3. 半屏接入不支持时会自动降级为全屏打开。
4. 因半屏接入是微信新上的能力，少部分机型还存在兼容性问题，请确保进行过完善的测试后再上线。

4. 调试 SDK

请在微信开发者工具中使用手机“预览”模式进行调试，请勿使用“真机调试”。

5. 注意事项

1. 从 eID 数字身份小程序返回接入方小程序。 当接入方小程序在初始化E证通 SDK 的时候，E证通 SDK 会通过 wx.onAppShow 注册一个监听从 eID 数字身份小程序跳转回接入方小程序的事件，从而根据情况触发接入方传入的核身完成的回调函数。
2. 由于微信的机制，用户在 eID 数字身份小程序跳转回接入方小程序的时候，同时也会触发接入方小程序 app.js 中的 onShow 方法。为了避免冲突，如果接入方小程序在 onShow 中有执行逻辑的话，需要排除掉从 eID 数字身份小程序跳转回接入方小程序这个场景。可通过以下方法实现：

```
// app.js

onShow(options) {
    const { referrerInfo, scene } = options;
    /* 判断是否从eID数字身份小程序返回 */
    const { appId } = referrerInfo;
    if (scene === 1038 && appId === 'wx0e2cb0b052a91c92') {
        return;
    } else {
        // 执行接入方小程序原本的逻辑
    }
},
```

错误码列表

最近更新时间：2024-07-03 18:06:31

服务端错误码列表

错误码	描述
InternalError.EncryptSystemError	加密失败。
FailedOperation.DecryptSystemError	解密失败。
FailedOperation.StsUnAuthErrError	STS 未授权。
FailedOperation.UnKnown	内部未知错误。
FailedOperation.UnregisteredEid	该用户未注册E证通，请先注册并跟权威库核验。
InvalidParameter	参数错误。
InvalidParameterValue	参数值错误。
InvalidParameter.UnsupportEncryptField	存在不加密的字段，请参考文档修改。
InvalidParameterValue.RuleIdDisabled	该 ruleid 已被您停用，请确认后重试。
InvalidParameterValue.RuleIdNotExist	RuleId 不存在，请到人脸核身控制台申请。
InvalidParameterValue.BizTokenExpired	BizToken 过期。
InvalidParameterValue.BizTokenIllegal	BizToken 不合法。
UnauthorizedOperation.Arrears	账号已欠费。
UnauthorizedOperation.NonAuthorize	账号未实名。
UnauthorizedOperation.Nonactivated	未开通服务。
InternalError	内部错误。

小程序端错误码列表

查看错误码

uni-app（小程序）

最近更新时间：2024-06-07 11:57:31

接入准备

申请商户 ID 及获取 EidToken

详情请参考 E证通小程序接入 。

前置条件

1. 添加服务器域名 名单
   程序前端接 请求有域名 名单限制，未添加 名单的域名只能在调试模式下运 。您需要在 程序上线前需要将以下域名添加 服务器域名 名单。

```
// request 合法域名
eid.faceid.qq.com
eid-enhance.faceid.qq.com
```

2. 添加业务域名 名单
   在小程序配置业务域名中，将下载后的校验文件在控制台随商户 ID 提交审核时上传，待腾讯侧审核通过后，将以下域名添加 业务域名 名单。

```
eid.faceid.qq.com
eid-enhance.faceid.qq.com
```

uni-app 接入

步骤一：注册并创建 uni-app 开发环境

uni-app 开发接 具体参照 uni 官 。

步骤二：下载并配置 mp_ecard_sdk 源码

1. 请到 控制台商户 ID 列表页 下载E证通小程序 uni-app版SDK。
2. 配置 sdk 源码。
   法 ：项 根 录配置（推荐）
   2.1 将 sdk 源码包 mp_ecard_sdk 件夹拷 到项 根 录。
   2.2 在 pages.json 中配置相关 。

```
 "pages": [
{
"path": "mp_ecard_sdk/index/index",
"style": {
"navigationBarTitleText": "腾讯云E证通授权"
}
},
// v1.0.3 以下版本需配置以下路径
{
"path": "mp_ecard_sdk/protocol/eid/eid",
"style": {
"navigationBarTitleText": "eID数字身份⍰程序服务协议",
"enablePullDownRefresh": false
}
},
{
"path": "mp_ecard_sdk/protocol/privacy/privacy",
"style": {
"navigationBarTitleText": "腾讯隐私政策",
"enablePullDownRefresh": false
}
},
```

```
{
"path": "mp_ecard_sdk/protocol/service/service",
"style": {
"navigationBarTitleText": "腾讯云□证通服务协议",
"enablePullDownRefresh": false
}
},
{
"path": "mp_ecard_sdk/protocol/userAccredit/userAccredit",
"style": {
"navigationBarTitleText": "□户授权协议",
"enablePullDownRefresh": false
}
}
]
```

法 ：任意位置 件夹配置

2.3 将 sdk 源码放置在其他 件夹下，例如 pagse/mp_ecard_sdk。

2.4 在 pages.json 中配置相关 ，pages 的 path 应为 mp_ecard_sdk 的相对位置路径，例如：pages/。

```
"pages": [
{
"path": "pages/mp_ecard_sdk/index/index",
"style": {
"navigationBarTitleText": "腾讯云□证通授权"
}
},
// v1.0.3 以下版本需配置以下路径
{
"path": "pages/mp_ecard_sdk/protocol/eid/eid",
"style": {
"navigationBarTitleText": "eID数字身份□程序服务协议",
"enablePullDownRefresh": false
}
},
{
"path": "pages/mp_ecard_sdk/protocol/privacy/privacy",
"style": {
"navigationBarTitleText": "腾讯隐私政策",
"enablePullDownRefresh": false
}
},
{
"path": "pages/mp_ecard_sdk/protocol/service/service",
"style": {
"navigationBarTitleText": "腾讯云□证通服务协议",
"enablePullDownRefresh": false
}
},
{
"path":
"pages/mp_ecard_sdk/protocol/userAccredit/userAccredit",
"style": {
"navigationBarTitleText": "□户授权协议",
"enablePullDownRefresh": false
}
}
],
```

2.5 修改 mp_ecard_sdk 下的 globalConfig.js 的 normalPath 参数,修改为对应相对路径，例如：'/pages'。注意，开头需要加'/'。

```
export default {
 normalPath: '/pages'
 }
```

3. 初始化
   - 法 ：可以在 App.vue 中全局初始化。

```
import { initEid } from './mp_ecard_sdk/main';
export default {
 onLaunch: function() {
 initEid();
 },
};
```

   - 法 ：在需要调 到的 法之前初始化即可。
4. 调
   在需要进 核身的地 引 调 SDK 的 法 startEid。

> ⚠ **注意**
> startEid 调 需要在 initEid 初始化之后。

```
import { startEid } from './mp_ecard_sdk/main'
// 示例法
startEid({
data: {
token, needJumpPage
},
verifyDoneCallback(res) {
const { token, verifyDone } = res;
console.log('收到核身完成的res:', res);
console.log('核身的token是:', token);
console.log('是否完成核身:', verifyDone);
},
});
```

startEid() 参数说明：

- `startEid(options)` ：进 实名认证 。
- `options` ：Object required 接 传 的参数。
- `options.data.token` ：String required 接 程序从接 服务端获取 EidToken。
- `options.data.needJumpPage:` 是否需要中转页，默认为 false（uni-sdk v1.0.3 及以上支持）
- `options.verifyDoneCallback` ：Function(res) required 核身完成的回调。res 包含验证成功的 token，是否完成的布尔值标志 verifyDone。请根据 res 返回的结果进 业务处理判断。

步骤三：半屏接入指引（可选）

如您需要实现小程序半屏模式，可参考3.4步骤进行配置，如不需要实现半屏模式，则可直接跳到 获取 E证通的核验结果信息 。
小程序的半屏模式可以省去小程序跳转的弹窗步骤，实现如图所示的半屏页面展示，流程体验可以联系 慧眼小助手 获取：

接入准备

申请半屏调用 eID 数字身份小程序

打开小程序微信管理平台后台，选择设置，第三方设置，打开半屏小程序管理，单击**添加**。

输入 eID 数字身份的 AppID（联系 慧眼小助手 获取 AppID 并通过半屏申请）。

> **注意：**
> 每个小程序最多可添加10个半屏打开的小程序。

SDK 接入

1. 请先完成 步骤一 及 步骤二 接入，并下载较新版本的 SDK（uni-app SDK v1.0.5及以上版本支持半屏接入）。
2. app.json 中添加如下代码，其中 AppID 为 `eID 数字身份` 的 AppID。

```
{
  "embeddedAppIdList": ["AppID"]
}
```

3. SDK 接入时传入允许半屏打开参数。

```
import { startEid } from './mp_ecard_sdk/main';

// 示例方法
goSDK(token) {
    startEid({
        data: {
            token,
            enableEmbedded: true,
            allowFullScreen: false
        },
        verifyDoneCallback(res) {
            const { token, verifyDone } = res;
            console.log('收到核身完成的res:', res);
            console.log('核身的token是:', token);
            console.log('是否完成核身:', verifyDone);
        },
    });
},
```

startEid() 参数说明：

- 半屏接入不支持时会自动降级为全屏打开。
- 因半屏接入是微信新上的能力，少部分机型还存在兼容性问题，请谨慎开启。
- `options.data.token`：String required 接入方小程序从接入方服务端获取 EidToken。
- `options.data.needJumpPage`：是否需要中转页，默认为 false（sdk v1.0.7 及以上支持）
- `options.data.enableEmbedded`：是否支持半屏打开，默认为 false（uni-app 版 sdk1.0.5及以上支持）
- `options.data.allowFullScreen`：半屏打开后是否支持全屏，默认为 true（uni-app 版 sdk1.0.5及以上支持）
- `options.verifyDoneCallback`：Function(res) required 核身完成的回调。res 包含验证成功的 token，是否完成的布尔值标志 verifyDone。请根据 res 返回的结果进行业务处理判断。

注意事项

1. 每个小程序最多可添加10个半屏打开的小程序。
2. 半屏接入需向E证通 `eID数字身份` 发送半屏申请，审核通过后才会生效。
3. 半屏接入不支持时会自动降级为全屏打开。
4. 因半屏接入是微信新上的能力，少部分机型还存在兼容性问题，请确保进行过完善的测试后再上线。

获取 E 证通核验结果信息

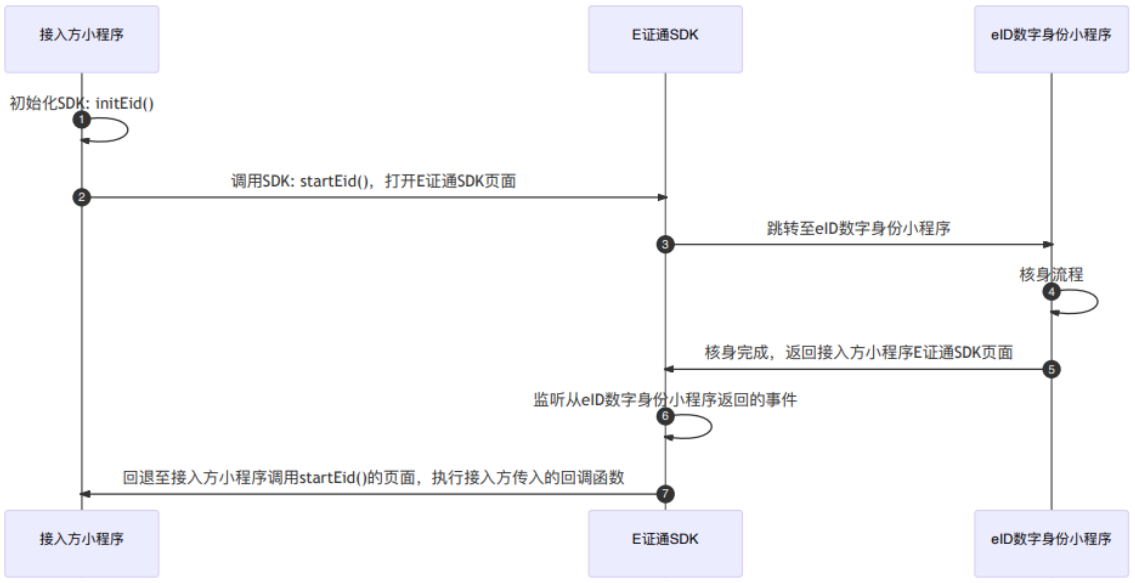
户完成 脸核身后，会以回调形式返回 EidToken 以及其他信息，接 程序将 EidToken 传给接 的服务端，接 服务端即可凭借 EidToken 参数调获取 程序核身结果信息 GetEidResult 接 去获取本次核身的详细信息，最后将核身结果返回给接 程序。

> **说明**
> EidToken 作为 次核身流程的标识，有效时间为600秒；完成核身后，可 该标识获取3天内验证结果信息。

卸载 sdk

删除 `mp_ecard_sdk` 件夹。

接 时序图



完整示例参考

- Vue2 Demo 示例
- Vue3 Demo示例

注意事项

- 从 eID 数字身份　程序返回接　　程序
  当接　　程序在初始化 E 证通 SDK 的时候，E 证通 SDK 会通过 wx.onAppShow 注册　个监听从 eID 数字身份　程序跳转回接　　程序的事件，从　根据情况触发接　　传　的核身完成的回调函数，由于微信的机制，　户在 eID 数字身份　程序跳转回接　　程序的时候，同时也会触发接　　程序 app.js 中的 onShow　法。为了避免冲突，如果接　　程序在 onShow 中有执　逻辑的话，需要排除掉从 eID 数字身份　程序跳转回接　　程序这个场景。
  可通过以下　法实现：

```
// app.vue
onShow(options) {
 const { referrerInfo, scene } = options;
 /* 判断是否从eID数字身份□程序返回 */
 const { appId } = referrerInfo;
 if (scene === 1038 && appId === 'wx0e2cb0b052a91c92') {
 return;
 } else {
 // 执□接□□□程序原本的逻辑
 }
},
```

- 基于对个人隐私信息的保护，按照最小必要返回身份信息的要求，E证通服务不再返回姓名和身份证号的明文信息。如因业务需要返回，请查看E证通获取实名信息指引 。

H5（移动端浏览器）

最近更新时间：2025-01-15 17:14:53

本文为您描述如何接入E证通 H5，接入相关指引链接如下：

1. 开通E证通
2. 申请商户 ID
3. 获取 EidToken
4. 跳转至 H5 进行核身
5. 获取E证通核验结果信息
6. 查看错误码

详细接入操作如下：

申请商户 ID

1. 登录腾讯云慧眼 人脸核身控制台，单击**自助接入>E证通服务>申请商户 ID**。

2. 完全填写相关信息，然后单击提交。审核时间从您提交当天开始计算，需要3个工作日，请您预留好时间并耐心等待。

> **注意**
> 如果审核后需要您修改资料，审核预估时间会从下次提交时重新计算。

3. 审核通过后，可以在自助接入列表页，查看商户 ID，后续需要使用商户 ID 获取E证通服务流程唯一标识 EidToken。

获取 EidToken 和 URL

接入方服务端调用 获取E证通 Token 接口，传入E证通服务所需信息获取到 EidToken 以及核身 URL。该 URL 有效为十分钟，且仅能使用一次。

跳转至 H5 进行核身

用户跳转至第二步获取到的 URL 完成核身之后，接入方可以有两种方式获取核身结果事件：接入方后台轮询、E证通重定向。

接入方后台轮询方式

如果接入方业务全程都在 PC 端的网页进行，可以将第二步获取到的核身 URL 转换为二维码进行展示，提示用户使用手机微信扫码进入E证通业务流程。接入方通过后台轮询 CheckEidTokenStatus 接口的方式实时检查用户的验证状态，然后根据返回的状态合理设计业务流程。

> **注意**
> 建议轮询间隔为1秒。遇到网络错误等异常状态连续三次以上才认为失败，否则均保持上一次状态。

重定向方式

如果接入方有自己的 H5，需要在第二步获取 EidToken 传入 `RedirectUrl`，在用户核身完成之后，Eid 会在微信 H5 中重定向到 RedirectUrl 并在 query 参数携带 `token={EidToken}`。接入方 H5 接收到参数后可以进行后续流程。

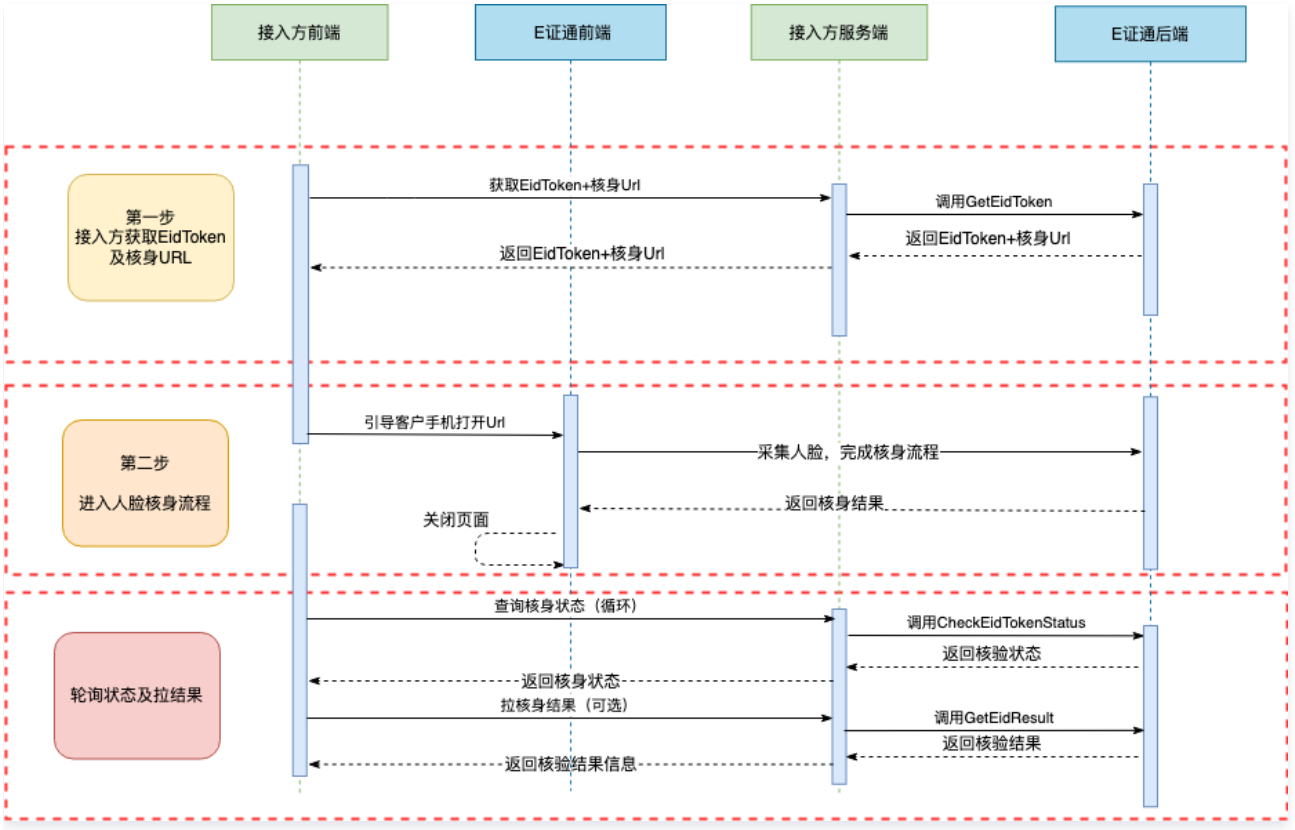
> **注意**
> 建议接入方重定向地址为后端接口，并在 URL 中有自身的鉴权信息，核身结果根据 EidToken 从腾讯云后台拉取。

获取 E证通核验结果信息

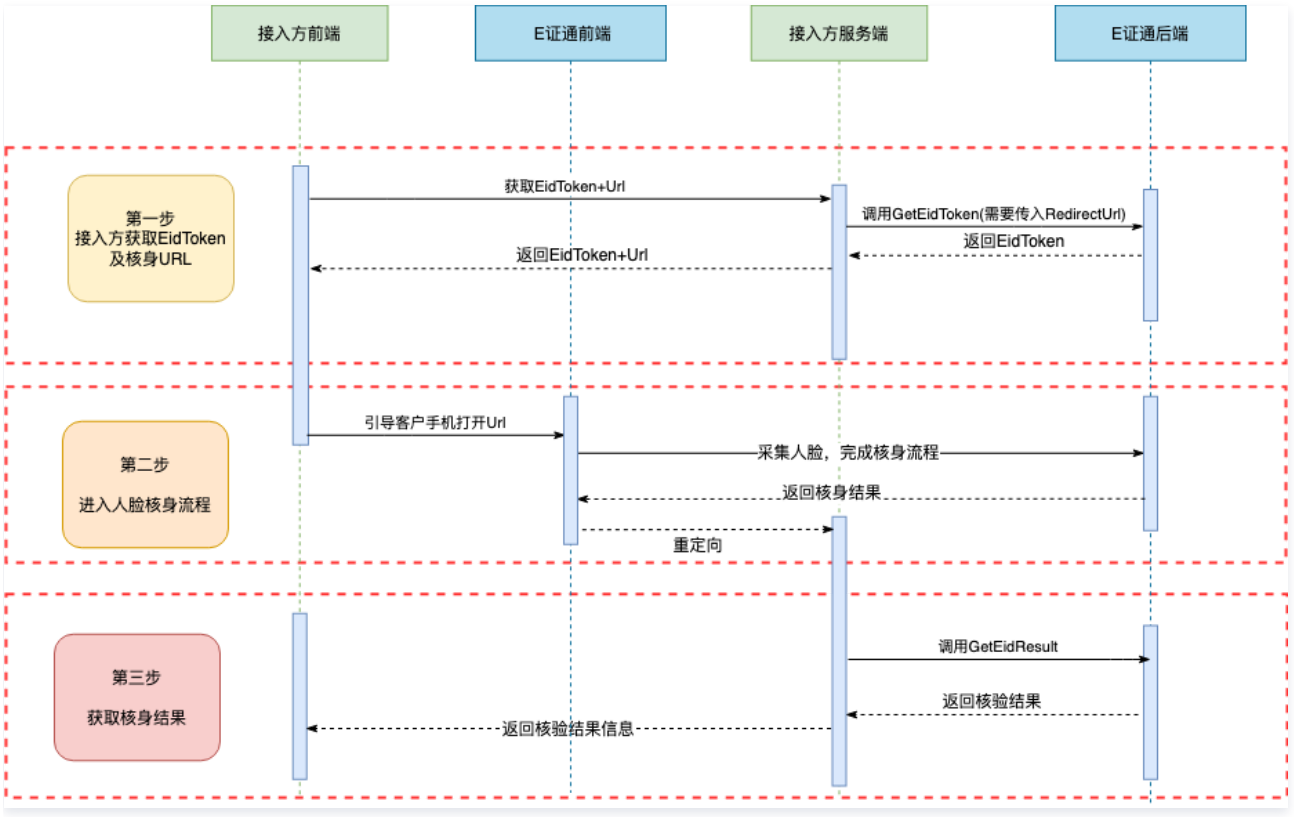
接入方获取到完成事件后，调用 获取 Eid 核身结果信息 接口去获取本次核身的详细信息。

接入时序图（轮询）

> **注意**
> 轮询应从获取 EidToken 之后就开始，对 EidToken 状态相应处理，如果核身完成就可以拉取核身结果。



接入时序图（重定向）



说明：
基于对个人隐私信息的保护，按照最小必要返回身份信息的要求，E证通服务不再返回姓名和身份证号的明文信息。如因业务需要返回，请查看E证通获取实名信息指引。

意愿核身
微信小程序
接入准备

最近更新时间：2025-06-25 09:50:22

意愿核身

意愿核身产品是腾讯云慧眼结合人脸核身与实时音视频技术打造的一款满足实名、实人、真意愿需求的产品，用户只需对准摄像头拍摄人脸视频并进行语音朗读/语音问答/点头确认，即可完成本人真实身份的认证以及真实意愿的确认，通过便捷的身份认证方式让业务办理更加安全、更加合规。

前置条件

1. 确认小程序类目

- 意愿核身基于微信小程序原生体验，要求符合以下行业要求且具备相关资质的客户才能申请，接入前请确认小程序的主体和类目是否符合以下范围。
- 接入前请登录 微信公众平台，打开**设置 > 基本设置 > 服务类目**，确认小程序服务类目满足以下类目要求，并准备好对应的资质文件，资质审核需要5 – 7个工作日，请预留好上线时间。

> **注意**
> 请确认接入方小程序同时符合以下两个类目要求的类目范围，如不符合，请选择使用意愿核身 E 证通小程序或是其他非小程序类的人脸核身服务。如有疑问，欢迎您 联系我们 进行询问。

1.1 类目要求1

- 政务民生：政府机构或事业单位
- 教育：学历教育（学校）
- 医疗：公立医疗机构、互联网医院、私立医疗机构
- 金融业：银行、信托、公募基金、证券/期货、保险、消费金融
- 快递业与邮政：寄件/收件
- 交通服务：打车（网约车）、航空、地铁、水运、城市交通卡、城市共享交通、火车、公交公司、长途客运、租车、高速服务
- 生活服务：生活缴费
- IT 科技：基础电信运营商、转售移动通信
- 旅游：酒店服务
- 商业服务：公证
- 社交：直播

详细类目限制及所需审核材料请参阅：微信 HTML5 及小程序资质文件列表 。

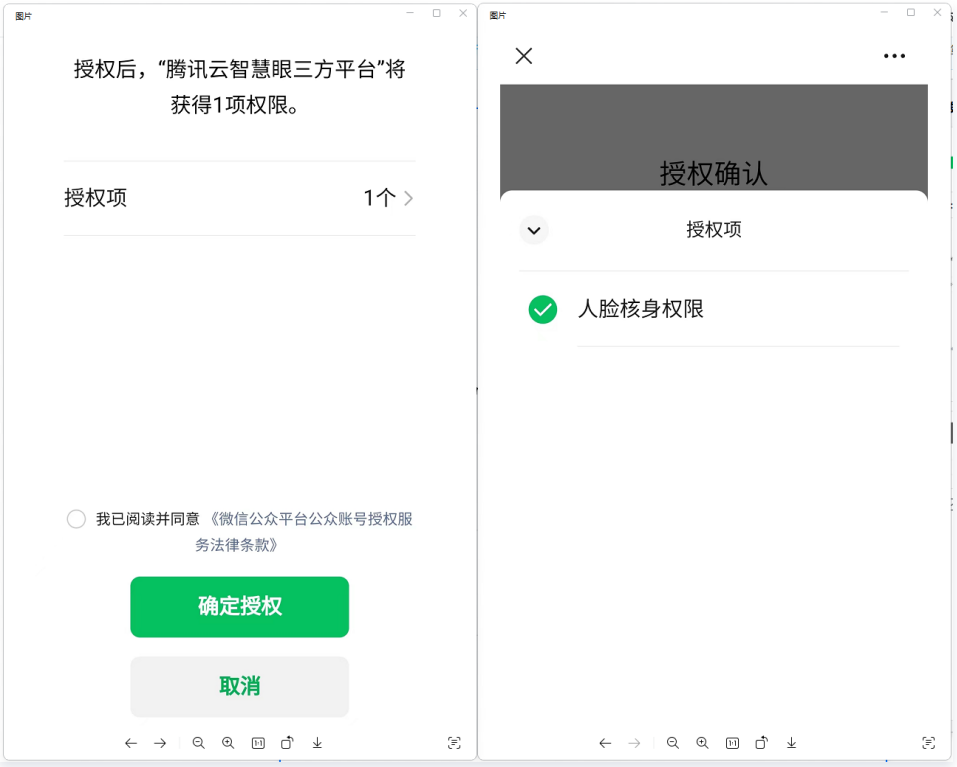
1.2 类目要求2

- 社交：直播
- 教育：在线视频课程
- 医疗：互联网医院，公立医疗机构，私立医疗机构
- 金融：银行、信托、公募基金、私募基金、证券/期货、证券、期货投资咨询、保险、征信业务、新三板信息服务平台、股票信息服务平台（港股/美股）、消费金融
- 汽车：汽车预售服务
- 政府主体账号：政府相关工作推广直播、领导讲话直播等
- 工具：视频客服
- IT 科技：多方通信；音视频设备

详细类目限制及所需审核材料请参阅：小程序-媒体组件说明 。

2. 设置授权

小程序开发需要授权，打开二维码，小程序管理员扫码后，单击授权项，只勾选人脸核身权限，将该权限授权给慧眼第三方平台，如下图所示：



3. 接口设置

意愿核身核验流程使用到微信小程序的 `live-pusher` 和 `live-player` 标签，因此需要您在“开发 > 接口设置”中开启实时播放音视频流和实时录制音视频流功能。

登录 微信公众平台 > 开发 > 开发管理 > 接口设置 > 其他接口：

4. 添加白名单

小程序前端接口请求有域名白名单限制，未添加白名单的域名只能在调试模式下运行，您需要在小程序上线前需要添加白名单限制。

登录 微信公众平台 > 开发 > 开发管理 > 开发设置 > 服务器域名：

Step 1. 添加服务器域名白名单

小程序前端接口请求有域名白名单限制，未添加白名单的域名只能在调试模式下运行。接入方需要在小程序上线前将以下域名添加至服务器域名白名单：

- **Request 合法域名：**

```
https://events.tim.qq.com;
https://faceid.qq.com;
https://grouptalk.c2c.qq.com;
https://pingtas.qq.com;
https://web.sdk.qcloud.com;
https://webim.tim.qq.com;
https://yun.tim.qq.com;
```

- **Socket 合法域名：**

```
wss://wss.im.qcloud.com;
wss://wss.tim.qq.com;
wss://faceid.qq.com;
// v1.0.9及以上版本身份校验环节如需NFC方式读取证件，需要添加以下socket域名
wss://idcloudread.eidlink.com
```

- **uploadFile 合法域名：**

```
https://cos.ap-shanghai.myqcloud.com;
https://faceid.qq.com;
```

- **downloadFile 合法域名：**

```
https://cos.ap-shanghai.myqcloud.com;
https://faceid.qq.com;
```

Step 2. 添加业务域名白名单

在小程序配置业务域名中，将域名 `faceid.qq.com` 添加至业务域名白名单，联系 小助手 提交校验文件。

5. 注册腾讯云企业账号

意愿核身服务目前只向已完成企业实名认证的腾讯云用户开放，使用意愿核身服务前需要您注册腾讯云账号并完成企业实名认证。

- **注册腾讯云账号 （如果您已在腾讯云注册，可忽略此步骤）。**
- **参考 企业实名认证操作指引 完成账号企业实名认证。**

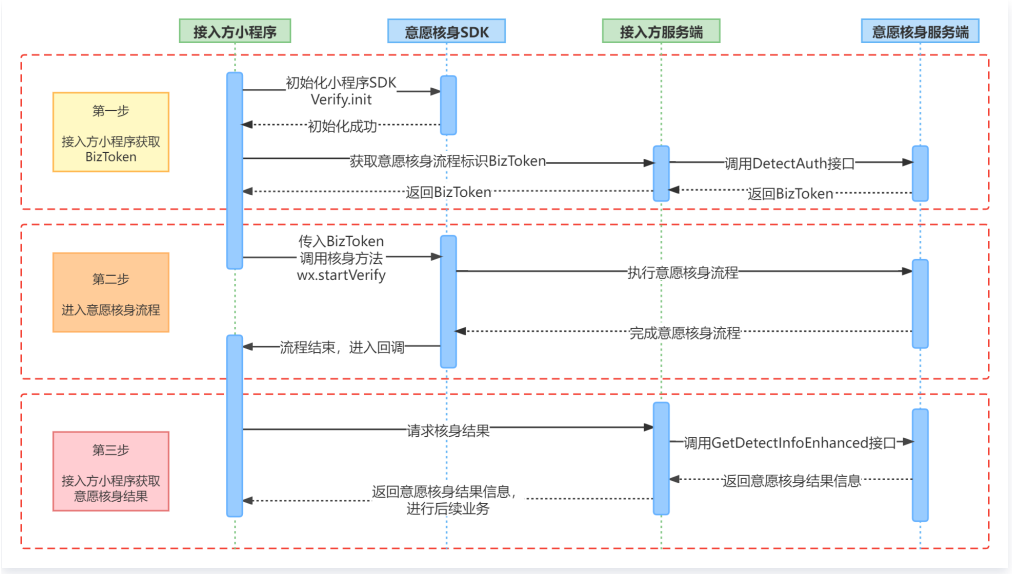
意愿核身小程序接入流程

最近更新时间：2024-11-11 11:28:12

1. 前提条件

1.已注册腾讯云账号，并完成企业实名认证。
2.已申请开通腾讯云人脸核身服务。
如果还未完成以上操作，可参考 流程指引 完成操作。

2. 接入时序图



3. 接入流程

下面将为您描述如何接入意愿核身小程序，详细接入操作如下：

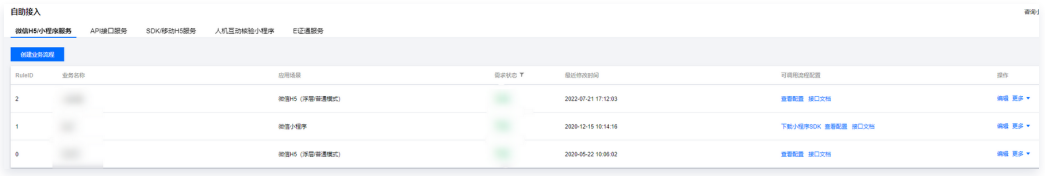
步骤1：接入前准备

确认小程序类目、配置授权以及域名白名单，详情请参见 接入准备 。

步骤2：创建 RuleID

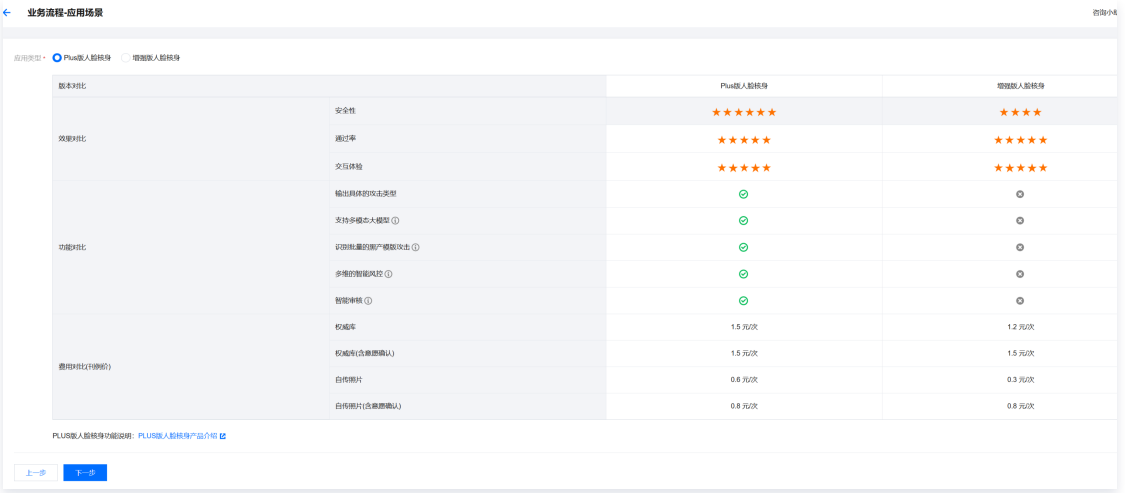
RuleId 用于调用配置的业务流程，创建步骤如下：

1. 登录人脸核身控制台，在 自主接入 页面，单击**创建业务流程**。



2. 选择应用场景：选择**微信小程序**并填写相关信息。

3. 选择应用类型，根据您的实际业务情况选择：
   - 如您的业务对安全性要求比较高可选择 **Plus 版人脸核身**。
   - 如对安全等级没有那么高，可选择**增强版人脸核身**，选好后单击**下一步**。

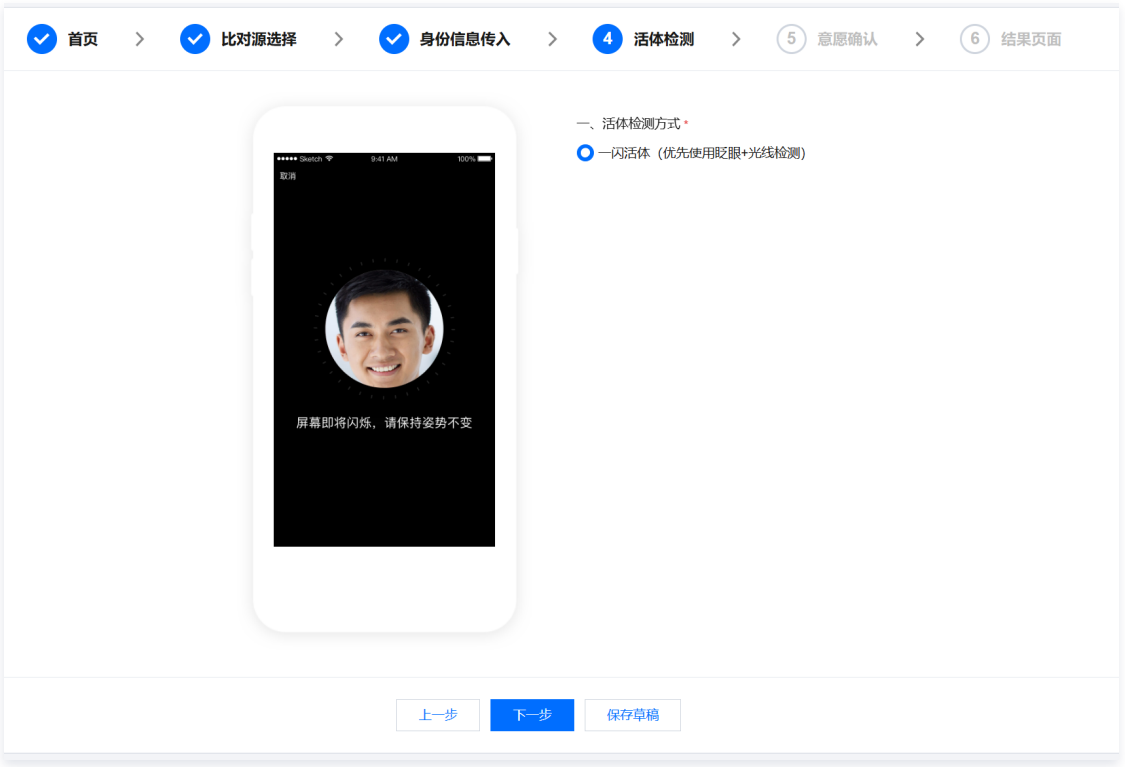


4. 进入接入配置，根据您业务的实际场景填写页面标题、业务名称、业务描述信息后单击**下一步**。

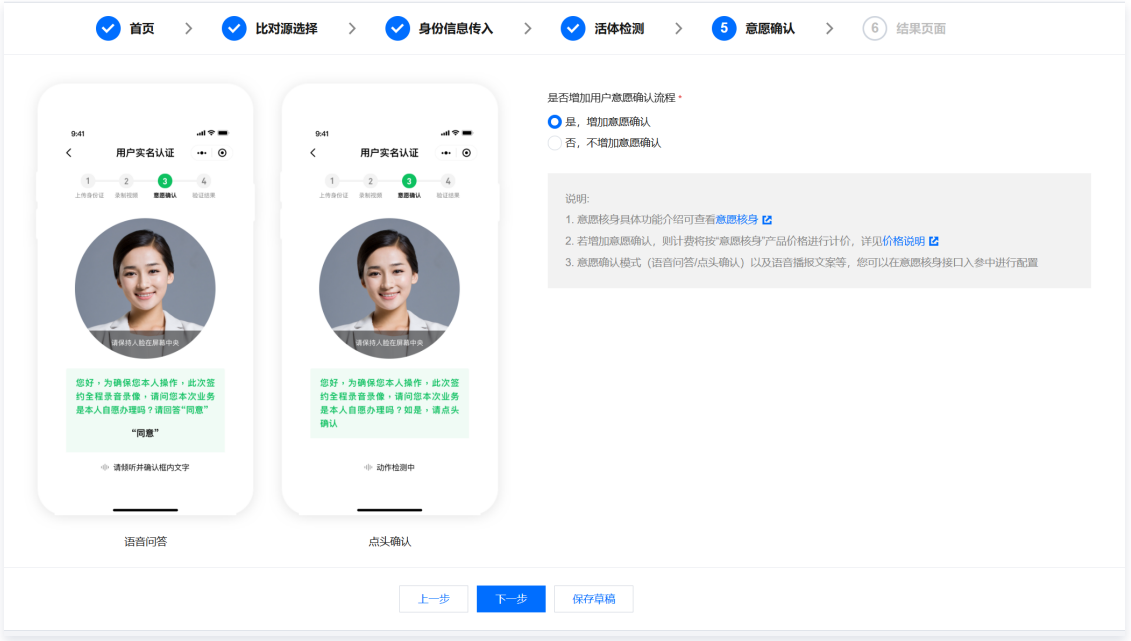
5. 选择人脸比对库源。人脸核身支持两种方式：跟权威库比对和跟上传照片比对。
   - 其中**跟权威库比对**收费价格为意愿核身（权威库）的价格。
   - 跟**上传照片比对**收费的价格为意愿核身（自传照片）的价格，详细价格请见 价格说明 。

6. 配置身份证 OCR 功能，如果不需要则勾选“**不需要用户在验证时上传**”，然后单击**下一步**。

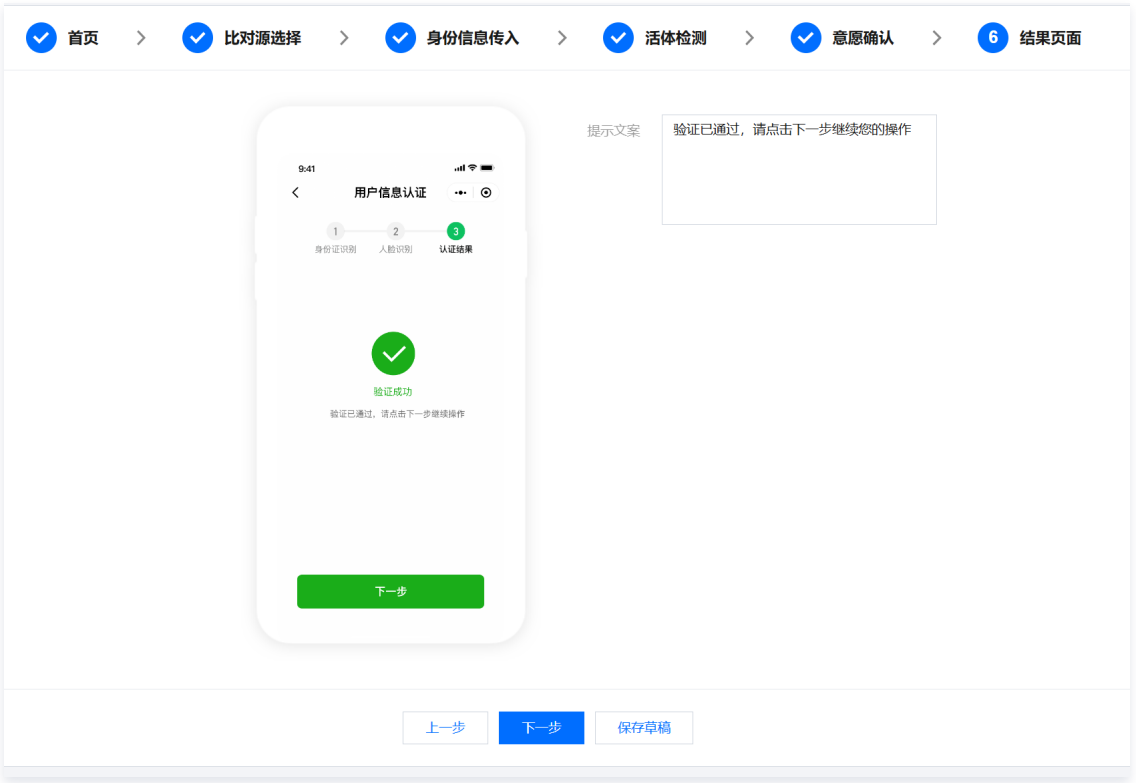
7. 配置活体检测方式，勾选后单击**下一步**。



8. 配置意愿确认时请勾选**是，增加意愿确认**，勾选后单击**下一步**。配置后产品将按意愿核身价格计费，详情请参见 价格说明 。



9. 配置结果页的文案描述，然后单击**下一步**。



10. 业务信息填写完成后，确认您的配置信息然后单击**确认并提交审核**。

步骤2：获取 BizToken

- 下载 意愿核身小程序 SDK，并在小程序代码中引入，调用 init 方法进行初始化。
- 接入方服务端调用实名核身鉴权 DetectAuth 接口，传入意愿核身所需配置参数，获取到核身流程标识（BizToken）。
  - 可在Config参数中选择的所需意愿核身类型（问答模式/点头确认模式），使用 IntentionQuestions.N 或 IntentionActions.N 传入问答模式或点头确认模式的语音播报内容。
  - 建议开启 Config 中的 IntentionRecognition（意图识别）开关，可提升语音识别通过率。

步骤3：跳转人脸核身 URL 完成核验

接入方服务端将 BizToken 返回给接入方小程序，然后小程序调用核身方法 startVerify 进入核身流程。

步骤4：查询核验结果信息

用户完成人脸核身后，会以回调函数形式返回 BizToken ，接入方小程序将 BizToken 传给接入方服务端，接入方服务端即可凭借 BizToken 参数调用获取实名核身结果信息增强版 GetDetectInfoEnhanced 接口去获取本次核身的详细信息，最后将核身结果返回给接入方小程序。

4. SDK 接入

4.1 开发准备

- **下载 SDK**
  下载 意愿核身小程序 SDK，并在小程序代码中引入，调用 init 方法进行初始化。
- **安装 SDK**
  将小程序 SDK 文件夹放在小程序根目录下，使用 require 函数引入。
  `const Verify = require('/verify_mpsdk/main');`
- **卸载 SDK**
  卸载时删除 `verify_mpsdk` 文件夹，移除相应 require 代码即可。
- **调试 SDK**
  微信开发者工具中使用预览模式调试。

4.2 快速入门

1. 将 verify_mpsdk 文件夹放到小程序项目根目录。
2. 初始化意愿核身 SDK。
   在 App.js 的 onLaunch() 中加入相应代码，在 App.json 文件里添加意愿核身页面 `verify_mpsdk/index/index` 。

```
//app.js
App({
    onLaunch: function () {
        // 初始化意愿核身组件
        const Verify = require('/verify_mpsdk/main');
        Verify.init();
    }
})
// app.json
{
    "pages":[
        "verify_mpsdk/index/index"
    ]
}
```

3. 调用 SDK 功能函数 wx.startVerify()。
   在需要意愿核身的地方调用 wx.startVerify() 进入意愿核身页面，认证完成会触发对应的回调函数。

```
// 单击某个按钮时，触发该函数
gotoVerify: function () {
    // 去接入方服务端调用DetectAuth接口获取BizToken，需要接入方服务端自行实现
    let BizToken = getBizToken();
    // 调用实名核身功能
    wx.startVerify({
        data: {
            token: BizToken // BizToken
        },
        success: (res) => { // 验证成功后触发
            // res 包含验证成功的token
        },
        fail: (err) => {  // 验证失败时触发
            // err 包含错误码，错误信息
        }
    });
}
```

4.3 基本 API 描述

- `Verify.init(options)` ：初始化插件。
- `options` ：Object required 初始化的参数。
- `wx.startVerify(options)` ：进入意愿核身页面。
- `options` ：Object required 初始化的参数。

- `options.data.token` ：String required 客户后端调用 DetectAuth 接口获取的 BizToken。
- `options.success` ：Function(res) required 验证成功的回调。res 包含验证成功的 token。
- `options.fail` ：Function(err) required 验证失败的回调。err 包含错误码、错误信息。

5. 完整示例参考

DEMO 示例：意愿核身 demo

6. 已有项目切换成意愿核身

1. 完成 接入准备 中的前置条件配置。
2. 下载并更新 SDK 为 意愿核身小程序 SDK 。
3. 参考步骤2，在第7点配置意愿确认时勾选**是，增加意愿确认。**
4. 接入方服务端调用实名核身鉴权 DetectAuth 接口，传入意愿核身所需字段，获取到核身流程标识（BizToken）。
   - 可在 Config 参数中选择的所需意愿核身类型（问答模式/点头确认模式），使用 IntentionQuestions.N 或 IntentionActions.N 传入问答模式或点头确认模式的语音播报内容。
   - 建议开启 Config 中的 IntentionRecognition（意图识别）开关，可提升语音识别通过率。
5. 用户完成人脸核身后，会以回调函数形式返回 BizToken ，接入方小程序将 BizToken 传给接入方服务端，接入方服务端即可凭借 BizToken 参数调用获取实名核身结果信息增强版 GetDetectInfoEnhanced 接口去获取本次核身的详细信息，最后将核身结果返回给接入方小程序。

意愿核身 uni-app 小程序接入流程

最近更新时间：2025-04-25 17:43:22

接入准备

授权指引及接入准备参见 意愿核身微信小程序接入准备 。
接入流程参见 意愿核身小程序接入流程 步骤1-3。

uni-app接入

步骤一：注册并创建 uni-app 开发环境

uni-app开发接入具体参考 uni 官网 。

步骤二：下载 SDK

控制台下载最新版本的意愿核身 SDK。

步骤三：并配置 verify_mpsdk

本 SDK 仅支持 uni 微信小程序端。

1. **将 verify-mpsdk 文件夹拷贝到项目根目录的 wxcomponents 文件夹下。**
2. **创建一个空页面调用组件。**

   pages.json 注册组件地址，如pages/verify/index 。

```
{
  "pages": [
    ...
    {
      "path": "pages/verify/index",
      "style": {
        "navigationBarTitleText": "uni-app",
        "usingComponents": {
              "verify-mp": "/wxcomponents/verify_mpsdk/indexCom"
          }
      }
    }
  ]
}
```

   在注册的地址生成小程序sdk页，调用verify-mp组件。

> **注意：**
> 该页面为空内容组件调用页，不需要其他逻辑代码。

```
<template>
    <verify-mp></verify-mp>
</template>

<script>
export default {
  data() {
    return {   }
  },
  onLoad() {},
  methods: {},
}
</script>
```

3. **初始化。**

- 方法一：可以在 App.vue 中全局初始化。

```
export default {
 onLaunch: function() {
   const verify = require('/wxcomponents/verify_mpsdk/main');
   verify.init();
 },
};
```

- 方法二：在需要调用到的页面方法之前初识化即可。

4. 调用 startVerify。

> ⚠ 注意：
> startVerify 调用需要在 init 初始化之后。

```
// 业务发起调用页面
wx.startVerify({
  data: {
    token: '', // 必要参数，BizToken
    startPath: 'pages/verify/index' // 必要参数，配置了verify核身组件的页面地址
  },
  success: (res) => { // 验证成功后触发
    // res 包含验证成功的token，这里需要加500ms延时，防止iOS下不执行后面的逻辑
    // 验证成功后，拿到token后的逻辑处理，具体以客户自身逻辑为准
  },
  fail: (err) => {  // 验证失败时触发
    // err 包含错误码，错误信息，弹窗提示错误
  }
});
```

基本 API 描述

- Verify.init(options)：初始化插件。
  - options：Object required 初始化的参数。
- wx.startVerify(options)：进入实名认证页面。
  - options：Object required 初始化的参数。
  - options.data.startPath：组件初始化页面，必填。
  - options.data.token：String required 客户后端调用 DetectAuth 接口获取的 BizToken。
  - options.success：Function(res) required 验证成功的回调。res 包含验证成功的 token。
  - options.fail：Function(err) required 验证失败的回调。err 包含错误码、错误信息。

卸载 SDK

卸载时删除 verify_mpsdk 文件夹，移除相应代码即可。

示例 demo

示例 demo 下载。

E证通
意愿核身 E证通接入流程

最近更新时间：2025-01-15 17:14:53

本文为您描述如何接入意愿核身 E证通小程序，详细接入操作如下：

1. 开通 E证通
2. 申请商户 ID
3. 使用商户 ID 获取 E证通服务流程唯一标识 EidToken
4. 下载、安装小程序 SDK
5. 获取意愿核身（E证通）核验结果信息
6. 查看错误码

详细接入操作如下：

1. 申请商户 ID

1. 登录腾讯云慧眼 人脸核身控制台，单击**自助接入>E证通服务>申请商户 ID**。
2. 完整填写相关信息，然后单击**提交**。审核时间从您提交当天开始计算，需要3个工作日，请您预留好时间并耐心等待。

> **注意：**
> 如果审核后需要您修改资料，审核预估时间会从下次提交时重新计算。

3. 审核通过后，可以看到商户 ID 和 E证通小程序 SDK，后续需要使用商户 ID 获取 E证通服务流程唯一标识 EidToken。

2. 获取 EidToken

接入方服务端调用获取E证通 Token GetEidToken 接口，传入 E证通服务所需参数及意愿核身所需参数 Config 信息，获取到 EidToken。

参数名称	必选	类型	描述
Action	是	String	公共参数，本接口取值：GetEidToken。
Version	是	String	公共参数，本接口取值：2018-03-01。
Region	否	String	公共参数，本接口不需要传递此参数。
MerchantId	是	String	EID商户id，字段长度最长50位。
IdCard	否	String	身份标识（未使用OCR服务时，必须传入）。 规则：a-zA-Z0-9组合。最长长度32位。
Name	否	String	姓名。（未使用OCR服务时，必须传入）最长长度32位。中文请使用UTF-8编码。
Extra	否	String	透传字段，在获取验证结果时返回。最长长度1024位。
Config	否	GetEidTokenConfig	小程序模式配置，包括如何传入姓名身份证的配置，以及是否使用意愿核身。
RedirectUrl	否	String	最长长度1024位。用户从Url中进入核身认证结束后重定向的回调链接地址。EidToken会在该链接的query参数中。
Encryption	否	Encryption	敏感数据加密信息。对传入信息（姓名、身份证号）有加密需求的用户可使用此参数，详情请点击左侧链接。

3. 内嵌小程序 SDK

3.1 前置条件

step 1. 添加服务器域名白名单

小程序前端接口请求有域名白名单限制，未添加白名单的域名只能在调试模式下运行。接入方需要在小程序上线前将以下域名添加至服务器域名白名单：

```
// request 合法域名
eid.faceid.qq.com
```

step 2. 添加业务域名白名单

> **说明：**
> SDK V1.0.7及以上版本不需要添加业务域名白名单。

在小程序配置业务域名中，将下载后的**校验文件**在控制台随商户 ID 提交审核时上传，待腾讯侧审核通过后，将域名 `eid.faceid.qq.com` 添加至业务域名白名单。

3.2 安装 SDK

商户 ID 申请通过后，在 控制台商户 ID 列表页 可以下载 E证通小程序 SDK。下载完成后将 E证通小程序 SDK 文件夹放在小程序根目录下。

3.3 SDK 调用步骤

step 1. 初始化 SDK

1. 在 app.js 文件中引入初始化 SDK 的方法 initEid。
2. 在 App.js 的 onLaunch() 中加入相应代码，在 App.json 文件里添加E证通 SDK 页面。
3. 在 onLaunch 方法中调用 initEid。

```
//app.js
import { initEid } from './mp_ecard_sdk/main';

App({
    onLaunch() {
        initEid();
    },
});

// app.json
{
    "pages":[
        "mp_ecard_sdk/index/index",
// sdk v1.0.4及以下版本还需添加以下路径配置
        "mp_ecard_sdk/protocol/service/index",
        "mp_ecard_sdk/protocol/privacy/index",
        "mp_ecard_sdk/protocol/userAccredit/index",
        "mp_ecard_sdk/protocol/eid/index",
    ]
}
```

step 2. 调用 SDK

> **注意：**
> **接入v1.0.5及以上版本，调用 startEid 需注意：**调用 startEid 需通过按钮的点击形式发起调用（因v1.0.5及以上版本 sdk 更新，取消了授权页和其他按钮，使用了 wx.navigateToMiniProgram 主动拉起授权小程序）。

```
import { startEid } from './mp_ecard_sdk/main';

// 示例方法
goSDK(token) {
    startEid({
        data: {
            token,
        },
        verifyDoneCallback(res) {
            const { token, verifyDone } = res;
            console.log('收到核身完成的res:', res);
            console.log('核身的token是:', token);
            console.log('是否完成核身:', verifyDone);
        },
    });
},
```

startEid() 参数说明：

- `startEid(options)`：进入实名认证页面。
- `options`：Object required 接入方传入的参数。
- `options.data.token`：String required 接入方小程序从接入方服务端获取 EidToken。
- `options.data.needJumpPage`：是否需要中转页，默认为 false（sdk v1.0.7 及以上支持）
- `options.verifyDoneCallback`：Function(res) required 核身完成的回调。res 包含验证成功的 token，是否完成的布尔值标志 verifyDone。请根据 res 返回的结果进行业务处理判断。

3.4 半屏接入指引（可选）

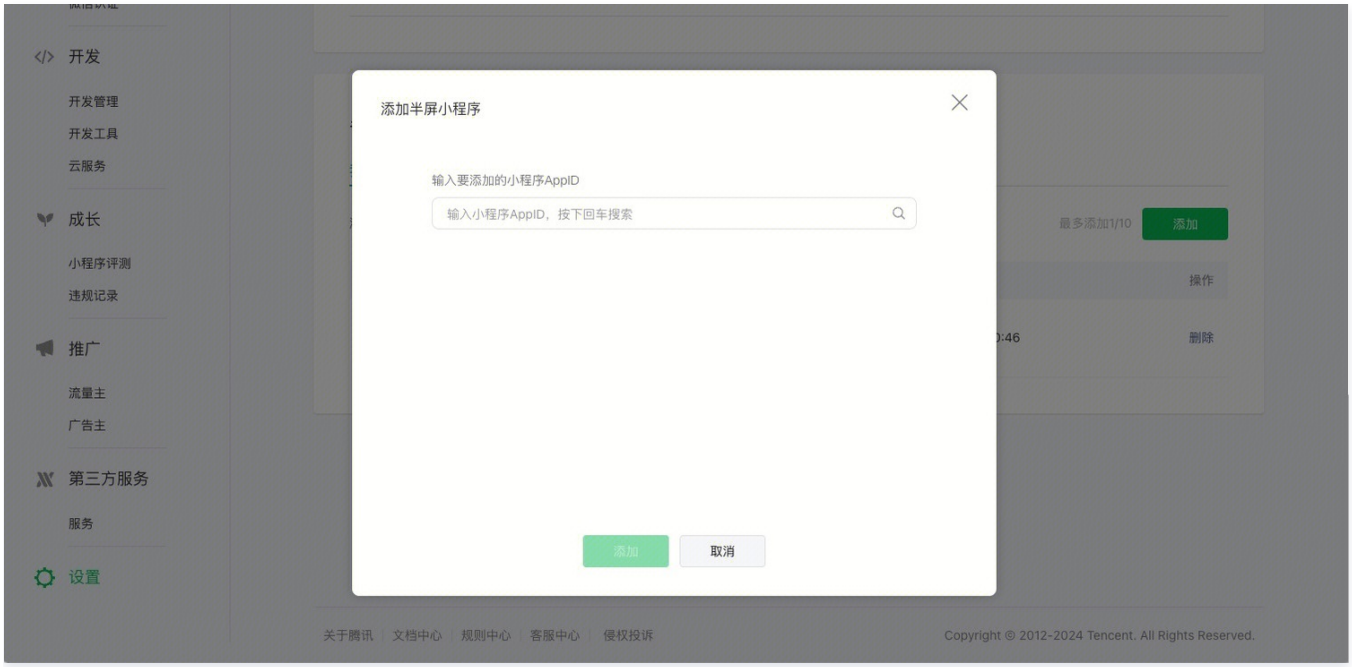
如您需要实现小程序半屏模式，可参考3.4步骤进行配置，如不需要实现半屏模式，则可直接跳到 获取 E证通的核验结果信息 。

小程序的半屏模式可以省去小程序跳转的弹窗步骤，实现如图所示的半屏页面展示，流程体验可以联系 慧眼小助手 获取：

接入准备

申请半屏调用 eID 数字身份小程序

打开小程序微信管理平台后台，选择设置，第三方设置，打开半屏小程序管理，单击**添加**。



输入 eID 数字身份的 AppID（联系 慧眼小助手 获取 AppID 并通过半屏申请）。

> **注意：**
> 每个小程序最多可添加10个半屏打开的小程序。

SDK 接入

1. 请先完成上述步骤3.1 – 3.3的接入，并下载最新版本的 SDK（SDK 1.0.9及以上版本支持半屏接入）。

2. app.json 中添加如下代码，其中 AppID 为 `eID 数字身份` 的 AppID。

```
{
  "embeddedAppIdList": ["AppID"]
}
```

3. SDK 接入时传入允许半屏打开参数。

```
import { startEid } from './mp_ecard_sdk/main';

// 示例方法
goSDK(token) {
    startEid({
        data: {
            token,
            enableEmbedded: true,
            allowFullScreen: false
        },
        verifyDoneCallback(res) {
            const { token, verifyDone } = res;
            console.log('收到核身完成的res:', res);
            console.log('核身的token是:', token);
            console.log('是否完成核身:', verifyDone);
        },
    });
},
```

startEid() 参数说明：

- 半屏接入不支持时会自动降级为全屏打开。
- 因半屏接入是微信新上的能力，少部分机型还存在兼容性问题，请谨慎开启。
- `options.data.token`：String required 接入方小程序从接入方服务端获取 EidToken。
- `options.data.needJumpPage`：是否需要中转页，默认为 false（sdk v1.0.7 及以上支持）
- `options.data.enableEmbedded`：是否支持半屏打开，默认为 false（sdk v1.0.9 及以上支持）
- `options.data.allowFullScreen`：半屏打开后是否支持全屏，默认为 true（sdk v1.0.9 及以上支持）
- `options.verifyDoneCallback`：Function(res) required 核身完成的回调。res 包含验证成功的 token，是否完成的布尔值标志 verifyDone。请根据 res 返回的结果进行业务处理判断。

注意事项

1. 每个小程序最多可添加10个半屏打开的小程序。
2. 半屏接入需向E证通 `eID数字身份` 发送半屏申请，审核通过后才会生效。
3. 半屏接入不支持时会自动降级为全屏打开。
4. 因半屏接入是微信新上的能力，少部分机型还存在兼容性问题，请确保进行过完善的测试后再上线。

4. 获取意愿核身（E证通）核验结果信息

1. 用户完成人脸核身后，会以回调形式返回 EidToken 以及其他信息，接入方小程序将 EidToken 传给接入方的服务端，接入方服务端即可凭借 EidToken 参数调用获取小程序核身结果信息 GetEidResult 接口去获取本次核身的详细信息，最后将核身结果返回给接入方小程序。

> **说明**
> EidToken 作为一次核身流程的标识，有效时间为600秒；完成核身后，可用该标识获取3天内验证结果信息。

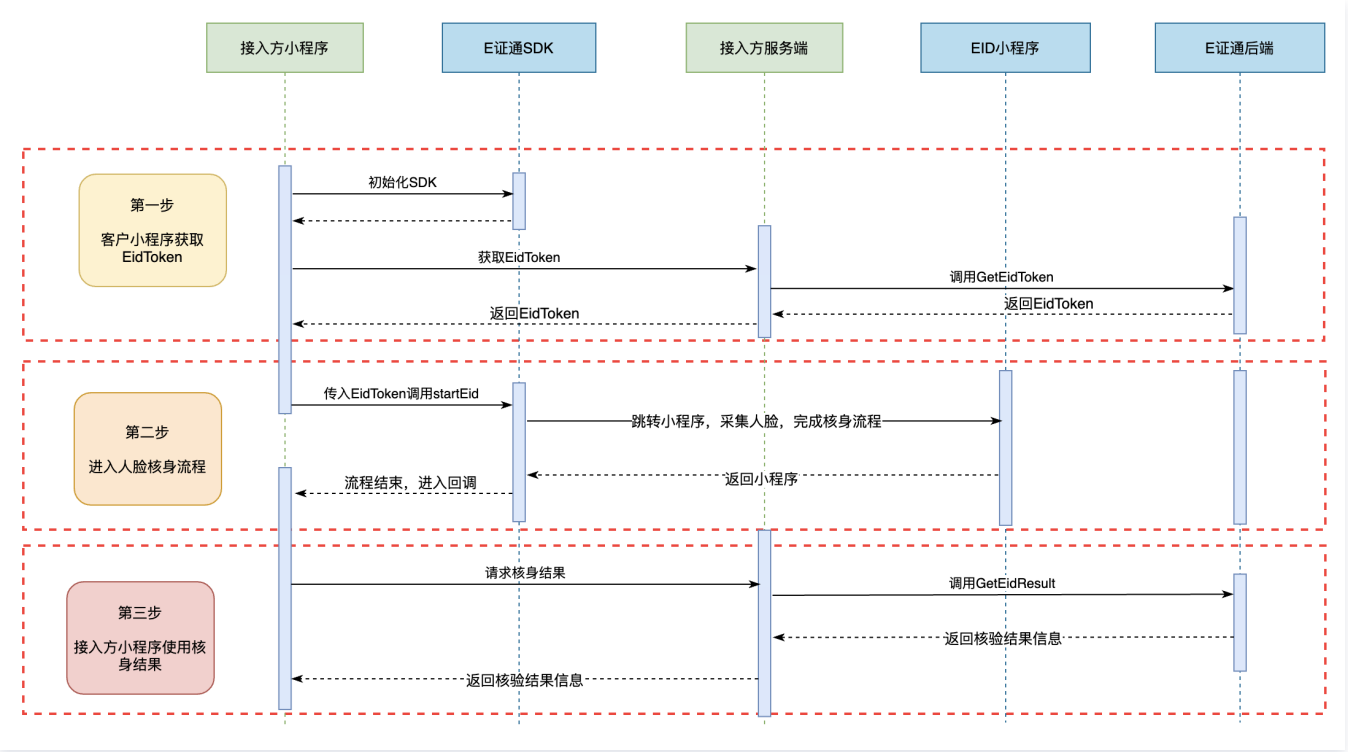
5. 调试 SDK

请在微信开发者工具中使用手机“预览”模式进行调试，请勿使用“真机调试”。

6. 卸载 SDK

删除 mp_ecard_sdk 文件夹。

接入时序图



注意事项

1. 从 eID 数字身份小程序返回接入方小程序。
   当接入方小程序在初始化E证通 SDK 的时候，E证通 SDK 会通过 wx.onAppShow 注册一个监听从 eID 数字身份小程序跳转回接入方小程序的事件，从而根据情况触发接入方传入的核身完成的回调函数。
2. 由于微信的机制，用户在 eID 数字身份小程序跳转回接入方小程序的时候，同时也会触发接入方小程序 app.js 中的 onShow 方法。为了避免冲突，如果接入方小程序在 onShow 中有执行逻辑的话，需要排除掉从 eID 数字身份小程序跳转回接入方小程序这个场景。可通过以下方法实现：

```
// app.js

    onShow(options) {
        const { referrerInfo, scene } = options;
        /* 判断是否从eID数字身份小程序返回 */
        const { appId } = referrerInfo;
        if (scene === 1038 && appId === 'wx0e2cb0b052a91c92') {
            return;
        } else {
            // 执行接入方小程序原本的逻辑
        }
    },
```

3. 基于对个人隐私信息的保护，按照最小必要返回身份信息的要求，E证通服务不再返回姓名和身份证号的明文信息。如因业务需要返回，请查看E证通获取实名信息指引 。

App SDK
合作方初始化上传信息

最近更新时间：2025-02-28 11:30:02

生成签名

准备步骤

- 下载 SDK，请到控制台自助接入列表页获取密码。
- 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 。
- SIGN 类型的 ticket，其有效期为60分钟，且可多次使用。
- 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。

合作方为人脸核身服务生成签名，需要具有以下参数：

参数名	说明	来源
appId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
userId	用户唯一标识	合作方自行分配，需跟用户绑定，32位以内，不包含特殊字符
version	参数值为：1.0.0	-
ticket	合作伙伴服务端获取的 ticket，注意是 SIGN 类型	获取方式见 SIGN ticket 获取
nonce	必须是32位随机数	合作方自行生成

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 appId、userId、version 连同 ticket、nonce 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	IDAXXXXX
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
ticket	XO99QfxIti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPlucaMS

字典排序后的参数为：

```
[1.0.0,IDAXXXXX,XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS,
kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551]
```

拼接后的字符串为：

```
1.0.0IDAXXXXXXO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7Tus
```

```
erID19959248596551
```

计算 SHA1 得到签名：

```
D7606F1741DDCF90757DA924EDCF152A200AC7F0
```

> **说明**
> 该字串就是最终生成的签名（40 位），不区分大小写。

合作方后台上传信息

请求

请求 URL： `https://kyc1.qcloud.com/api/server/getWillFaceId?orderNo=xxx`

> **注意**
> 为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。

请求方法： POST
报文格式： `Content-Type: application/json`
请求参数：

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 appid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
orderNo	订单号，由合作方上传，字母/数字组成的字符串，每次唯一，不能超过32位	String	不能超过32位	是
version	默认参数值为：1.0.0	String	20	是
sign	签名：使用上面生成的签名	String	40	是
nonce	随机数	String	32	是
name	姓名	String	-	使用权威源比对时：姓名+证件号 必须输入 使用自带源比对时：姓名+证件号 可不输入
idNo	证件号码	String	-	使用权威源比对时：姓名+证件号 必须输入 使用自带源比对时：姓名+证件号 可不输入
userId	用户 ID，用户的唯一标识（不能带有特殊字符），需要跟生成签名的 userId 保持一致	String	不能超过32位	是
sourcePhotoStr	比对源照片，**注意：原始图片不能超过500k，且必须为 JPG 或 PNG、BMP 格式** • 参数有值：使用合作伙伴提供的比对源照片进行比对，必须注意是正脸可信照片，照片质量由合作方保证 • 参数为空：根据身份证号 + 姓名使用权威数据源比对	BASE64String	1048576	否，非必填请使用标准的图片转 base64方法，base64编码不可包含换行符，不需要加前缀
sourcePhotoType	比对源照片类型： • 参数值为1 时是：水纹正脸照 • 参数值为 2 时是：高清正脸照 **重要提示：照片上无水波纹的为高清照，请勿传错，否则影响比对准确率。如有疑问，请 联系我们**	String	1	否，提供比对源照片需要传
liveService	应用模式，意愿核身默认上传2	String	32	是
willType	意愿核身类型：产品服务类型为“2”时，需要填写参数	string	32	否

	配置参数： • 0：问答模式 • 1：播报模式 • 2：点头模式 默认配置为0			
willLanguage	意愿核身语言：产品服务类型为“2”时，需要填写 参数配置：0-中文普通话，默认配置为0	string	32	否
willContentList	意愿核身过程中播报文本/问题、用户朗读/回答的文本，当前支持一个播报文本+回答文本	-	List	是
willContentList.id	从下标0开始，问答模式下支持多轮问答，最大支持 5 轮	string	32	是
willContentList.question	系统播报问题文本/问题长度为250个字以内 示例：“ 您好，为确保您本人操作，此次签约全程录音录像。请问您本次业务是本人自愿办理吗？请回答：我确认 ” 注意：多轮问答的总字数之和需要小于或等于500字。	string	32	是
willContentList.answer	用户朗读/回答的文本，用于识别用户朗读/回答的语音与文本是否一致。长度为10个字以内支持多个文本作为识别内容，文本之间用“|”符号分割。示例：“我确认”	string	32	是
speed	播报问题文字语速：默认为1.0倍速 "-1"代表0.8倍，"0"代表1.0倍，"1"代表1.2倍，"1.5"代表1.35倍，"2"代表1.5倍	string	32	否

响应

响应参数：

参数	类型	说明
code	String	0：成功，非0：失败 详情请参见 SaaS 服务错误码
msg	String	请求结果描述
bizSeqNo	String	请求业务流水号
orderNo	String	订单编号
faceId	String	此次唯一标识，调 SDK 时传入

响应示例：

```
{
  "code": 0,
  "msg": "成功",
  "result": {
        "bizSeqNo":"业务流水号",
        "orderNo":"合作方订单号",
        "faceId":"cc1184c3995c71a731357f9812aab988"
  }
}
```

说明：

success：false 无意义，合作伙伴可忽略，无需处理。

faceId 有效期为5分钟，每次进行人脸核身都需要重新获取。

生成 SDK 接口调用步骤使用签名

最近更新时间：2024-09-10 17:56:33

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 。
NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。

- 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。
- 合作方为人脸核身服务生成签名，需要具有以下参数：

参数名	说明	来源
appId	业务流程唯一标识，即 WBappid	参考 获取 WBappid 指引在人脸核身控制台内申请
userId	用户唯一标识	合作方自行分配（和 SDK 里面定义的 userId 保持一致）
version	参数值为：1.0.0	-
ticket	合作伙伴服务端获取的 ticket，注意是 NONCE 类型	获取方式见 NONCE ticket 获取
nonce	必须是32位随机数	合作方自行生成（和 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 nonce（其为字母和数字，登录时也要用到）。
2. 将 appId、userId、version 连同 ticket、nonce 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	IDAXXXXX
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
ticket	XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

字典排序后的参数为：

```
[1.0.0, IDAXXXXX, XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS ,
kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551]
```

拼接后的字符串为：

```
1.0.0IDAXXXXXXO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7Tus
erID19959248596551
```

计算 SHA1得到签名：

```
D7606F1741DDCF90757DA924EDCF152A200AC7F0
```

说明

该字串就是最终生成的签名（40 位），不区分大小写。

Android SDK 接入

最近更新时间：2025-06-06 17:12:52

开发准备

CPU 平台设置

目前 SDK 支持 armeabi-v7a 和 arm64-v8a，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，如下所示：

```
defaultConfig {
    ndk {
          //设置支持的so库框架
          abiFilters 'armeabi-v7a', 'arm64-v8a',
     }
}
```

配置流程

接入配置

意愿核身 SDK（WbCloudFaceLiveSdk、WbCloudFaceWillSdk）最低支持到 **Android API 19: Android 4.4 (KitKat)** ，请在构建项目时注意。
意愿核身 SDK 将以 AAR 文件的形式提供。另外 意愿核身 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 {
    //1. 云刷脸SDK
    implementation(name: 'WbCloudFaceLiveSdk-v4.5.5.0-3b725d95', ext: 'aar')
    //2. 云common SDK
    implementation(name: 'WbCloudNormal-v5.1.3-0f08e6d', ext: 'aar')
//3.意愿性 SDK
implementation(name: 'WbCloudFaceWillSdk-v1.0.0-3b725d95', ext: 'aar')
   }
```

混淆配置

混淆规则已经嵌入 sdk 内，此处不另外提供。

接口调用

1. 标准模式 SDK 接口调用方法
   SDK 代码调用的入口为 `com.tencent.cloud.huiyansdkface.facelight.api.WbCloudFaceVerifySdk` 这个类。

```
public class WbCloudFaceVerifySdk{

/**
* 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 WbCloudFaceVerifySdk getInstance() {
       //    ...
       }
/**
* 在使用SDK前先初始化，传入需要的数据data
* 由 WbCloudFaceVerifyLoginListener返回是否登录SDK成功
```

```
 * 关于传入数据data见后面的说明
 */
public void initWillSdk(Context context,Bundle data,
WbCloudFaceVerifyLoginListener loginListerner){
//     ...
}
    /**
     * 登录成功后，调用此函数拉起sdk页面。
     * 由 FaceVerifyResultForSecureListener返回刷脸结果。
     */
public void startWbFaceVerifySdk(Context ctx,
WbCloudFaceVerifyResultListener listener) {     // ...
}
```

WbCloudFaceVerifySdk.initSdk() 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：

```
String faceId; //此次刷脸用户标识，合作方需要向人脸识别后台拉取获得，详见获取faceId接口
String agreementNo;   //订单号
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
//刷脸类别：分级活体 FaceVerifyStatus.Mode.GRADE
FaceVerifyStatus.Mode verifyMode;
String keyLicence;    //在人脸核身控制台内申请
```

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

2. 登录接口

```
/**
  * 登录回调接口
  */
public interface WbCloudFaceVerifyLoginListener {
    void onLoginSuccess();
    void onLoginFailed(WbFaceError error);
}
```

3. 返回第三方接口

```
/**
  * 退出SDK,返回第三方的回调,同时返回识别结果
  */
public interface WbCloudFaceVerifyResultListener {
    /**
     * @PARAM result 刷脸结果
     */
    void onFinish(WbFaceVerifyResult result);
}
```

识别结果类

WbFaceVerifyResult 是 SDK 用来给合作方传递身份识别结果的对象，在 WbCloudFaceVerifyResultListener 回调中作为参数返回给合作方 App。
WbFaceVerifyResult 对象的各个字段意义如下表所示：

字段名	类型	字段含义	说明
isSuccess	boolean	人脸核身是否成功	True 代表人脸核身对比成功；false 代表人脸核身失败，具体的失败原因请参考 WbFaceError 对象说明

sign	String	签名	供 App 校验人脸核身结果的安全性
liveRate	String	活体检测分数	-
similarity	String	人脸比对分数	“仅活体检测” 类型不提供此分数
userImageString	String	用户人脸核身图片	经过 Base64 编码后的用户人脸核身图片，仅用户成功通过验证时返回
WbFaceError	自定义对象	人脸核身错误	人脸核身成功时为 null
WbFaceWillModeResult	自定义对象	意愿性结果信息	-

WbFaceError 对象说明

WbFaceError 是 SDK 用来给合作方传递人脸核身错误信息的对象，在 WbCloudFaceVerifyLoginListener 回调和 WbFaceVerifyResult 对象中作为参数返回给合作方 App。WbFaceError 对象的各个字段意义如下表所示，各个字段的内容取值详情请参见 SaaS 服务 错误码 。

字段名	类型	字段含义	说明
domain	String	错误发生的阶段	只有当domain=WBFaceErrorDomainCompareServer时表示用户完成了刷脸，可以通过接口去拉取刷脸结果。其他domain表示用户刷脸中途退出或命中了风控逻辑，后端无法查询到刷脸结果
code	String	错误码	-
desc	String	错误描述	如有需求，可以展示给用户
reason	String	错误信息内容	错误的详细实际原因，主要用于定位问题

WbFaceWillModeResult 对象说明

WbFaceWillModeResult 是 SDK 用来给合作方传递意愿性表达综合信息的对象，在 WbCloudFaceVerifyResultListener 中的 WbFaceVerifyResult 对象中作为参数返回给合作方 App。WbFaceWillModeResult 对象的各个字段意义如下表所示，各个字段的内容取值详情请参见 SaaS 服务 错误码 。

字段名	类型	字段含义	说明
faceCode	String	人脸识别结果码	-
faceMsg	String	人脸识别结果信息	-
willCode	String	ASR 结果码	-
willMsg	String	ASR 结果信息	-
videoPath	String	意愿性存证视频存储地址	如果打开了本地存储意愿性视频开关，将在此处返回意愿性视频地址

接口参数说明

InputData 对象说明

InputData 是用来给 SDK 传递一些必须参数所需要使用的对象（WbCloudFaceVerifySdk.initWillSdk() 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刷脸 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（字节）	是否必填
faceId	刷脸 id 号，由合作方向人脸识别后台拉取获得	String	-	是
agreementNo	订单号，合作方订单的唯一标识	String	32	是
openApiAppId	业务流程唯一标识，即 WBappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	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	User Id，每个用户唯一的标识	String	32	是
openApiSign	获取方式请参考 生成SDK接口调用步骤使用签名	String	40	是
verifyMode	刷脸类型：分级模式 FaceVerifyStatus.Mode.GRADE	FaceVerifyStatus.Mode	–	是
keyLicence	在人脸核身控制台内申请	String	以实际申请为准	是

个性化参数设置（可选）

WbCloudFaceVerifySdk.initSdk() 里 Bundle data，除了必须要传的 InputData 对象之外，还可以由合作方为其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。

是否录制意愿性视频存证

SDK 默认不录制视频存证。如果合作方需要视频存证，可以通过该字段进行设置。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(WbCloudFaceContant.INPUT_DATA, inputData);
  //设置是否录制视频进行存证，默认不录制存证。
  //此处设置为录制存证
  data.putBoolean(WbCloudFaceContant.RECORD_WILL_VIDEO, true);
```

是否对录制视频进行检查

> **说明**
> 如果上一节的 **是否录制视频存证** 的设置为录制存证，则目前该字段有效，否则不录制视频的话，也不会对视频进行检查，设置无效。

在 SDK 使用过程中，发现视频录制在性能不太好的手机上可能会报错，导致刷脸中断，影响用户体验。
为了减少因为录制视频原因导致的刷脸中断问题，SDK 默认设置对录制的视频不作检测。如果合作方对刷脸安全有进一步的更加严格的要求，可以考虑打开这一选项。但打开这个字段可能导致某些低性能手机上用户刷脸不能进行，请慎重考虑。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(WbCloudFaceContant.INPUT_DATA, inputData);
  //设置是否对录制的视频进行检测，默认不检测
  //此处设置为检测
  data.putBoolean(WbCloudFaceContant.CHECK_WILL_VIDEO, true);
```

错误码描述

SDK 在登录以及返回人脸服务结果时，如果发生错误或者识别失败会返回 WBFaceError 对象，该对象的字段结构意义见 接口参数说明，其中各个字段的内容如下：

WBFaceErrorDomainParams

Code（错误码）	Description（描述）	Reason（详细实际原因）
11000	传入参数为空	传入的xx为空
11001	传入的 keyLicence	不可用 传入的 keyLicence 不可用
11002	报文加解密失败	报文加解密失败

WBFaceErrorDomainLoginNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
21100	网络异常	登录时网络异常（请求未到达后台）
21200	网络异常	登录时后台返回参数有误（请求到达后台）

WBFaceErrorDomainLoginServer

Code（错误码）	Description（描述）	Reason（详细实际原因）
其他错误码	透传后台错误码	例如签名问题等

WBFaceErrorDomainGetInfoNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
31100	网络异常	获取活体类型/光线/意愿性表达资源，网络异常（请求未到达后台）
31200	网络异常	获取活体类型/光线/意愿性表达资源，后台返回参数有误（请求到达后台）

WBFaceErrorDomainNativeProcess

Code（错误码）	Description（描述）	Reason（详细实际原因）
41000	用户取消	回到后台/单击 home/左上角/上传时左上角取消
41001	无法获取唇语数据	获取数字活体的数字有问题
41002	权限异常，未获取权限	相机
41003	相机运行中出错	–
41004	视频录制中出错	不能存/启动失败/结束失败
41005	请勿晃动人脸，保持姿势	未获取到最佳图片
41006	视频大小不满足要求	视频大小不满足要求
41007	超时	预检测/动作活体
41008	检测中人脸移出框外	活体/数字/反光
41009	光线活体本地错误	–
41010	风险控制超出次数	用户重试太多次
41011	没有检测到读数声音	数字活体过程中没有发声
41012	初始化模型失败，请重试	初始化算法模型失败
41013	初始化 sdk 异常	WbCloudFaceVerifySdk 未被初始化
41014	简单模式本地加密失败	编码转换异常/加解密编码失败
41101	音频录制中出错	意愿性录音失败
41102	没有检测到麦克风声音	意愿性检测音量过低
41103	播报音频文件加载失败	意愿性播放音频失败

WBFaceErrorDomainCompareNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
51100	网络异常	对比时，网络异常（请求未到达后台）
51200	网络异常	对比时，后台返回参数有误（请求到达后台）

WBFaceErrorDomainCompareServer

Code（错误码）	Description（描述）	Reason（详细实际原因）
其他错误码	透传后台错误码	–

接入示例

权威库网纹图片比对、自带对比源对比接入示例：

```
# 在MainActivity中单击某个按钮的代码逻辑：
//先填好数据
Bundle data = new Bundle();
WbCloudFaceVerifySdk.InputData inputData = new WbCloudFaceVerifySdk.InputData(
                faceId,
                agreementNo,
                openApiAppId,
                openApiAppVersion,
                openApiNonce,
                userId,
                userSign,
                verifyMode,
                keyLicence);
  data.putSerializable(WbCloudFaceContant.INPUT_DATA, inputData);

//设置选择的比对类型  默认为权威库网纹图片对比
//此处设置权威库网纹图片对比
  data.putString(WbCloudFaceContant.COMPARE_TYPE, WbCloudFaceContant.ID_CRAD);

//初始化 SDK，得到是否登录 SDK 成功的结果,由 WbCloudFaceVerifyLoginListener  返回登录结果
//【特别注意】建议对拉起人脸识别按钮做防止重复点击的操作
//避免用户快速点击导致二次登录，二次拉起刷脸等操作引起问题
WbCloudFaceVerifySdk.getInstance().initWillSdk(DemoActivity.this, data, new WbCloudFaceVerifyLoginListener() {
        @Override
        public void onLoginSuccess() {
                //登录成功，拉起 sdk 页面,由 FaceVerifyResultListener   返回刷脸结果
                WbCloudFaceVerifySdk.getInstance().startWbFaceVerifySdk(DemoActivity.this, new
WbCloudFaceVerifyResultListener() {
                        @Override
                        public void onFinish(WbFaceVerifyResult result) {
                                if (result != null) {
                                        if (result.isSuccess()) {
                                                Log.d(TAG, "刷脸成功！");
                                        } else {
                                                Log.d(TAG, "刷脸失败！");
                                        }
                                }
                                //刷脸结束后，及时释放资源
                                WbCloudFaceVerifySdk.getInstance().release();
                        }
                });
        }
        @Override
        public void onLoginFailed(WbFaceError error) {
                Log.d(TAG, "登录失败！");
                //刷脸结束后，及时释放资源
                WbCloudFaceVerifySdk.getInstance().release();
        }
});
```

详细接入代码，请参考 SDK 附的 KnowYourCustomerDemo 工程。

iOS SDK 接入

最近更新时间：2025-02-28 11:30:03

> **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。

配置流程

使用 Cocoapod 集成

SDK 最低支持到 iOS11.0，请在构建项目时候注意，目前仅支持 Xcode11.0及更高版本编译。
以下为接入配置的步骤：

1. 将 TencentCloudHuiyanSDKFace_framework 文件夹拷贝到自己项目的 podfile 文件所在的同一目录。
2. 在 podfile 使用如下配置(请注意 target 后面内容根据自己项目配置，参考 demo)：

```
target 'WBCloudReflectionFaceVerify-Demo' do
pod 'TencentCloudHuiyanSDKFace_framework', :path=>'./ TencentCloudHuiyanSDKFace_framework'
end
```

3. 使用 pod install 命令
4. SDK 需要使用相机以及麦克风权限，请在 info.plist 中添加：

```
Privacy - Camera Usage Description
Privacy - Microphone Usage Description
```

直接引用 framework

SDK 最低支持到 iOS11.0，请在构建项目时候注意。
以下为接入配置的步骤：

1. 引用以下资源文件到项目：

```
TencentCloudHuiyanSDKFace.framework
TuringShieldCamRisk.framework
YTCommonLiveness.framework
YTCv.framework
YTFaceAlignmentTinyLiveness.framework
YTFaceDetectorLiveness.framework
YTFaceLiveReflect.framework
YTFaceTrackerLiveness.framework
YTPoseDetector.framework
YtSDKKitFrameworkTool.framework
tnnliveness.framework
TencentCloudHuiyanSDKFace.bundle
TencentCloudHuiyanSDKWill.bundle
face-tracker-v003.bundle
```

2. SDK 依赖以下系统框架,需要在 **BuildPhases > Link Binary With Libraries** 中添加，可以参考 Demo，具体依赖的系统库如下：

```
UIKit.framewrok
AVFoundation.framework
CoreVideo.framework
Security.framework
SystemConfiguration.framework
CoreMedia.framework
CoreTelephony.framework
ImageIO.framework
```

```
MediaPlayer.framework
Accelerate.framework
WebKit.framework
libc++.tbd
libz.tbd
videoToolbox.framework
```

3. SDK 需要使用相机权限，请在 info.plist 中添加：

```
Privacy - Camera Usage Description
Privacy - Microphone Usage Description
```

4. 需要在 **BuildSettings > Other Linker Flags** 中设置：`-ObjC` 。

接口调用

SDK 接口调用方法

SDK 的功能通过 WBFaceVerifyCustomerService 这个类的方法进行调用 ，其中 SDK 中使用的 nonce，sign 等重要信息，需要合作方从自己后台拉取，并且两者不能缓存，只能使用一次即失效，详细接口说明如下，其他的操作请参考 Demo 中的登录接口的参数说明：
版本号及宏定义说明：

```
#import <UIKit/UIKit.h>
#ifndef WBFaceVerifyConst_h
#define WBFaceVerifyConst_h
#define WBCloudReflectionFaceVerifyVersion

UIKIT_EXTERN NSString *const WBCloudFaceVerifySDKVersion;

#endif /* WBFaceVerifyConst_h */
```

入口方法说明：
NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。

```
（1）faceID +活体检测+人脸比对服务（身份证的网纹照片进行对比）
/*
意愿性SDK入口，注意传入的faceId不能为空

@param userid 用户唯一标识，由合作方自行定义（具体要求，参考接入文档）
@param nonce  满足接入要求的32位随机数（具体要求，参考接入文档）
@param sign 满足接入要求的40位签名值（具体要求，参考接入文档）
@param appid 腾讯服务分配的appid
@param orderNo 每次人脸身份认证请求的唯一订单号：建议为32位字符串(不超过32位)
@param apiVersion 后台api接口版本号(不是SDK的版本号)，默认请填写@"1.0.0"
@param licence 腾讯给合作方派发的前端使用的licence(该licence同app当前使用的bundle id绑定)
@param faceId 合作方必须要先获取*增强级*faceId，再送入sdk，不允许为空(参考接入文档)
@param sdkConfig SDK基础配置项目
@param success 服务登录成功回调，登录成功以后开始进行活体和检测服务
@param failure 服务登录失败回调，具体参考错误码文档(参考接入文档)
*/
-(void)initWillSDKWithUserId:(NSString *)userid
                       nonce:(NSString *)nonce
                        sign:(NSString *)sign
                       appid:(NSString *)appid
                     orderNo:(NSString *)orderNo
                  apiVersion:(NSString *)apiVersion
                     licence:(NSString *)licence
                      faceId:(NSString *)faceId
                   sdkConfig:(WBFaceVerifySDKConfig *)sdkConfig
```

```
                success:(void (^)())success
                failure:(void (^)(WBFaceError * _Nonnull))failure;
/**
 以上一次的登录结果拉起刷脸页面，必须先登录再拉起刷脸页面

 @return 拉起是否成功
 */
- (BOOL)startWbFaceVerifySdk;
```

个性化参数设置：

SDK 登录接口 initSDK 方法中需要传入 WBFaceVerifySDKConfig 字段，通过该对象可以配置 SDK 中其他基础配置。

```
/**
 人脸识别SDK 基础配置类
 */
@interface WBFaceVerifySDKConfig : NSObject


#pragma mark - common
/**
 sdk中拉起人脸活体识别界面中使用UIWindow时的windowLevel配置，默认配置是1 + UIWindowLevelNormal

 如果接入放app中有其他自定义UIWindow， 为了防止界面覆盖，可以酌情设置该参数
 */
@property (nonatomic, assign) NSUInteger windowLevel;

/**
 人脸识别服务是否进行通过录像， 从而进行视频存证

 default: NO
 */
@property (nonatomic, assign) BOOL recordVideo;

/**
 是否由SDK内部处理sdk网络请求的cookie

 默认值: YES
 */
@property (nonatomic, assign) BOOL manualCookie;

/**
 是否静音
 默认值: YES
 */
@property (nonatomic, assign) BOOL mute;


/*
 送入自定义提示文案的位置
 默认: WBFaceCustomTipsLoc_Bottom
 */
@property (nonatomic, assign) WBFaceCustomTipsLoc tipsLoc;

/*
 检测过程中展示的文案
 默认为空
 */
@property (nonatomic, copy) NSString *customTipsInDetect;

/*
 上传过程中展示的文案
 默认为空
```

```
 */
@property (nonatomic, copy) NSString *customTipsInUpload;

/*
 底部提示文案，长度不超过70字
 */
@property (nonatomic, copy) NSString *bottomCustomTips;

/*
 退出二次确认UI配置
 */
@property (nonatomic, copy) NSString *exitAlertTitle; //标题
@property (nonatomic, copy) NSString *exitAlertMessage; //内容
@property (nonatomic, copy) NSString *exitAlertRight; //右边按钮
@property (nonatomic, copy) NSString *exitAlertLeft; //左边按钮

/*
 如果有使用苹果分屏模式（UIWindowScene），打开此开关
 Xcode11新建工程有使用Scene，可以参考资料自行调整
 默认为NO
 */
@property (nonatomic, assign) BOOL useWindowSecene;

/*
意愿性表达视频录制参数
*/
@property (nonatomic, assign) BOOL recordWillVideo;
@property (nonatomic, assign) BOOL checkWillVideo;

/**
 默认sdk配置
 */
+(instancetype)sdkConfig;

@end
```

核身结果返回

IOS SDK 中，意愿性结果返回给接入方有两种方式，两种结果回调都在 Main Thread 完成：
您可以信任前端 SDK 意愿性结果，返回中的 isSuccess 字段 YES 代表意愿性流程成功；NO 代表意愿性流程失败。意愿性 SDK 和核身后台服务之间通信采用加密方式，可以有效防止结果篡改。

1. 通过实现 WBFaceVerifyCustomerServiceDelegate 的 WBFaceVerifyCustomerServiceDelegate 方法,通过该方法返回WBFaceVerifyResult 对象。
2. 无需实现 delegate 方法，通过注册 WBFaceVerifyCustomerServiceDidFinishedNotification 通知，在接受到该通知时，进行结果处理。

WBFaceVerifyCustomerServiceDidFinishedNotification 通知中，通过获取该通知的 userInfo，获取字典中 key 为faceVerifyResult 对应的 value 对象就是 WBFaceVerifyResult。

WBFaceVerifyResult 对象说明

字段名	类型	字段含义	说明
isSuccess	BOOL	人脸核身是否成功	YES 代表认证成功，NO 代表认证失败，具体原因参考 error 描述
sign	NSString	签名	供 App 校验意愿性结果的安全性
liveRate	NSString	活体检测分数	–
similarity	NSString	人脸比对分数	“仅活体检测” 类型不提供此分数
userImageString	NSString	用户人脸核身图片	经过 Base64编码后的用户人脸核身图片，用来做比对认证的最优图
WbFaceError	自定义对象	人脸核身错误	意愿性成功时为 nil

WBWillModeResult 说明

字段名	类型	字段含义	说明
faceCode	NSString	人脸识别结果码	-
faceMsg	NSString	人脸识别结果信息	-
willCode	NSString	ASR 结果码	-
willMsg	NSString	ASR 结果信息	-
videoURL	NSURL	意愿性存证视频存储地址	如果打开了本地存储意愿性视频开关,将在此处返回意愿性视频地址

WBFaceError 对象说明

字段名	类型	字段含义	说明
domain	NSString	错误发生的阶段	只有当 domain=WBFaceErrorDomainCompareServer 时表示用户完成了刷脸，可以通过接口去拉取刷脸结果。其他 domain 表示用户刷脸中途退出或命中了风控逻辑，后端无法查询到刷脸结果
code	NSString	错误码	-
desc	NSString	错误描述	如有需求，可以展示给用户
reason	NSString	错误信息内容	错误的详细实际原因，主要用于定位问题

WBFaceVerifyResult 说明

```
@interface WBFaceWillModeResult : NSObject
/*
 核身结果的对应错误码
 */
@property (nonatomic, copy, readonly) NSString *faceCode;
/*
 核身结果的对应错误描述
 */
@property (nonatomic, copy, readonly) NSString *faceMsg;
/*
 意愿性结果的对应错误码
 */
@property (nonatomic, copy, readonly) NSString *willCode;
/*
 意愿性结果的对应错误描述
 */
@property (nonatomic, copy, readonly) NSString *willMsg;

@end
/**
 人脸服务返回结果对象
 */
@interface WBFaceVerifyResult : NSObject
/**
 人脸比对结果是否通过:

 YES: 表示人脸服务通过
 NO:  表示人脸服务不通过lo
 */
@property (nonatomic, assign, readonly) BOOL isSuccess;

/**
 结果对应的订单号
 */
@property (nonatomic, copy, readonly) NSString *orderNo;
```

```
/**
 isSuccess == YES 时, sign有值,可以去后台拉取本次人脸服务的照片,视频存证
 isSuccess == NO  时, sign 无意义
 */
@property (nonatomic, copy, readonly) NSString * sign;
/**
 活体检测服务得分

 isSuccess == YES 时, liveRate 有值:
        1. liveRate 可能是 @"分数为空", 这种情况具体咨询合作方
        2. float类型的字符串, 请调用 [liveRate floatValue] 获取具体分数
 isSuccess == NO  时, liveRate 无意义
 */
@property (nonatomic, copy, readonly) NSString * liveRate;

/**
 人脸比对服务得分

 isSuccess == YES 时, similarity 有值:
         1. similarity 可能是 @"分数为空", 这种情况具体咨询合作方
         2. float类型的字符串, 请调用 [similarity floatValue] 获取具体分数
 isSuccess == NO  时, similarity 无意义
 */
@property (nonatomic, copy, readonly) NSString * similarity;

/**
 人脸比对图片之一

 isSuccess == YES 时, 该属性是上送比对图片之一UIImage的base64编码字符串(图片格式是jpg)

 isSuccess == NO  时, 如果是SDK返回的错误,该属性为nil,如果是后端返回的错误,该属性是上送比对图片之一UIImage的base64编码字符
串(图片格式是jpg)
 */
@property (nonatomic, copy, readonly) NSString * userImageString;

/**
 isSuccess == YES 时候, error 无意义
 isSuccess == NO  时, error中存储的具体错误信息,参考 WBFaceError.h
 */
@property (nonatomic, strong, readonly) WBFaceError * error;

#pragma mark - 意愿性SDK返回参数
@property (nonatomic, strong, readonly) WBFaceWillModeResult *willModeResult;

#pragma mark -

-(NSString *)description;
@end
```

错误码描述

SDK 在登录以及返回人脸服务结果时，如果发生错误或者识别失败会返回 WBFaceError 对象，该对象的结构如下, 并且在我们判断具体错误时, 需要先根据 domain 字段判断错误发生在 sdk 服务运行中的位置, 然后根据 code 判断具体的错误：

```
*/
NS_ASSUME_NONNULL_BEGIN
/*
 错误domain划分成两类:

 出现在登录时, 通过调用startXXXX 方法的failure block进行回调返回:
```

```
 WBFaceErrorDomainInputParams, WBFaceErrorDomainLoginNetwork, WBFaceErrorDomainLoginServer

 人脸服务结果返回(封装在WBFaceVerifyResult中):
 WBFaceErrorDomainGetInfo, WBFaceErrorDomainNativeProcess, WBFaceErrorDomainCompareNetwork,
WBFaceErrorDomainCompareServer
 */

/* 登录时传入参数有误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainInputParams;
/* 登录时网络请求错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainLoginNetwork;
/* 登录时后台拒绝登录，具体参考后台word版本错误码，这里直接透传 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainLoginServer;
/* 拉取有效信息时候网络错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainGetInfo;
/* native本地在活体检测中，某些原因导致错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainNativeProcess;
/* 上送后台比对时,网络错误 */
UIKIT_EXTERN NSString *const WBFaceErrorDomainCompareNetwork;
/* 后台比对完成,返回比对结果的错误原因*/
UIKIT_EXTERN NSString *const WBFaceErrorDomainCompareServer;

@interface WBFaceError: NSObject
/**
 错误发生的地方，具体的发生地方由上面定义的 WBFaceErrorDomainXXXX 指定
 */
@property (nonatomic, readonly, copy) NSString *domain;
/**
 每个domain中有相应的错误代码，具体的错误代码见
 */
@property (nonatomic, readonly, assign) NSInteger code; //错误代码
@property (nonatomic, readonly, copy) NSString *desc; //获取本地化描述
@property (nonatomic, readonly, copy) NSString *reason; // 错误出现的真实原因原因
@property (nonatomic, readonly, copy) NSDictionary * _Nullable otherInfo;// 预留接口
@end
```

以下是 Domain 为不同值时，对应的 code 和错误描述信息：

若 Domain 为 WBFaceErrorDomainInputParams

Code（错误码）	Description（描述）	Reason（详细实际原因）
12000	传入参数为空	传入的 xx 为空
12001	传入的 keyLicence 不可用	传入的 keyLicence 不可用
12002	身份证格式不正确	身份证格式不正确
12003	使用自带对比源，传入参数错误，非 base64	传入的 srcPhotoString 不是base64
12004	使用自带对比源，传入参数错误，超过 1 MB	传入的 srcPhotoString 超过 1 MB
12005	SDK 资源引入版本不匹配	没有引入资源包或者引入的资源包版本与当前 SDK 版本不匹配
12006	订单号不能为 0 或者超过 32 位	-
12007	nonce 字符串位数不为32 位	-
12009	定制化SDK生成参数失败	-
12010	定制化参数校验错误	-

Domain 为：WBFaceErrorDomainLoginNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）

22100	网络异常	登录时网络异常（请求未到达后台）
22200	网络异常	登录时后台返回参数有误（请求到达后台）
22003	请勿晃动人脸，保持姿势	未获取到最佳图片

Domain 为：WBFaceErrorDomainLoginServer

Code（错误码）	Description（描述）	Reason（详细实际原因）
其他错误码	透传后台错误码	例如签名问题等

Domain 为：WBFaceErrorDomainGetInfo

Code（错误码）	Description（描述）	Reason（详细实际原因）
32100	网络异常	获取数字/活体类型/光线阈值，网络异常(请求未到达后台)
32200	网络异常	获取数字/活体类型/光线阈值，后台返回参数有误（请求到达后台）

Domain 为：WBFaceErrorDomainNativeProcess

Code（错误码）	Description（描述）	Reason（详细实际原因）
42000	用户取消	回到后台/单击 home/左上角/上传时左上角取消
42001	网络环境不满足认证需求	无网络/2g 网络
42002	权限异常，未获取权限	相机/麦克风/read phone/external/storage
42003	相机运行中出错	–
42004	视频录制中出错	不能存/启动失败/结束失败
42005	请勿晃动人脸，保持姿势	未获取到最佳图片
42006	视频大小不满足要求	视频大小不满足要求
42007	超时	预检测/动作活体
42008	检测中人脸移出框外	活体/数字/反光
42009	光线活体本地错误	–
42010	风险控制超出次数	用户重试太多次
42011	没有检测到读数声音	数字活体过程中没有发声
42012	模型初始化失败	–
42015	报文解密失败	请求报文解密失败
42101	音频录制中出错	–
42102	没有检测到麦克风声音	–
42103	播报音频文件加载失败	–
42104	麦克风被占用，音频播报失败	–
42105	视频录制中出错	
42106	音量过低，用户主动退出	–

Domain 为：WBFaceErrorDomainCompareNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
52100	网络异常	对比时，网络异常（请求未到达后台）

52200	网络异常	对比时，后台返回参数有误（请求到达后台）

Domain 为：WBFaceErrorDomainCompareServer

Code（错误码）	Description（描述）	Reason（详细实际原因）
其他错误码	透传后台错误码	-

Harmony NEXT 意愿核身接入

最近更新时间：2024-12-06 18:12:12

开发准备

权限

SDK 需要用到相机权限、麦克风权限和网络权限，如下示例：

```
"requestPermissions": [
  {
    "name": "ohos.permission.INTERNET",
  },
  {
    "name": "ohos.permission.GET_NETWORK_INFO"
  },
{
    "name": "ohos.permission.GET_WIFI_INFO"
},
  {
    "name": "ohos.permission.CAMERA",
    "reason": "face verify",
    "usedScene": {
      "abilities": [
        "EntryAbility",
      ],
      "when": "inuse"
    }
  },
  {
    "name": "ohos.permission.MICROPHONE",
    "usedScene": {
      "abilities": [
        "EntryAbility",
      ],
      "when": "inuse"
    },
    "reason": "face verify"
  }
]
```

CPU平台配置

当前 Harmony NEXT 手机都是64位手机，目前 SDK 只支持 armeabi-v8a 平台。

配置流程

> **注意：**
> 以下文章出现 “刷脸” 一词均可以表示为 “意愿核身”。

接入配置

SDK（WbCloudFaceVerifySdk-xx.har, 具体版本号以SDK交付件为准）最低支持到 5.0.0(12)，请在构建项目时注意。
刷脸 SDK 将以 HAR 文件的形式提供。
将提供的 HAR 文件添加到工程的libs目录下，并且在 **oh-package.json5** 中添加下面的配置，**请注意大小写：**

```
  "dependencies": {
    "wbcloudfaceverifysdk": "file:../libs/WbCloudFaceVerifySdk-xxx.har"

  }
```

接口调用

本章将展示 SDK 核心接口的调用方法。具体的代码示例参考交付 Demo(demo/entry/src/main/ets/pages/DemoPage.ets)。

> ⚠ **注意：**
> 以下文章出现 “刷脸” 一词均可以表示为 “意愿核身”。

SDK 接口调用方法

SDK 代码调用的入口为：**WbCloudFaceVerifySdk** 这个类。

```
export class WbCloudFaceVerifySdk {

  /**
   * 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 getInstance(): WbCloudFaceVerifySdk {
    …
  }

  /**
   * 在使用SDK前先初始化，传入需要的数据WbFaceVerifyConfig
   * 由 WbCloudFaceVerifyLoginCallback返回是否登录SDK成功
   * 关于传入WbFaceVerifyConfig见后面的说明
   */
  public initWillSdk(context: Context, config: WbFaceVerifyConfig, loginCallback:
WbCloudFaceVerifyLoginCallback) {
    …
  }

  /**
   * 登录成功后，调用此函数拉起sdk页面。
   * 由 WbCloudFaceVerifyResultCallback返回刷脸结果。
   */
  public startWbFaceVerifySdk(context: Context, resultCallback: WbCloudFaceVerifyResultCallback) {
    …
  }

  /**
* 拿到刷脸结果后，释放资源， 防止内存泄漏
   * 【注意】请务必在拿到刷脸结果后释放资源，否则可能会发生丢失回调的情况！
   */
  public release() {
    …
  }
}
```

WbCloudFaceVerifySdk.initSdk() 的第二个参数 config: WbFaceVerifyConfig 用来传递 SDK 初始化必需数据和 SDK 配置项参数。

```
export class WbFaceVerifyConfig {
    //sdk初始化必需数据
  public inputData: InputData;
  //是否开启日志，默认不打开sdk日志，
  //【注意】上线请务必关闭sdk日志！
  public isEnableLog: boolean = false;
}
```

必须传递的参数包括（参考要求详见本页接口参数说明的描述）：

```
//这些都是InputData对象里的字段，是需要传入的数据信息
export class InputData {
//此次刷脸用户标识，合作方需要向意愿识别后台拉取获得，详见获取faceId接口
public faceId: string;
//订单号
public orderNo: string;
//APP_ID
public appId: string;
  //openapi Version
public version: string;
//32位随机字符串
public nonce: string;
//User id
public userId: string;
//签名信息
public sign: string;
////在意愿核身控制台内申请
public licence: string;

}
```

关于接口调用的示例可参考 接入示例 。

接口参数说明

InputData 对象说明

InputData 是用来给 SDK 传递必须参数所需要使用的对象，合作方需要往里塞入SDK 需要的一些数据以便启动刷脸 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（字节）	是否必填
faceId	刷脸id号，由合作方向意愿识别后台拉取获得	String	-	是
orderNo	订单号，合作方订单的唯一标识	String	32	是
appId	业务流程唯一标识，即wbappid，可参考 获取WBappid 指引在意愿核身控制台内申请	String	8	是
version	接口版本号，默认填1.0.0	String	20	是
nonce	32 位随机字符串，每次请求需要的一次性 nonce	String	32	是
userId	User Id，每个用户唯一的标识	String	30	是
sign	获取方式请参考 生成SDK接口调用步骤使用签名	String	40	是
licence	在意愿核身控制台内申请	String	以实际申请为准	是

个性化参数设置（可选）
WbCloudFaceVerifySdk.initSdk() 里 WbFaceVerifyConfig，除了必须要传的 InputData 对象之外，还可以由合作方为其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。(暂不支持，后续 SDK 版本将逐步更新)

核身结果返回

SDK 结果分别由以下两个回调返回：
如果您只需要获取刷脸结果，不需要拉取视频和照片，您可以信任前端 SDK 核身结果，返回中isSuccess字段True代表意愿核身对比成功；false代表意愿核身失败。核身SDK和核身后台服务之间通信采用加密方式，可以有效防止结果篡改。

```
/**
  * 登录回调接口   返回登录 sdk 是否成功
  */
```

```
export interface WbCloudFaceVerifyLoginCallback {
  onLoginSuccess: () => void;
  onLoginFail: (error: WbFaceError) => void;
}
/**
  * 刷脸结果回调接口
  */
export interface WbCloudFaceVerifyResultCallback {
  onFinish: (result: WbFaceVerifyResult) => void;
}
```

WBFaceVerifyResult 对象说明

WbFaceVerifyResult 是 SDK 用来给合作方传递身份识别结果的对象，在WbCloudFaceVerifyResultListener回调中作为参数返回给合作方 App。WbFaceVerifyResult 对象的各个字段意义如下表所示：

字段名	类型	字段含义	说明
isSuccess	boolean	意愿核身是否成功	True代表意愿核身对比成功；false代表意愿核身失败，具体的失败原因请参考WbFaceError对象说明
sign	String	签名	供 App 校验意愿核身结果的安全性
liveRate	String	活体检测分数	-
similarity	String	意愿比对分数	“仅活体检测” 类型不提供此分数
WbFaceError	自定义对象	意愿核身错误	意愿核身成功时为 null

WbFaceError 对象说明

WbFaceError 是 SDK 用来给合作方传递意愿核身错误信息的对象，在WbCloudFaceVerifyLoginListener回调和 WbFaceVerifyResult 对象中作为参数返回给合作方 App。WbFaceError 对象的各个字段意义如下表所示，各个字段的内容取值详情请参见 SaaS 服务 错误码 。

字段名	类型	字段含义	说明
domain	String	错误发生的阶段	只有当domain=WBFaceErrorDomainCompareServer时表示用户完成了刷脸，可以通过接口去拉取刷脸结果。其他domain表示用户刷脸中途退出或命中了风控逻辑，后端无法查询到刷脸结果
code	String	错误码	-
desc	String	错误描述	如有需求，可以展示给用户
reason	String	错误信息内容	错误的详细实际原因，主要用于定位问题

错误码描述

> ⚠ **注意：**
> 以下文章出现 “刷脸” 一词均可以表示为 “意愿核身”。

SDK 在登录以及返回意愿服务结果时，如果发生错误或者识别失败会返回 WBFaceError 对象，该对象的字段结构意义见 接口参数说明，其中各个字段的内容如下：

WBFaceErrorDomainParams

Code（错误码）	Description（描述）	Reason（详细实际原因）
13000	传入参数为空	传入的 xx为空
13001	传入的keyLicence不可用	传入的keyLicence不可用
13002	报文加解密失败	报文加解密失败

WBFaceErrorDomainLoginNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
23100	网络异常	登录时网络异常（请求未到达后台）
23200	网络异常	登录时后台返回参数有误（请求到达后台）

WBFaceErrorDomainLoginServer

Code（错误码）	Description（描述）	Reason（详细实际原因）
其他错误码	透传后台错误码	例如签名问题等等

WBFaceErrorDomainGetInfoNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
33100	网络异常	获取活体类型/光线资源，网络异常（请求未到达后台）
33200	网络异常	获取活体类型/光线资源，后台返回参数有误（请求到达后台）

WBFaceErrorDomainNativeProcess

Code（错误码）	Description（描述）	Reason（详细实际原因）
43000	用户取消	回到后台/单击home/左上角/上传时左上角取消
43001	无法获取唇语数据	获取数字活体的数字有问题
43002	权限异常，未获取权限	相机
43003	相机运行中出错	-
43004	视频录制中出错	不能存/启动失败/结束失败
43005	请勿晃动人脸,保持姿势	未获取到最佳图片
43006	视频大小不满足要求	视频大小不满足要求
43007	超时	预检测/动作活体
43008	检测中人脸移出框外	活体/数字/反光
43009	光线活体本地错误	-
43010	风险控制超出次数	用户重试太多次
43011	没有检测到读数声音	数字活体过程中没有发声
43012	初始化模型失败，请重试	初始化算法模型失败
43013	初始化sdk异常	WbCloudFaceVerifySdk未被初始化
43014	简单模式本地加密失败	编码转换异常/加解密编码失败

WBFaceErrorDomainCompareNetwork

Code（错误码）	Description（描述）	Reason（详细实际原因）
53100	网络异常	对比时，网络异常（请求未到达后台）
53200	网络异常	对比时，后台返回参数有误（请求到达后台）

WBFaceErrorDomainCompareServer

Code（错误码）	Description（描述）	Reason（详细实际原因）
其他错误码	透传后台错误码	-

接入示例

权威库网纹图片比对、自带对比源对比接入示例：

```
# 在DemoPage中单击某个按钮的代码逻辑：
//先填好数据
let inputData = new InputData(
orderNo,
this.appId,
'1.0.0',
this.nonce,
this.userId,
sign,
this.licence,
faceId
)

//设置必需参数
let wbFaceVerifyConfig = new WbFaceVerifyConfig(inputData);
//是否打开sdk日志开关
wbFaceVerifyConfig.isEnableLog = true;

WbCloudFaceVerifySdk.getInstance().initWillSdk(getContext(), wbFaceVerifyConfig, {
onLoginSuccess: () => {
DemoLog.i(this.TAG, `onLoginSuccess:`)
WbCloudFaceVerifySdk.getInstance().startWbFaceVerifySdk(getContext(), {
onFinish: (_result: WbFaceVerifyResult) => {
DemoLog.i(this.TAG, `WbCloudFaceVerifySdk onFinish`)
//todo 处理刷脸结果
……
//处理完后释放sdk
//【特别注意】请在拿到sdk结果后对sdk进行释放，不要在页面结束时释放，避免未能获取刷脸回调结果的情况
WbCloudFaceVerifySdk.getInstance().release();
}
})
},
onLoginFail: (error: WbFaceError) => {
DemoLog.e(this.TAG, `onLoginFailed:JSON.stringify(error)`)
//todo 处理登录错误逻辑
……
//【特别注意】请在拿到sdk结果后对sdk进行释放，不要在页面结束时释放，避免未能获取刷脸回调结果的情况
WbCloudFaceVerifySdk.getInstance().release();
}
})
```

意愿核身查询结果

最近更新时间：2024-11-13 18:06:32

合作伙伴服务端验证结果

此方式用于：合作伙伴服务端生成签名，并调用意愿核身服务端查询结果，鉴权完成后返回结果（服务端上送 orderNo 和 appId 查询）。

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 。
合作方为意愿核身查询服务生成签名，需要具有以下参数：

参数	说明	来源
appId	腾讯服务分配的 Appid	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，本次意愿核身合作伙伴上送的订单号，唯一标识，字母/数字组成的字符串	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 ticket，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 appId、orderNo、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 。

意愿核身识别结果查询接口

请求

请求 URL： `https://kyc1.qcloud.com/api/server/getWillFaceResult?orderNo=xxxxx`
请求方法： POST
报文格式： `Content-Type: application/json`
请求参数：

参数	说明	类型	长度（字节）	是否必填
appId	腾讯服务分配的 Appid	字符串	腾讯服务分配	是
orderNo	订单号，合作方订单的唯一标识，字母/数字组成的字符串	字符串	32	是
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
getFile	是否需要获取人脸识别的视频和文件，值为1则返回视频和照片、值为2则返回照片、值为3则返回视频；其他则不返回	字符串	40	是
getWillFile	是否需要获取意愿表达的音视频文件，默认值为1 值为1-返回所有音视频文件，值为2-不返回完整的视频文件，返回其他音视频文件	字符串	40	是

响应参数：

参数	类型	说明
code	String	0代表成功
msg	String	返回结果描述
faceCode	String	人脸核身结果
faceMsg	String	人脸核身结果描述
willCode	String	意愿表达结果
willMsg	String	意愿表达结果描述
bizSeqNo	String	业务流水号
transactionTime	String	请求接口的时间
orderNo	String	订单编号
liveRate	String	活体检测得分
riskInfo	object	后台返回的刷脸风险信息
similarity	String	人脸比对得分
occurredTime	String	进行刷脸的时间
photo	String	人脸核身时的照片，base64 位编码
video	String	人脸核身时的视频，base64 位编码 注：该视频为活体阶段的1秒短视频。
sdkVersion	String	人脸核身时的 sdk 版本号
appId	String	腾讯云控制台申请的 appid
willUserAudio	String	意愿表达用户音频 意愿表达阶段的音频文件，base64 位编码 注：该字段仅在单轮问答情况下返回，多轮问答请查看字段 willTurnsData。
willReadAudio	String	意愿表达播报音频 意愿表达阶段的音频文件，base64 位编码 注：该字段仅在单轮问答情况下返回，多轮问答请查看字段 willTurnsData。
willScrnShotImage	String	意愿表达阶段的屏幕画面图，base64 位编码
willUserVideo	String	意愿核身完整视频：从用户播报音频到回复音频过程，base64 位编码 注：如果功能配置为 SDK 返回视频，则本字段返回为空。
willStandText	String	ASR 客户初始化上送的文字信息
willStandAnswer	String	ASR 客户初始化上送的答案文字信息
willUserAnswer	String	ASR 识别结果文本信息
willTurnsData	object	多轮问答模式下，问答内容明细结果信息

willTurnsData 字段明细内容如下：

参数	类型	说明
willTurnsData：id	String	问答轮次
willUserAudio	String	意愿表达用户音频 意愿表达阶段的音频文件，base64 位编码
willReadAudio	String	意愿表达播报音频 意愿表达阶段的音频文件，base64 位编码
willStandText	String	ASR 客户初始化上送的文字信息
willStandAnswer	String	ASR 客户初始化上送的答案文字信息
willUserAnswer	String	ASR 识别结果文本信息

```
{
    "code": "0",
    "msg": "请求成功",
    "bizSeqNo": "22071420001184453211072324316763",
    "result": {
        "bizSeqNo": "22071420001184453211072324316763",
        "transactionTime": "20220714110723",
        "appId": "IDAXXXXX",
        "orderNo": "13213545132",
        "similarity": "94.94",
        "liveRate": "99",
        "occurredTime": "20220714110147",
     "riskInfo": {
            "deviceInfoLevel": "1",
            "deviceInfoTag": ""
      },
        "faceCode": "0",
        "faceMsg": "请求成功",
        "willCode": "0",
        "willMsg": "请求成功",
        "willUserAudio": "UklGRiR2AgBXQVZFZm10IBAAAAABAAIAQB8AAAB9AAAEABAAZGF0YQB2AgAAAAAAAACAAgAGwA7AFEA",

"willReadAudio": "UklGRqRIBQBXQVZFZm10IBAAAAABAAEAgD4AAAB9AAACABAAZGF0YYBIBQAAAAAAAAAAAAAAAAAAAAAAAAAA",
        "willUserVideo": "AAAAIGZ0eXBpc29tAAACAGlzb21pc28yYXZjMW1wNDEAAAAIZnJlZQAOLGttZGF0AAACXQYF//9Z3EXpve",
        "willStandText": "本人知情以下号码开户，知晓出售、出租、转让等方式提供电话卡供他人使用的法律风险。请回答：同意",
        "willStandAnswer": "嗯|好的|是的",
        "willUserAnswer": "是的",
        "trtcFlag": "Y",
        "success": false
        "willTurnsData": [
       {
         "id": "0",
         "willUserAudio": "UklGRiSwBABXQVZFZm10IBAAAAABAAEAgLsAAAB3AQ",
         "willReadAudio": "UklGRqRpAABXQVZFZm10IBAAAAABAAEAQB8AAIA+AAAC",
         "willStandText": "请回答:是的请回答",
         "willStandAnswer": "是的",
         "willUserAnswer": "是的"
       },
       {
         "id": "1",
         "willUserAudio": "UklGRiSRBQBXQVZFZm10IBAAAAABAAEAgLsAAAB3AQAC",
         "willReadAudio": "UklGRoSxAABXQVZFZm10IBAAAAABAAEAQB8AAIA+AAAC",
         "willStandText": "这是第2轮这是第2轮：请回答:是的请回答",
         "willStandAnswer": "是的",
         "willUserAnswer": "是的"
         }
        ]
 }
    "transactionTime": "20220714110723"
}
```


注意事项

- code 非0时，有时不返回图片和视频。
- 照片和视频信息作为存证，合作伙伴可以通过此接口拉取视频等文件，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，默认不返回视频和照片信息。为确保用户操作整体流程顺利完成，部分情况下获取视频和照片会有1秒左右的延迟。
- 由于照片和视频信息有可能超过1M，考虑传输的影响，建议合作伙伴在使用时注意，建议获取比对结果用于后续流程处理和存证使用分开调用。避免网络传输带来的影响。目前我们的查询接口是异步查询，返回的文件可能会有1/2s的延迟。
- 如果合作方是完成人脸核身流程后马上去拉取视频/照片，建议在查询接口加上一个查询机制：判断图片是否存在，轮询3次，每次2s。
- 照片和视频均为 base64位编码，其中照片解码后格式一般为 JPG 和 PNG。视频格式解码后一般为 MP4。
- 服务端验证结果接口返回66660011无此查询结果的可能原因：66660017 验证次数超限后被风控，风控后的订单为无效，查询结果为无此查询结果。

- 用户中途退出刷脸，没有完成人脸验证的流程，没有比对，查询结果为无此查询结果。
- 操作超时，请退出重试 无此 ID 的用户身份信息，H5faceid 5分钟有效期，失效后无法进入刷脸流程，查询结果为无此查询结果。
- 66660016 视频格式或文件大小不合法，无法进行比对，查询结果为无此查询结果。
- 上传的视频非实时录制,被时间戳校验拦截，查询结果为无此查询结果。
- 查询超过3天的订单返回无此查询结果。

- 响应参数请勿做强校验，后续可能会有扩展。

RiskInfo 对象说明

RiskInfo 是用来给合作方传递刷脸风险信息的对象。其中 RiskInfo 对象的各个字段意义如下表所示：

字段名	类型	字段含义	说明
deviceInfoLevel	String	设备风险等级	设备风险级别，如果命中风险则显示
deviceInfoTag	String	设备风险标签	设备风险等级描述
riskInfoLevel	String	身份风险等级	如果命中身份风险则显示风险等级
riskInfoTag	String	身份风险标签	身份风险等级描述

详情见 SaaS 错误码风险标签说明 。

H5（微信浏览器）
微信H5（微信浏览器）

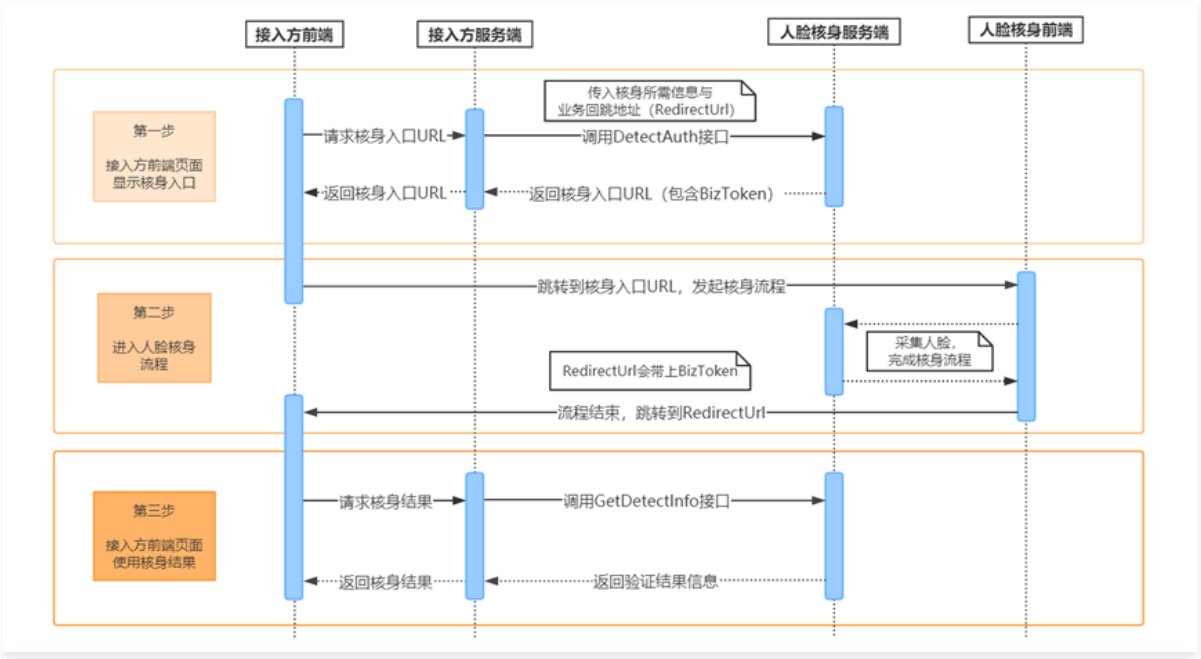
最近更新时间：2025-01-15 17:14:53

前提条件

1. 已注册腾讯云账号，并完成企业实名认证。
2. 已申请通过腾讯云人脸核身服务。

如果还未完成以上操作，可参考 流程指引 完成操作。

接入时序图

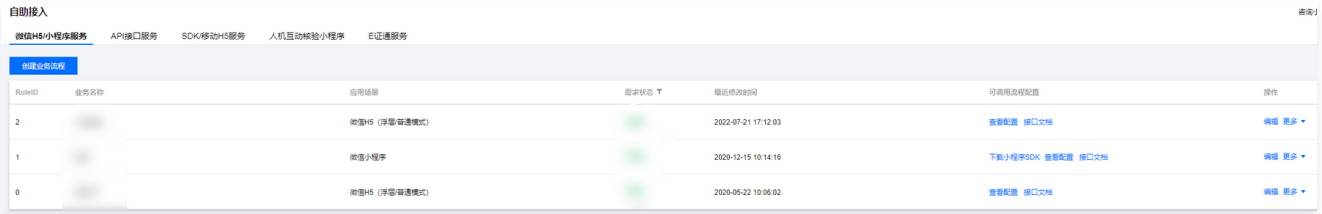


操作步骤

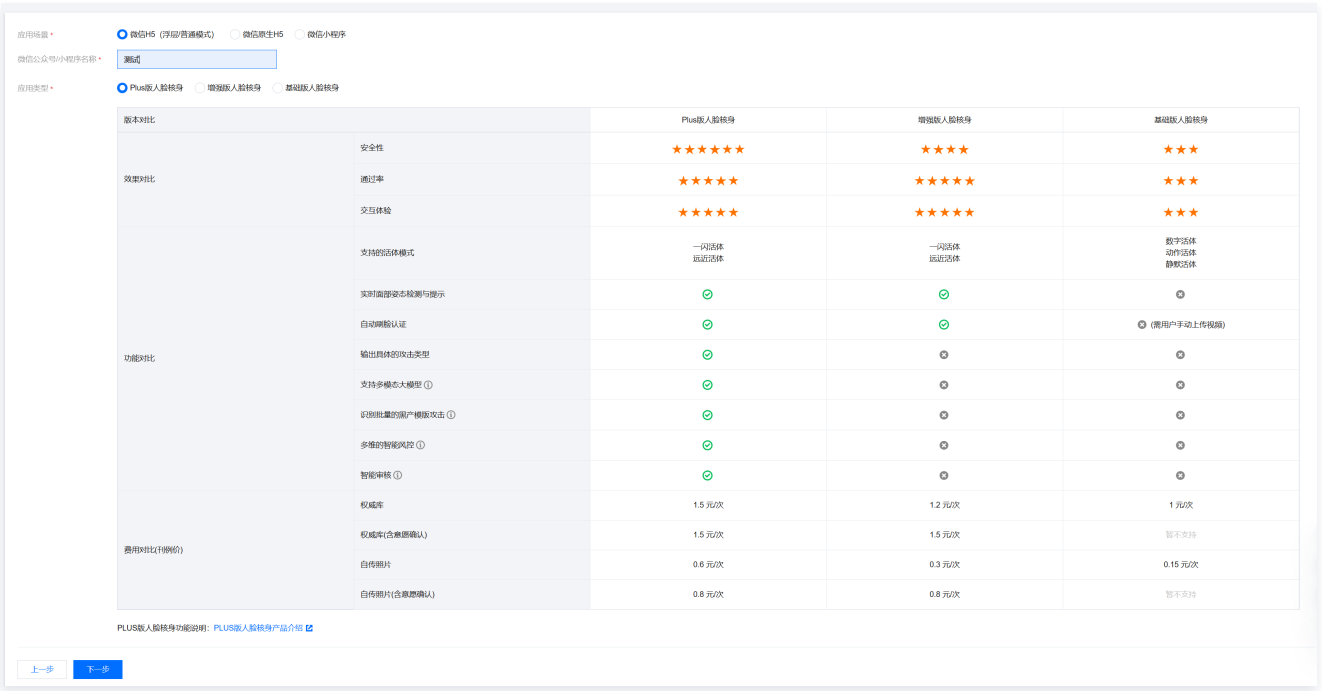
步骤1：创建 RuleId

RuleId 用于调用配置的业务流程，创建步骤如下：

1. 登录腾讯云人脸核身控制台，在 自主接入 页面，单击**创建业务流程**。



2. 选择应用场景，意愿核身微信 H5 支持两种接入模式，微信原生 H5 和微信浮层 H5：
   - 如您的公众号符合微信原生要求的行业类目及资质，推荐使用微信原生 H5。资质相关详情请参见 微信 HTML5 及小程序资质文件列表 。
   - 如您的公众号不满足原生资质，推荐使用微信浮层 H5。
3. 选择场景后再选择应用类型，您可选择 **Plus 版人脸核身**和**增强版人脸核身**。不可选**基础版人脸核身**，因为该类型无意愿确认的功能。填写好相关信息，完成后单击**下一步**。



4. 进入接入配置，根据您业务的实际场景填写页面标题、业务名称、业务描述信息后单击**下一步**。

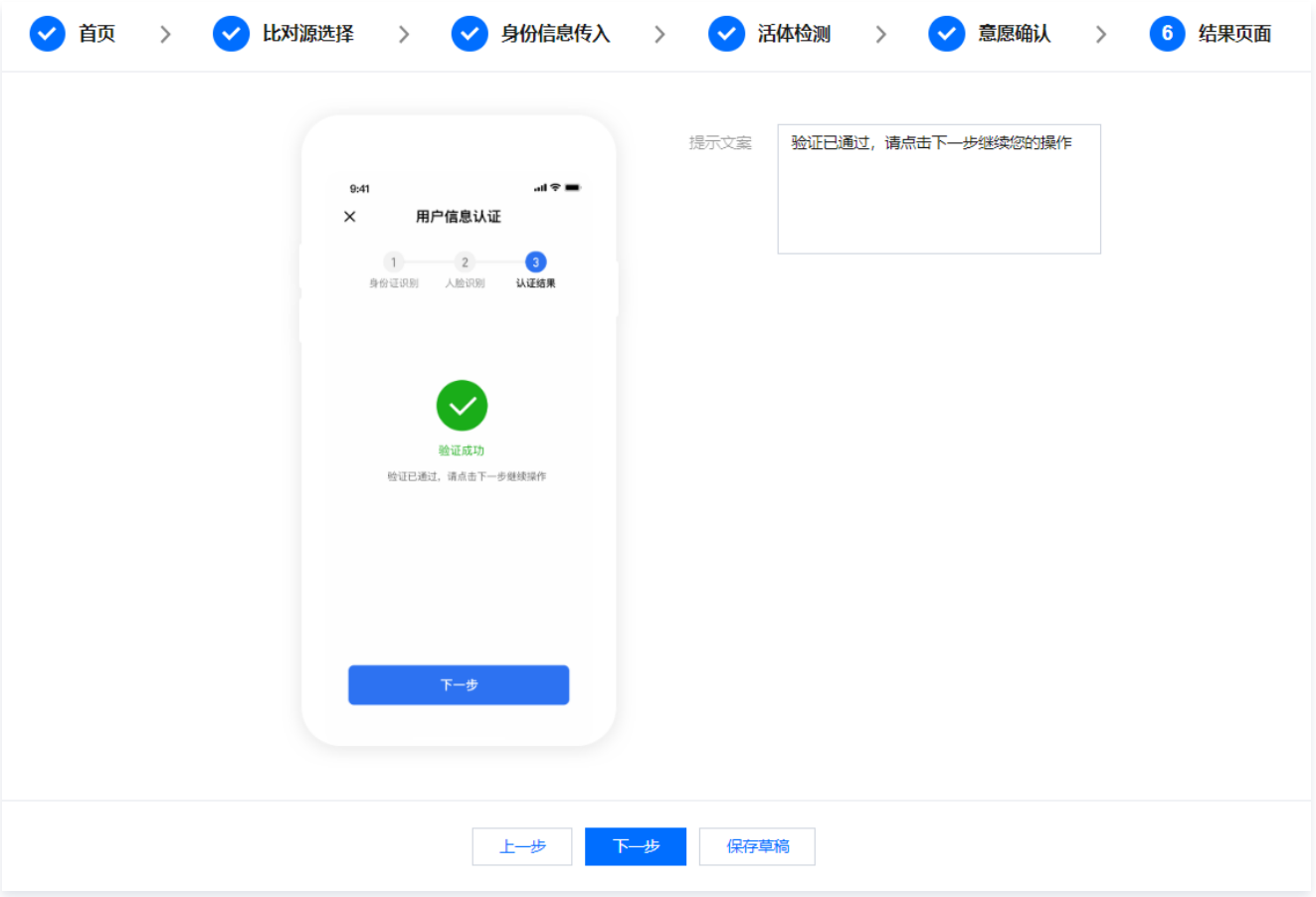
5. 选择人脸比对库源。人脸核身支持两种方式：**跟权威库比对**和**跟上传照片比对**，
   - 其中**跟权威库比对**收费价格为意愿核身（权威库）的价格。
   - **跟上传照片比对**收费的价格为意愿核身（自传照片）的价格，详细价格请见 **价格说明** 。

6. 配置身份证 OCR 功能，如果不需要则勾选**不需要用户在验证时上传**，如果需要则勾选**需要用户验证时上传身份信息**，并且配置好使用模式、是否支持港澳台居住证识别等，然后单击**下一步**。

7. 配置活动模式请勾选**一闪活体**，然后根据业务需要勾选**是否自动降级**和**降级后的活体模式**，然后单击**下一步**。

8. 配置意愿确认时请勾选**是，增加意愿确认**，勾选后单击**下一步**。配置后产品将按意愿核身价格计费，详情请参见 **价格说明** 。

9. 配置结果页的文案描述，然后单击**下一步**。



10. 业务信息填写完成后，确认您的配置信息然后单击**确认并提交审核**。

步骤2：获取 BizToken

- 完成 RuleId 创建后，需获取 BizToken，用于调用您配置的人脸核身验证的流程。
- 调用 实名核身鉴权 接口，得到核验流程唯一密钥（BizToken）和用于发起核身流程的 URL。
  - 传入 步骤1 生成的 RuleId，和认证结束后重定向的回调链接地址 RedirectUrl。
  - 可在 Config 参数中选择的所需意愿核身类型（问答模式/点头确认模式），使用 IntentionQuestions.N 或 IntentionActions.N 传入问答模式或点头确认模式的语音播报内容。
  - 建议开启 Config 中的 IntentionRecognition（意图识别）开关，可提升语音识别通过率。
- 在线调试 。

> **注意**
> - BizToken 是仅一次核身流程的标识，有效时间为7,200秒；用户完成核身后，开发者可用该标识获取验证结果信息，结果信息仅保留3天。
> - 如果输入参数 RedirectUrl 为空，则用户完成核验后默认跳转腾讯云人脸核身产品介绍页。

步骤3： 跳转人脸核身 URL 完成核验

您在前端通过地址跳转方式重定向至 步骤2 中获取的核身入口 URL，用户进入核身流程，完成身份证拍摄识别、录制视频等操作完身份核验。

步骤4： 查询核验结果信息

用户完成人脸核身后，页面会跳转到 RedirectUrl 上，地址中会带上此次验证流程使用的 BizToken，您在服务端即可凭借 BizToken 参数调用 获取实名核身结果信息 接口去获取本次核身的详细信息。获取到用户验证过程数据，包括文本信息、识别分数、照片和视频。也可以通过访问 人脸核身控制台 查看服务调用情况。

> **注意**
> - 为了保证用户的隐私，结果信息人脸核身侧仅保留3天，请您及时拉取。
> - 在开发、产品使用、费用、合同等问题有任何疑问，欢迎您 点此链接 扫描二维码添加腾讯云人脸核身小助手进行询问。

H5（移动端浏览器）
意愿核身移动 H5 接入步骤

最近更新时间：2024-06-17 17:09:41

序号	接入步骤	描述
1	合作方后台上传身份信息	合作方后台上传意愿核身信息，包括 appId、orderNo、name、idNo、userId 等。
2	启动意愿核身 H5	合作方上传 faceId 以及 sign，后台校验 sign 通过之后重定向到 H5 意愿核身。
3	H5 结果返回及跳转	认证完成后，认证结果页会回调启动 H5 意愿核身入参中指定的回调 URL 地址，使用回调参数中的 code 判断是否核身通过。
4	意愿核身结果查询	当用户完成核身认证后，如果合作方需要拉取意愿核身的视频用于存证等其他需要，可以调用查询核身结果接口来获取。
5	兼容性配置	如是 App 调用，请务必按照 App 调用 H5 兼容性配置 进行 iOS 及 Android 手机的兼容性适配，否则将影响正常使用。

合作方初始化上传信息

最近更新时间：2025-02-28 11:30:03

生成签名

前置条件

> **说明**
> 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。

请合作方确保 SIGN ticket 已经正常获取，获取方式见 获取 SIGN ticket 。合作方根据本次意愿核身的如下参数生成签名，需要签名的参数信息如下：

参数名	说明	来源
appId	业务流程唯一标识	参考 获取 WBappid 指引在意愿核身控制台内申请
userId	用户唯一标识	合作方自行分配，需跟用户绑定，32位以内，不包含特殊字符
version	参数值为：1.0.0	-
ticket	合作伙伴服务端获取的 ticket，注意是 SIGN 类型	获取方式见 获取 SIGN ticket
nonce	必须是 32 位随机数	合作方自行生成

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 appId、userId、version 连同 ticket、nonce 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	IDAXXXXX
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
ticket	XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

字典排序后的参数为：

```
[1.0.0,IDAXXXXX,XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS,
kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551]
```

拼接后的字符串为：

```
1.0.0IDAXXXXXXO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7Tus
erID19959248596551
```

计算 SHA1 得到签名：

```
D7606F1741DDCF90757DA924EDCF152A200AC7F0
```

该字串就是最终生成的签名（40 位），不区分大小写。

合作方后台上传信息

请求 URL： `https://kyc1.qcloud.com/api/server/getWillFaceId?orderNo=xxx`

> **注意**
> 为方便查询耗时，该请求url后面请拼接 orderNo 订单号参数。

请求方法：POST报文格式：Content-Type: application/json请求参数：

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 appid，可参考 获取 WBappid 指引在意愿核身控制台内申请	String	8	是
orderNo	订单号，由合作方上传，字母/数字组成的字符串,每次唯一，不能超过 32 位	String	不能超过 32 位	是
version	默认参数值为：1.0.0	String	20	是
sign	签名：使用上面生成的签名	String	40	是
nonce	随机数	String	32	是
name	姓名	String	-	使用权威源比对时：姓名+证件号，必须输入 使用自带源比对时：姓名+证件号，可不输入
idNo	证件号码	String	-	使用权威源比对时：姓名+证件号，必须输入 使用自带源比对时：姓名+证件号，可不输入
userId	用户 ID，用户的唯一标识（不能带有特殊字符），需要跟生成签名的 userId 保持一致	String	不能超过 32 位	是
sourcePhotoStr	比对源照片，**注意：原始图片不能超过500k，且必须为 JPG 或 PNG、BMP 格式** • 参数有值：使用合作伙伴提供的比对源照片进行比对，必须注意是正脸可信照片，照片质量由合作方保证 • 参数为空：根据身份证号 + 姓名使用权威数据源比对	BASE64String	1048576	否，非必填请使用标准的图片转 base64方法，base64编码不可包含换行符，不需要加前缀
sourcePhotoType	比对源照片类型参数值为1 时是：水纹正脸照 参数值为 2 时是：高清正脸照 **重要提示：照片上无水波纹的为高清照，请勿传错，否则影响比对准确率。如有疑问，请 联系我们**	String	1	否，提供比对源照片需要传
liveService	应用模式意愿核身默认上传 2	String	32	是
liveInterType	• 参数值为1时，表示仅使用实时检测模式，不兼容的情况下回调错误码3005 • 参数值为0时，表示优先使用实时检测模式，不兼容情况自动降级为视频录制模式 • 不传或传入为空，默认为仅使用强制实时检测模式	String	32	否

willType	意愿核身类型产品服务类型为“2”时，需要填写参数。 配置参数： • 0：问答模式 • 1：播报模式 • 2：点头模式 默认配置为0	String	32	否
willLanguage	意愿核身语言产品服务类型为“2”时，需要填写。 参数配置：0-中文普通话默认配置为0	String	32	否
willContentList	意愿核身过程中播报文本/问题、用户朗读/回答的文本，当前支持一个播报文本+回答文本	List	-	是
willContentList.id	从下标0开始，问答模式下支持多轮问答，最大支持 5 轮	String	32	是
willContentList.question	系统播报问题文本/问题长度为250个字以内 示例：“ 您好，为确保您本人操作，此次签约全程录音录像。请问您本次业务是本人自愿办理吗？请回答：我确认 ” 注意：多轮问答的总字数之和需要小于或等于500字。	String	32	是
willContentList.answer	用户朗读/回答的文本，用于识别用户朗读/回答的语音与文本是否一致。长度为10个字以内支持多个文本作为识别内容，文本之间用"|"分割。示例：我确认	String	32	是
speed	播报问题文字语速：默认为1.0倍速 "-1"代表0.8倍，"0"代表1.0倍，"1"代表1.2倍，"1.5"代表1.35倍，"2"代表1.5倍	string	32	否
willMidAnswer	liveInterType=0 时为必填项，用于意愿降级模式用户回答提示语，请保持与播报内容让用户回答的内容一致	string	32	否

响应

响应参数：

参数	类型	说明
code	String	0：成功，非0：失败 详情请参见 SaaS 服务错误码
msg	String	请求结果描述
bizSeqNo	String	请求业务流水号
orderNo	String	订单编号
faceId	String	此次唯一标识，调用时传入
optimalDomain	String	启动 H5 意愿核身步骤中调用 login 接口使用的域名

响应示例：

```
{
  "code": 0,
  "msg": "成功",
  "result": {
        "bizSeqNo":"业务流水号",
        "orderNo":"合作方订单号",
        "faceId":"cc1184c3995c71a731357f9812aab988"
        "optimalDomain":"kyc1.qcloud.com"
  }
}
```

说明：

success：false 无意义，合作伙伴可忽略，无需处理。
faceId 有效期为5分钟，每次进行人脸核身都需要重新获取。

启动意愿核身移动 H5

最近更新时间：2025-01-15 17:14:53

前置条件

合作方如果使用 App 内调起 H5 意愿核身，需要 App 平台的 webkit/blink 等组件支持调用摄像头录视频，方可正常使用意愿核身功能。

> **说明**
> - App 内调用 H5 意愿核身或者短信链接调用 H5 意愿核身等场景，若当前设备无法正常调用摄像头进行视频录制，前端将返回特定的错误码（如下）告知合作方，合作方拿到错误码后可选择用备用方案核身处理。
> - 在 App、微信公众号、浏览器中调用H5实时检测意愿核身，需要用户允许使用摄像头权限（授权操作形式包括弹窗、动作菜单、应用设置等，具体形式视手机型号而异）。如用户误操作导致拒绝授权，需要退出意愿核身并重新进入和允许授权。部分浏览器会缓存用户之前的授权操作，故如果退出意愿核身并重新进入时仍然出现无摄像头权限的情况，可以尝试清除浏览器缓存。

返回码	返回信息	处理措施
3001	该浏览器不支持视频录制	请使用其它验证方案
3002	登录态异常，cookie 参数缺失	重新进入
3003	意愿核身中途中断	重新进入
3004	无摄像头权限	重新进入并授权摄像头
3005	该浏览器不支持实时检测模式	请使用其它浏览器
3006	该功能仅支持微信环境打开	请使用微信进入
300101	报文包体问题	重新进入

摄像头授权操作示例：

生成签名

前置条件

请合作方确保 NONCE ticket 已经正常获取，获取方式见 获取 NONCE ticket 。合作方根据本次意愿核身的如下参数生成签名,需要签名的参数信息如下：

> **说明**

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	业务流程唯一标识	参考 获取 WBappid 指引在意愿核身控制台内申请
orderNo	订单号，本次意愿核身合作伙伴上送的订单号，字母/数字组成的字符串，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（与接口中使用的 userId 保持一致）
version	参数值为：1.0.0	-
faceId	getWillFaceId 接口返回的唯一标识	-
ticket	合作伙伴服务端获取的 tikcet，注意是 NONCE 类型	获取方式见 获取 NONCE ticket（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（与接口中的随机数保持一致）

基本步骤

1. 生成一个32位的随机字符串（字母和数字）nonce（接口请求时也要用到）。
2. 将 appId、userId、orderNo、version、faceId 连同 ticket、nonce 共7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）。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	appId001
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
faceId	bwiwe1457895464
orderNo	aabc1457895464
ticket	zxc9Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

字典排序后的参数为：

```
[1.0.0, aabc1457895464, appId001, bwiwe1457895464, kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551,
zxc9Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS]
```

拼接后的字符串为：

```
1.0.0aabc1457895464appId001bwiwe1457895464kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7TuserID19959248596551zxc9Qfxlti9iTVgHA
jwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS
```

计算 SHA1 得到签名：

```
4E9DFABF938BF37BDB7A7DC25CCA1233D12D986B
```

该字串就是最终生成的签名（40位），不区分大小写。

启动 H5 意愿核身

合作方上送 faceId 以及 sign，后台校验 sign 通过之后重定向到 H5 意愿核身。

> **注意**
> 为了保证服务的高可用，全面消除单点风险，我们启用了多域名服务。启动 H5 意愿核身需要使用 合作方初始化上送信息 接口返回内容中 optimalDomain 字段的域名。

请求 URL： `https://kyc1.qcloud.com/api/web/willLogin`
optimalDomain 使用 合作方后台上送接口返回的 optimalDomain 域名，如果 optimalDomain 字段返回为空或者取不到，默认使用域名 kyc1.qcloud.com。该跳转 url 不能直接暴露在前端 html 页面的&lta&gt标签中。某些浏览器会预加载&lta&gt标签中的 url，导致用户点击访问该 url 时，因 url 已经被预加载访问过，于是签名失效，页面报错“签名不合法”。
请求方法：GET
请求参数：

参数	说明	类型	长度	是否必填
appId	参考 获取 WBappid 指引在意愿核身控制台内申请	String	8	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，字母/数字组成的字符串,每次唯一，此信息为本次意愿核身上送的信息，不能超过32位	String	32	是
faceId	getWillFaceId 接口返回的唯一标识	String	32	是
url	H5 意愿核身完成后回调的第三方 URL，需要第三方提供完整 URL 且做 URL Encode。完整 URL Encode 示例： 原 URL：`https://cloud.tencent.com` Encode 后：`https%3a%2f%2fcloud.tencent.com`	String	-	是
resultType	是否显示结果页面，参数值为“1”时直接跳转到 url 回调地址，null 或其他值跳转提供的结果页面	String	-	否
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
from	browser：表示在浏览器启动刷脸 App：表示在 App 里启动刷脸，默认值为 App	String	-	是
redirectType	跳转模式，参数值为“1”时，刷脸页面使用 replace 方式跳转，不在浏览器 history 中留下记录；不传或其他值则正常跳转	String	-	否

请求示例：

```
https://kyc1.qcloud.com/api/web/willLogin?appId=appId001
&version=1.0.0
&nonce=4bu6a5nv9t678m2t9je5819q46y9hf93
&orderNo=161709188560917432576916585
&faceId=wb04f10695c3651ce155fea7070b74c9
&url=https%3a%2f%2fcloud.tencent.com
&from=browser
&userId=23333333333333
&sign=5DD4184F4FB26B7B9F6DC3D7D2AB3319E5F7415F
&redirectType=1
```

H5 结果返回及跳转

最近更新时间：2023-08-31 16:53:50

认证结束后，认证结果页会回调启动 H5 意愿核身入参中指定的回调 URL 地址。并带有 code、orderNo、faceId 等参数。
您可以根据我们刷脸完成后的回调请求参数中的 code 判断是否核身通过，code 0 表示意愿核身成功，其他错误码 表示失败。如果您需要二次验证结果准确性，您可以通过服务端查询接口获取最终认证结果。服务端获取验证结果能够返回意愿核身的结果以及相应的视频和图片用于您存证等其他需要。（请注意：务必在收到完成刷脸的回调后，再来调用服务端获取验证结果。）
参数说明：

参数	说明
code	意愿核身结果的返回码，0 表示成功，其他错误码表示失败
faceCode	意愿核身结果返回码
willCode	意愿表达结果返回码
orderNo	订单号 ，本次意愿核身上送的订单号
h5faceId	本次请求返回的唯一标识，此信息为本次意愿核身上送的信息
newSignature	对 URL 参数 App ID、orderNo 和 SIGN ticket、code 的签名
liveRate	本次意愿核身的活体检测分数

注意

- 若 H5 浏览器不支持调用摄像头录制视频，则直接返回前端错误码 code（详见 启动意愿核身 H5 的前置条件）。
- 在 H5 实时检测模式中，若意愿核身过程中发生中断，例如 App 被切换到后台等情况，前端会返回特定的错误码3003（意愿核身中途中断）。

意愿核身查询结果

最近更新时间：2024-11-13 17:57:42

合作伙伴服务端验证结果

此方式用于合作伙伴服务端生成签名，并调用意愿核身服务端查询结果，鉴权完成后返回结果（服务端上送 orderNo 和 appId 查询）。

合作方后台生成签名

准备步骤

1. 前置条件：
   请合作方确保 SIGN ticket 已经正常获取，获取方式见 获取 SIGN ticket。
2. 合作方为意愿核身查询服务生成签名，需要具有以下参数：

参数	说明	来源
appId	腾讯服务分配的 Appid	参考 获取 WBappid 指引在意愿核身控制台内申请
orderNo	订单号，本次意愿核身合作伙伴上送的订单号，唯一标识，字母/数字组成的字符串	合作方自行分配
version	默认值：1.0.0	-
ticket	合作伙伴服务端缓存的 ticket，注意是 SIGN 类型	获取方式见 获取 SIGN ticket
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 appId、orderNo、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 。

识别结果查询接口

请求 URL： `https://kyc1.qcloud.com/api/server/getWillFaceResult?orderNo=xxxx`
请求方法： POST
报文格式： Content-Type: application/json
请求参数：

参数	说明	类型	长度（字节）	是否必填
appId	腾讯服务分配的 Appid	字符串	腾讯服务分配	是
orderNo	订单号，合作方订单的唯一标识，字母/数字组成的字符串	字符串	32	是
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
getFile	是否需要获取人脸识别的视频和文件，值为1则返回视频和照片、值为2则返回照片、值为3则返回视频；其他则不返回	字符串	40	是
getWillFile	是否需要获取意愿表达的音频文件，默认值为1，值为1-返回，其他值为不返回	字符串	40	是

响应参数：

参数	类型	说明
code	String	0代表成功
msg	String	返回结果描述
faceCode	String	人脸核身结果
faceMsg	String	人脸核身结果描述
willCode	String	意愿表达结果
willMsg	String	意愿表达结果描述
bizSeqNo	String	业务流水号
transactionTime	String	请求接口的时间
orderNo	String	订单编号
liveRate	String	活体检测得分
similarity	String	人脸比对得分
occurredTime	String	进行刷脸的时间
photo	String	意愿核身时的照片，base64 位编码
video	String	意愿核身时的视频，base64 位编码
appId	String	腾讯云控制台申请的 appid
willUserAudio	String	意愿核身用户音频 意愿表达阶段的音频文件，base64位编码 注：该字段仅在单轮问答情况下返回，多轮问答请查看字段willTurnsData
willReadAudio	String	意愿核身播报音频 意愿表达阶段的音频文件，base64位编码 注：该字段仅在单轮问答情况下返回，多轮问答请查看字段willTurnsData
willUserVideo	String	意愿核身完整视频：从用户播报音频到回复音频过程，base64位编码
willScrnShotImage	String	意愿表达阶段的屏幕画面图，base64位编码
willStandText	String	客户初始化上送的文字信息
willStandAnswer	String	客户初始化上送的答案文字信息
willUserAnswer	String	识别结果文本信息
willTurnsData	object	多轮问答模式下，问答内容明细结果信息

willTurnsData 字段明细内容如下：

参数	类型	说明
willTurnsData：id	String	问答轮次
willUserAudio	String	意愿表达用户音频 意愿表达阶段的音频文件，base64 位编码
willReadAudio	String	意愿表达播报音频 意愿表达阶段的音频文件，base64 位编码
willStandText	String	ASR 客户初始化上送的文字信息
willStandAnswer	String	ASR 客户初始化上送的答案文字信息
willUserAnswer	String	ASR 识别结果文本信息


```
{
    "code": "0",
    "msg": "请求成功",
    "bizSeqNo": "2207142000118445321107232431676 3",
```

```
    "result": {
        "bizSeqNo": "2207142000118445321107232431676 3",
        "transactionTime": "20220714110723",
        "appId": "IDAXXXXX",
        "orderNo": "13213545132",
        "similarity": "94.94",
        "liveRate": "99",
        "occurredTime": "20220714110147",
        "faceCode": "0",
        "faceMsg": "请求成功",
        "willCode": "0",
        "willMsg": "请求成功",
        "willUserAudio": "UklGRiR2AgBXQVZFZm10IBAAAAABAAIAQB8AAAB9AAAEABAAZGF0YQB2AgAAAAAAAACAAgAGwA7AFEA",
        "willReadAudio": "UklGRqRIBQBXQVZFZm10IBAAAAABAAEAgD4AAAB9AAACABAAZGF0YYBIBQAAAAAAAAAAAAAAAAAAAAAAAAAA"
,
        "willUserVideo": "AAAAIGZ0eXBpc29tAAACAGlzb21pc28yYXZjMW1wNDEAAAAIZnJlZQAOLGttZGF0AAACXQYF//9Z3EXpve",
        "willStandText": "本人知情以下号码开户，知晓出售、出租、转让等方式提供电话卡供他人使用的法律风险。请回答：同意",
        "willStandAnswer": "嗯|好的|是的",
        "willUserAnswer": "是的",
        "trtcFlag": "Y",
        "success": false
     {
          "id": "0",
          "willUserAudio": "UklGRiSwBABXQVZFZm10IBAAAAABAAEAgLsAAAB3AQ",
          "willReadAudio": "UklGRqRpAABXQVZFZm10IBAAAAABAAEAQB8AAIA+AAAC",
          "willStandText": "请回答:是的请回答",
          "willStandAnswer": "是的",
          "willUserAnswer": "是的"
        },
        {
          "id": "1",
          "willUserAudio": "UklGRiSRBQBXQVZFZm10IBAAAAABAAEAgLsAAAB3AQAC",
          "willReadAudio": "UklGRoSxAABXQVZFZm10IBAAAAABAAEAQB8AAIA+AAAC",
          "willStandText": "这是第2轮这是第2轮：请回答:是的请回答",
          "willStandAnswer": "是的",
          "willUserAnswer": "是的"
          }
        ]
  }
    "transactionTime": "20220714110723"
}
```


注意事项

- code 非0时，有时不返回图片和视频。
- 照片和视频信息作为存证，合作伙伴可以通过此接口拉取视频等文件，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，默认不返回视频和照片信息。为确保用户操作整体流程顺利完成，部分情况下获取视频和照片会有1秒左右的延迟。
- 由于照片和视频信息有可能超过 1M，考虑传输的影响，建议合作伙伴在使用时注意，建议获取比对结果用于后续流程处理和存证使用分开调用。避免网络传输带来的影响。目前我们的查询接口是异步查询，返回的文件可能会有1~2s的延迟。
- 如果合作方是完成意愿核身流程后马上去拉取视频/照片，建议在查询接口加上一个查询机制：判断图片是否存在，轮询3次，每次2s。
- 照片和视频均为 base64 位编码，其中照片解码后格式一般为 JPG 和 PNG。视频格式解码后一般为 MP4。
- 服务端验证结果接口返回66660011无此查询结果的可能原因：66660017 验证次数超限后被风控，风控后的订单为无效，查询结果为无此查询结果。
  - 用户中途退出刷脸，没有完成人脸验证的流程，没有比对，查询结果为无此查询结果。
  - 操作超时，请退出重试 无此 ID 的用户身份信息，faceid 5分钟有效期，失效后无法进入刷脸流程，查询结果为无此查询结果。
  - 66660016 视频格式或文件大小不合法，无法进行比对，查询结果为无此查询结果。
  - 上传的视频非实时录制,被时间戳校验拦截，查询结果为无此查询结果。
  - 查询超过3天的订单返回无此查询结果。
- 响应参数请勿做强校验，后续可能会有扩展。

服务响应码

详情请参见 SaaS 服务错误码 。

App 调用 H5 兼容性配置

最近更新时间：2025-06-16 17:02:32

说明

> 接入前务必注意：请参见 移动 H5 意愿核身接入步骤 进行接入，如是 App 调用，务必按照 App 调用 H5 兼容性配置 进行兼容性适配，否则将影响正常使用。

兼容性说明

因网页端实时音视频技术是在2017年初次提出，对浏览器和手机系统存在兼容性要求，经过大规模测试，腾讯云 H5 实时检测意愿核身的兼容性情况如下：

手机平台	应用端	应用端说明	兼容性要求	机型支持率*
iOS	App	在客户自有 App 中使用，适用于在客户 App 中触达用户的业务场景	iOS 系统版本14.3+ 需按照 App 调用 H5 兼容性配置 做适配	90%
	浏览器	在手机浏览器使用，适用于通过短信或其他方式发送链接触达用户的业务场景	iOS 系统版本11.1.2+ 支持 Safari 浏览器，浏览器版本11+	100%
Android	App	在客户自有 App 中使用，适用于在客户 App 中触达用户的业务场景	Android 系统版本7+ 需按照 App 调用 H5 兼容性配置做适配	95%
	自带浏览器	在手机系统自带浏览器使用，适用于通过短信或其他方式发送链接触达用户的业务场景	Android 系统版本7+ 华为、OPPO、VIVO、魅族、荣耀、三星等自带浏览器兼容性较好（支持率80%） 小米自带浏览器兼容性一般（支持率30%）	
	QQ 浏览器	在 QQ 浏览器使用，适用于通过短信或其他方式发送链接触达用户的业务场景	Android 系统版本7+ 支持 QQ 浏览器，不支持 UC、百度浏览器	100%

- 采用达到版本要求的具有一定市场占有率的主流手机型号，测试机型共计829款。
- Android 手机测试品牌名单（按拼音排序）：HTC、OPPO、OPPO_REALME、VIVO、VIVO_IQOO、锤子-坚果、黑鲨、华硕、华为、荣耀、金立、联想、魅族、魅族-魅蓝、努比亚、努比亚-红魔、三星-GALAXY、小米、小米-红米、一加、中兴。

请按照下文兼容性配置指引进行 iOS 及 Android 手机的兼容性适配。

iOS 接入

iPhone 的兼容性适配，需在配置里加上摄像头和麦克风的使用权限。App 的 info.plist 中加入：

```
.NSMicrophoneUsageDescription
.NSCameraUsageDescription
```

使用 WKWebView 时，需要通过 WKWebViewConfiguration 配置允许使用相机：

```
WKWebViewConfiguration *config = [[WKWebViewConfiguration alloc] init];
config.allowsInlineMediaPlayback = YES;
```

Android 接入

由于 Android 机器碎片化严重，用系统 WebView 调起系统摄像头完成视频录制可能存在很多兼容性问题，如部分机器出现调不起摄像头、调起摄像头无法录制视频等。因此整理了接入指引。H5 刷脸包括 trtc 和录制模式，合作方需要对这两种模式都做兼容性配置。
请合作方**务必**按照如下步骤顺序，实现兼容性处理：

1. 引入工具类
   将 WBH5FaceVerifySDK.java 文件拷贝到项目中。该文件下载地址：人脸核身控制台（请到控制台自助接入列表页获取密码）。
2. 申请权限
   在 Manifest.xml 文件中增加申请以下权限：

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
```

   动态申请权限：
   - 如果第三方编译的 targetSdkVersion >= 23，则需要动态申请权限。
   - 如果第三方编译的 targetSdkVersion < 23，则不需要动态申请权限。

- 权限代码申请处理逻辑，demo 仅供参考，合作方可根据自身业务需求自行处理。
- 一定要在动态权限申请成功后，才能去调用 WBH5FaceVerifySDK的enterOldModeFaceVerify() 录制模式或 enterTrtcFaceVerify() trtc 模式体验 h5 刷脸功能。

3. 设置 WebSettings
调用 WebView.loadUrl(String url) 前一行添加如下代码设置 WebSettings。

```
/**
* 对 WebSettings 进行设置：添加 ua 字段和适配 h5 页面布局等
* @param mWebView  第三方的 WebView 对象
* @param context  第三方上下文
*/
WBH5FaceVerifySDK.getInstance().setWebViewSettings(mWebView,getApplicationContext());
```

4. 重写 WebChromeClient
调用 WebView.loadUrl(String url) 前，WebView 必须调用 setWebChromeClient(WebChromeClient webChromeClient)，并重写 WebChromeClient 的如下2个函数：

```
/**
 H5意愿刷脸模式配置，这里负责处理来自H5页面发出的相机权限申请
  * @param request 来自H5页面的权限请求
  */
 @Override
 public void onPermissionRequest(PermissionRequest request) {
   if
(request!=null&&request.getOrigin()!=null&&WBH5FaceVerifySDK.getInstance().isTencentH5FaceVerify(request.get
Origin().toString())){ //通过url判断是腾讯意愿性h5刷脸的域名
     this.request=request;
     if (activity!=null){
         activity.requestCameraAndRecordPermissions(false,true);//申请相机和录音权限，申请权限的代码demo仅供参考，合
作方可根据自身业务定制
     }
    }
 }

 /**
  * 相机权限申请成功后，拉起H5意愿刷脸模式进行实时刷脸验证
  */
 public void enterWillFaceVerify(){
    if (Build.VERSION.SDK_INT>Build.VERSION_CODES.LOLLIPOP){ // android sdk 21以上
         if (request!=null&&request.getOrigin()!=null){
             //根据腾讯域名授权，如果合作方对授权域名无限制的话，这个if条件判断可以去掉，直接进行授权即可。
             Log.d(TAG,"enterTrtcFaceVerify getOrigin()!=null");
             if (WBH5FaceVerifySDK.getInstance().isTencentH5FaceVerify(request.getOrigin().toString())){
                 //授权
                 request.grant(request.getResources());
                 request.getOrigin();
              }
           }else {
              if (request==null){
                 Log.d(TAG,"enterTrtcFaceVerify request==null");
                 if (webView!=null&&webView.canGoBack()){
                     webView.goBack();
                 }
              }
           }
      }
 }


  // For Android >= 4.1  降级为录制模式，收到h5页面发送的录制请求
 public void openFileChooser(ValueCallback<Uri> uploadMsg, String acceptType, String capture) {
```

```
    Log.d(TAG,"openFileChooser-------");
    if (WBH5FaceVerifySDK.getInstance().isTencentH5FaceVerify(null,null,acceptType)){ //true 说明是腾讯的h5刷脸页面
        this.uploadMsg=uploadMsg;
        this.acceptType=acceptType;
        if (activity!=null){ //申请系统的相机、录制、sd卡等权限
            activity.requestCameraAndSomePermissions(true);
        }
    }

 }

 // For Lollipop 5.0+ Devices  降级为录制模式，收到h5页面发送的录制请求
 @TargetApi(21)
 @Override
 public boolean onShowFileChooser(WebView webView, ValueCallback<Uri[]> filePathCallback, FileChooserParams fileChooserParams) {
    Log.d(TAG,"onShowFileChooser-------");
    if (WBH5FaceVerifySDK.getInstance().isTencentH5FaceVerify(webView,fileChooserParams,null)){ //true说 明是腾讯的h5刷脸页面
        this.webView=webView;
        this.filePathCallback=filePathCallback;
        this.fileChooserParams=fileChooserParams;
        if (activity!=null){ //申请系统的相机、录制、sd卡等权限
            activity.requestCameraAndSomePermissions(false);
        }
    }
    return true;

 }

 //降级为录制模式，拉起系统相机进行录制视频
 public boolean enterOldModeFaceVerify(boolean belowApi21){
    Log.d(TAG,"enterOldFaceVerify");
    if (belowApi21){ // For Android < 5.0
        if (WBH5FaceVerifySDK.getInstance().recordVideoForApiBelow21(uploadMsg, acceptType, activity)) { //腾讯的h5刷脸
            return true;
        }else {
            // todo 合作方如果其他的h5页面处理，则再次补充其他页面逻辑
        }
    }else { // For Android >= 5.0
        if (WBH5FaceVerifySDK.getInstance().recordVideoForApi21(webView, filePathCallback, activity,fileChooserParams)) {  //腾讯的h5刷脸
            return true;
        }else {
            // todo 合作方如果其他的h5页面处理，则再次补充其他页面逻辑

        }
    }
    return false;
 }
```

> **注意**
> - 如果第三方已重写以上函数，只要将如上述所示的函数体内容添加至第三方的对应函数体首行即可。
> - 如果第三方没有重写以上函数，则直接按上述所示重写。
> - WebView 不要使用 layerType 属性，否则导致刷脸界面白屏。

5. 重写 Activity

> **注意**

- 如果第三方 WebView 所属的 Activity 已重写以下函数，则将如下所示的函数体内容添加至第三方的对应函数体首行即可。
- 如果第三方 WebView 所属的 Activity 没有重写以下函数，则直接按以下代码重写。

WebView 所属的 Activity 必须重写如下函数：

```
Override
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
     Log.d(TAG, "onActivityResult --------"+requestCode);
     super.onActivityResult(requestCode, resultCode, data);
      if (requestCode == 0x11) { //传统录制模式，收到录制模式调用系统相机录制完成视频的结果
         Log.d(TAG, "onActivityResult recordVideo");
         if (WBH5FaceVerifySDK.getInstance().receiveH5FaceVerifyResult(requestCode, resultCode, data)) {
             return;
         }
     } else if (requestCode == PERMISSION_QUEST_WILL_CAMERA_RECORD_VERIFY) { //意愿性刷脸模式，从设置界面返回。
         Log.d(TAG, "onActivityResult willface cameraAndRecord");
         requestCameraAndRecordPermission();
     }else if (requestCode == PERMISSION_QUEST_CAMERA_RECORD_VERIFY) { //传统录制模式，从设置界面返回。
         Log.d(TAG, "onActivityResult cameraAndSome");
         requestCameraAndSomePermissions(false);
     }
 }
```

6. 权限拒绝的处理

在录制模式中，当用户点击了**开始录制**后，如果没有授权相机权限，会弹框让用户授权。如果用户拒绝授权的话，一定要调用下面的方法：

```
WBH5FaceVerifySDK.getInstance().resetReceive()    //原理参考https://www.teachcourse.cn/2224.html
```

调用可参见 demo，文件下载地址：人脸核身控制台（请到控制台自助接入列表页获取密码）。

Harmony Next 接入

1. 申请权限

a.在 model.json5 文件中增加申请以下权限

```
"requestPermissions": [
  {
    "name": "ohos.permission.INTERNET",
  },
  {
    "name" : "ohos.permission.CAMERA",
    "reason": "$string:app_name",
    "usedScene": {
      "abilities": [
        "EntryAbility"
      ],
      "when":"inuse"
    }
  },
  {
    "name" : "ohos.permission.MICROPHONE",
    "reason": "$string:app_name",
    "usedScene": {
      "abilities": [
        "EntryAbility"
      ],
      "when":"inuse"
    }
  }
]
```

2. 创建 WebviewController

```
@State controller: webview.WebviewController = new webview.WebviewController();
```

3. 创建 Web 组件

```
Web({
  src: 'https://kyc.qcloud.com/s/web/h5/#/entry?will=true&appId=IDALfjoD', //设置加载的意愿刷脸h5页面的url地址
  controller: this.controller  //给web设置controller
})
  .onControllerAttached(() => {
    this.controller.setCustomUserAgent(this.controller.getCustomUserAgent() + ';kyc/h5face;hoskyc/2.0') //设置
ua
  })
  .fileAccess(true)//web的配置，不能少
  .javaScriptAccess(true)//web的配置，不能少
  .domStorageAccess(true)//意愿性h5刷脸一定要添加这个配置，否则刷脸页面无法进行意愿性朗读
```

4. web 组件配置意愿模式

```
 .onPermissionRequest((event) => { // 终端页面收到h5刷脸意愿模式的请求
          if (event) {
            this.checkPermission().then(() => { //1. 终端申请相机权限成功
              event.request.grant(event.request.getAccessibleResource()) // 2. 授权h5页面
            }).catch((error: BusinessError) => { // 终端申请相机权限失败
              console.error(TAG, "申请权限异常" + error.message)
              event.request.deny() //2.告诉h5页面没有权限
            })
          }
        })
```

> **注意：**
> checkPermission() 是申请相机和麦克风权限的方法，接入方可以根据自身业务需求发挥，demo 的这个方法仅供参考。

5. web 组件配置传统录制模式

```
.onShowFileSelector((event) => { //终端页面收到h5刷脸传统录制模式的请求
  if (event) {
      let result = picCamera().then(result => { //1.打开系统相机
        let str: string[] = [result.resultUri]
        event.result.handleFileList(str) // 2. 把录制的视频传给h5页面
      }).catch((error:BusinessError)=>{//打开系统相机异常
        console.error(TAG,"打开相机异常"+error.message)
      });
  }
  return true;
})
```

实证 NFC App SDK 合作方初始化上传信息

最近更新时间：2024-06-17 17:09:41

生成签名

准备步骤

- 下载 SDK，请到控制台自助接入列表页获取密码。
- 前置条件：请合作方确保 SIGN ticket 已经正常获取，获取方式见 获取 SIGN ticket 。
- SIGN 类型的 ticket，其有效期为60分钟，且可多次使用。
- 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。

合作方为人脸核身服务生成签名，需要具有以下参数：

参数名	说明	来源
appId	业务流程唯一标识	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	业务订单号	订单号，由合作方上传，字母/数字组成的字符串,每次唯一，不能超过 32 位
version	参数值为：1.0.0	-
ticket	合作伙伴服务端获取的 ticket，注意是 SIGN 类型	获取方式见 获取 SIGN ticket
nonce	必须是 32 位随机数	合作方自行生成

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 appId、orderNo、version 连同 ticket、nonce 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	IDAXXXXX
orderNo	orderNo596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
ticket	XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

字典排序后的参数为：

```
[1.0.0, IDAXXXXX, XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS , kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, orderNo596551]
```

拼接后的字符串为：

```
1.0.0IDAXXXXXXO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7TorderNo596551
```

计算 SHA1 得到签名：

```
6CD5F0DBCFA1155E2A66754B33C2E67DD358393B
```

> **说明**
> 该字串就是最终生成的签名（40 位），不区分大小写。

合作方后台上传信息

请求

请求 URL： `https://kyc1.qcloud.com/api/server/getOcrCertId?orderNo=xxx`

> **注意**
> 为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。

请求方法：POST。
报文格式：Content-Type: application/json。
请求参数：

参数	说明	类型	长度（字节）	是否必填
appId	业务流程唯一标识，即 appid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	是
orderNo	订单号，由合作方上传，字母/数字组成的字符串,每次唯一，不能超过32位	String	不能超过32位	是
userId	用户 ID ，用户的唯一标识（不能带有特殊字符）	String	不能超过 32 位	是
version	默认参数值为：1.0.0	String	20	是
sign	签名：使用上面生成的签名	String	40	是
nonce	随机数，需要跟生成签名的 nonce 保持一致	String	32	是
nfcType	调用类型；1：二代身份证；3：港澳居民回乡证	String	32	是

响应

响应参数：

参数	类型	说明
code	String	0：成功，非0：失败；详情请参见 SaaS 服务错误码
msg	String	请求结果描述
bizSeqNo	String	请求业务流水号
orderNo	String	订单编号
ocrCertId	String	此次唯一标识，调 SDK 时传入

响应示例：

```
{
  "code": 0,
  "msg": "成功",
  "result": {
          "bizSeqNo":"业务流水号",
          "orderNo":"合作方订单号",
          "ocrCertId":"cc1184c3995c71a731357f9812aab988"
  }
}
```

生成 SDK 接口调用步骤使用签名

最近更新时间：2024-09-10 17:56:33

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 。
NONCE 类型的 ticket，其有效期为120秒，且一次性有效，即每次启动 SDK 刷脸都要重新请求 NONCE ticket，重新算 sign。同时建议合作方做前端保护，防止用户连续点击，短时间内频繁启动 SDK。

- 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。
- 合作方为人脸核身服务生成签名，需要具有以下参数：

参数名	说明	来源
appId	业务流程唯一标识，即 WBappid	参考 获取 WBappid 指引在人脸核身控制台内申请
userId	用户唯一标识	合作方自行分配（和 SDK 里面定义的 userId 保持一致）
version	参数值为：1.0.0	-
ticket	合作伙伴服务端获取的 ticket，注意是 NONCE 类型	获取方式见 NONCE ticket 获取
nonce	必须是 32 位随机数	合作方自行生成（和 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 nonce（其为字母和数字，登录时也要用到）。
2. 将 appId、userId、version 连同 ticket、nonce 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	IDAXXXXX
userId	userID19959248596551
nonce	kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T
version	1.0.0
ticket	XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS

字典排序后的参数为：

```
[1.0.0, IDAXXXXX, XO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMS ,
kHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7T, userID19959248596551]
```

拼接后的字符串为：

```
1.0.0IDAXXXXXXO99Qfxlti9iTVgHAjwvJdAZKN3nMuUhrsPdPlPVKlcyS50N6tlLnfuFBPIucaMSkHoSxvLZGxSoFsjxlbzEoUzh5PAnTU7Tus
erID19959248596551
```

计算 SHA1 得到签名：

```
D7606F1741DDCF90757DA924EDCF152A200AC7F0
```

该字串就是最终生成的签名（40 位），不区分大小写。

Android SDK 接入

最近更新时间：2024-05-13 10:35:21

开发准备

CPU 平台设置
目前 SDK 支持 armeabi，armeabi-v7a，arm64-v8a，x86， x86_64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，如下所示：

```
defaultConfig {
    ndk {
          //设置支持的so库框架
          abiFilters 'armeabi-v7a', 'armeabi', 'arm64-v8a', 'x86', 'x86_64'
     }
}
```

配置流程

1. 接入配置

云 NFC SDK（WbCloudNfcSdk）最低支持到 **Android API 14: Android 4.0(ICS)**，请在构建项目时注意。
云 NFC 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 {
        //1. 云NFC SDK
        implementation(name: 'WbCloudNfcSdk-pro-v2.0.0-1fec32a', ext: 'aar')
        //2.云公共组件
        implementation(name: 'WbCloudNormal-v5.1.3-90776e2', ext: 'aar')    }
```

2. 混淆配置

云 NFC 产品的混淆规则，对接入方而言，只需加载云公共组件的混淆规则。
您可以将 SDK 中的 `txkyc-cloud-normal-consumer-proguard-rules.pro` 拷贝到主工程根目录下，然后通过 `-include txkyc-cloud-normal-consumer-proguard-rules.pro` 加入到您的混淆文件中。

接口调用

1. 标准模式 SDK 接口调用方法
   SDK 代码调用的入口为 **com.tencent.cloud.huiyansdknfc.WbCloudNfcSDK** 这个类。

```
public class WbCloudNfcSDK{

/**
* 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 WbCloudNfcSDK getInstance() {
       //     ...
       }
/**
* 在使用SDK前先初始化，传入需要的数据data
*由 nfcLoginListener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, NfcLoginListener nfcLoginListerner){
```

```
//     ...
}
   /**
     * 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 nfcResultListener          返回到第三方的接口
     */
    Public void startActivityForNfc(Context context, NfcResultListener){
    // ...
 }
/**
  * 登录回调接口
  */
public interface NfcLoginListener {
        void onLoginSuccess();
        void onLoginFailed(String errorCode, String errorMsg);
    }

/**
  * 退出SDK，返回第三方的回调，同时返回NFC识别结果
  */
public interface NfcResultListener{
        /**
         * @RARAM wbNfcResult    SDK返回的NFC识别结果
         */
        void onFinish(WBNfcResult wbNfcResult);
}
```

WbCloudNfcSdk.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：

```
String orderNo;  //订单号，字母/数字组成的字符串
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
String openOcrCertId; //ocrCertId
```

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

2. 登录接口

```
/**
  * 登录回调接口
  */
public interface NfcLoginListener {
        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);
    }
```

3. 返回第三方接口

```
/**
  * 退出SDK，返回第三方的回调，同时返回NFC识别结果
  */
public interface NfcResultListener{
        /**
         * 退出SDK，返回第三方的回调，同时返回NFC识别结果
```

```
         * @param wbNfcResult    返回NFC 识别结果
        */
        void onFinish(WBNfcResult wbNfcResult);
}
```

NFC 识别结果类

NFC 识别结果，封装在 WBNfcResult 类中，该类属性如下所示：

```
    public String code;//识别结果的code
    public String message; //识别结果的msg
    public String orderNo;//识别结果的订单号
    public String ocrCertId;//订单结果的ocrCertId
    public String reqId; //订单结果的reqId
```

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	String	32	必填，合作方订单的唯一标识，字母/数字组成的字符串
openApiAppId	业务流程唯一标识，即 wbappid，可参考 获取 WBappid 指引在人脸核身控制台内申请	String	8	必填
openApiAppVersion	接口版本号	String	20	必填，默认填 1.0.0
openApiNonce	32位随机字符串	String	32	必填，每次请求需要的一次性 nonce
openApiUserId	User Id	String	30	必填，每个用户唯一的标识
openApiSign	合作方后台服务器通过 ticket 计算出来的签名信息	String	40	必填
openApiOcrCertId	合作方后台服务器通过 ticket 计算出来的 ocrCertId	String	40	必填

个性化参数设置

WbCloudOcrSdk.init() 里 Bundle data，除了必须要传的 InputData 对象（详情见上节）之外，还可以由合作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。

错误码描述

终端返回错误码	描述
UNABLE_SUPPORT_NFC_ERROR = "600100";	设备不支持
USER_CANCEL_NFC_ERROR =" 600101 "	用户取消操作
ILLEGAL_PARAM_ERROR =" 600102 ";	传入的参数不合法
RECOGNISE_TIME_OUT =" 600103 ";	识别超时
NO_NET_ERROR =" 600104 ";	网络不给力
DECODE_ENMSG_ERROR =" 600105 ";	解码异常
INIT_NFC_ERROR =" 600106 ";	初始化 NFC SDK 出错

接入示例

```
# 在 合作方的页面中单击某个按钮的代码逻辑：
//先填好数据
  Bundle data = new Bundle();
        WbCloudNfcSDK.InputData inputData = new WbCloudNfcSDK.InputData(
                orderNo,
                openApiAppId,
                openApiAppVersion,
                openApiNonce,
                openApiUserId,
                openApiSign,
 openApiOcrCertId);
data.putSerializable(WbCloudNfcSDK.INPUT_DATA, inputData);
        data.putLong(WbCloudNfcSDK.NFC_TIME, nfcTimeOut);//识别超时时间。身份证默认20s,回乡证默认60s
        data.putString(WbCloudNfcSDK.NFC_TYPE, cardType);//1表示身份证，3表示回乡证。默认身份证
        if ("3".equals(cardType)){
            data.putString("travel_card_no",travelCardnoEt.getText().toString());
            data.putString("travel_card_birth",travelBirthEt.getText().toString());
            data.putString("travel_card_validate",travelValidateEt.getText().toString());
        }
        WbCloudNfcConfig.getInstance().setEnableLog(true);
        final long startLoginTime = System.currentTimeMillis();
        WbCloudNfcSDK.getInstance().init(MainActivity.this, data, new WbCloudNfcSDK.NfcLoginListener() { //返退
出 SDK 回调接口
                    @Override
                    public voidonFinish(WBNfcResult wbNfcResult) {
           // resultCode为0，则NFC识别成功；否则识别失败
                if ("0".equals(wbNfcResult.code)) {
//识别成功后，可以拿wbNfcResult.reqId去查询身份证结果
                WLogger.d(TAG, "识别成功，NFC的结果是:"+wbNfcResult.toString());
                }else{  //
                WLogger.d(TAG, "NFC识别失败 code="+wbNfcResult.code+" message="+wbNfcResult.message);
                }
  });
}
@Override
public void onLoginFailed(String errorCode, String errorMsg) {
if(errorCode.equals(ErrorCode.ILLEGAL_PARAM_ERROR)) {
Toast.makeText(MainActivity.this, "传入参数有误！" + errorMsg, Toast.LENGTH_SHORT).show();
} else {
Toast.makeText(MainActivity.this, "登录 NFC SDK 失败！" + "errorCode= " + errorCode + " ;errorMsg=" + errorMsg,
Toast.LENGTH_SHORT).show();
}
}
})
```

详细接入代码，请参考 SDK 附的 wbcloudnfcdemo 工程。

iOS SDK 接入

最近更新时间：2024-11-18 14:50:12

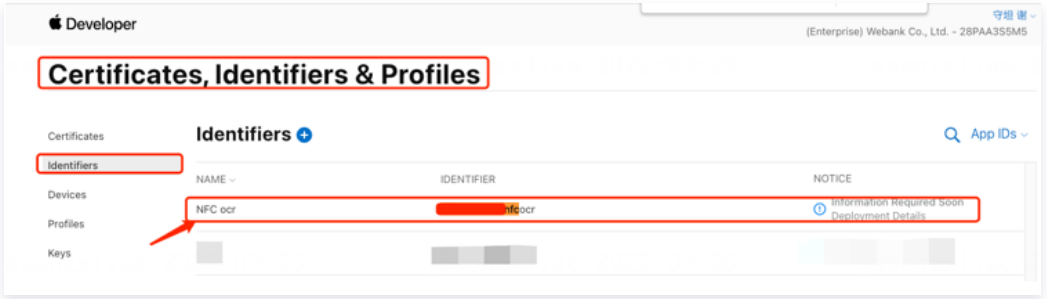
> **注意**
> 接入之前，请详细阅读 SDK 中的 readme 和接入指引。

以下为接入配置的步骤。

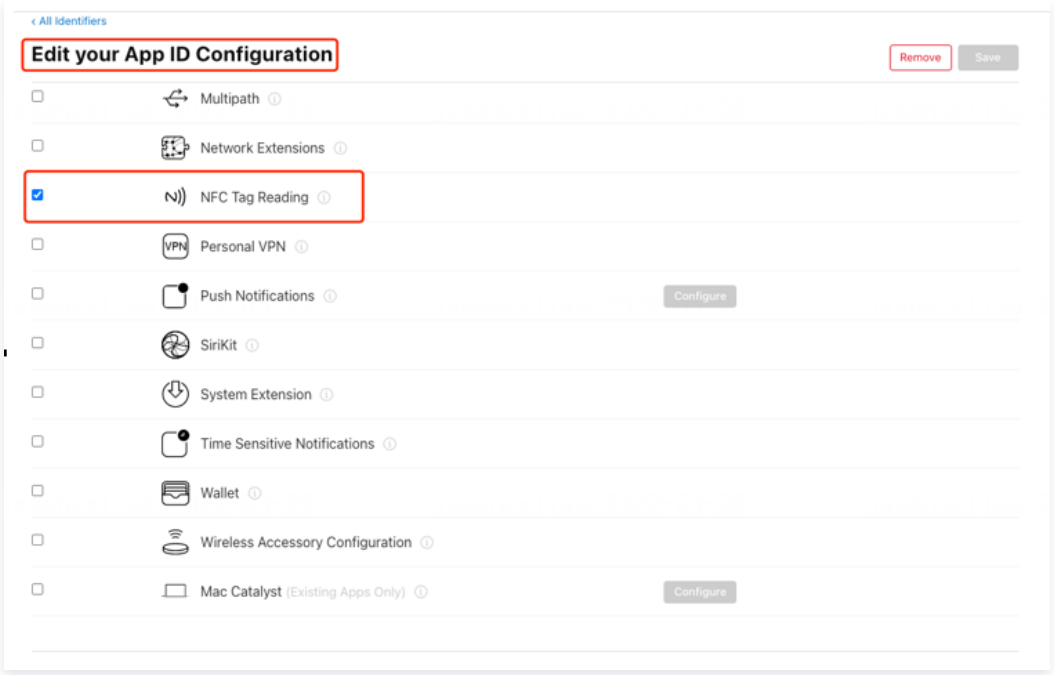
基础配置

本文档介绍了接入 NFC OCR SDK 接口，NFC 读取证件需要 iPhone 7 及以上，iOS14.5 及以上。

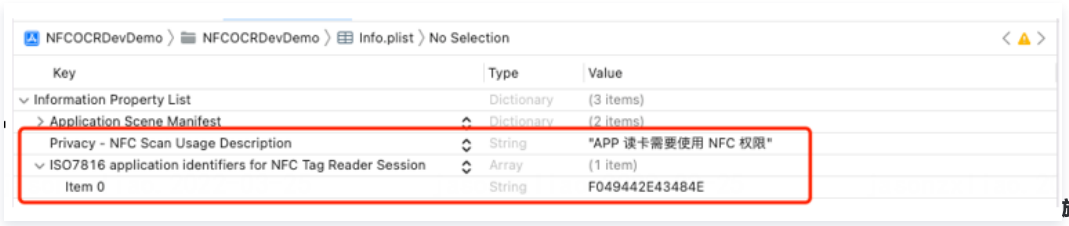
1. 在苹果开发者管理平台配置 Bundle Id。
   请登录苹果开发者管理平台，确认当前 bundle ID 下的 NFC Tag Reading 已经选中。
   配置路径 Certificates，Identifiers & Profiles > Identifiers > 选中需要配置接入 SDK 的 App 对应的 ID。



   单击进去之后，勾选 NFC Tag Reading。

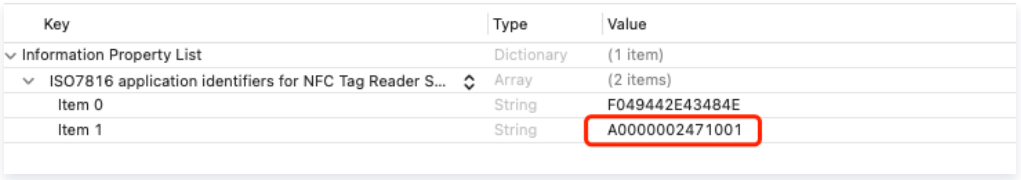


2. 在 Xcode 中配置 Info.plist 文件。
   实证 NFC 配置
   在 info.plist 中添加 NFCReaderUsageDescription 和 ISO7816 application identifiers for NFC Tag Reader Session，在 item0 中填写 F049442E43484E。

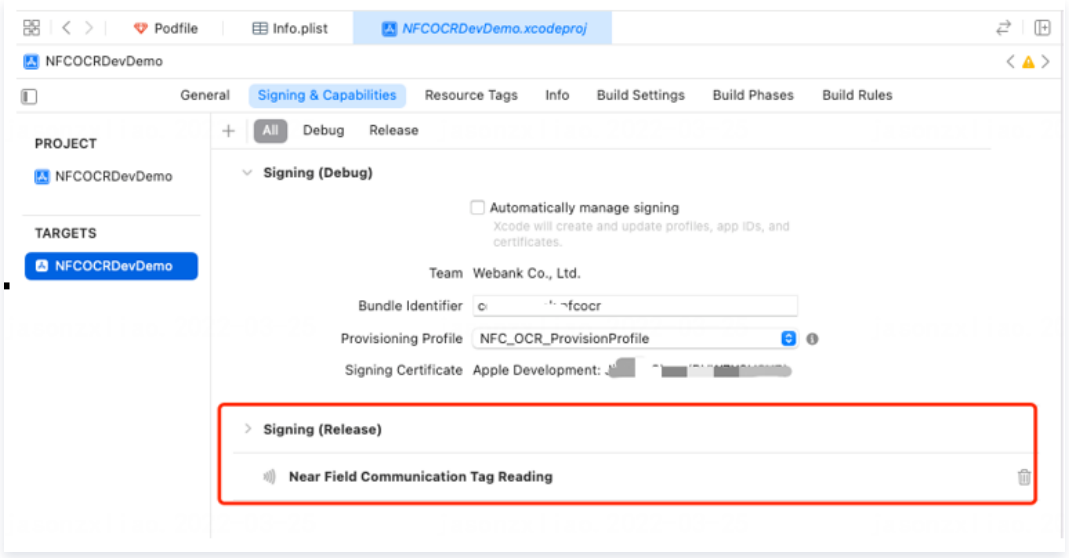


旅行证 NFC 配置

在 http://info.plist 中添加 NFCReaderUsageDescription 和 ISO7816 application identifiers for NFC Tag Reader Session，在 item0 中填写 A0000002471001（下图配置了两个 item，同时支持身份证和旅行证件识别）。



3. 在 Xcode 中配置 Capabilities。
在 IDE 中，依次选择 target > Signing&Capabilities > All ，添加 Near Field Communication Tag Reading。



SDK 集成

SDK 文件目录如下：

```
.
├── NFCOCRDevDemo
├── Readme
└── WBNFCReaderService
```

- WBNFCReaderService 目录下提供了 SDK。
- NFCOCRDevDemo 目录下面提供了 SDK 开发接入的 Demo。
- Readme 是版本信息。

Cocoapods 集成

下面介绍 Cocoapods 集成 SDK。
参考 NFCOCRDevDemo，在项目的 Podfile 中添加引用语句，并指明 SDK 的相对路径。

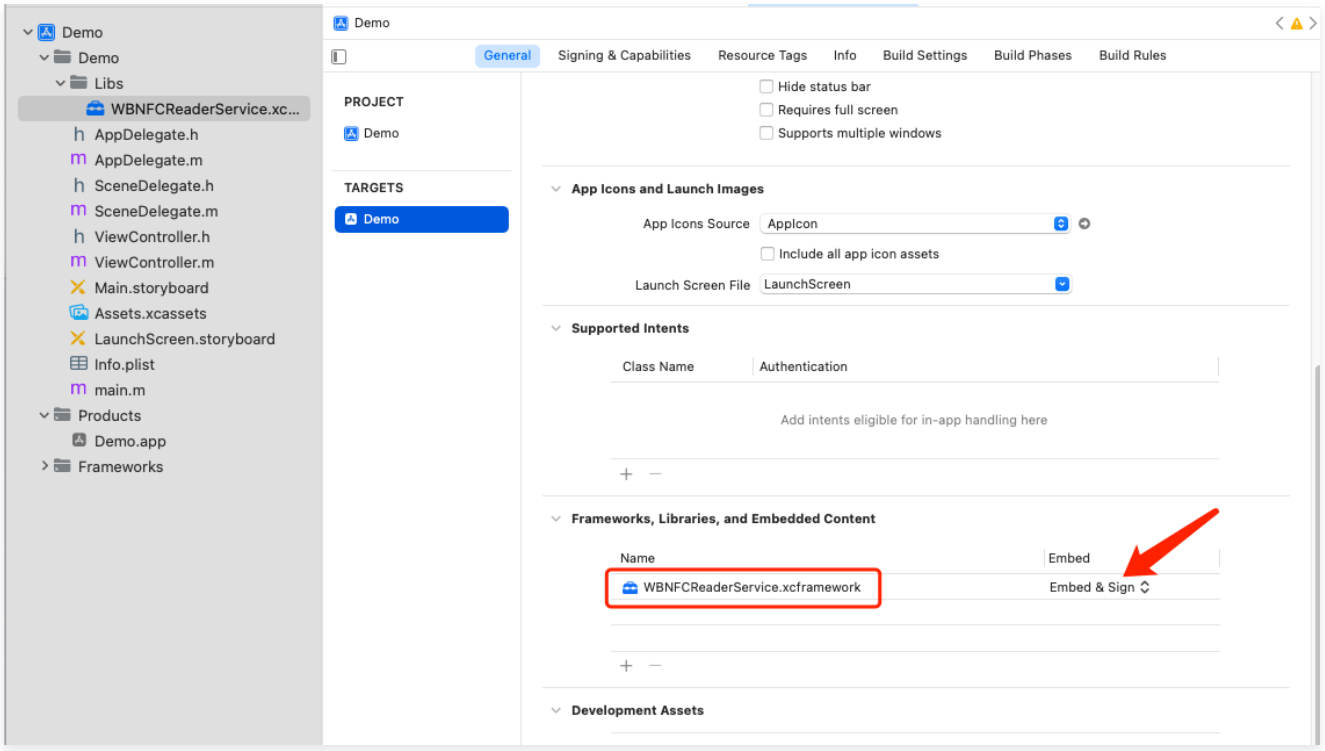
```
target 'NFCOCRDevDemo' do
  use_frameworks!
      # Pods for NFCOCRDevDemo
  pod 'WBNFCReaderService', :path => '../WBNFCReaderService'
```

```
end
```

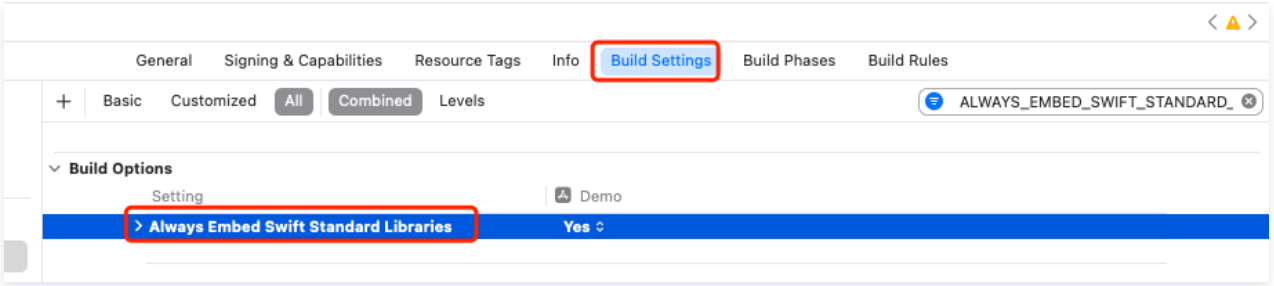
执行 pod install，便可完成 SDK 的集成。

手动集成

1. 将 WBNFCReaderService.xcframework 添加到项目。
   将 WBNFCReaderService.xcframework 添加到项目，并勾选 Embed & Sign。

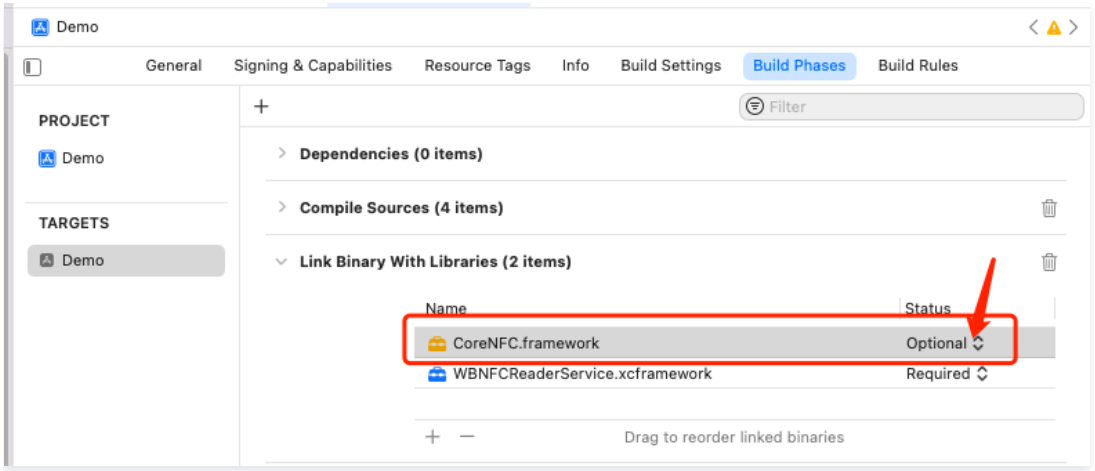


2. 配置 Build Settings。
   设置 ALWAYS_EMBED_SWIFT_STANDARD_LIBRARIES为 YES. 这个配置很重要，SDK 使用 swift 语言开发，不设置的话，在低版本 iOS 系统上会出现启动 crash。
   配置方式如下，Build Settings > ALWAYS_EMBED_SWIFT_STANDARD_LIBRARIES，设置为 YES。



3. CoreNFC.framework 配置。
4. SDK 使用到系统库 CoreNFC.framework, 在 Build Phases > Link Binary 中添加 CoreNFC.framework。

> ⚠ 注意
> status 要选 Optional，否则在不支持 CoreNFC 的低版本手机上会出现 crash。



SDK 调用

调用前准备

App 接入 SDK 前，需要获取腾讯服务分配的接入参数，通过 App 后台计算签名。

SDK 主要 API 介绍

1. WBOCRReaderService 类提供 NFC SDK 入口和回调。
1. WBOCRReaderService 是 SDK 的核心类，通过这个类对外提供 NFC 证件识别能力，它是个单例类，通过 sharedInstance来实例化。

```
+ (WBOCRReaderService * _Nonnull)sharedInstance;
```

SDK 的入口方法如下, 入参通过 WBOCRReaderParam 类传入。

- 实证 NFC 识别入口

```
/// SDK 入口方法1 -- 身份证 NFC（带 UI 的入口）
/// - Parameters:
/// - param: 请求 SDK 的业务参数，字段参考 `WBOCRReaderParam` 类
/// - fromVC: 跳转的 UIViewController 或者 UINavigationController，SKD 基于这个 VC 跳转
/// - loginSucceedBlock: SDK 登录成功回调，收到这个回调之后，即将进入 SDK 页面
/// - readSucceedBlock: SDK 识别成功回调，接收到这个回调之后，SDK 即将退出，回到 APP 页面
/// - failedBlock: SDK 异常回调，接收到这个回调之后，SDK 即将退出，回到 APP 页面
- (void)startReaderServiceWith:(WBOCRReaderParam *_Nonnull)param fromVC:(UIViewController
*_Nonnull)fromVC loginSucceedBlock:(void (^ _Nonnull)(void))loginSucceedBlock readSucceedBlock:(void (^
_Nonnull)(WBOCRReaderResult *_Nonnull))readSucceedBlock failedBlock:(void (^ _Nonnull)(WBOCRReadError
*_Nonnull))failedBlock SWIFT_AVAILABILITY(ios,introduced=14.5);

/// SDK 入口方法2 -- 身份证 NFC（无 UI 的入口）
/// - Parameters:
/// - param: 请求 SDK 的业务参数，字段参考 `WBOCRReaderParam` 类
/// - loginSucceedBlock: SDK 登录成功回调，收到这个回调之后，即将进入 SDK 页面
/// - readSucceedBlock: SDK 识别成功回调，接收到这个回调之后，SDK 即将退出，回到 APP 页面
/// - failedBlock: SDK 异常回调，接收到这个回调之后，SDK 即将退出，回到 APP 页面
- (void)startReaderServiceWith:(WBOCRReaderParam *_Nonnull)param loginSucceedBlock:(void (^ _Nonnull)
(void))loginSucceedBlock readSucceedBlock:(void (^ _Nonnull)(WBOCRReaderResult
*_Nonnull))readSucceedBlock failedBlock:(void (^ _Nonnull)(WBOCRReadError * _Nonnull))failedBlock
SWIFT_AVAILABILITY(ios,introduced=14.5);
```

- 旅行证识别入口

和身份证识别相比，旅行证识别需要额外传入三个参数：旅行证编号、出生日期和截止日期。

```
/// SDK 入口方法3 -- 旅行证识别入口（带 UI 入口）
/// - Parameters:
/// - param: 请求 SDK 的业务参数，字段参考 `WBOCRReaderParam` 类
```

```
/// - passportNumber: 旅行证件编号
/// - dateOfBirth: 出生日期 6位数字
/// - expiryDate: 截止日期 6位数字
/// - fromVC: 跳转的 UIViewController 或者 UINavigationController, SKD 基于这个 VC 跳转
/// - loginSucceedBlock: SDK 登录成功回调，收到这个回调之后，即将进入 SDK 页面
/// - readSucceedBlock: SDK 识别成功回调，接收到这个回调之后，SDK 即将退出，回到 APP 页面
/// - failedBlock: SDK 异常回调，接收到这个回调之后，SDK 即将退出，回到 APP 页面
- (void)startPassportReaderServiceWith:(WBOCRReaderParam *_Nonnull)param passportNumber:(NSString
*_Nonnull)passportNumber dateOfBirth:(NSString *_Nonnull)dateOfBirth expiryDate:(NSString
*_Nonnull)expiryDate fromVC:(UIViewController *_Nonnull)fromVC loginSucceedBlock:(void (^ _Nonnull)
(void))loginSucceedBlock readSucceedBlock:(void (^ _Nonnull)(WBOCRReaderResult
*_Nonnull))readSucceedBlock failedBlock:(void (^ _Nonnull)(WBOCRReadError * _Nonnull))failedBlock;

/// SDK 入口方法4-- 旅行证识别入口
/// - Parameters:
/// - param: 请求 SDK 的业务参数，字段参考 `WBOCRReaderParam` 类
/// - passportNumber: 旅行证件编号
/// - dateOfBirth: 出生日期 6位数字
/// - expiryDate: 截止日期 6位数字
/// - loginSucceedBlock: SDK 登录成功回调，收到这个回调之后，即将进入 SDK 页面
/// - readSucceedBlock: SDK 识别成功回调，接收到这个回调之后，SDK 即将退出，回到 APP 页面
/// - failedBlock: SDK 异常回调，接收到这个回调之后，SDK 即将退出，回到 APP 页面
- (void)startPassportReaderServiceWith:(WBOCRReaderParam *_Nonnull)param passportNumber:(NSString
*_Nonnull)passportNumber dateOfBirth:(NSString *_Nonnull)dateOfBirth expiryDate:(NSString
*_Nonnull)expiryDate loginSucceedBlock:(void (^ _Nonnull)(void))loginSucceedBlock readSucceedBlock:(void
(^ _Nonnull)(WBOCRReaderResult *_Nonnull))readSucceedBlock failedBlock:(void (^ _Nonnull)(WBOCRReadError
*_Nonnull))failedBlock;
```

通过 sdkVersion 属性来查看 SDK 的版本号。

```
@property (nonatomic, readonly, copy) NSString * _Nonnull sdkVersion;
```

2. WBOCRReaderParam 类定义 SDK 入参模板。
   WBOCRReaderParam 描述了 SDK 的入参格式。
   每个字段的含义以及格式如下：

```
@interface WBOCRReaderParam : NSObject
@property (nonatomic, copy) NSString * _Nonnull appId;    // appId 由腾讯服务分配的
@property (nonatomic, copy) NSString * _Nonnull version; // openAPI 接口版本号，由腾讯服务统一分配，当前传入 1.0.0
@property (nonatomic, copy) NSString * _Nonnull nonce;    // 每次请求需要的一次性nonce，一次有效
@property (nonatomic, copy) NSString * _Nonnull userId;   // 每个用户唯一的标识
@property (nonatomic, copy) NSString * _Nonnull sign;     // 签名信息，有接入方后台提供，一次有效
@property (nonatomic, copy) NSString * _Nonnull orderNo; // 订单号，长度不能超过32位的字符串
@property (nonatomic, copy) NSString * _Nonnull ocrCertId;// NFC流程唯一id，后台生成
@end
```

3. WBOCRReaderResult 类定义 SDK 返回结果模板。
   SDK 识别成功之后，会通过 readSucceedBlock 将识别结果回调给 App，WBOCRReaderResult 描述了返回值格式。

```
@interface WBOCRReaderResult : NSObject
@property (nonatomic, readonly, copy) NSString * orderNo;  // 订单号
@property (nonatomic, readonly, copy) NSString * reqId;    // 卡片识别结果
@property (nonatomic, readonly, copy) NSString * ocrCertId;// NFC 流程唯一 id
@end
```

4. WBOCRReadError 类定义 SDK 异常结果模板.
   SDK 异常的时候，会通过 failedBlock 将错误信息返回给 App ，并退出 SDK。

```
@interface WBOCRReadError : NSObject
@property (nonatomic) NSInteger code;                          // 错误码
```

```
@property (nonatomic, copy) NSString * _Nonnull message;          // 错误信息
@property (nonatomic, copy) NSString * _Nonnull localDescription;// 错误描述
@end
```

错误码对照表参见 SDK 错误码章节。

接入示例

1. 引入头文件。

```
#import <WBNFCReaderService/WBNFCReaderService-Swift.h>
```

2. 调起 SDK 并接收回调。

```
[[WBOCRReaderService sharedInstance] startReaderServiceWith:param
                                                    fromVC:self
                                            loginSucceedBlock:^{
                                              // 1. SDK 登录成功回调
} readSucceedBlock:^(WBOCRReaderResult * _Nonnull result) {
                                              // 2. SDK 识别成功回调
} failedBlock:^(WBOCRReadError * _Nonnull error) {
                                              // 3. SDK 识别异常回调
}];
```

详细接入代码，请参考 SDK 附的 NFCOCRDevDemo 工程。

常见问题 Q&A

1. dyld: Library not loaded
   集成 SDK 之后的 App，在低版本手机上运行，App 启动时候，可能会报如下错误，App 表现为 crash。

```
dyld: Library not loaded: @rpath/xxx/WBNFCReaderService
dyld: Library not loaded: @rpath/WBNFCReaderService.framework/WBNFCReaderService
  Referenced from: /var/containers/Bundle/Application/07B7698F-2A05-45C5-B07B-3C9BA532CF7E/Demo.app/Demo
  Reason: image not found
```

   出现这个问题，请参考 手动集成 步骤1 的配置。
2. dyld: Library not loaded: @rpath/libswiftCore.dylib

```
dyld: Library not loaded: @rpath/libswiftCore.dylib
  Referenced from: /private/var/containers/Bundle/Application/D775FC77-5694-4B8B-B6D7-
17F0D04DB960/Demo.app/Frameworks/WBNFCReaderService.framework/WBNFCReaderService
  Reason: image not found
```

   出现这个问题，请参考 手动集成 步骤2 的配置。
3. 在支持 NFC 的设备上，报错 Device doesn't support NFC tag reading。
   在 iPhone 7 及以上，iOS14.5 及以上 的设备上运行 SDK，仍然报 Device doesn't support NFC tag reading 错误。日志如下：

```
2022-06-09 16:26:57.222541+0800 Demo[11049:1234104] [CoreNFC] -[NFCHardwareManager
areFeaturesSupported:outError:]:166 XPC Error: Error Domain=NSCocoaErrorDomain Code=4099 "The connection to
service named com.apple.nfcd.service.corenfc was invalidated from this process." UserInfo=
{NSDebugDescription=The connection to service named com.apple.nfcd.service.corenfc was invalidated from this
process.}
2022-06-09 16:26:57.328858+0800 Demo[11049:1234104] [CoreNFC] -[NFCHardwareManager
areFeaturesSupported:outError:]:166 XPC Error: Error Domain=NSCocoaErrorDomain Code=4099 "The connection to
service named com.apple.nfcd.service.corenfc was invalidated: failed at lookup with error 159 - Sandbox
restriction."
```

   请参考本文档的 项目基础配置 检查配置项。

实证 NFC 查询结果

最近更新时间：2024-06-17 17:09:41

合作伙伴服务端验证结果

此方式用于：合作伙伴服务端生成签名，并调用实证 NFC 服务端查询结果，鉴权完成后返回结果（服务端上送 orderNo 和 appId 查询）。

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 获取 SIGN ticket 。
合作方为实证 NFC 识别服务生成签名，需要具有以下参数：

参数	说明	来源
appId	腾讯服务分配的 Appid	参考 获取 WBappid 指引在人脸核身控制台内申请
orderNo	订单号，本次 NFC 识别合作伙伴上送的订单号，唯一标识，字母/数字组成的字符串	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 获取 SIGN ticket
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 appId、orderNo、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 。

实证 NFC 识别结果查询接口

请求

请求 URL：`https://kyc1.qcloud.com/api/v2/nfcpaas/getIdcardNfcResult?orderNo=xxx`

> **注意**
> 为方便查询耗时，该请求 url 后面请拼接 orderNo 订单号参数。

请求方法：POST。
报文格式：`Content-Type: application/json` 。
请求参数：

参数	说明	类型	长度（字节）	是否必填
appId	腾讯服务分配的 Appid	字符串	腾讯服务分配	是
orderNo	订单号，合作方订单的唯一标识，字母/数字组成的字符串	字符串	32	是
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

reqId	本次实证 NFC读取唯一标识	字符串	40	是
getPhoto	是否需要获取 NFC 识别的证件图片文件，值为1则返回图片	字符串	1	否

响应

响应参数：

参数	类型	说明
code	String	0：成功；非0：失败；详情请参见 SaaS 服务错误码
msg	String	请求结果描述
result	NfcQueryRsp	nfc 查询加密结果集
transactionTime	String	交易时间戳

NfcQueryRsp 响应参数结构：

参数	类型	说明
code	String	0：成功；非0：失败；详情请参见 SaaS 服务错误码
msg	String	请求结果描述
nfcEnResult	String	加密的 nfc 识别结果
transactionTime	String	交易时间戳

nfcEnResult 响应参数结构内容：

参数	类型	说明	居民身份证	港澳回乡证
orderNo	string	订单编号	✓	✓
reqId	string	本次实证 NFC 读取唯一标识	✓	✓
name	string	姓名	✓	✓
enName	string	英文名	–	✓
sex	string	性别	✓	✓
nation	string	民族	✓	–
birth	string	出生日期	✓	✓
address	string	地址	✓	–
idcard	string	证件号	✓	✓
validDateBegin	string	证件的有效期起始时间	✓	–
validDateEnd	string	证件的有效期结束时间	✓	✓
signingOrganization	string	发证机关	✓	–
frontPhoto	Base 64 string	证件正面照	✓	–
backPhoto	Base 64 string	证件反面照	✓	–
portraitPhoto	Base 64 string	证件人像照	✓	✓


```
{
    "code": "0",
    "msg": "请求成功",
    "result": {
"bizSeqNo": "1607280FD011410000000000000009003",
```

```
"nfcEnResult":SgoeKZt5nIHQmiLluNxxsdfasdfasdfasdfa1uOswIiebhOPe0SgoeKZt5nIHQmiLluN123123MdsdfKsmiLluN123123Mdfs
IHQmiLluN123123MdsdfKdf1uOswIiebhOPe0SgoeKZt5nIH"
},
    "bizSeqNo": "1607280FD011410000000000000009003",
    "transactionTime": "20160728075617"
}
```


注意

- 照片均为 base64 位编码，其中照片解码后格式一般为 JPG。
- 返回报文加密nfcEnResult部分，解密后为响应参数。

解密示例：

第一步先 base64解码 nfcEnResult 字段，第二步通过 SM4国密算法根据下发的处理后密钥(主力方式：将十六进制密钥转为字节数组)解密第一步解码结果，第三步将第二步结果 json 解析为 nfcEnResult 解密结构内容。

解密代码示例(JAVA)：

```
import com.google.common.io.BaseEncoding;
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
import java.security.Key;
import java.security.Security;
import java.util.Random;
/**
 *     项目所依赖的类：
 *      List utils = [
 *      "commons-io:commons-io:2.7",
 *      "com.google.guava:guava:16.0.1",
 *      "org.bouncycastle:bcprov-jdk15to18:1.66",
 *      ]
 *
 * @date 2022-03-04-10:27
 */
public class SM4 {
    private static final String ENCODING = "UTF-8";
    public static final String ALGORITHM_NAME = "SM4";
    // 加密算法/分组加密模式/分组填充方式
    // PKCS5Padding-以8个字节为一组进行分组加密
    // 定义分组加密模式使用：PKCS5Padding
    public static final String ALGORITHM_NAME_ECB_PADDING = "SM4/ECB/PKCS5Padding";
    /**
     * 生成 SM4 Cipher
     * @return
     * @throws Exception
     */
    protected static Cipher generateCipher(int mode, byte[] keyData) throws Exception {
        Cipher cipher = Cipher.getInstance(ALGORITHM_NAME_ECB_PADDING, BouncyCastleProvider.PROVIDER_NAME);
        Key sm4Key = new SecretKeySpec(keyData, ALGORITHM_NAME);
        cipher.init(mode, sm4Key);
        return cipher;
    }
    /**
     * 加密模式之
     * @param key  密钥
     * @param data 待加密的内容
     * @return
     * @throws Exception
     * @explain
     */
    public static byte[] encrypt(byte[] key, byte[] data) throws Exception {
        Cipher cipher = generateCipher(Cipher.ENCRYPT_MODE, key);
```

```
        return cipher.doFinal(data);
    }
    /**
     * sm4解密
     * @param
     * @param cipherText 16进制的加密字符串（忽略大小写）
     * @return 解密后的字符串
     * @throws Exception
     * @explain 解密模式：采用ECB
     */
    public static String decrypt(String key, String cipherText) throws Exception {
        // 用于接收解密后的字符串
        String decryptStr = "";
        byte[] keyData = key.getBytes();
        byte[] cipherData = cipherText.getBytes();
        // 解密
        byte[] srcData = decrypt(keyData, cipherData);
        // byte[]-->String
        decryptStr = new String(srcData, ENCODING);
        return decryptStr;
    }
    /**
     * 解密
     * @param key
     * @param cipherText
     * @return 解密后的内容  byte[]
     * @throws Exception
     * @explain
     */
    public static byte[] decrypt(byte[] key, byte[] cipherText) throws Exception {
        Cipher cipher = generateCipher(Cipher.DECRYPT_MODE, key);
        return cipher.doFinal(cipherText);
    }
    /**
     * 将byte转换为16进制字符串
     * @param src
     * @return
     */
    public static String byteToHexString(byte[] src) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < src.length; i++) {
            int v = src[i] & 0xff;
            String hv = Integer.toHexString(v);
            if (hv.length() < 2) {
                sb.append("0");
            }
            sb.append(hv);
        }
        return sb.toString();
    }
    /**
     * 将16进制字符串转换为字节数组
     * @param str 16进制字符串
     * @return byte[]
     */
    public static byte[] hexStringToBytes(String str) {
        byte[] bytes;
        bytes = new byte[str.length() / 2];
        for (int i = 0; i < bytes.length; i++) {
            bytes[i] = (byte) Integer.parseInt(str.substring(2 * i, 2 * i + 2), 16);
        }
        return bytes;
    }
```

```
    /**
     * 接入方拿到key（即控制台申请的 NFC 秘钥） 先可以跑通下面的案例，并认真阅读 1，2，3，4
     * 项目所依赖的类：
     * List utils = [
     * "commons-io:commons-io:2.7",
     * "com.google.guava:guava:16.0.1",
     * "org.bouncycastle:bcprov-jdk15to18:1.66",
     * ]
     * @param args
     * @throws Exception
     */
    public static void main(String[] args) throws Exception {
        // 1. 系统初始话的时候，需要注入所使用的 provider
        Security.addProvider(new BouncyCastleProvider());
        // 2.收到key后，先从hex转成byte[]，加解密就是用byte[]输入
        byte[] key = hexStringToBytes("7F26724F72D48EBA5AD3848B30B81122");
        // 3.测试的文本内容
        String src = "168，FM973 全角半角测试，你好，大家好，你好，美丽人生Ｅｆｇｋｙｍｋ";
        byte[] enSrc = encrypt(key, src.getBytes());
        String base64EnStr = BaseEncoding.base64().encode(enSrc);
        System.out.println("base64 en:" + base64EnStr);
        // 4.解密，用户只需要执行解密即可（先base64 转成byte[] 然后执行解密）
        byte[] decSrc = decrypt(key, BaseEncoding.base64().decode(base64EnStr));
        System.out.println("===== base64 dec:" + new String(decSrc, ENCODING));
    }
}
```

计费错误码
Plus 版人脸核身服务错误码

最近更新时间：2025-02-07 14:41:42

本文只列出 Plus 版人脸核身服务相关返回码及其对应的计费关系。增强版、基础版、E 证通、意愿核身、实证 NFC 等的服务返回码及其他对应计费关系请查阅 增强版人脸核身服务错误码 、基础版人脸核身服务错误码 、E 证通服务错误码 、意愿核身服务错误码 、实证 NFC 服务错误码 。

微信小程序/H5（微信公众号）/H5（微信浏览器）返回码

以下为微信小程序、H5（微信公众号）、H5（微信浏览器）的返回码及收费关系：

系统返回码

返回码	含义	是否收费
0	成功	是
1	调用方式出错，请参考人脸核身 DetectAuth 文档进行调用	否
2	调用方式出错，请参考人脸核身 DetectAuth 文档进行调用	否
3	调用方式出错，请参考人脸核身 DetectAuth 文档进行调用	否
4	服务异常，请稍后重试或联系客服人员	否
8	服务异常，请稍后重试或联系客服人员	否
10	服务异常，请稍后重试或联系客服人员	否
12	服务异常，请稍后重试或联系客服人员	否
13	服务异常，请稍后重试或联系客服人员	否
14	本次校验已完成	否
15	token 过期，请重试	否
16	已达到重试上限	否
17	本次流程未完成	否

微信相关返回码

返回码	含义	是否收费
201	从微信拉取图片失败，请重试	否
202	视频获取失败，请重试	否
203	无此 API 调用权限	否
204	当前公众号/小程序未完成对慧眼授权，请参照慧眼接入文档进行扫码授权，如有疑问可微信联系慧眼小助手（faceid001）	否
301	当前认证人身份证号码与业务传入验证证件号码不符，请重新上传	否
302	身份证已过期，请使用有效身份证进行核验	否
303	扫描识别失败，请重新拍摄	否

活体检测返回码

活体通用返回码

返回码	含义	是否收费
1002	疑似非真人录制	是

1001	引擎超时，请稍后重试	否
1003	人脸识别验证未成功，请重试	否
1004	人脸验证失败，请重试	否
1005	人脸验证失败，请重试	否
1006	证照库出错，请重试	否
1007	未拉取微信活体结果或拉取微信活体结果失败	否

证件图像比对返回码

返回码	含义	是否收费
2006	姓名和身份证号不匹配，请核实身份后重试	是
2016	无法判断为同一人，请确认身份后重试（相似度未达到通过阈值）	是
2001	引擎超时，请稍后重试	否
2002	姓名有误，请确认后重试	否
2003	身份证号有误，请确认后重试	否
2004	视频提取照片出错，请重试或更换手机进行验证	否
2005	未查询到身份信息，请通过其他渠道验证	否
2007	未查询到照片信息，请通过其他渠道验证	否
2008	未查询到照片信息，请通过其他渠道验证	否
2009	查询到照片分辨率过低，请通过其他渠道验证	否
2010	脸部未完整露出，请保持正脸面对屏幕	否
2011	无法判断为同一人，请确认身份后重试（疑似非活体）	否
2012	检测到多张人脸，请保持验证视频中只有本人	否
2013	脸部未完整露出，请保持正脸面对屏幕	否
2014	视频像素太低无法检测，请更换手机进行验证	否
2015	无法判断为同一人，请确认身份后重试（比对失败）	否
2017	权威数据源升级中，请稍后再试	否
2018	操作过于频繁，请第二天0点后重试	否

App SDK/ H5（移动端浏览器）返回码

以下为 App SDK、H5（移动端浏览器）的返回码及收费关系：

返回码	返回信息	是否收费
0	认证通过	是
66660004	无法确认为同一人	是
66660010	姓名和身份证不一致，请确认	是
-5008	声音未能识别，请大声慢读一遍数字	否
-5018	未识别到指定动作	否
-5023	光线太强，请到室内识别	否
999999	网络不给力，请稍后重试	否

66660000	请求参数异常	否
66660002	服务不可用（已欠费或主动停服，请联系技术支持确认）	否
66660003	试用版最大刷脸次数已超	否
66660005	此证件信息不支持校验	否
66660011	未完成认证或认证结果已过期	否
66660015	姓名或身份证号格式不正确	否
66660016	视频或图片异常	否
66660017	验证次数超限	否
66660018	操作超时，请退出重试	否
66660023	疑似非真人，请确保本人操作且正脸对框	是
66660035	未识别到人脸，请确保正脸对框且清晰完整	否
66660037	照片出现多张脸	否
66660041	脸部有遮挡或闭眼，请重试	否
66660044	应用服务版本过低，请升级	否
66660046	比对源照片无法处理	否
66662001	无此授权结果	否
66665001	疑似非真人，请确保本人操作且正脸对框	是
400101	签名校验不通过	否
400103	IP 未加入白名单列表（请线下联系技术支持进行配置）	否
400106	不合法请求	否
400506	请求频率过高，请稍后再试	否
400600	微信下载视频失败，请重试	否
400601	视频或图片异常	否

增强级 riskinfo 返回码说明

返回码	说明
deviceInfoLevel	0、1、2等级为低风险设备，3、4为高风险设备，会进行拦截

增强版人脸核身服务错误码

最近更新时间：2025-02-07 14:41:42

本文只列出增强版人脸核身服务相关返回码及其对应的计费关系。Plus 版、基础版、E 证通、意愿核身、实证 NFC 等的服务返回码及其他对应计费关系请查阅 Plus 版人脸核身服务错误码 、基础版人脸核身服务错误码 、E 证通服务错误码 、意愿核身服务错误码 、实证 NFC 服务错误码 。

微信小程序/H5（微信公众号）/H5（微信浏览器）返回码

以下为 H5（微信公众号）、H5（微信浏览器）、微信小程序的返回码及收费关系：

系统返回码

返回码	含义	是否收费
0	成功	是
1	调用方式出错，请参考人脸核身 DetectAuth 文档进行调用	否
2	调用方式出错，请参考人脸核身 DetectAuth 文档进行调用	否
3	调用方式出错，请参考人脸核身 DetectAuth 文档进行调用	否
4	服务异常，请稍后重试或联系客服人员	否
8	服务异常，请稍后重试或联系客服人员	否
10	服务异常，请稍后重试或联系客服人员	否
12	服务异常，请稍后重试或联系客服人员	否
13	服务异常，请稍后重试或联系客服人员	否
14	本次校验已完成	否
15	token 过期，请重试	否
16	已达到重试上限	否
17	本次流程未完成	否

微信相关返回码

返回码	含义	是否收费
201	从微信拉取图片失败，请重试	否
202	视频获取失败，请重试	否
203	无此 API 调用权限	否
204	当前公众号/小程序未完成对慧眼授权，请参照慧眼接入文档进行扫码授权，如有疑问可微信联系慧眼小助手（faceid001）	否
301	当前认证人身份证号码与业务传入验证证件号码不符，请重新上传	否
302	身份证已过期，请使用有效身份证进行核验	否
303	扫描识别失败，请重新拍摄	否

活体检测返回码

活体通用返回码

返回码	含义	是否收费
1001	引擎超时，请稍后重试	否
1002	疑似非真人录制	否

1003	人脸识别验证未成功，请重试	否
1004	人脸验证失败，请重试	否
1005	人脸验证失败，请重试	否
1006	证照库出错，请重试	否
1007	未拉取微信活体结果或拉取微信活体结果失败	否

证件图像比对返回码

返回码	含义	是否收费
2006	姓名和身份证号不匹配，请核实身份后重试	是
2016	无法判断为同一人，请确认身份后重试（相似度未达到通过阈值）	是
2001	引擎超时，请稍后重试	否
2002	姓名有误，请确认后重试	否
2003	身份证号有误，请确认后重试	否
2004	视频提取照片出错，请重试或更换手机进行验证	否
2005	未查询到身份信息，请通过其他渠道验证	否
2007	未查询到照片信息，请通过其他渠道验证	否
2008	未查询到照片信息，请通过其他渠道验证	否
2009	查询到照片分辨率过低，请通过其他渠道验证	否
2010	脸部未完整露出，请保持正脸面对屏幕	否
2011	无法判断为同一人，请确认身份后重试（疑似非活体）	否
2012	检测到多张人脸，请保持验证视频中只有本人	否
2013	脸部未完整露出，请保持正脸面对屏幕	否
2014	视频像素太低无法检测，请更换手机进行验证	否
2015	无法判断为同一人，请确认身份后重试（比对失败）	否
2017	权威数据源升级中，请稍后再试	否
2018	操作过于频繁，请第二天0点后重试	否

App SDK/ H5（移动端浏览器）返回码

以下为 APPSDK、H5（移动端浏览器）的返回码及收费关系：

返回码	返回信息	是否收费
0	认证通过	是
66660004	无法确认为同一人	是
66660010	姓名和身份证不一致，请确认	是
-5008	声音未能识别，请大声慢读一遍数字	否
-5018	未识别到指定动作	否
-5023	光线太强，请到室内识别	否
999999	网络不给力，请稍后重试	否
66660000	请求参数异常	否

66660002	服务不可用（已欠费或主动停服，请联系技术支持确认）	否
66660003	试用版最大刷脸次数已超	否
66660005	此证件信息不支持校验	否
66660011	未完成认证或认证结果已过期	否
66660015	姓名或身份证号格式不正确	否
66660016	视频或图片异常	否
66660017	验证次数超限	否
66660018	操作超时，请退出重试	否
66660023	疑似非真人，请确保本人操作且正脸对框	否
66660035	未识别到人脸，请确保正脸对框且清晰完整	否
66660037	照片出现多张脸	否
66660041	脸部有遮挡或闭眼，请重试	否
66660044	应用服务版本过低，请升级	否
66660046	比对源照片无法处理	否
66662001	无此授权结果	否
66665001	疑似非真人，请确保本人操作且正脸对框	否
400101	签名校验不通过	否
400103	IP 未加入白名单列表（请线下联系技术支持进行配置）	否
400106	不合法请求	否
400506	请求频率过高，请稍后再试	否
400600	微信下载视频失败，请重试	否
400601	视频或图片异常	否

增强级 riskinfo 返回码说明

返回码	说明
deviceInfoLevel	0、1、2等级为低风险设备，3、4为高风险设备，会进行拦截

基础版人脸核身服务错误码

最近更新时间：2025-01-15 17:14:53

本文只列出基础版人脸核身服务相关返回码及其对应的计费关系。Plus 版、增强版、E 证通、意愿核身、实证 NFC 等的服务返回码及其他对应计费关系请查阅 Plus 版人脸核身服务错误码 、增强版人脸核身服务错误码 、E 证通服务错误码 、意愿核身服务错误码 、实证 NFC 服务错误码 。

微信小程序/H5（微信公众号）/H5（微信浏览器）返回码

以下为微信小程序、 H5（微信公众号）、H5（微信浏览器）的返回码及收费关系：

系统返回码

返回码	含义	是否收费
0	成功	是
1	调用方式出错，请参考人脸核身 DetectAuth 文档进行调用	否
2	调用方式出错，请参考人脸核身 DetectAuth 文档进行调用	否
3	调用方式出错，请参考人脸核身 DetectAuth 文档进行调用	否
4	服务异常，请稍后重试或联系客服人员	否
8	服务异常，请稍后重试或联系客服人员	否
10	服务异常，请稍后重试或联系客服人员	否
12	服务异常，请稍后重试或联系客服人员	否
13	服务异常，请稍后重试或联系客服人员	否
14	本次校验已完成	否
15	token 过期，请重试	否
16	已达到重试上限	否
17	本次流程未完成	否

微信相关返回码

返回码	含义	是否收费
201	从微信拉取图片失败，请重试	否
202	视频获取失败，请重试	否
203	无此 API 调用权限	否
204	当前公众号/小程序未完成对慧眼授权，请参照慧眼接入文档进行扫码授权，如有疑问可微信联系慧眼小助手（faceid001）	否
301	当前认证人身份证号码与业务传入验证证件号码不符，请重新上传	否
302	身份证已过期，请使用有效身份证进行核验	否
303	扫描识别失败，请重新拍摄	否

活体检测返回码

活体通用返回码

返回码	含义	是否收费
1001	引擎超时，请稍后重试	否
1002	疑似非真人录制	否

1003	人脸识别验证未成功，请重试	否
1004	人脸验证失败，请重试	否
1005	人脸验证失败，请重试	否
1006	证照库出错，请重试	否
1007	未拉取微信活体结果或拉取微信活体结果失败	否

数字活体返回码

返回码	含义	是否收费
1101	声音未能识别，请大声慢读一遍数字	否
1102	脸部未完整露出，请保持正脸面对屏幕	否
1103	声音未能识别，请大声慢读一遍数字	否
1104	视频格式出错无法检测，请更换手机进行验证	否
1105	视频拉取失败，请重试	否
1106	声音太小，请大声慢读一遍数字	否
1107	视频为空，或大小不合适，请控制录制时长在6s左右	否
1108	视频像素太低无法检测，请更换手机进行验证	否
1109	未识别到唇动，请大声慢读一遍数字	否

动作活体返回码

返回码	含义	是否收费
1201	光线不佳，建议在光源充足的室内环境验证	否
1202	光线不佳，建议在光源充足的室内环境验证	否
1203	离屏幕太近或太远，请确保人脸清晰完整	否
1204	离屏幕太近或太远，请确保人脸清晰完整	否
1205	离屏幕太近或太远，请确保人脸清晰完整	否
1206	离屏幕太近或太远，请确保人脸清晰完整	否
1207	动作未能识别，请按顺序缓慢做动作	否
1208	动作未能识别，请按顺序缓慢做动作	否
1209	动作未能识别，请按顺序缓慢做动作	否
1210	动作未能识别，请按顺序缓慢做动作	否

静默活体返回码

返回码	含义	是否收费
1301	疑似非真人录制，请重试	否
1302	人脸验证失败，请重试	否
1303	视频录制时间过短，请重新录制2秒以上的视频	否

证件图像比对返回码

返回码	含义	是否收费

2006	姓名和身份证号不匹配，请核实身份后重试	是
2016	无法判断为同一人，请确认身份后重试（相似度未达到通过阈值）	是
2001	引擎超时，请稍后重试	否
2002	姓名有误，请确认后重试	否
2003	身份证号有误，请确认后重试	否
2004	视频提取照片出错，请重试或更换手机进行验证	否
2005	未查询到身份信息，请通过其他渠道验证	否
2007	未查询到照片信息，请通过其他渠道验证	否
2008	未查询到照片信息，请通过其他渠道验证	否
2009	查询到照片分辨率过低，请通过其他渠道验证	否
2010	脸部未完整露出，请保持正脸面对屏幕	否
2011	无法判断为同一人，请确认身份后重试（疑似非活体）	否
2012	检测到多张人脸，请保持验证视频中只有本人	否
2013	脸部未完整露出，请保持正脸面对屏幕	否
2014	视频像素太低无法检测，请更换手机进行验证	否
2015	无法判断为同一人，请确认身份后重试（比对失败）	否
2017	权威数据源升级中，请稍后再试	否
2018	操作过于频繁，请第二天0点后重试	否

App SDK/ H5（移动端浏览器）返回码

以下为 APPSDK、H5（移动端浏览器）的返回码及收费关系：

返回码	返回信息	是否收费
0	认证通过	是
66660004	无法确认为同一人	是
66660010	姓名和身份证不一致，请确认	是
-5008	声音未能识别，请大声慢读一遍数字	否
-5018	未识别到指定动作	否
-5023	光线太强，请到室内识别	否
999999	网络不给力，请稍后重试	否
66660000	请求参数异常	否
66660002	服务不可用（已欠费或主动停服，请联系技术支持确认）	否
66660003	试用版最大刷脸次数已超	否
66660005	此证件信息不支持校验	否
66660011	未完成认证或认证结果已过期	否
66660015	姓名或身份证号格式不正确	否
66660016	视频或图片异常	否
66660017	验证次数超限	否

66660018	操作超时，请退出重试	否
66660023	请确保本人操作且正脸对框	否
66660035	未识别到人脸，请确保正脸对框且清晰完整	否
66660037	照片出现多张脸	否
66660041	脸部有遮挡或闭眼，请重试	否
66660044	应用服务版本过低，请升级	否
66660046	比对源照片无法处理	否
66662001	无此授权结果	否
66665001	请确保本人操作且正脸对框	否
400101	签名校验不通过	否
400103	IP 未加入白名单列表（请线下联系技术支持进行配置）	否
400106	不合法请求	否
400506	请求频率过高，请稍后再试	否
400600	微信下载视频失败，请重试	否
400601	视频或图片异常	否

E 证通服务错误码

最近更新时间：2025-01-15 17:14:53

本文只列出 E 证通服务相关返回码及其对应的计费关系。Plus 版、增强版、基础版、意愿核身、实证 NFC 等的服务返回码及其他对应计费关系请查阅 Plus 版人脸核身服务错误码 、增强版人脸核身服务错误码 、基础版人脸核身服务错误码 、意愿核身服务错误码 、实证 NFC 服务错误码 。

小程序返回码

以下为微信小程序、uni-app（小程序）方式接入 E 证通的返回码及收费关系：

系统返回码

返回码	含义	是否收费
0	成功	是
1	调用方式出错，请参考人脸核身 DetectAuth 文档进行调用	否
2	调用方式出错，请参考人脸核身 DetectAuth 文档进行调用	否
3	调用方式出错，请参考人脸核身 DetectAuth 文档进行调用	否
4	服务异常，请稍后重试或联系客服人员	否
8	服务异常，请稍后重试或联系客服人员	否
10	服务异常，请稍后重试或联系客服人员	否
12	服务异常，请稍后重试或联系客服人员	否
13	服务异常，请稍后重试或联系客服人员	否
14	本次校验已完成	否
15	token 过期，请重试	否
16	已达到重试上限	否
17	本次流程未完成	否

微信相关返回码

返回码	含义	是否收费
201	从微信拉取图片失败，请重试	否
202	视频获取失败，请重试	否
203	无此 API 调用权限	否
204	当前公众号/小程序未完成对慧眼授权，请参照慧眼接入文档进行扫码授权，如有疑问可微信联系慧眼小助手（faceid001）	否
301	当前认证人身份证号码与业务传入验证证件号码不符，请重新上传	否
302	身份证已过期，请使用有效身份证进行核验	否
303	扫描识别失败，请重新拍摄	否

活体检测返回码

活体通用返回码

返回码	含义	是否收费
1001	引擎超时，请稍后重试	否
1002	疑似非真人录制	否

1003	人脸识别验证未成功，请重试	否
1004	人脸验证失败，请重试	否
1005	人脸验证失败，请重试	否
1006	证照库出错，请重试	否
1007	未拉取微信活体结果或拉取微信活体结果失败	否

证件图像比对返回码

返回码	含义	是否收费
2006	姓名和身份证号不匹配，请核实身份后重试	是
2016	无法判断为同一人，请确认身份后重试（相似度未达到通过阈值）	是
2001	引擎超时，请稍后重试	否
2002	姓名有误，请确认后重试	否
2003	身份证号有误，请确认后重试	否
2004	视频提取照片出错，请重试或更换手机进行验证	否
2005	未查询到身份信息，请通过其他渠道验证	否
2007	未查询到照片信息，请通过其他渠道验证	否
2008	未查询到照片信息，请通过其他渠道验证	否
2009	查询到照片分辨率过低，请通过其他渠道验证	否
2010	脸部未完整露出，请保持正脸面对屏幕	否
2011	无法判断为同一人，请确认身份后重试（疑似非活体）	否
2012	检测到多张人脸，请保持验证视频中只有本人	否
2013	脸部未完整露出，请保持正脸面对屏幕	否
2014	视频像素太低无法检测，请更换手机进行验证	否
2015	无法判断为同一人，请确认身份后重试（比对失败）	否
2017	权威数据源升级中，请稍后再试	否
2018	操作过于频繁，请第二天0点后重试	否

意愿核身服务错误码

最近更新时间：2025-01-15 17:14:53

本文只列出意愿核身服务相关返回码及其对应的计费关系。Plus 版、增强版、基础版、E 证通、实证 NFC 等的服务返回码及其他对应计费关系请查阅 Plus 版人脸核身服务错误码 、增强版人脸核身服务错误码 、基础版人脸核身服务错误码 、E 证通服务错误码 、实证 NFC 服务错误码 。

微信小程序/E 证通/H5（微信浏览器）返回码

以下为 微信小程序、E 证通、H5（微信浏览器）方式接入的的返回码及收费关系：

系统返回码

返回码	含义	是否收费
0	成功	是
1	调用方式出错，请参考人脸核身 DetectAuth 文档进行调用	否
2	调用方式出错，请参考人脸核身 DetectAuth 文档进行调用	否
3	调用方式出错，请参考人脸核身 DetectAuth 文档进行调用	否
4	服务异常，请稍后重试或联系客服人员	否
8	服务异常，请稍后重试或联系客服人员	否
10	服务异常，请稍后重试或联系客服人员	否
12	服务异常，请稍后重试或联系客服人员	否
13	服务异常，请稍后重试或联系客服人员	否
14	本次校验已完成	否
15	token 过期，请重试	否
16	已达到重试上限	否
17	本次流程未完成	否

微信相关返回码

返回码	含义	是否收费
201	从微信拉取图片失败，请重试	否
202	视频获取失败，请重试	否
203	无此 API 调用权限	否
204	当前公众号/小程序未完成对慧眼授权，请参照慧眼接入文档进行扫码授权，如有疑问可微信联系慧眼小助手（faceid001）	否
301	当前认证人身份证号码与业务传入验证证件号码不符，请重新上传	否
302	身份证已过期，请使用有效身份证进行核验	否
303	扫描识别失败，请重新拍摄	否

活体检测返回码

活体通用返回码

返回码	含义	是否收费
1001	引擎超时，请稍后重试	否
1002	疑似非真人录制	否

1003	人脸识别验证未成功，请重试	否
1004	人脸验证失败，请重试	否
1005	人脸验证失败，请重试	否
1006	证照库出错，请重试	否
1007	未拉取微信活体结果或拉取微信活体结果失败	否

证件图像比对返回码

返回码	含义	是否收费
2006	姓名和身份证号不匹配，请核实身份后重试	是
2016	无法判断为同一人，请确认身份后重试（相似度未达到通过阈值）	是
2001	引擎超时，请稍后重试	否
2002	姓名有误，请确认后重试	否
2003	身份证号有误，请确认后重试	否
2004	视频提取照片出错，请重试或更换手机进行验证	否
2005	未查询到身份信息，请通过其他渠道验证	否
2007	未查询到照片信息，请通过其他渠道验证	否
2008	未查询到照片信息，请通过其他渠道验证	否
2009	查询到照片分辨率过低，请通过其他渠道验证	否
2010	脸部未完整露出，请保持正脸面对屏幕	否
2011	无法判断为同一人，请确认身份后重试（疑似非活体）	否
2012	检测到多张人脸，请保持验证视频中只有本人	否
2013	脸部未完整露出，请保持正脸面对屏幕	否
2014	视频像素太低无法检测，请更换手机进行验证	否
2015	无法判断为同一人，请确认身份后重试（比对失败）	否
2017	权威数据源升级中，请稍后再试	否
2018	操作过于频繁，请第二天0点后重试	否

App SDK/ H5（移动端浏览器）返回码

以下为 APPSDK、H5（移动端浏览器）方式接入的返回码及收费关系：

返回码	返回信息	是否收费
0	认证通过	是
66660004	无法确认为同一人	是
66660010	姓名和身份证不一致，请确认	是
-5008	声音未能识别，请大声慢读一遍数字	否
-5018	未识别到指定动作	否

-5023	光线太强，请到室内识别	否
999999	网络不给力，请稍后重试	否
66660000	请求参数异常	否
66660002	服务不可用（已欠费或主动停服，请联系技术支持确认）	否
66660003	试用版最大刷脸次数已超	否
66660005	此证件信息不支持校验	否
66660011	未完成认证或认证结果已过期	否
66660015	姓名或身份证号格式不正确	否
66660016	视频或图片异常	否
66660017	验证次数超限	否
66660018	操作超时，请退出重试	否
66660023	请确保本人操作且正脸对框	否
66660035	未识别到人脸，请确保正脸对框且清晰完整	否
66660037	照片出现多张脸	否
66660041	脸部有遮挡或闭眼，请重试	否
66660044	应用服务版本过低，请升级	否
66660046	比对源照片无法处理	否
66662001	无此授权结果	否
66665001	请确保本人操作且正脸对框	否
400101	签名校验不通过	否
400103	IP 未加入白名单列表（请线下联系技术支持进行配置）	否
400106	不合法请求	否
400506	请求频率过高，请稍后再试	否
400600	【微信】下载视频失败，请重试	否
400601	视频或图片异常	否

意愿核身返回码

返回码	返回信息	处理措施	是否收费
66666004	音频文件下载失败，请重试	下载用户 answer 音频或播报音频失败	否
66666006	意愿表达结果不正确	用户回答音频 asr 识别和标准答案不一致	是
66666010	当前订单识别结果不存在	查询结果时，意愿表达记录没有结果	否

实证 NFC 服务错误码

最近更新时间：2025-01-03 17:58:52

本文只列出实证 NFC 服务相关返回码及其对应的计费关系。Plus版、增强版、基础版、E 证通、意愿核身等的服务返回码及其他对应计费关系请查阅 Plus 版人脸核身服务错误码 、增强版人脸核身服务错误码 、基础版人脸核身服务错误码 、E 证通服务错误码 、意愿核身服务错误码 。

实证 NFC 服务端返回码

返回码	返回信息	处理措施	是否收费
0	请求成功	相同的 reqId 有效期为5分钟，有效期内重复查询不重复计费	是
66664002	ocrCertId 已失效，请校验参数	ocrCertId 有效期为5分钟，重新获取	否
66664003	请求频率过高，请稍后重试	相同的 reqId 有效期为5分钟内仅支持10次以内的重复查询	否
66664004	该 reqId 不存在，请校验参数	该 reqId 不存在，请校验参数	否
66664005	reqId 已失效或不存在	reqId 的查询有效期为5分钟	否
66664006	网络不给力,请稍后重试	联系技术支持	否
66664007	网络不给力,请稍后重试	联系技术支持	否

实证 NFC 终端返回码

返回码	返回信息	处理措施	是否收费
0	识别成功	识别成功	是
600100	该设备不支持 NFC 功能	NFC 读取证件需要 iPhone 7 及以上，iOS14.5 及以上	否
600101	用户退出本次识别	用户点击返回/退出按钮，主动退出 SDK	否
600102	请求参数异常，请校验参数	登录参数校验不通过	否
600103	本次识别超时，请重试	超过重试的次数，仍然未识别到结果	否
600104	网络不给力，请稍后重试	网络异常 Error，请求未到达服务器	否
600106	初始化失败，请重试	Android 特有	否
600105	网络不给力，请重试	enmsg 为空/enmsg 解码失败	否
66664020	读卡时勿移动卡片，请重试	读卡时请勿移动，请将证件对准 NFC 区域，重新读卡	否
66664021	设备不支持 NFC，请更换手机重试	设备不支持 typeb 卡，或检测卡类型错误会出现，个别安卓手机会出现	否
66664022	网络不给力,请稍后重试	网络异常，重新读卡	否
66664023	设备 NFC 异常，请重启 NFC 后重试	NFC 连接异常，重新读卡	否
66664024	证件类型不正确，请重试	识读的证件类型不对，例如贴银行卡进行识读，重新读卡	否
66664025	该设备不支持 NFC 功能	设备不支持 NFC 功能，更换手机识读	否
66664026	读卡时勿移动卡片，请重试	iOS nfc 弹窗超时消失情况下出现此结果码，请重试	否
66664027/66664029	网络不给力,请稍后重试	网络服务异常，联系客服	否
66664028/66664030	请求参数异常，请重试	调试参数异常，联系客服	否

核身及要素 API 服务错误码

最近更新时间：2025-01-03 17:58:52

本文只列出 AI人脸防护盾、人脸核验类、要素类相关返回码及其对应的计费关系。Plus 版、增强版、基础版、E 证通、意愿核身、实证 NFC 等的服务返回码及其对应计费关系请查阅 Plus 版人脸核身服务错误码 、增强版人脸核身服务错误码 、基础版人脸核身服务错误码 、E 证通服务错误码 、意愿核身服务错误码 、实证 NFC 服务错误码 。

AI 人脸防护盾

以下仅列出了接口业务逻辑相关的错误码，其他错误码详见 公共错误码 。

错误码	描述	是否收费
Success	成功	是
FailedOperation.CoveredFace	图中人脸存在遮挡，请传入无遮挡人脸图片	否
FailedOperation.DetectEngineSystemError	服务引擎调用失败，请重试	否
FailedOperation.ImageDecodeFailed	图片解码失败。	否
FailedOperation.IncompleteFace	未检测到完整人脸，请传入完整人脸图片	否
FailedOperation.PoorImageQuality	图片质量过差，请检查图片质量	否
FailedOperation.UnKnown	内部未知错误。	否
FailedOperation.VideoDecodeFailed	视频解码异常	否
FailedOperation.VideoDurationExceeded	视频时长过长，当前接口最大支持的视频时长为20s。	否
InvalidParameter	参数错误。	否
InvalidParameterValue	参数取值错误。	否
UnauthorizedOperation.Arrears	账号已欠费。	否
UnauthorizedOperation.ChargeStatusException	计费状态异常。	否
UnauthorizedOperation.Nonactivated	未开通服务。	否

人脸核验类

下表仅列出了活体人脸核身、活体人脸比对、照片人脸核身和身份证人像照片验真接口的收费返回码，其他未收费的接口错误码请参阅对应接口文档，完整错误码对照表请参考 API3.0 文档 > 错误码 。

返回码	含义	是否收费
Success	检测通过	是
FailedOperation.IdNameMisMatch	姓名和身份证号不一致，请核实后重试。	是
FailedOperation.CompareLowSimilarity	比对相似度未达到通过标准。	是
FailedOperation.IdPhotoPoorQuality	证件图片分辨率太低，请重新上传。	否
FailedOperation.LifePhotoDetectFaces	检测到多张人脸。	否
FailedOperation.LifePhotoDetectFake	实人比对没通过。	否
FailedOperation.LifePhotoDetectNoFaces	未能检测到完整人脸。	否
FailedOperation.CompareFail	比对失败。	否
FailedOperation.CoveredFace	图中人脸存在遮挡，请传入无遮挡人脸图片	否
FailedOperation.DetectEngineSystemError	服务引擎调用失败，请重试	否

FailedOperation.ImageDecodeFailed	图片解码失败。	否
FailedOperation.IncompleteFace	未检测到完整人脸，请传入完整人脸图片	否
FailedOperation.PoorImageQuality	图片质量过差，请检查图片质量	否
FailedOperation.UnKnown	内部未知错误。	否
FailedOperation.VideoDecodeFailed	视频解码异常	否
InvalidParameter	参数错误。	否
InvalidParameterValue	参数取值错误。	否
UnauthorizedOperation.Arrears	账号已欠费。	否
UnauthorizedOperation.ChargeStatusException	计费状态异常。	否
UnauthorizedOperation.Nonactivated	未开通服务。	否

要素类

身份证要素、银行卡要素和运营商要素的接口错误码及对应计费关系，请参阅各自接口文档。

接口名称	错误码及是否收费详见链接接口文档
身份信息认证（二要素核验）	https://cloud.tencent.com/document/product/1007/33188
身份证识别及信息核验	https://cloud.tencent.com/document/product/1007/37980
身份信息及有效期核验	https://cloud.tencent.com/document/product/1007/60075
银行卡基础信息查询	https://cloud.tencent.com/document/product/1007/47837
银行卡二要素核验	https://cloud.tencent.com/document/product/1007/35776
银行卡三要素核验	https://cloud.tencent.com/document/product/1007/33848
银行卡四要素核验	https://cloud.tencent.com/document/product/1007/35775
手机号在网时长核验	https://cloud.tencent.com/document/product/1007/40546
手机号状态查询	https://cloud.tencent.com/document/product/1007/40545
手机号三要素核验	https://cloud.tencent.com/document/product/1007/39765
手机号三要素核验（移动）	https://cloud.tencent.com/document/product/1007/68559
手机号三要素核验（电信）	https://cloud.tencent.com/document/product/1007/68558
手机号三要素核验（联通）	https://cloud.tencent.com/document/product/1007/68557
手机号二要素核验	https://cloud.tencent.com/document/product/1007/50364