阿里云
内容安全

开发指南

文档版本：20190704

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **警告:** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	 **说明:** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 **确定**。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

1 人脸识别基本概念

本文介绍了内容安全人脸识别功能的相关概念。

人脸

人脸（Face）在人脸识别技术中特指从待检测图像中发现的人脸。当系统对一张图片进行人脸检测时，会将检测到的人脸记录下来，包括人脸在图片中的位置信息。通常，多张人脸在同一张图像中只要特征点数足够均能够被识别出来。

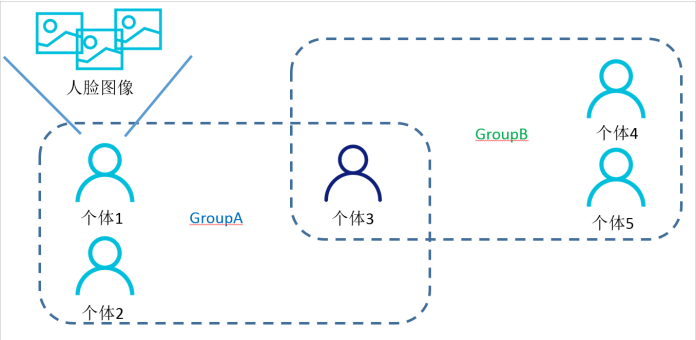
个体

个体（Person）在人脸识别技术中通常用来表示某张图像所代表的人物信息。个体信息一般在上传人脸图像的时候由用户自行指定。通常，同一个个体所关联的人脸图像需要保证是相同的人，并且图像中的人脸数不超过一个。

人脸库

人脸库（Group）在人脸识别技术中通常是用来保存个体信息的逻辑分组。当用户需要从一批给定的人脸图像中检索到与给定的人脸图像相同/相似的人脸时，需要指定人脸库。

人脸库与个体和人脸图像的关系如下图所示，个体可以属于一个或者多个人脸库，一个个体可以有多张图像作为个体的人脸底库。



人脸属性检测

人脸属性检测可以识别出给定图像中的一个/多个人脸，并对个体的年龄、性别、表情等属性做出判断，通常用于检测一张图像中是否有人脸/有一张或多张人脸。

活体检测

活体检测在人脸识别技术中通常是用来判断给定的图像/图像序列中出现的人脸是真人还是翻拍的照片，常见的活体检测方式有两种：

- 静默式翻拍检测：能够根据给定的图像判断该图像中的人是否来自于翻拍照片。
- 交互式活体检测：能够根据用户做出的动作判断出图像序列中出现的人是否是真人。

人脸比对

人脸比对是指利用人脸识别技术判断给定的两张图像中出现的人脸是否为同一个个体，需要保证给定的两张图像中都只有一张人脸，通常用于打卡签到场景。

人脸搜索

人脸搜索是指利用人脸识别技术从给定的人脸图像集合中，检索出与给定的图像最相似的个体列表，通常用于安检闸机身份验证场景。

2 离线活体检测SDK

内容安全提供离线的、基于手机端的人脸活体检测功能。您可以通过离线SDK将活体检测集成到APP中，实时获取被检测对象的动作状态，如眼睛、嘴巴、头部等姿态。集成活体检测SDK后，客户APP可以指示用户完成相关动作，从而判断对象是否为活体。

获取SDK及授权

您需要在内容安全控制台上传需要集成SDK的APP，才能获取相应SDK及授权。参照以下步骤，获取SDK及授权：

1. 登录云盾内容安全控制台。
2. 前往设置 > 活体检测页。
3. 单击添加应用。
4. 在新增应用包对话框中，上传要集成活体检测SDK的APK包。

> **说明：**
> 请确保APK包已经有正确的签名。

上传APK包后，您可以获得对应的SDK包，该SDK包中已包含相应授权。

每个新应用默认有1天的免费SDK使用期，过期后请联系客户经理或以工单形式联系我们进行续费。一个UID最多支持上传10个应用。

集成Android SDK

参照以下步骤将Android SDK集成到您的应用中：

1. 在工程中导入SDK。解压*liveness_sdk_xxx.zip*，将以下Android依赖包引入到您的应用工程中。

   - aar依赖文件

     ```
     aligreen-release-1.0.1.aar
     FaceLivenessOpen-1.1.6.11.aar
     NoCaptchaSDK-external-release-5.4.26.aar
     SecurityBodySDK-external-release-5.4.66.aar
     SecurityGuardSDK-external-release-5.4.94.aar
     ```

     a. 按如下方式配置*gradle*：

     ```
     repositories {
             flatDir {
                 dirs '../libs'
             }
         }
     ```

     b. 设定引入的本地库所在路径。将需要引入的依赖包都放在*../libs*目录，包含所有需要的库。

     c. 在*gradle*文件中引入以下需要的库依赖：

     ```
     compile (name:'aligreen-release-1.0.1',ext:'aar')
         compile (name:'FaceLivenessOpen-1.1.6.11',ext:'aar')
         compile (name:'NoCaptchaSDK-external-release-5.4.26',ext:'
     aar')
         compile (name:'SecurityBodySDK-external-release-5.4.66',ext
     :'aar')
         compile (name:'SecurityGuardSDK-external-release-5.4.94',
     ext:'aar')
     ```

   - yw_1222_xxx.jpg

     a. 把该图片文件导入到工程中的*res\drawable*目录。

     说明：

     如果没有这个文件夹，请先在工程中创建，否则将无法正常工作。

     b. 如果在安卓工程打包时启用了`shrinkResources true`，还需要在*keep.xml*文件中添加以下内容：

     ```
     <resources xmlns:tools="http://schemas.android.com/tools" tools
     :keep="@drawable/yw_1222_*"/>
     ```

   - aligreen-license.bin

     把该文件导入到工程中的*assets*目录。

     说明：

如果没有这个文件夹，请先在工程中创建，否则将无法完成初始化。

- 关于权限

  SDK的使用需要添加以下权限：

  - <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
  - <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
  - <uses-permission android:name="android.permission.CAMERA" />

2. 使用SDK。

   a. 启动应用后，执行以下操作：

```
import com.alibaba.security.aligreenv2.AligreenSdkManager;
AligreenSdkManager.getInstance().init(appContext);
```

   b. 执行以下操作进行人脸采集。

```
/**
* 此接口固定采集两个动作的照片
*
* @param isAutoCollect boolean类型 是否开启活体自动采集
*/
AligreenSdkManager.getInstance().startFaceLiveness(Context,
isAutoCollect, callback);

/**
* 此接口可选择静默式活体（无动作）或是采集一个活体动作照片或是采集两个活体动作照片
*
* @param actionCount int类型 动作个数，可选范围（0，1，2）
* @param isAutoCollect boolean类型 是否开启活体自动采集
*/
AligreenSdkManager.getInstance(). startFaceLivenessWithAction(
Context, actionCount, isAutoCollect, callback);
```

   说明：

   采集人脸中包含耗时操作，建议您不要在主线程中执行，否则会导致采集失败。

   c. 获取人脸结果。您可以在callback的onFinish方法中获取采集结果。

集成iOS SDK

参照以下步骤将iOS SDK集成到您的应用中：

1. 在应用工程中导入SDK

   解压离线活体SDK-IOS.zip压缩包，将以下Framework依赖包引入您的应用工程：

   a. 关于Framework依赖文件的操作

      A. 导入以下Framework依赖文件：

      - Aligreen.framework
      - FaceLivenessOpen.framework
      - SGMain.framework
      - SGNoCaptcha.framework
      - SecurityGuardSDK.framework
      - SGSecurityBody.framework

      B. 确认您的工程中已包含以下系统依赖：

      - MediaPlayer.framework
      - CoreMedia.framework
      - WebKit.framework
      - CoreMotion.framework
      - SystemConfiguration.framework
      - CoreLocation.framework
      - CoreTelephony.framework
      - libc++.tbd
      - libz.tbd

   b. 关于yw_1222_xxx.jpg，aligreen-license.bin以及FaceLivenessSDK.bundle文件的操作

      A. 将这三个资源文件导入应用工程中。

      B. 导入完成后，确认在Build Phases的Copy Bundle Resources中已存在这些资源文件。

   c. 关于权限的操作

      在应用的info.plist中添加Privacy - Photo Library Additions Usage Description和Privacy - Camera Usage Description权限。

   d. 关于编译选项

      在应用工程的Other Linker Flags选项中添加-ObjC。

2. 使用SDK

a. 在应用启动时添加SDK调用：

```
#import <Aligreen/AligreenSdkManager.h>
[AligreenSdkManager setup]
```

b. 进行人脸采集：

```
/**
* 此接口固定采集两个动作的照片
*
* @param isAutoCollect boolean类型 是否开启活体自动采集
*/
[AligreenSdkManager startFaceLiveness:(bool)isAutoCollect
withCallback:(id<FaceImageResultCallback>)faceImageResultCallback
withVC:(UIViewController *)vc];

/**
* 此接口可选择静默式活体（无动作）或是采集一个活体动作照片或是采集两个活体动作照片
*
* @param actionCount int类型 动作个数，可选范围（0，1，2）
* @param isAutoCollect boolean类型 是否开启活体自动采集
*/
[AligreenSdkManager startFaceLivenessWithActionCount:(int)
actionCount isAutoCollect:(bool)isAutoCollect withCallback:(
id<FaceImageResultCallback>)faceImageResultCallback withVC:(
UIViewController *)vc];
```

c. 获取人脸结果：

在callback的onFinish方法中获取人脸采集结果。

错误码说明

错误信息	错误代码	说明
SDK_NOT_INIT	-1001	SDK未初始化
GET_LICENSE_ERROR	-1002	获取服务端License失败
LICENSE_EXPIRED	-1003	License已过期
LICENSE_NOT_RIGHT	-1004	License不正确
ACTION_COUNT_TOO_LARGE	-1005	执行动作数量超标 说明: 最多执行两个动作。
ERROR_LICENSE	5000	授权错误
ERROR_LICENSE_INPUT	5001	输入参数有误，例如env、ctx、licenseData参数为空等

错误信息	错误代码	说明
ERROR_LICENSE_DATA_FORMAT	5002	licenseData参数格式错误
ERROR_LICENSE_SIGN	5003	签名校验失败
ERROR_LICENSE_APKPKG	5004	包名称校验失败
ERROR_LICENSE_PUBKEY	5005	公钥校验失败
ERROR_LICENSE_EXPIRE	5006	授权过期
ERROR_LICENSE_NOT_CHECK	5007	未执行授权检查
ERROR_LICENSE_BUNDLEID	5008	iOS BundleID校验错误
ERROR_LICENSE_CLIENT_ID	5009	终端标识校验错误
ERROR_LICENSE_EXPIRE_DATE_MODIFIED	5010	授权过期且日期被修改

3 人脸检索简介

本文介绍了接入人脸检索功能的操作流程。

初次接入

如果您初次接入人脸检索功能，想快速进行测试，请参照以下步骤进行操作：

1. 创建个体（person），接口说明请参见文档新建个体。

   
   说明：
   如果分组（group）不存在，创建个体的接口会自动帮您创建分组；如果分组已经存在，则该接口会将新创建的个体添加到分组中。

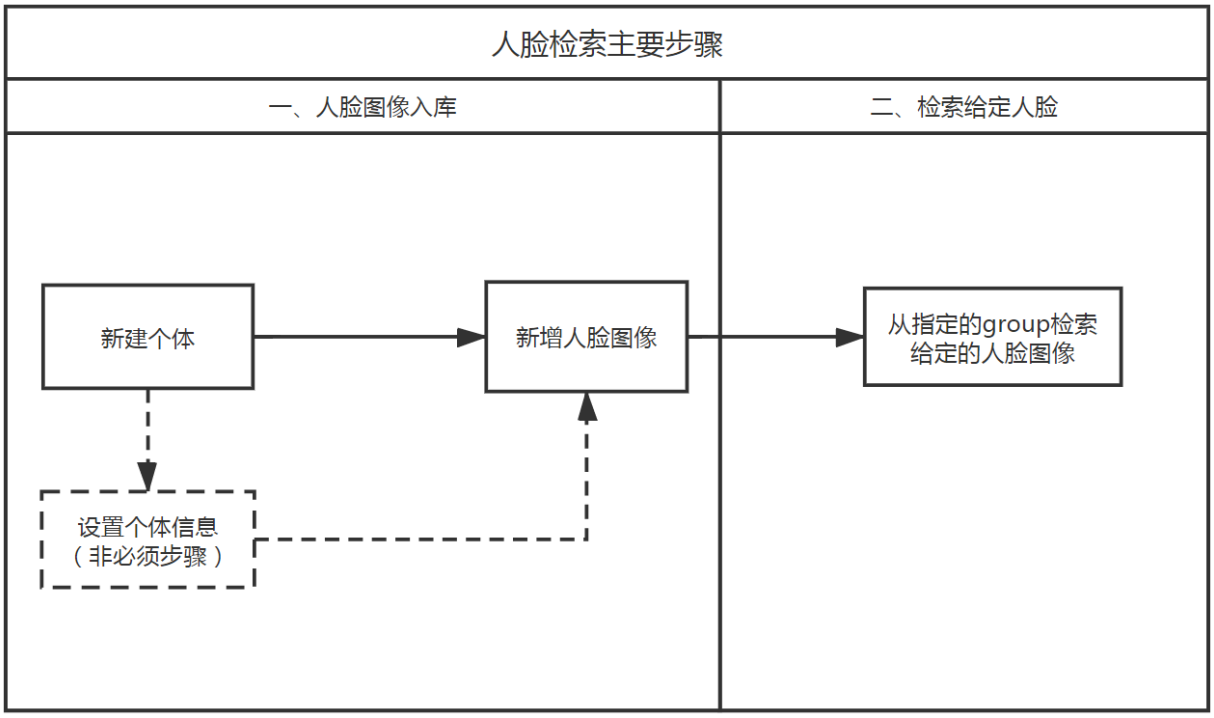
2. 新增人脸图像，接口说明请参见文档新增人脸。

   
   说明：
   人脸图像支持传递图像的HTTP/HTTPS协议的URL链接（建议您使用这种方式），也支持传递本地图片文件，具体请参见SDK示例。

3. 从指定的分组中检索给定的人脸图像，接口说明请参见文档自定义人脸检索。

   
   说明：
   一次检索请求中只能指定一个分组，如果您需要检测多个分组，则需要多次调用该接口传递不同的分组ID。

4. 设置个体信息。如果您需要给个体添加额外的信息，可以调用此接口，接口说明请参见文档设置个体。在检索到对应的个体后，我们会返回您设置的相关信息。



使用限制

- 待检索的图像中，支持出现多张人脸（默认最大支持二张人脸）。如果您需要检索超过二张人脸的图像，请通过工单联系我们调整。
- 对于检索到的每一张人脸图像，默认会返回相似度最高的五个个体的ID，建议您取分值最高的个体ID作为结果使用。如果五个结果不能满足您的需求，您可以通过工单联系我们调整。
- 检索不会返回对应的图像，您需要自行存储底库的图像与个体之间的对应关系。

修改个体所属分组

如果您需要修改某个个体所属的分组，请参照以下步骤进行操作：

1. 将个体从分组中移除，请参见接口文档删除分组中个体。

   说明：
   从分组中移除某个个体不会删除该个体下的任何一张图像。

2. 将个体添加到分组，请参见接口文档添加个体到分组。

   说明：
   调用该接口前请确保分组已存在。如果分组不存在，则接口调用会返回错误。

4 人脸识别

5 离线授权认证SDK

5.1 产品简介

阿里云内容安全提供离线授权认证SDK，帮助您实现在弱网或离网环境下的人脸认证。

说明:

离线授权认证 SDK 的实际应用效果与硬件配置和设备所处环境密切相关，目前只通过项目合作方式输出，前期需要评估方案可行性。

什么是离线授权认证SDK

离线授权认证SDK在Android 设备终端集成，支持在弱网或离线环境下的人脸认证应用。离线授权认证SDK包含人脸检测、活体检测、人脸1:1比对、人脸1：N检索、人脸库管理等能力，并全部离线化、本地化。SDK在授权激活后，可完全在无网环境下工作，所有数据皆在Android设备本地运行处理，可根据业务需要进行灵活的上层业务开发。

核心功能

离线授权认证SDK提供以下核心能力：

- 人脸检测

  输入摄像头采集的数据，返回检测到的人脸位置。

- RGB可见光活体检测

  针对视频流，通过采集人像的破绽（摩尔纹、成像畸形等）来判断目标对象是否为活体，可有效防止屏幕二次翻拍和复印二次翻拍等作弊攻击。

- NIR近红外活体检测

  针对视频流，利用近红外成像原理，实现夜间或无自然光条件下的活体判断。

- 人脸比对

  本地1:1人脸比对功能，针对一张预设的图片，和摄像头实时采集的符合条件的人脸进行比对，比对是否为同一人。

- 人脸检索

  本地1：N人脸检索功能，针对摄像头实时采集的符合条件的人脸，从本地人脸底库中检索是否存在此人的信息。

- 人脸库管理

  提供本地人脸底库的新增、删除等管理维护功能。

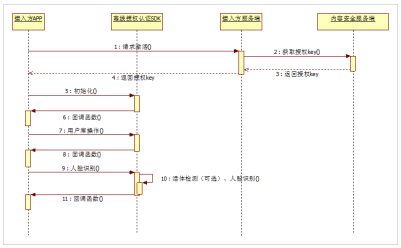
人脸算法对接入设备的要求

- 硬件设备
  - 系统：Android 4.4及以上
  - 系统类型：32位、64位
  - 处理器：4核，1.45GHz及以上
  - 内存：1G及以上
- 镜头
  - 分辨率：RGB镜头1080p， 红外镜头800X600及以上（使用红外功能时需要关注）
  - 固定30帧，无拖影，RGB镜头建议是宽动态
  - 照片清晰可辨认，人脸区域宽度大于100像素，人脸角度yaw ≤ ±15°， pitch ≤ ±15°

5.2 接入时序图

介绍离线授权认证SDK的接入流程。

图 5-1: 离线授权认证SDK接入时序图



时序图说明：

- Step1~Step4：可选。若设备未激活（如首次使用SDK，或授权有效期过期等），则需要调用内容安全服务端接口`CreateAuthKey`获得授权key，再进行初始化。若设备已激活，则直接从step5开始，传入授权key进行SDK初始化。
- Step5~Step6：调用SDK的`initWithToken`函数，传入授权key进行初始化，SDK会回调初始化结果。
- Step7~Step8：可选。若使用人脸1：N检索，则需要进行用户库操作。若使用人脸1:1比对，则不需要该步骤。
- Step9~Step11：进行人脸比对或检索，其中活体检测（翻拍和红外活体）根据接入方需要可开启或关闭，比对/检索结果SDK会回调告知。

5.3 SDK端接入

介绍离线授权认证SDK的SDK端接入方式。

授权方式说明

离线授权认证SDK目前支持在线联网激活，以获得SDK使用授权，被授权的设备和APP，在有效期内可以运行SDK，离线完成包括人脸库管理、人脸检测、人脸比对、人脸检索、活体判断等功能。授权结束之后，将无法使用任何功能，需要重新激活获得授权。

授权有效，不会消耗新的授权，仅需联网重新拉取授权：

- 删除SDK
- 设备刷机
- 清除数据
- 恢复出厂设置

授权无效，需要联网激活，且会消耗新的授权：

- 授权有效期到期
- 新设备首次使用

免费测试：我们为每个接入方提供2个免费测试授权，用于SDK接入试用，测试授权在激活后的1个月内有效。测试授权的开通需要联系我们。

正式购买：客户根据业务需要购买相应时长的SDK使用授权，具体定价可联系阿里云客户经理咨询。

步骤一：SDK下载

离线授权认证SDK可通过服务端接口进行下载，具体请参见下载离线授权认证SDK。

步骤二：在工程中导入SDK

解压离线授权认证SDK包，将以下Android依赖包引入到您的应用工程中：

- verifysdk-2.0.1.3-20190329-model-release.aar
- SecurityBodySDK-external-release.aar
- SecurityGuardSDK-external-release.aar

1. 设定依赖包所在的路径：

```
apply plugin: 'com.android.application'
 repositories {
     flatDir {
         dirs '../libs'
     }
 }
```

2. 设定引入的本地库所在路径，将需要引入的依赖包都放在../libs目录，包含所有需要的库。
3. 在gradle文件中引入以下需要的库依赖：

```
compile (name:'verifysdk-2.0.1.3-20190329-model-release',ext:'aar')
compile (name:'SecurityGuardSDK-external-release-5.4.121',ext:'aar')
 compile (name:'SecurityBodySDK-external-release-5.4.79',ext:'aar')
```

说明：

未来各依赖库的版本号会有所变化，实际版本号以下载到的SDK为准。

关于SDK签名图片

- 解压已下载的离线授权认证SDK包，获得yw_1222_*.jpg签名图片文件，该文件在SDK授权时需要使用。
- 把该图片文件导入到工程中res\drawable\目录，如果没有这个文件夹，请先在工程中创建，否则将无法正常工作。
- 如果在安卓工程打包时启用了shrinkResources true，还需要在keep.xml文件中添加以下内容：

```
<resources xmlns:tools="http://schemas.android.com/tools" tools:keep
="@drawable/yw_1222_*" />
```

- 离线授权认证SDK会与APP包名（package name）+签名（keystore）绑定，所以若APP的package name或keystore有变化都需要重新下载SDK，若无变化，则不需要重新下载。

关于 ABI 类型

离线授权认证SDK 目前支持 armeabi、armeabi-v7a、arm64-v8a，请接入方按需在build.gradle中增加abifilters配置。例如，接入方仅需要支持其中 armeabi 和 arm64-v8a，则配置如下：

```
defaultConfig {
    ndk {
        abiFilters "armeabi", "arm64-v8a"
    }
}
```

步骤三：SDK初始化及相关配置

接入方只需关注 SDK 提供的 VerifySDKManager 类，上层与 SDK 的交互都是通过 VerifySDKManager 进行调用。

在使用SDK前必须通过setNeedDeviceManage开启设备管理功能，且通过initWithToken方法进行初始化。

- **设置是否开启设备管理功能**

  接口名：`setNeedDeviceManage`

  接口描述：使用`initWithToken`**初始化方法，需要开启设备管理功能。**

  表 5-1: setNeedDeviceManage参数说明

名称	类型	描述
needDeviceManage	boolean	是否开启设备管理，true为开启，false为不开启。

  示例代码：

```
VerifySDKManager.getInstance().setNeedDeviceManage(true);
```

- **使用授权key进行初始化**

  接口名：`initWithToken`

  接口描述：使用授权key，初始化结果通过回调告知。

  表 5-2: initWithToken参数说明

名称	类型	描述
context	Context	当前activity的上下文。
token	String	激活使用的授权key，从内容安全服务端申请获得。
initWithTokenCallback	InitWithTokenCallback	回调函数，获取初始化结果。

  说明:
  如果设备目录/sdcard/verify/verifysdk/ab.bin存在而且有效，可以使用token=”“直接进行本地断网初始化。如果本地初始化失败，需要联网在线初始化，激活使用的授权key，从内容安全服务端CreateAuthKey接口获得，在线联网初始化会将最新的授权文件ab.bin下载到本地，完成SDK授权。

  示例代码：

```
// 初始化结果字段
private int regCode = -1;
// 回调函数
private InitWithTokenCallback initWithTokenCallback = new InitWithTo
kenCallback(){
    @Override
```

```
    public void onInitResult(int code) {
        regCode = code;
        Log.i("initData","return code: " + code);
    }
};
// activity onCreate方法中进行SDK初始化
VerifySDKManager.getInstance().setNeedDeviceManage(true);
VerifySDKManager.getInstance().initWithToken(getApplicationContext
(),"",initWithTokenCallback);
if(regCode !=0){
      // TODO 调用接入方服务端，服务端通过调用内容安全服务端接口CreateAuth
Key获取
      // String token =
VerifySDKManager.getInstance().initWithToken(getApplicationContext
(),token,initWithTokenCallback);
  }
```

SDK提供相关设置函数，可以修改人脸检测的最小尺寸、是否开启翻拍检测、是否开启红外活体检测、设置翻拍检测阈值、设置红外活体阈值、设置人脸检索匹配阈值等，接入方可以按需设置。

- 设置最小检测人脸

  接口名：setMinFaceDetectSize

  接口描述：设置最小检测人脸，图像中的人脸大小超过该值时，才能检测到人脸，否则无法检测到人脸。

  表 5-3: setMinFaceDetectSize参数说明

名称	类型	描述
minFaceDetectSize	float	人脸检测的最小尺寸，取值范围0-1，定义为占图像min(宽，高）的比例，默认值0.1。

  示例代码：

```
VerifySDKManager.getInstance().setMinFaceDetectSize(0.05f);
```

- 设置人脸位置有效区域边界

  接口名：setBorders

  接口描述：设置人脸位置有效区域边界。

  表 5-4: setBorders参数说明

名称	类型	描述
leftBorder	float	人脸位置有效区域的左边界，定义为占旋转后图像宽度的比例，默认值0.1。

名称	类型	描述
rightBorder	float	人脸位置有效区域的右边界，定义为占旋转后图像宽度的比例，默认值0.9。
topBorder	float	人脸位置有效区域的上边界，定义为占旋转后图像高度的比例，默认值0.1。

名称	类型	描述
bottomBorder	float	人脸位置有效区域的下边界，定义为占旋转后图像高度的比例，默认值0.9。

示例代码：

```
VerifySDKManager.getInstance().setBorders(0.1f,0.9f,0.1f,0.9f);
```

- 设置是否开启离线翻拍检测

  接口名：setNeedRecapCheck

  接口描述： 设置是否开启离线翻拍检测。

  表 5-5: setNeedRecapCheck参数说明

名称	类型	描述
needRecapCheck	boolean	是否开启离线翻拍检测。

  示例代码：

```
VerifySDKManager.getInstance().setNeedRecapCheck(true);
```

- 设置是否开启红外防彩色图片翻拍检测

  接口名：setNeedNirSeniorRecapCheck

  接口描述：设置是否开启红外防彩色图片翻拍检测。

  表 5-6: setNeedNirSeniorRecapCheck参数说明

名称	类型	描述
needNirSeniorRecapCheck	boolean	是否开启红外防彩色图片翻拍检测。

  示例代码：

```
VerifySDKManager.getInstance().needNirSeniorRecapCheck(true);
```

- 设置是否需要红外活体能力

  接口名：setNeedNirLiveness

接口描述：设置是否需要红外活体能力，若设置需要红外活体能力，那么setNirFrameW、setNirFrameH、setNirAngle、setRgbAngle为必要参数，最后再调用init方法进行生效。

表 5-7: setNeedNirLiveness参数说明

名称	类型	描述
needNirLiveness	boolean	是否需要红外活体能力。

示例代码：

```
VerifySDKManager.getInstance().needNirLiveness(true).setNirAngle(0).
setNirFrameH(480).setNirFrameW(640).setRgbAngle(0).init(getApplica
tionContext());
```

- 设置红外图像的帧高度

  接口名：setNirFrameH

  接口描述：设置红外图像的帧高度，默认值为720。

  表 5-8: setNirFrameH参数说明

名称	类型	描述
nirFrameH	int	红外图像的帧高度，默认值为720。

- 设置红外图像的帧宽度

  接口名：setNirFrameW

  接口描述：设置红外图像的帧宽度，默认值为720。

  表 5-9: setNirFrameW参数说明

名称	类型	描述
nirFrameW	int	红外图像的帧宽度，默认值为720。

- **设置红外图像的旋转角度**

  接口名：`setNirAngle`

  接口描述：设置红外图像的旋转角度，默认值为0。

  表 5-10: setNirAngle参数说明

名称	类型	描述
`nirAngle`	**int**	**红外图像的旋转角度，默认值为**`0`**。**

- **设置RGB图像的旋转角度**

  接口名：`setRgbAngle`

  接口描述：设置RGB图像的旋转角度，默认值为0。

  表 5-11: setRgbAngle参数说明

名称	类型	描述
`rgbAngle`	**int**	**RGB图像的旋转角度，默认值为**`0`**。**

- **设置翻拍检测阈值**

  接口名：`setRecapThreshold`

  接口描述：设置翻拍检测阈值，目前翻拍默认分值是80f，取值范围0~100，翻拍得分大于等于80f则认为是翻拍，小于80f则认为是真人。接入方可根据业务实际需要进行阈值调整。

  表 5-12: setRecapThreshold参数说明

名称	类型	描述
recapThreshold	float	设置翻拍检测阈值，默认值80f。

  示例代码：

```
VerifySDKManager.getInstance().setRecapThreshold(80f);
```

- **设置红外活体阈值**

  接口名：`setNirScoreThreshold`

  接口描述：设置红外活体阈值，目前默认分值是0f，取值范围0~1，接入方可根据业务实际需要进行阈值调整。

  表 5-13: setNirScoreThreshold参数说明

名称	类型	描述
nirScoreThreshold	float	设置红外活体阈值，默认值0f。

  示例代码：

```
VerifySDKManager.getInstance().setNirScoreThreshold(0f);
```

- **设置人脸匹配阈值**

  接口名：`setFaceMatchThreshold`

  接口描述：设置人脸匹配阈值，目前默认分值是0.44f，取值范围0~1，接入方可根据业务实际需要进行阈值调整。

表 5-14: setFaceMatchThreshold参数说明

名称	类型	描述
faceMatchThreshold	float	设置人脸匹配阈值，默认值0.44f。

示例代码：

```
VerifySDKManager.getInstance().setFaceMatchThreshold(0f);
```

步骤四：SDK用户库操作

初始化成功后，如果要使用SDK的人脸1：N检索能力，需要进行相应用户库操作，向人脸库中添加人脸图片，否则人脸检索将无法匹配。

如果要使用SDK的人脸1:1比对能力，则可以不关注用户库操作，参见人脸识别部分的说明。

- 加载用户库

  接口名：loadUserLib

  接口描述：用于初始时加载全量用户库。

  表 5-15: loadUserLib参数说明

名称	类型	描述
listener	VerifyLibEventListener	用户库操作回调。

  示例代码：

```
// 加载用户数据到内存
VerifySDKManager.getInstance().loadUserLib(verifyLibEventListener);
```

- 添加本地用户照片

  接口名：addUser

  接口描述：添加本地用户照片到用户库，操作结果以回调方式通知，完成后不必重新loadUserLib。添加后，用户照片数据会存储在设备本地数据库，适用于人脸底库数量不大或设备存储空间相对充裕的情况。

  说明：

使用该函数添加用户照片，优势是当人脸算法模型更新升级时，接入方不需要再重新入库一次用户照片，SDK会根据本地已有的照片数据重新提取特征值，支持在新模型下的运行；劣势是照片数据会占用一定的设备存储空间。

表 5-16: addUser参数说明

名称	类型	描述
isSync	boolean	是否同步调用，建议上层调用入库自己管理线程，该参数传true，否则传false。
id	String	用户ID，需要保证唯一性。
type	int	生物特征类型，目前只支持Type.BIOLOGY_FACE。
featureData	byte[]	用户照片的byte源数据，支持jpg、png格式。
verifyLibEventListener	VerifyLibEventListener	加入库操作回调。

示例代码：

```
VerifySDKManager.getInstance().addUser(true, String.valueOf(i), Type
.BIOLOGY_FACE,FileUtil.getFileBuffer(path),verifyLibEventListener);
```

接口名：addUserWithoutFeatureData

接口描述：添加本地用户照片，操作结果以回调方式通知。添加后，用户照片数据不会存储在设备本地数据库，适用于人脸底库数量大或设备存储空间有限的情况。

说明：

使用该函数添加用户照片，优势是不占用设备存储空间；劣势是当人脸算法模型更新升级时，本地已有的特征库无法适配新模型，SDK会运行失败，需要接入方再重新入库一次用户照片，以便新模型重新提取特征值，确保SDK正常运行。

说明：

使用该函数的接入方，需要在onUserLibLoaded回调函数中处理errorCode=104的错误码，详见onUserLibLoaded调用示例代码。

表 5-17: addUserWithoutFeatureData参数说明

名称	类型	描述
isSync	boolean	是否同步调用，建议上层调用入库自己管理线程，该参数传true，否则传false。
id	String	用户ID，需要保证唯一性。
type	int	生物特征类型，目前只支持Type.BIOLOGY_FACE。
featureData	byte[]	用户照片的byte源数据，支持jpg、png格式。
verifyLibEventListener	VerifyLibEventListener	加载用户库操作回调。

示例代码：

```
VerifySDKManager.getInstance().addUserWithoutFeatureData(true,
String.valueOf(i), Type.BIOLOGY_FACE,FileUtil.getFileBuffer(path),
verifyLibEventListener);
```

- 删除指定用户生物数据

  接口名：removeUser

  接口描述：删除指定用户生物数据，操作结果以回调方式通知，无需重新loadUserLib。

  表 5-18: removeUser参数说明

名称	类型	描述
isSync	boolean	是否同步调用，建议上层调用入库自己管理线程，该参数传true，否则传false。
id	String	用户ID。

名称	类型	描述
verifyLibEventListener	VerifyLibEventListener	用户库操作回调。

示例代码：

```
VerifySDKManager.getInstance().removeUser(false,id,verifyLibEventListener);
```

- 清空所有用户数据

  接口名：clearUserLib

  接口描述：清除本地用户库。

  表 5-19: clearUserLib参数说明

名称	类型	描述
isSync	boolean	是否同步调用，建议上层调用入库自己管理线程，该参数传true，否则传false。
verifyLibEventListener	VerifyLibEventListener	用户库操作回调。

  示例代码：

```
VerifySDKManager.getInstance().clearUserLib(false,verifyLibEventListener);
```

- 单用户数据更新回调

  接口名：onSingleUserLibUpdate

  接口描述：用户库操作回调，单用户数据更新。

  表 5-20: onSingleUserLibUpdate参数说明

名称	类型	描述
id	String	用户ID。
errorCode	int	错误码，0为正确。

- 用户库加载完成回调

  接口名：onUserLibLoaded

  接口描述：loadUserLib操作回调，用户库加载完成。

  表 5-21: onUserLibLoaded参数说明

名称	类型	描述
errorCode	int	错误码，0为正确。  说明: 使用addUserWithoutFeatureData函数添加用户的接入方，需要特殊处理该回调函数errorCode=104的错误码，详见下方示例代码中说明。

- 用户库清除回调

  接口名：onUserLibEmpty

  接口描述：clearUserLib操作回调，用户库清除完成。

  表 5-22: onUserLibEmpty参数说明

名称	类型	描述
errorCode	int	错误码，0为正确。

  示例代码：

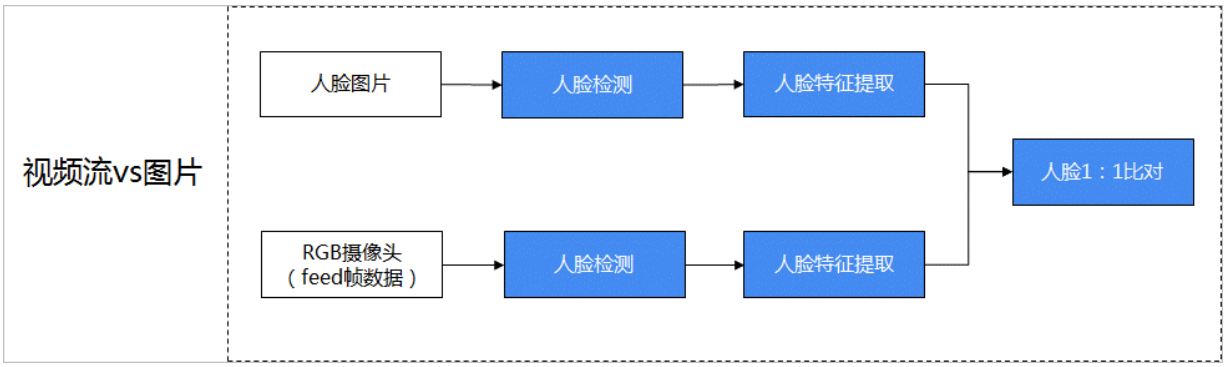
```
// 用户库操作回调
final VerifySDKManager.VerifyLibEventListener verifyLibEventListener
 = new VerifySDKManager.VerifyLibEventListener() {
        @Override
        public void onSingleUserLibUpdate(String id, final int
errorCode) {
        //TODO 单用户更新操作完成后处理
        }
        @Override
        public void onBatchUserLibUpdate(int errorCode) {
        //TODO 批量用户更新，暂时不支持
        }
       @Override
        public void onUserLibLoaded(int errorCode) {
        //TODO loadUserLib操作完成后处理
        //特殊说明：使用addUserWithoutFeatureData函数添加用户的接入方，需要
在该回调中特殊处理errorCode=104的错误码，操作步骤如下：
        //第一步：调用clearUserLib清除本地数据
        //第二步：接入方重新入库用户数据
        }
        @Override
        public void onUserLibEmpty(int errorCode) {
        //TODO clearUserLib操作完成后处理
```

```
        }
};
```

步骤五：SDK人脸识别

人脸1:1比对说明

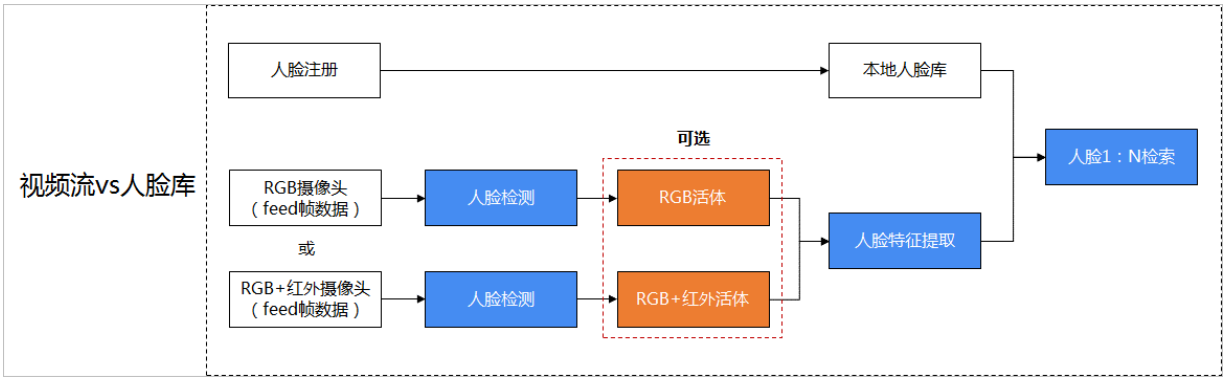
SDK支持实时视频流和人脸图片进行比对，这是最为常见的1：1对比类型。针对一张事先获取的图片（通常为身份证芯片照、证件照片等），与摄像头实时采集的符合条件的人脸图片进行比对。通常适用于有人值守的场景。



人脸1：N检索

将需要识别的人脸图片注册到本地人脸库中，当有用户需要识别身份时，从视频流中实时采集符合条件的人脸图片，与人脸库中的人脸集合比对，得到检索结果。如果是无人值守的情况，建议可以开启翻拍检测或红外活体保障安全性。

如果在初始化时设置开启了翻拍检测或红外活体检测，则摄像头采集的人脸图片必须同时通过活体检测，才能进入人脸检索环节，任一活体检测未通过，都不会进行人脸检索。



- **人脸1:1比对调用**

  接口名：doFaceMatchWithImage

  接口描述：摄像头帧数据与人脸图片进行1:1比对。

  表 5-23: doFaceMatchWithImage参数说明

名称	类型	描述
data	byte[]	检测到合格人脸时的RGB帧数据。
width	int	帧数据宽度。
height	int	帧数据高度。
cameraRotation	int	读取到合格人脸时的相机旋转角度，取值0/90/180/270，从摄像头的回调接口中获取。
targetBitmap	Bitmap	待比对人脸图片。
listener	FaceMatchWithImageListener	1:1比对回调。

  示例代码：

```
//1:1匹配后的回调
VerifySDKManager.FaceMatchWithImageListener faceMatchWithImageLi
stener = new
VerifySDKManager.FaceMatchWithImageListener() {
    @Override
    public void onMatchResult(FaceMatchResult faceMatchResult) {
     //TODO 与目标图片匹配后的处理
}
};
//1:1比对调用
VerifySDKManager.getInstance().doFaceMatchWithImage(data,width,
height,cameraRotation,data, faceMatchWithImageListener)
```

- **人脸1:1比对结果回调**

  接口名：`onMatchResult`

  接口描述： 人脸1:1比对结果回调，可获得相应比对分。

  表 5-24: onMatchResult参数说明

名称	类型	描述
`result`	FaceMatchResult	人脸1:1比对结果。

  示例代码：见上述”人脸1:1比对调用”代码示例。

- **人脸1:N检索调用**

  - **接口名：**`feedPreviewFrame`

    接口描述：RGB摄像头帧数据与人脸库进行1：N检索，结果以回调方式通知。适用于设备上只有RGB单目摄像头，或者有双目摄像头但未开启红外活体检测的情况，可以使用该接口进行检索比对。

    表 5-25: feedPreviewFrame参数说明

名称	类型	描述
`data`	byte[]	检测到合格人脸时的RGB帧数据。
`width`	int	帧数据宽度。
`height`	int	帧数据高度。
`imageFormat`	int	帧数据格式，Android默认选择`ImageFormat.NV21`。
`imageAngle`	int	读取到合格人脸时的图片旋转角度，取值0/90/180/270，从摄像头的回调接口中获取。
`cameraRotation`	int	读取到合格人脸时的相机旋转角度，取值0/90/180/270，从摄像头的回调接口中获取。
`faceDetectWithMatchListener`	FaceDetectWithMatchListener	人脸检测、检索比对、翻拍检测回调。

    示例代码：

```
// 当未开启红外活体功能时，使用FaceDetectWithMatchListener类型做回调
final VerifySDKManager.FaceDetectWithMatchListener
```

```
faceDetectListener = new VerifySDKManager. FaceDetectWithMatchL
istener () {
        @Override
        public void onFaceDetected(byte[] data, final int width,
final int height, int imageFormat, int imageAngle, int cameraRota
tion, final FaceInfo faceInfo) {
    //TODO 识别到人脸时
        }
        @Override
        public void onNofaceDetected(byte[] data, int width, int
height, int imageFormat, int imageAngle, int cameraRotation) {
    //TODO 没有识别到人脸时
        }
        @Override
        public void onFaceMatched(byte[] data, int width, int
height, int imageAngle, int cameraRotation, final FaceMatchResult
matchResult, final long costTime) {
    //TODO 识别到人脸并在用户库中匹配用户
        }
        @Override
        public void onRecapDetected(byte[] data, int width, int
height, int imageAngle, int cameraRotation, float recapScore) {
    //TODO 检查到人脸翻拍
        }
        @Override
        public void onFaceMoving(boolean isMoving) {
    //TODO 检查到人脸移动
        }
//1: N检索接口调用，不带红外活体
VerifySDKManager.getInstance().feedPreviewFrame(data,width, height
,ImageFormat.NV21,degree,cameraRotation, faceDetectWithMatchL
istener);
```

- **接口名：** `feedPreviewFrameWithNir`

  接口描述：RGB和红外摄像头帧数据与人脸库进行1：N检索，结果以回调方式通知。适用于设备上有双目红外的摄像头，在开启红外活体检测的情况下，可以使用该接口进行检索比对。

  表 5-26: feedPreviewFrameWithNir参数说明

名称	类型	描述
`nirData`	byte[]	检测到合格人脸时的红外帧数据。
`rgbData`	byte[]	检测到合格人脸时的RGB帧数据。
`width`	int	RGB帧数据宽度。
`height`	int	RGB帧数据高度。
`imageFormat`	int	帧数据格式，Android默认选择`ImageFormat.NV21`。
`imageAngle`	int	读取到合格人脸时的图片旋转角度，取值0/90/180/270，从摄像头的回调接口中获取。

名称	类型	描述
cameraRotation	int	读取到合格人脸时的相机旋转角度，取值0/90/180/270，从摄像头的回调接口中获取。
nirFaceDetectListener	FaceMatchWithImageListener	人脸检测、检索比对、翻拍检测回调。

示例代码：

```
//当开启红外检测时，使用NirFaceDetectListener类型做回调
final VerifySDKManager.NirFaceDetectListener nirFaceDetectListener
 = new VerifySDKManager.NirFaceDetectListener() {
        @Override
        public void onFaceDetected(byte[] data, final int width,
final int height, int imageFormat, int imageAngle, int cameraRota
tion, final FaceInfo faceInfo) {
    //TODO 识别到人脸时
        }
        @Override
        public void onNofaceDetected(byte[] data, int width, int
height, int imageFormat, int imageAngle, int cameraRotation) {
    //TODO 没有识别到人脸时
        }
        @Override
        public void onFaceMatched(byte[] data, int width, int
height, int imageAngle, int cameraRotation, final FaceMatchResult
matchResult, final long costTime) {
    //TODO 识别到人脸并在用户库中匹配用户
        }
        @Override
        public void onRecapDetected(byte[] data, int width, int
height, int imageAngle, int cameraRotation, float recapScore) {
    //TODO 检查到人脸翻拍
        }
        @Override
        public void onFaceMoving(boolean isMoving) {
    //TODO 检查到人脸移动
        }
        @Override
        public void onFaceDetectedInNIR(byte[] nirData, int width
, int height, int nirAngle, NirFaceInfo nirFaceInfo){
            //TODO 红外摄像头检查到人脸
        }
        @Override
        public void onNofaceDetectedInNIR(byte[] nirData, int
width, int height, int nirAngle){
    //TODO 红外摄像头没有检查到人脸
        }
    };
//1: N检索接口调用，带红外活体
VerifySDKManager.getInstance().feedPreviewFrameWithNir(nirData
,data,width, height,ImageFormat.NV21,degree,cameraRotation,
nirFaceDetectListener);
```

- **人脸1：N检测到人脸回调**
  - **接口名：**onFaceDetected

    接口描述：RGB帧数据中检测到人脸时，会回调该函数，调用方可根据此回调绘制人脸框。

    表 5-27: onFaceDetected参数说明

名称	类型	描述
data	byte[]	RGB帧数据。
width	int	RGB帧数据宽度。
height	int	RGB帧数据高度。
imageFormat	int	帧数据格式，Android默认选择ImageFormat.NV21。
imageAngle	int	图片旋转角度。
cameraRotation	int	相机旋转角度。
faceInfo	FaceInfo	检测到的人脸信息在帧中的坐标，宽高。

    示例代码：见1：N检索调用代码。
  - **接口名：**onFaceDetectedInNIR

    接口描述：红外帧数据中检测到人脸时，会回调该函数，调用方可根据此回调做相应处理。

    表 5-28: onFaceDetectedInNIR参数说明

名称	类型	描述
nirData	byte[]	红外帧数据。
width	int	红外帧数据宽度。
height	int	红外帧数据高度。
nirAngle	int	图片旋转角度。
nirFaceInfo	NirFaceInfo	检测到的红外人脸信息在帧中的坐标，宽高。

    示例代码：见1：N检索调用代码。

- **人脸1：N未检测到人脸回调**
  - **接口名：**onNofaceDetected

    接口描述：RGB帧数据中未检测到人脸，可以在这个回调中更新UI，移除之前绘制的人脸框。

    表 5-29: onNofaceDetected参数说明

名称	类型	描述
data	byte[]	RGB帧数据。
width	int	RGB帧数据宽度。
height	int	RGB帧数据高度。
imageFormat	int	帧数据格式，Android默认选择ImageFormat.NV21。
imageAngle	int	图片旋转角度。
cameraRotation	int	相机旋转角度。

    示例代码：见1：N检索调用代码。
  - **接口名：**onNofaceDetectedInNIR

    接口描述：红外帧数据中未检测到人脸。

    表 5-30: onNofaceDetectedInNIR参数说明

名称	类型	描述
nirData	byte[]	红外帧数据。
width	int	红外帧数据宽度。
height	int	红外帧数据高度。
nirAngle	int	图片旋转角度。

    示例代码：见1：N检索调用代码。

- **人脸1：N检索结果回调**

  接口名：onFaceMatched

  接口描述：人脸检索匹配结果回调。

  表 5-31: onFaceMatched参数说明

名称	类型	描述
data	byte[]	RGB帧数据。
width	int	RGB帧数据宽度。
height	int	RGB帧数据高度。
imageAngle	int	图片旋转角度。
cameraRotation	int	相机旋转角度。
matchResult	FaceMatchResult	匹配结果，极端情况下可能存在多个人的情况，比如双胞胎。
costTime	long	人脸检索比对耗时，性能分析调试用。

  示例代码：见1：N检索调用代码。

- **检测到翻拍回调**

  接口名：onRecapDetected

  接口描述：检测到翻拍，且初始化时开启了翻拍检测时，会执行这个回调。

  表 5-32: onRecapDetected参数说明

名称	类型	描述
data	byte[]	RGB帧数据。
width	int	RGB帧数据宽度。
height	int	RGB帧数据高度。
imageAngle	int	图片旋转角度。
cameraRotation	int	相机旋转角度。
recapScore	float	翻拍得分。

  示例代码：见1：N检索调用代码。

- **人脸移动时的回调**

  接口名：onFaceMoving

  接口描述：检测到人脸在移动时，会回调该函数。

  表 5-33: onFaceMoving参数说明

名称	类型	描述
isMoving	boolean	人脸是否移动。

  示例代码：见1：N检索调用代码。

SDK错误码

错误信息	错误描述	错误码
VERIFYSDK_ERR_CODE_BAD_PARAM	参数错误。	100
VERIFYSDK_ERR_CODE_ACCESS_WORK_FAILED	访问工作路径错误。	101
VERRORCODE_NEED_MANUAL_UPDATE	人脸特征库与算法不匹配，SDK启动失败。	104
VERIFYSDK_ERR_CODE_INVALID_PARAM	人脸引擎参数无效。	1000
VERIFYSDK_ERR_CODE_SDK_NOT_INIT	人脸引擎未初始化。	1001
VERIFYSDK_ERR_CODE_INIT_MULT_TIME	人脸引擎初始化多次。	1002
VERIFYSDK_ERR_CODE_FILE_NOT_EXIST	人脸模型文件不存在。	1100
VERIFYSDK_ERR_CODE_FILE_SIZE_ERROR	人脸模型文件大小错误。	1101
VERIFYSDK_ERR_CODE_VERSION_NOT_MATCH	人脸模型版本不匹配。	1102
VERIFYSDK_ERR_CODE_VERSION_NOT_MATCH_FD	人脸模型版本不匹配子错误。	11020
VERIFYSDK_ERR_CODE_VERSION_NOT_MATCH_LDM	人脸模型版本不匹配子错误。	11021

错误信息	错误描述	错误码
VERIFYSDK_ERR_CODE_VERSION_NOT_MATCH_LDC	人脸模型版本不匹配子错误。	11022
VERIFYSDK_ERR_CODE_L_VERSION_NOT_MATCH_FE	人脸模型版本不匹配子错误。	11023
VERIFYSDK_ERR_CODE_VERSION_NOT_MATCH_FEENC	人脸模型版本不匹配子错误。	11024
VERIFYSDK_ERR_CODE_CANCEL	取消。	1103
VERIFYSDK_ERR_CODE_BUSY	数据库并发操作。	1104
VERIFYSDK_ERR_CODE_FEATURE_SIZE_ERROR	人脸特征大小错误。	1201
VERIFYSDK_ERR_CODE_LICENCE	授权错误。	5000
VERIFYSDK_ERR_CODE_LICENCE_INPUT	输入参数错误。	5001
VERIFYSDK_ERR_CODE_LICENCE_DATA_FORMAT	授权文件格式错误。	5002
VERIFYSDK_ERR_CODE_LICENCE_SIGN	签名校验错误。	5003
VERIFYSDK_ERR_CODE_LICENCE_APKPKG	包名校验错误。	5004
VERIFYSDK_ERR_CODE_LICENCE_PUBKEY	公钥校验错误。	5005
VERIFYSDK_ERR_CODE_LICENCE_EXPIRE	授权过期。	5006
VERIFYSDK_ERR_CODE_LICENCE_NOT_CHECK	未进行授权检查。	5007
VERIFYSDK_ERR_CODE_LICENCE_CLIENT_ID	终端标识校验错误。	5009
VERIFYSDK_ERR_CODE_LICENCE_EXPIRE_DATE_MODIFIED	授权失效时间被修改。	5010

错误信息	错误描述	错误码
VERIFYSDK_ERR_CODE_NOT_SUPPORT_DEVICE	不支持的硬件型号。	6000
VERIFYSDK_ERR_CODE_LOAD_LIBARY_FAILED	加载so失败。	6001
VERIFYSDK_ERR_CODE_LICENCE_NOT_EXIST	授权文件不存在。	6002
VERIFYSDK_ERR_CODE_SECURITY_TOKEN_SOCKET_TIMEOUT	获取激活key超时，即初始化失败或未完成。	7001
VERIFYSDK_ERR_CODE_GET_STATIC_DATA_STORE_COMP	SDK安全组件获取appkey失败。	7003
VERIFYSDK_ERR_CODE_GET_SECURE_SIGNATURE_COMP	SDK安全组件获取加签失败。	7004
VERIFYSDK_ERR_CODE_SG_SECEXCEPTION	SDK安全组件异常。	7002
VERIFYSDK_ERR_CODE_GET_LICENSE_INFO_TOKEN_NULL	带授权key激活时，key为空。	8001
VERIFYSDK_ERR_CODE_GET_LICENSE_INFO_NATIVE_FAILED	获取授权失败。	8002
VERIFYSDK_ERR_CODE_GET_LICENSE_CHARSETNAME_EXCEPTION	授权key编码转换出错。	8003
VERIFYSDK_ERR_CODE_USER_LIB_NOT_LOADED	人脸库还未初始化完成。	9001
VERIFYSDK_ERR_CODE_UNKNOWN	人脸模块未知错误。	9999
VERIFYSDK_ERR_CODE_ACTIVATE_LICENSE_REQUEST_FAILED	请求失败，激活SDK失败。	10001
VERIFYSDK_ERR_CODE_ACTIVATE_LICENSE_DEVICE_NOT_AUTH	未授权，激活SDK失败。	10002

错误信息	错误描述	错误码
VERIFYSDK_ERR_CODE_ACTIVATE_LICENSE_DEVICE_AUTH_EXPIRED	授权到期失效，激活SDK失败。	10003
VERIFYSDK_ERR_CODE_ACTIVATE_LICENSE_DATA_NULL	授权文件为空，激活SDK失败。	10004

5.4 服务端接入

5.4.1 服务端接入准备

介绍离线授权SDK的服务端接入接口。

服务端接入准备

- 获取AccessKey
- API调用方式

离线授权认证SDK服务端API接口概览

API	描述
CreateVerifySDK	提交接入方应用，异步生成离线授权认证SDK。
DescribeVerifySDK	根据生成离线授权认证SDK的任务ID获取SDK生成结果。

API	描述
CreateAuthKey	获取授权key，用于离线授权认证SDK激活。
DescribeDeviceInfo	查询设备信息，包括设备授权的有效期等。

5.4.2 获取AccessKey

介绍如何创建RAM子用户，并授权其访问API和进行控制台操作。

阿里云会对每个访问请求进行身份验证，采用Access Key ID和Access Key Secret对称加密的方法来验证请求的发送者身份。其中，Access Key ID和Access Key Secret统称为AccessKey，是阿里云颁发给用户的访问服务所用的密钥。

内容安全支持阿里云账号或其 RAM 子用户的 AccessKey，但因为阿里云账号的 AccessKey 具有所有云产品 API 的访问权限，一旦泄露将导致极大的安全风险，所以强烈建议您根据最小权限原则创建并使用 RAM 子用户来进行 API 访问和控制台操作，具体操作步骤如下：

1. 创建 RAM 子用户。具体操作请参见创建 RAM 用户。
2. 给 RAM 子用户授权系统策略`AliyunYundunCloudAuthFullAccess`。具体操作请参见为 RAM 用户授权。
3. 给 RAM 子用户创建 AccessKey。具体操作请参见创建AccessKey。

5.4.3 API调用方式

对离线授权认证SDK的服务端API接口调用是通过向API的服务端地址发送HTTPS GET/POST请求，并按照接口说明在请求中加入相应请求参数来完成的；根据请求的处理情况，系统会返回处理结果。

请求结构

服务地址： 离线授权认证SDK服务端API的服务接入地址为：`cloudauth.aliyuncs.com`。

通信协议： 为了保证安全性，只支持通过HTTPS通道进行请求通信。

请求方法： 支持HTTPS GET/POST方法发送请求，这种方式下请求参数需要包含在请求的URL中。

请求参数： 每个请求的参数都由两部分组成：

- 公共请求参数：用于指定API版本号、返回值类型、签名等等，具体字段请参见下文。
- Action 及其特有的请求参数：Action参数对应具体要调用的接口名称（例如GetVerifyToken），其特有的请求参数即为该接口所要求的参数，这一部分的具体字段请参见各个API的说明文档。

字符编码： 请求及返回结果都使用 UTF-8 字符集进行编码。

说明:

为了便于进行开发，阿里云还提供Java、PHP、Python、.NET、Node.js、Go语言的SDK开发包，可以免去拼接 HTTPS 请求和对签名算法进行编码的麻烦。

公共参数

公共请求参数

公共请求参数是指每个接口都需要使用到的请求参数。

名称	类型	是否必需	描述
Format	String	否	返回值的类型，支持 JSON 与 XML，默认为 XML。
Version	String	是	API 版本号，为日期形式：YYYY-MM-DD，具体请参见API Release Notes。
AccessKeyId	String	是	阿里云颁发给用户的访问服务所用的密钥 ID。
Signature	String	是	用 AccessKeySecret 签名的结果串，关于签名的计算方法请参见RPC风格API的签名机制。
SignatureMethod	String	是	签名方式，目前支持 HMAC-SHA1。
Timestamp	String	是	请求的时间戳。日期格式按照 ISO8601 标准表示，并需要使用 UTC 时间 0 时区的值。格式为 YYYY-MM-DDThh:mm:ssZ。例如，2017-11-11T12:00:00Z（为北京时间 2017年11月11日20点0分0秒）。
SignatureVersion	String	是	签名算法版本，目前版本是1.0。
SignatureNonce	String	是	唯一随机数，用于防止网络重放攻击。用户在不同请求间要使用不同的随机数值。

请求示例

```
https://cloudauth.aliyuncs.com/?Format=xml
&Version=2017-11-17
&Signature=Pc5WB8gokVn0xfeu%2FZV%2BiNM1dgI%3D
&SignatureMethod=HMAC-SHA1
&SignatureNonce=15215528852396
&SignatureVersion=1.0
&AccessKeyId=key-test
&Timestamp=2017-11-11T12:00:00Z
&<[Action及其特有的请求参数]>
```

公共返回参数

用户发送的每次接口调用请求，无论成功与否，系统都会返回一个唯一识别码RequestId给用户。

- **XML返回示例**

```
<?xml version="1.0" encoding="UTF-8"?>
<!—结果的根结点-->
<接口名称+Response>
    <!—返回请求标签-->
    <RequestId>4C467B38-3910-447D-87BC-AC049166F216</RequestId>
    <!—返回结果数据-->
</接口名称+Response>
```

- **JSON返回示例**

```
{
    "RequestId": "4C467B38-3910-447D-87BC-AC049166F216",
    /* 返回结果数据 */
}
```

返回结果

调用 API 服务后返回数据采用统一格式，返回的 HTTP 状态码为 2xx，代表调用成功。返回 4xx 或 5xx 的 HTTP 状态码代表调用失败。

调用成功返回的数据格式主要有 XML 和 JSON 两种，外部系统可以在请求时传入参数来制定返回的数据格式，默认为 XML 格式。

本文档中的返回示例为了便于用户查看，做了格式化处理，实际返回结果是没有进行换行、缩进等处理的。

- **成功结果**
  - **XML返回示例（XML返回结果包括请求是否成功信息和具体的业务数据）**

```
<?xml version="1.0" encoding="UTF-8"?>
<!—结果的根结点-->
<接口名称+Response>
    <!—返回请求标签-->
```

```
    <RequestId>4C467B38-3910-447D-87BC-AC049166F216</RequestId>
    <!—返回结果数据-->
</接口名称+Response>
```

- JSON返回示例

```
{
    "RequestId": "4C467B38-3910-447D-87BC-AC049166F216",
    /* 返回结果数据 */
}
```

· 错误结果

调用接口出错后，将不会返回结果数据。调用方可根据接口文档中的错误码说明来定位错误原因。

当调用出错时，HTTP 请求返回一个 4xx 或 5xx 的 HTTP 状态码。返回的消息体中是具体的错误代码及错误信息。另外还包含一个全局唯一的请求 ID RequestId和一个您该次请求访问的站点 ID HostId。在调用方找不到错误原因时，可以联系阿里云客服，并提供该HostId和RequestId，以便我们尽快帮您解决问题。

- XML返回示例

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
<RequestId>8906582E-6722-409A-A6C4-0E7863B733A5</RequestId>
   <HostId>cloudauth.aliyuncs.com</HostId>
   <Code>UnsupportedOperation</Code>
   <Message>The specified action is not supported.</Message>
</Error>
```

- JSON返回示例

```
{
    "RequestId": "8906582E-6722-409A-A6C4-0E7863B733A5",
    "HostId": "cloudauth.aliyuncs.com",
    "Code": "UnsupportedOperation",
    "Message": "The specified action is not supported."
}
```

5.4.4 API Release Notes

下表记录了历次API的更新说明。

API版本号	发布说明
2018-09-16	更新内容如下： · 发布下载离线授权认证SDK接口（CreateVerifySDK） · 发布申请授权key接口（CreateAuthKey） · 发布查询设备信息接口（DescribeDeviceInfo）

5.4.5 下载离线授权认证SDK

异步提交SDK下载任务

描述

接口名：CreateVerifySDK

提交无线应用，异步生成离线授权认证SDK，一般可在1分钟内生成完成。

请求方法：支持以HTTPS POST和GET方法发送请求。

请求参数

名称	类型	参数位置	是否必需	描述
Url	String	Query	是	接入方无线应用程序的可访问链接。内容安全服务端会从该地址下载无线应用，进行SDK生成，接入方务必保证该地址可访问，否则SDK会生成失败。

返回参数

名称	类型	描述
TaskId	String	生成SDK的任务ID，用于查询SDK生成结果，一般生成可以在1分钟内完成。

获取SDK生成结果

描述

接口名：DescribeVerifySDK

根据生成离线授权认证SDK的任务ID获取SDK生成结果。

请求方法：支持以HTTPS POST和GET方法发送请求。

请求参数

名称	类型	参数位置	是否必需	描述
TaskId	String	Query	是	生成SDK的任务ID。

返回参数

名称	类型	描述
Url	String	SDK下载地址。不为空时，表示生成完成。

示例

使用SDK开发包的示例，请参见Java SDK、PHP SDK、Python SDK、.NET SDK、Node.js SDK、Go SDK中的接入说明及离线授权认证SDK下载示例部分的代码。（推荐以该方式接入）

以下是拼接HTTPS请求的相关示例：

异步提交SDK下载任务

- 请求示例

```
https://cloudauth.aliyuncs.com/?Action= CreateVerifySDK
&Url=https://www.xxx.com
&<[公共请求参数]>
```

其中，公共请求参数请参见API调用方式。

- 返回示例
  - XML格式

```
<?xml version='1.0' encoding='UTF-8'?>
<CreateVerifySDKResponse>
    <Data>
        <TaskId>1KQMcnLd4m37LN2D0F0WCD</TaskId>
    </Data>
    <Success>true</Success>
    <Code>1</Code>
</CreateVerifySDKResponse>
```

  - JSON格式

```
{
  "Data": {
    "TaskId":1KQMcnLd4m37LN2D0F0WCD
  },
  "Success": true
}
```

获取SDK生成结果

- 请求示例

```
https://cloudauth.aliyuncs.com/?Action=DescribeVerifySDK
```

```
&TaskId=1KQMcnLd4m37LN2D0F0WCD
&<[公共请求参数]>
```

- 返回示例
  - XML格式

```
<?xml version='1.0' encoding='UTF-8'?>
<DescribeVerifySDKResponse>
    <Data>
        <Url>https://www.xxx.com</Url>
    </Data>
    <Success>true</Success>
    <Code>1</Code>
</DescribeVerifySDKResponse>
```

  - JSON格式

```
{
  "Data": {
    "Url":https://www.xxx.com
  },
  "Success": true
}
```

5.4.6 申请授权key

描述

接口名：CreateAuthKey

获取授权key，用于离线授权认证SDK激活。

请求方法：支持以HTTPS POST和GET方法发送请求。

说明:
授权key在30分钟内有效，且不可重复使用，建议在每次激活前重新获取。

请求参数

名称	类型	参数位置	是否必需	描述
BizType	String	Query	否	业务类型。不超过64字符。可用于对具体业务进行备注，例如可以取接入方不同的人脸使用场景，或者待交付的客户标识等。建议传入该参数。
UserDeviceId	String	Query	否	用户设备ID。不超过64字符。可用于标识具体设备，建议可以使用设备物理编号。建议传入该参数。

名称	类型	参数位置	是否必需	描述
Test	Boolean	Query	否	测试标识。为true表示使用测试授权，授权时长默认为30天；为false，则授权时长根据AuthYears进行授权。
AuthYears	Integer	Query	否	当Test标识为false或空时，AuthYears必填，单位为年，范围为[1,100]，取值100表示永久授权。

返回参数

名称	类型	描述
AuthKey	String	可用于授权激活的key。授权key在30分钟内有效，且不可重复使用，建议在每次激活前重新获取。

示例

使用SDK开发包的示例，请参见Java SDK、PHP SDK、Python SDK、.NET SDK、Node.js SDK、Go SDK中的接入说明及离线授权认证SDK下载示例部分的代码。（推荐以该方式接入）

以下是拼接HTTPS请求的相关示例：

- 请求示例

```
https://cloudauth.aliyuncs.com/?Action=CreateAuthKey
&BizType=FACE_TEST
&UserDeviceId=3iJ1AY$oHcu7mC69
&Test=false
&AuthYears=1
&<[公共请求参数]>
```

其中，公共请求参数请参见API调用方式。

- 返回示例
  - XML格式

```
<?xml version='1.0' encoding='UTF-8'?>
<CreateAuthKeyResponse>
    <Data>
        <AuthKey>auth.1KQMcnLd4m37LN2D0F0WCD-1qtQI$</AuthKey>
    </Data>
    <Success>true</Success>
    <Code>1</Code>
</CreateAuthKeyResponse>
```

  - JSON格式

```
{
  "Data": {
    "AuthKey":auth.1KQMcnLd4m37LN2D0F0WCD-1qtQI$
```

```
  },
  "Success": true
}
```

5.4.7 查询设备信息

描述

接口名：DescribeDeviceInfo

查询设备相关信息，比如授权有效期、接入方自定义的业务标识和设备ID等。

请求方法：支持以HTTPS POST和GET方法发送请求。

请求参数

名称	类型	参数位置	是否必需	描述
DeviceId	String	Query	否	内容安全服务端为接入方设备生成的唯一ID，只有在设备成功激活后才会生成，该ID可通过离线授权认证SDK里的getLicenseExtraInfo函数获得。
UserDeviceId	String	Query	否	不超过64字符，由接入方自定义，可用于标识具体设备。
BizType	String	Query	否	业务类型。不超过64字符。
ExpiredStartDay	String	Query	否	查询的开始时间，即查询在ExpiredStartDay和ExpiredEndDay之间要过期的授权。示例：20180101。
ExpiredEndDay	String	Query	否	查询的结束时间。即查询在ExpiredStartDay和ExpiredEndDay之间要过期的授权。示例：20180101。
PageSize	Integer	Query	否	查询每页数目。
CurrentPage	Integer	Query	否	当前查询页数。

返回参数

名称	类型	描述
PageSize	Integer	每页数目。
CurrentPage	Integer	当前查询页数。
TotalCount	Integer	总数。

名称	类型	描述
Devices	Array	设备信息数组。每个元素如下表DeviceInfo。

表 5-34: DeviceInfo

名称	类型	描述
DeviceId	String	对应于请求中的DeviceId。
UserDeviceId	String	对应于请求中的UserDeviceId。
BizType	String	对应于请求中的BizType。
BeginDay	String	授权开始时间。示例：20180101。
ExpiredDay	String	授权到期时间。示例：20180101。

示例

以下是拼接HTTPS请求的相关示例：

- **请求示例**

```
https://cloudauth.aliyuncs.com/?Action= DescribeDeviceInfo
&DeviceId=wd.6ziUffspAeW5FVYbaqmexR-1qwNjM
&BizType=FACE_TEST
&UserDeviceId=3iJ1AY$oHcu7mC69
&ExpiredStartDay=20190401
&ExpiredEndDay=20200330
&PageSize=20
&CurrentPage=1
&<[公共请求参数]>
```

其中，公共请求参数请参见API调用方式。

- **返回示例**
  - **XML格式**

```
<?xml version='1.0' encoding='UTF-8'?>
<DescribeDeviceInfoResponse>
    <Data>
        <PageSize>20</PageSize>
    <CurrentPage>1</CurrentPage>
    <TotalCount>1</TotalCount>
    <Devices>
    <DeviceId>wd.6ziUffspAeW5FVYbaqmexR-1qwNjM</DeviceId>
             <UserDeviceId>3iJ1AY$oHcu7mC69</UserDeviceId>
             <BizType>FACE_TEST</BizType>
             <BeginDay>20190101</BeginDay>
    <ExpiredDay>20200101</ExpiredDay>
    </Devices>
    </Data>
    <Success>true</Success>
    <Code>1</Code>
```

```
</DescribeDeviceInfoResponse>
```

- JSON格式

```
{
  "Data": {
    "PageSize":20
    "CurrentPage":1
    "TotalCount":1
    "Devices":{
    "DeviceId":wd.6ziUffspAeW5FVYbaqmexR-1qwNjM
              "UserDeviceId":3iJ1AY$oHcu7mC69
              "BizType":FACE_TEST
              "BeginDay":20190101
    "ExpiredDay":20200101
    }
  },
"Success": true
}
```

5.5 服务端SDK开发包

5.5.1 Java SDK

获取地址

- *aliyun-java-sdk-core*: mvnrepository / maven.org / GitHub
- *aliyun-java-sdk-cloudauth*: mvnrepository / maven.org / GitHub

说明:

两个SDK都必须引入，其中*aliyun-java-sdk-core*为阿里云的核心SDK，*aliyun-java-sdk-cloudauth*为离线授权认证的SDK。

若您使用的是*aliyun-java-sdk-core* 4.0.0~4.0.2 版本，那么在调用https接口时需要在`profile`中额外加上以下内容：`profile.getHttpClientConfig().setIgnoreSSLCerts(true);`

安装说明

方法1：使用Maven（推荐）

如果您使用Maven管理Java项目，可以通过在*pom.xml*文件中添加Maven依赖：

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
    <version>4.1.2</version>
</dependency>
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-cloudauth</artifactId>
```

```
    <version>1.3.1</version>
</dependency>
```

说明:

`version`的值以SDK获取地址中的最新版本为准。

方法2：在集成开发环境（IDE）中导入jar文件

Eclipse安装

1. 将下载的*aliyun-java-sdk-xxx.jar*文件复制到您的项目文件夹中。
2. 在Eclipse中打开您的项目，右键单击该项目，单击Properties。
3. 在弹出的对话框中，单击Java Build Path > Libraries > Add JARs添加下载的JAR文件。
4. 单击Apply and Close。

IntelliJ 安装

1. 将下载的*aliyun-java-sdk-xxx.jar*文件复制到您的项目文件夹中。
2. 在IntelliJ中打开您的项目，在菜单栏中单击File > Project > Structure。
3. 单击Apply，然后单击OK。

离线授权认证SDK下载示例

```
DefaultProfile profile = DefaultProfile.getProfile(
"cn-hangzhou", // 可用区域id，目前只支持cn-hangzhou
"your access key id", // 您的Access Key ID
"your access secret"); // 您的Access Key Secret
DefaultProfile.addEndpoint("cn-hangzhou", "Cloudauth", "cloudauth.
aliyuncs.com"); //手动添加域名
client = new DefaultAcsClient(profile);

try {
CreateVerifySDKRequest createRequest = new CreateVerifySDKRequest();
createRequest.setAppUrl("https://app"); //接入方APP的公网可访问地址，内容安
全服务端会下载您的APK并为您APK生成离线授权认证SDK
CreateVerifySDKResponse createResponse = client.getAcsResponse(
createRequest);
String taskId = createResponse.getTaskId(); //获取生成sdk任务的taskId
String sdkUrl = null;
do {
//使用taskId轮询结果，一般生成可以在1分钟内完成
Thread.sleep(TimeUnit.SECONDS.toMillis(15));
DescribeVerifySDKRequest request = new DescribeVerifySDKRequest();
request.setTaskId(taskId);
DescribeVerifySDKResponse describeVerifySDKResponse = null;
describeVerifySDKResponse = client.getAcsResponse(request);
sdkUrl = describeVerifySDKResponse.getSdkUrl();
} while (sdkUrl == null || sdkUrl.isEmpty());
//sdkUrl为生成的sdk可访问链接，下载后进行集成
} catch (ClientException e) {
//生成异常
} catch (InterruptedException e) {
```

```
}
```

离线授权认证SDK获取授权key示例

```
DefaultProfile profile = DefaultProfile.getProfile(
"cn-hangzhou", // 可用区域id，目前只支持cn-hangzhou
"your access key id", // 您的Access Key ID
"your access secret"); // 您的Access Key Secret
DefaultProfile.addEndpoint("cn-hangzhou", "Cloudauth", "cloudauth.
aliyuncs.com"); //手动添加域名
client = new DefaultAcsClient(profile);

//发起获取授权key的请求
CreateAuthKeyRequest request = new CreateAuthKeyRequest();
request.setTest(Boolean.FALSE); //测试标识
request.setAuthYears(1);//授权年限
request.setBizType("biz type"); //业务类型
request.setUserDeviceId("device id"); //可自定义的用户设备id
CreateAuthKeyResponse createAuthKeyResponse = client.getAcsResponse(
request);
String authKey = createAuthKeyResponse.getAuthKey();
//获取到授权key调用离线授权认证SDK的initWithToken进行设备激活
```

5.5.2 PHP SDK

获取地址

GitHub，引入*aliyun-php-sdk-core*和*aliyun-php-sdk-cloudauth*。

> 说明:
> 两个SDK 都必须引入。其中*aliyun-php-sdk-core*为阿里云的核心SDK，*aliyun-php-sdk-cloudauth*为离线授权认证的SDK。

使用说明

1. 从GitHub中下载PHP SDK的源代码。
2. 在代码中根据实际文件路径添加PHP SDK引用，参见下方代码示例。

使用示例

完整示例正在准备中，请先参见Java SDK示例，或者参见阿里云OpenAPI调试平台里的示例代码进行调试。

附录

生成GUID

```
<?php
function guid(){
    if (function_exists('com_create_guid')){
        return com_create_guid();
    }else{
```

```
        mt_srand((double)microtime()*10000);//optional for php 4.2.0
and up.
        $charid = strtoupper(md5(uniqid(rand(), true)));
        $hyphen = chr(45);// "-"
        $uuid = chr(123)// "{"
            .substr($charid, 0, 8).$hyphen
            .substr($charid, 8, 4).$hyphen
            .substr($charid,12, 4).$hyphen
            .substr($charid,16, 4).$hyphen
            .substr($charid,20,12)
            .chr(125);// "}"
        return $uuid;
    }
}
```

5.5.3 Python SDK

获取地址

GitHub，引入*aliyun-python-sdk-core*和*aliyun-python-sdk-cloudauth*。

说明:

两个SDK都必须引入。其中*aliyun-python-sdk-core*为阿里云的核心SDK，*aliyun-python-sdk-cloudauth*为离线授权认证的SDK。

使用说明

参见GitHub中的说明。

使用示例

完整示例正在准备中，请先参见Java SDK示例，或者参见阿里云OpenAPI调试平台里的示例代码进行调试。

5.5.4 .NET SDK

获取地址

GitHub，引入*aliyun-net-sdk-core*和*aliyun-net-sdk-cloudauth*。

说明:

两个SDK都必须引入，其中*aliyun-net-sdk-core*为阿里云的核心SDK，*aliyun-net-sdk-cloudauth*为离线授权认证的SDK。

使用说明

参见GitHub中的说明。

使用示例

完整示例正在准备中，请先参见Java SDK示例，或者参见阿里云OpenAPI调试平台里的示例代码进行调试。

5.5.5 Node.js SDK

获取地址

GitHub，基于核心SDK进行开发，使用RPC调用风格。

使用说明

参见GitHub中的说明。

使用示例

完整示例正在准备中，请先参见Java SDK示例，或者参见阿里云OpenAPI调试平台里的示例代码进行调试。

5.5.6 Go SDK

获取地址

GitHub，使用泛化调用接口（CommonAPI）。

使用说明

参见GitHub中的说明。

使用示例

完整示例正在准备中，请先参见Java SDK示例，或者参见阿里云OpenAPI调试平台里的示例代码进行调试。