

集成文档-Android

合规性说明

一、SDK集成

- 1、本地集成
- 2、SDK 初始化

二、API 调用

1、活体检测

请求示例

将活体检测视频保存到自己的服务器（可选项，非必需）

响应数据

2、身份证 OCR

- 2.1 识别单面
- 2.2 识别双面

3、身份证二要素认证

请求参数

响应数据

合规性说明

SDK名称	场景描述	收集个人信息的类型	第三方机构名称	数据是否加密传输	第三方隐私政策链接
-------	------	-----------	---------	----------	-----------

实名认证SDK	活体检测	实名认证SDK提供活体检测服务过程中收集和使用的信息包括： 用户的面部识别信息（静态及/或动态）、用户的设备摄像头权限：用于活体检测服务中人脸检测、人脸检索、人脸比对；	上海创蓝云智信息科技股份有限公司	是	https://doc.chuanlan.com/document/6JFXVLDBQDXNK0HS
---------	------	---	------------------	---	---

一、SDK集成

1、本地集成

a.将libs目录下的aar包拷贝到您工程的libs目录下，如没有该目录需新建。

b.在build文件的dependencies中添加aar包依赖：

▼

Groovy

```

1 repositories {
2     flatDir {
3         dirs '../app/libs'
4     }
5 }
6 dependencies {implementation fileTree(include: ['*.aar'], dir: 'libs')}
```

2、SDK 初始化

a.建议在 Application 的 onCreate() 方法中进行初始化：

```
1  @Override
2  public void onCreate() {
3      super.onCreate();
4      // 设置调试模式，可以输出日志，方便调试，生产环境请关闭此开关
5      CLBaseManager.setDebuggable(true);
6      CLBaseManager.setWarnDeprecated(true);
7      CLBaseManager.init(getApplicationContext(),appId);
8  }
```

二、API 调用

1、活体检测

我们提供了一个 Demo 供您参考，如果觉得我们提供的界面文案、标题、文字大小、**titleBar** 的背景颜色，活体检测提示音，活体检测时的提示动画（GIF图片）等等不满足你们的需求，或者与你们的 **App** 页面风格样式不搭，想自定义修改这些配置当然也是可以的。

请求示例

```

1  SdkConfiguration configuration = new SdkConfiguration.Builder()
2      // 建议从服务端调用获取 authToken 接口获取
3      .setAuthToken(authToken)
4      // 设置动作数量 1-3 分别代表 1-3 个动作, 4 代表 1-3 个动作随机,
    不设置默认是 4
5      .setActionSize(1)
6      // 每个动作的执行时间, 单位为秒
7      .setActionSecond(3)
8      // 安全级别 0: 低 1: 高 (不设置默认为0) 低级别随机动作, 高级别将包
    含局部和全脸动作
9      .setSecurityLevel(0)
10     // 设置当前 Activity 对象
11     .setActivity(this)
12     // 人脸检测超时时间, 不大于 120s, 且不小于 10s, 不设置默认为 30s
13     .setTimeout(30)
14     // 设置相机配置, 比如相机预览的大小等等
15     .setICameraConfig(this::getCameraParameters)
16     .build();
17
18     OnlineAliveDetectorApi.getInstance().start(configuration, new IAl
iveDetectedListener() {
19         /**
20          * 当准备好的时候
21          */
22         @Override
23         public void onReady() {
24
25         }
26
27         /**
28          * 当收到下发的动作序列时
29          * @param actions 动作序列列表
30          */
31         @Override
32         public void onReceivedActions(List<OnlineAliveBean> actions)
33         {
34
35         }
36
37         /**
38          * 当提示动作改变时的回调, 比如提示做下一个动作时
39          * @param index 动作下标, 从 0 开始
40          */
41         @Override
42         public void onActionChanged(final int index) {

```

```

42
43     }
44
45     /**
46      * 当提示状态变化时的回调
47      * @param tip 提示信息
48      */
49     @Override
50     public void onStateTipChanged(String tip) {
51
52     }
53
54     /**
55      * 当脸部状态变化时
56      * @param isFullFace true: 预览框检测到全脸, false 不是全脸
57      */
58     @Override
59     public void onFaceStateChanged(boolean isFullFace) {
60
61     }
62
63     /**
64      * 人脸已经就位, 准备提示动作指令
65      */
66     @Override
67     public void onFaceReady() {
68
69     }
70
71     /**
72      * 服务端开始检测的回调, 这个时候可以显示一些动画或者 Loading
73      */
74     @Override
75     public void onStartDetect() {
76
77     }
78
79     /**
80      * 服务端检测完成的回调, 这个时候可以停止显示动画或者 Loading
81      */
82     @Override
83     public void onDetectComplete() {
84
85     }
86
87     /**
88      * 活体检测完成
89      */

```

```

90         @Override
91         public void onPassed(String data) {
92             // 返回数据示例: {"score":0.9667554498,"action_verify":"pass"}
93             // action_verify 为 pass 时代表动作验证通过, score 为活体检测
94             分数,
95             // 请自行根据自身业务场景设置阈值进行判断是否通过
96         }
97         /**
98          * 当活体检测有错误时的回调
99          */
100        @Override
101        public void onError(String errCode, String errMsg) {
102        }
103
104        /**
105         * 当活体检测超时时的回调
106         */
107        @Override
108        public void onOverTime() {
109        }
110
111    }
112    });

```

温馨提示：所有回调的方法都是在子线程中，所以，如果要处理 UI 相关的逻辑，需要切换到主线程处理。

当调用过一次后，如果失败了，可以调用重试方法，会重新开始活体检测：

```

▼ Java
1  // @Deprecated method
2  OnlineAliveDetectorApi.getInstance().tryAgain();
3  // 推荐使用
4  OnlineAliveDetectorApi.getInstance().tryAgain(authToken);

```

推荐使用带参数的 **tryAgain** 方法，这样可以设置新的 **authToken**，防止使用旧的 **authToken** 提示过期的问题。

将活体检测视频保存到自己的服务器（可选项，非必需）

活体检测 SDK 内部默认采用的是 BASE64 编码的方式上传视频，如果要将视频上传到自己的服务器或其他 OSS 服务器也是可以的，需要在调用活体开始之前先设置一个监听回调：

Java

```
1 // 设置视频录制的监听器，当需要自己上传是需要设置此监听
2 OnlineAliveDetectorApi.getInstance().setIVideoRecordListener(this);
3
4 // 并实现接口方法
5 public interface IVideoRecordListener {
6     void onComplete(String videoPath, IVideoUploadListener listener);
7 }
```

可以在 onComplete 回调方法中进行视频的上传操作，视频上传完成后将视频的 url 地址再通过回调方法中的监听器传递给 SDK 内部，此时会自动触发活体检测。

文件上传的代码可以参考 Demo。

响应数据

- ① 当活体检测通过时，会在 onPassed() 回调方法中收到回调；
- ② 当活体检测有错误的时候，会在 onError() 回调方法中回调 errCode、errMsg 信息，具体的错误信息如下表所示：

错误码	错误信息	备注说明
权限相关（1开头）		
10001	缺少相机权限	/
10002	缺少读写SD卡权限	/
参数相关（2开头）		
20001	context为空，SDK没有初始化或者初始化失败	/

20002	appId为空，SDK没有初始化或者初始化失败	/
20003	网络异常，请检查网络连接	/
20004	请求超时，请检查网络连接或重试！	/
业务相关（3、0或216开头）		
30001	动作验证未通过，请按提示完成动作	/
30002	活体检测超时，请在规定时间内完成提示动作	/
30003	当前网络不稳定，请切换网络后再试	/
30004	返回数据解析异常	/
30005	活体检测视频解析失败，请重新再试	
216434	人脸动作与提示动作不吻合，请重试	/
216501	没有检测到人脸，请重试	/
216507	检测到有多张人脸，请重试	/
216908	检测到人脸模糊，请重试	/
000400	令牌无效	authToken超过有效期或者使用的AppKey 和 AppSecret 不对
000998	账户金额预消耗失败	账号余额不足了
500003	活体检测视频太长	活体视频超过15秒

温馨提示：

- ① 调用活体检测方法之前，请确保已经获取到相机、读写 SD 卡的权限，SDK 内部不做权限申请的处理！！
- ② 我们提供了默认的声音文件和 GIF 图片文件，如果觉得我们的声音不好听或者图片不好看，也可以使用你们自己的，声音文件支持 mp3、wav 等常见格式，GIF 图片设计规范：123px * 111px；
- ③ 在调用之前需要获取到 **authToken**，为了安全，建议从服务端请求，具体参考服务端的接口文档，或者参考Demo 中的示例代码，Demo 请求是为了方便测试，生产环境建议从服务端请求；

2、身份证 OCR

2.1 识别单面

入参	说明	备注
authToken	授权token，建议通过服务端获取	/
side	front 为正面，back 为反面	/
imageBytes	身份证照片文件的 byte 数组	/
IDetectedListener	监听器	/

Java

```
1 IdCardApi.getInstance().singleSide(authToken, side, imageBytes, new IDetectedListener() {
2     @Override
3     public void onSuccess(String result) {
4         Log.d(TAG, "json:" + result);
5     }
6
7     @Override
8     public void onFailed(int errorCode, String errorMsg) {
9         Log.e(TAG, "onFailure():call=" + errorCode);
10    }
11 });
```

响应数据示例：

JSON

```
1 {
2   "code": "000000",
3   "message": "成功",
4   "data": {
5     "address": "江西省九江市庐山河南路xx号xx室",
6     "id_card_no": "360402200111133850",
7     "brith_day": "20011113",
8     "name": "黄xx",
9     "sex": "男",
10    "nation": "汉",
11    "issuing_authority": null,
12    "issuing_date": null,
13    "expire_date": null,
14    "msg": null
15  }
16 }
```

2.2 识别双面

入参	说明	备注
authToken	授权token，建议通过服务端获取	/
frontBytes	身份证正面文件的 byte 数组	/
backBytes	身份证反面文件的 byte 数组	/
IDetectedListener	监听器	/

传递身份证正、反面照片的文件字节数组：

```
1 IdCardApi.getInstance().doubleSide(authToken, frontBytes, backBytes, new I
  DetectedListener() {
2     @Override
3     public void onSuccess(String result) {
4         Log.d("TAG", "onSuccess() -> result:" + result);
5     }
6
7     @Override
8     public void onFailed(int errorCode, String errorMsg) {
9         Log.e("TAG", "onFailed() -> errorCode:" + errorCode + ",errorMsg:"
+ errorMsg);
10    }
11 });
```

温馨提示：

在调用之前，需要将图片做压缩处理，推荐每张照片压缩小于 1M 以内。

返回数据格式：

```
1 {
2   "code": "000000",
3   "message": "成功",
4   "data": {
5     "back": {
6       "address": null,
7       "id_card_no": null,
8       "brith_day": null,
9       "name": null,
10      "sex": null,
11      "nation": null,
12      "issuing_authority": "xxx县公安局",
13      "issuing_date": "20110123",
14      "expire_date": "20211123",
15      "msg": null
16    },
17    "front": {
18      "address": "河南省xx县xx乡xx村xxx组",
19      "id_card_no": "111111199012202638",
20      "brith_day": "19901220",
21      "name": "张三",
22      "sex": "男",
23      "nation": "汉",
24      "issuing_authority": null,
25      "issuing_date": null,
26      "expire_date": null,
27      "msg": null
28    }
29  }
30 }
```

当 OCR 成功时，返回如上的数据结构，当照片模糊、边角缺失或者正反面颠倒时，**front** 或 **back** 节点中的 **msg** 字段会提示模糊或者正反颠倒。

错误时的数据格式：

```
1 {
2   "code": "000000",
3   "message": "成功",
4   "data": {
5     "back": {
6       "address": null,
7       "id_card_no": null,
8       "brith_day": null,
9       "name": null,
10      "sex": null,
11      "nation": null,
12      "issuing_authority": "上海市公安局xx分局",
13      "issuing_date": "20051008",
14      "expire_date": "20151008",
15      "msg": null
16    },
17    "front": {
18      "address": null,
19      "id_card_no": "",
20      "brith_day": null,
21      "name": "",
22      "sex": null,
23      "nation": null,
24      "issuing_authority": null,
25      "issuing_date": null,
26      "expire_date": null,
27      "msg": "正反面颠倒"
28    }
29  }
30 }
```

温馨提示：

如果某一面识别成功了，某一面识别失败了，那么重新调用的时候，还是两张都要一起上传的。

3、身份证二要素认证

```
1 IdentityAuthApi.getInstance().verify(authToken, name, idNum, new IDetected
  Listener() {
2     @Override
3     public void onSuccess(String result) {
4         Log.d("TAG", "json:" + result);
5     }
6
7     @Override
8     public void onFailed(int errorCode, String errorMsg) {
9         Log.e("TAG", "onFailure():call=" + errorCode);
10    }
11    });
```

请求参数

authToken	授权token, 建议通过服务端获取	/
name	姓名	/
idNum	身份证号	/
IDetectedListener	监听器	/

响应数据

```
1 {  
2     "code": "000000",  
3     "message": "成功",  
4     "data": {  
5         "order_no": "011634609037212683",  
6         "handle_time": "2021-10-19 10:03:57",  
7         "province": "湖南省",  
8         "city": "邵阳市",  
9         "country": "邵阳县",  
10        "birthday": "19760320",  
11        "age": "46",  
12        "gender": "1",  
13        "remark": "一致",  
14        "result": "01"  
15    }  
16 }
```