FacePass Android版开发者指南_V3.2.0

本文档为旷视科技FacePass Android客户端SDK文档，面向使用客户，旨在全面详实介绍客户端SDK的类、成员变量、成员函数以及部分使用建议。

实际集成请配套示例Demo工程一起完整使用。

文档目录如下：

SDK使用说明

SDK简介

FacePassAndroidSDK是一个用于安卓应用开发的库。

使用SDK开发安卓应用需要在工程中加入SDK包FacePassAndroidSDK-release.aar，同时在工程的assets目录下放入算法所需的blurness.v5.l2rsmall.bin, pose.bin等.bin模型文件。

安卓端SDK的功能是检测出人脸，并进行人脸识别操作（包括活体检测和人脸底库检索）。

常用流程：

1，SDK初始化

2，人脸入库

3，人脸检测

4，人脸识别

SDK核心类和函数

FacePass SDK的接口使用到了多个内部定义的结构体，具体结构如下

FacePassConfig

概述

FacePassSDK handler的配置类。

主要场景应用：

1.初始化各类配置，并作为主要参数完成核心引擎FacePassHandler的初始化；

2.FacePassHandler提供get/set函数，供用户查询和修改配置。

FacePassConfig的的public参数主要分为四类：

1.阈值和开关类：主要提供人脸识别和活体识别的阈值，用于决策是否通过；

2.质量判断类：对于抓拍到的人脸帧，进行质量判断，包括角度、模糊、光照等，以选择合乎标准的人脸进行识别；

3.业务参数类：提供识别失败的重试次数、相机旋转角度、文件存储路径等参数；

4.模型文件：传入算法所需模型，用于配置初始化，会根据模式类型对模型文件进行校验。 抓拍版模式时加载3个模型文件，脱机版需要加载6个模型文件；

实际阈值和开关的逻辑说明主要放到FacePassSDK handler的feedFrame接口返回值中说明。

实际质量判断参数和其他参数说明将会在下表中着重解释。

实际模型文件相关的会在FacePassModel的数据结构出说明。

变量分类	分类说明	变量名	变量类型	变量说明

<table>
<tr><td>阈值和开关</td><td>一次完整的人脸识别流程包含两个算法：

• 人脸识别(Recognition)：从底库中检索分值最高的一个，看是否高于阈值，高则通过，反之则拒绝。
• 活体检测(Liveness)：对一张人脸进行活体判断，分值高于阈值则判定为真人，反之则视为攻击。

只有同时通过Search和Liveness的人脸才会让门禁机允许通行。

Search阈值越高，要求越严格，发生正识别和误识别的都会更难，是一个平衡；同时，不同底库大小也会有不同的建议阈值。Liveness的阈值也一样，越严格则判真核判假都会更难，也是一个平衡。

我们会提供某种默认的阈值出来，但是客户也可以自己跟进实际情况进行调整。

活体提供开关，可以关闭活体检测，以提升性能。</td><td>searchThreshold</td><td>float</td><td>识别阈值。

分布区间[0, 100]，float型，传入未限制位数。

极值0和100支持，但是实际使用过程强烈不建议取两端的值，这会导致效果不可用。

具体版本的建议阈值请查看对应Release note;

*注：以上通过率/误识率针对是基于正常场景下、底库质量合格的通道场景。实际识别率依赖底库质量、现场质量判断阈值、相机成像等综合因素，以体验效果为准。

阈值升高，底库内通过率会变低(坏事)，底库内误识率和底库外率误识率也会变低(好事)，底库内漏识率则会上升(坏事)。实际上要达到一个”完美的“阈值的调整是一个很精妙的过程，一般也不太需要那么严格。对于10000以内的底库，通常目前的建议阈值的精度就够了；如果的确存在某个率太离谱，我们可以协助进行数据分析和现场阈值重新建议。</td></tr>
<tr><td></td><td></td><td>livenessThreshold</td><td>float</td><td>活体阈值。

分布区间[0, 100]，float型，传入未限制位数。

极值0和100支持，但是实际使用过程强烈不建议取两端的值，这会导致效果不可用。

具体版本的建议阈值请查看对应Release note；

*注：实际通过率依赖于现场环境、攻击类型等。

这个阈值是针对门禁场景设计的，默认保证通过率（底库内人员的顺利通行）；如果不同场景有不同需求（比如要求超高的防攻击率），则可以自行调整阈值来实现。

方法是，抬高阈值，真人和攻击的通过率都会降低，到达需要的某个通过率即可；反之的思路是一样的。</td></tr>
<tr><td></td><td></td><td>livenessEnabled</td><td>bool</td><td>活体开关。

True or False，二选一。

活体检测会消耗一定的运算资源，在一些场景如果不需要进行活体检测（比如有人值守），则可以关闭活体开关，以提升算法系统整体的性能。</td></tr>
<tr><td></td><td></td><td>rgbIrLivenessEnabled</td><td>bool</td><td>红外活体开关

True or False，二选一。

需有RGB+IR双目相机。

活体检测会消耗一定的运算资源，在一些场景如果不需要进行活体检测（比如有人值守），则可以关闭活体开关，以提升算法系统整体的性能。

红外活体开关与活体开关，是两个独立的功能，同时打开，SDK只会使用一种活体算法，优先使用红外活体检测。</td></tr>
<tr><td>质量判断</td><td>质量判断(Quality)是一系列的指标，包括最小脸、角度、模糊、光照等。

质量判断的作用，是对于抓拍到的人脸，判断其质量分数，用于决策是否要进行人脸识别。

质量不高的图，无论对于识别、还是活体检测，都会产生负面影响。

理论上质量判断越严格，对识别越有利；但是越严格，花在检测上的时间也越长（因为会丢弃掉不合格的帧），这也是一个平衡关系。

我们会提供一套默认参数，但是也提供config中对每个参数值的自定义修改。</td><td>faceMinThreshold</td><td>int</td><td>最小人脸尺寸。

表示允许进行识别的最小人脸尺寸。

Int值，[0, 512]，理论上两端0和512可取，但是实际使用中强烈不建议选择这么极端的数值。

FaceMin是在检测到人脸后，进行大小比较的，同时对长和宽生效，也就是说需要同时满足：

• width >= faceMin
• height >= faceMin

如果不满足，这一帧会被直接丢弃，然后等待下一帧。

FaceMin直接的影响识别距离，faceMin越小，脸离摄像头越远，但是对算法识别和活体的准确度越差；faceMin越大，越适合算法，但是就得凑近脸。

这块算法效果不太好量化出来，所以我们给了一个综合的策略：

”建议客户保证faceMin大于旷视的建议、以保障最基本效果，如果需要降低faceMin来提升距离体验，则需要自行保障整体识别体验（比如降低底库大小、提升底库质量等）“。

目前旷视建议的值是150，基本可以保障识别效果。

附faceMin和识别距离的关系（基于主流720p摄像头的正常调校）：

<table><tr><th>faceMin</th><th>识别距离</th></tr><tr><td>100</td><td>1m</td></tr><tr><td>125</td><td>0.8m</td></tr><tr><td>150</td><td>0.5m</td></tr></table>
*注：实际情况依赖相机镜头参数、焦距、分辨率等诸多因素，请以实际为准。</td></tr>
</table>

<table>
<tr><td>poseThreshold</td><td>FacePassPose</td><td>人脸角度阈值。

角度，定义为人脸在三个轴维度上和“完全正脸”的差值，越大差值也越大。

具体细分有三个维度：

<table><tr><th>大类</th><th>子类</th><th>建议阈值</th><th>备注</th></tr><tr><td rowspan="3">pose</td><td>Roll</td><td>30°</td><td>旋转角度</td></tr><tr><td>Pitch</td><td>30°</td><td>垂直角度</td></tr><tr><td>Yaw</td><td>30°</td><td>水平角度</td></tr></table>
角度阈值，3个维度均采取角度值，范围[0, 90]，不限制小数点位数，然后不建议两端极值。

建议值在20°到30°之间，越大表示角度限制越“宽松”，所以越“容易过质量判断”，体验上是人脸更快的抓拍成功并送往端识别了；但是同步的，图的质量也容易越差，因此会不利于识别结果，体验上是识别相对容易漏识和误识。

目前建议是30°，小于30°的算通过，再大就容易影响识别了。一开始可以以30°为初始值，如果觉得精度不够的时候来下调。</td></tr>
<tr><td>blurThreshold</td><td>float</td><td>人脸模糊度阈值。

每张人脸都有模糊度，模糊度越大，模糊值（blur）越高。

有很多原因造成模糊，比如运动、失焦等。模糊了会影响识别和活体效果，所以需要加以限制。

阈值Float型，范围[0,1]，不限制小数点位数，也不建议两端极值。

<table><tr><th>分类</th><th>建议阈值</th></tr><tr><td>blur</td><td>0.2</td></tr></table>
目前算建议值0.2，低于0.2的算通过，高于0.2的认为太模糊了，建议不用于识别。

Blur阈值越小，对人脸模糊的判定越严格，所以对识别最有利，但是因检测严格，通过率下降，可能导致检测次数增多，通过时间会变长。实际操作中如果真的遇到模糊的问题，可以尝试根据需要来调整。</td></tr>
<tr><td>lowBrightnessThreshold</td><td>float</td><td rowspan="2">人脸平均照度阈值。

平均照度是人脸除去毛发和装饰物等杂物后，一个平均的亮度，数值越大越亮，不超过255。

对应的，阈值为Float，分布在[0, 255]之间，默认建议值是[70, 210]，在这个区间中的算通过。低于70则过暗，高于210则过量，均会开始影响识别结果。

<table><tr><th>分类</th><th>建议阈值</th></tr><tr><td>lowBrightness</td><td>70</td></tr><tr><td>highBrightness</td><td>210</td></tr></table>
如果要针对环境光照自行调整明暗阈值，也请务必同步保障识别结果。</td></tr>
<tr><td>highBrightnessThreshold</td><td>float</td></tr>
<tr><td>brightnessSTDThreshold</td><td>float</td><td>人脸照度标准差阈值。

照度标准差是用来衡量人脸光照对比度的参数，分布在[0, 255]，越大对比越强烈。

对应的，阈值为Float，分布在[0, 255]，建议取值80，80以下的算通过。

<table><tr><th>分类</th><th>建议阈值</th></tr><tr><td>brightnessSTD</td><td>80</td></tr></table>
越高越宽松，意味着放行高对比度的图，比如阴阳脸之类的，引起识别困难。

越低越严格，但是对图片质量就太苛刻，会降低检测的成功率和效率。</td></tr>
</table>

业务参数	SDK也提供部分业务逻辑的参数配置，供用户自定义某些逻辑。	retryCount	int	识别失败时的重试次数。 int型，>= 0，理论上上不封顶，如果足够大，就等于一直在重试，除非业务层中断、或者track不在出现等其他逻辑结束重试。 重试都是针对某个trackId而言的，这块具体的逻辑可以参见FacePassHandler中关于trackId逻辑的说明。 当前版本默认是2，意味着连续2次重试不过就不再识别当前trackId，也就是说，即使该trackId对应的人持续出现在画面中，那么FacePassDetectionResult中，也仅返回FaceList的“展示数据”，而不再提供用于识别的message和imageList的“识别数据”。 如果用户需要自定义修改，一般建议只有两种需要： 1.觉得不必重试3次，改到小于3的值，减少重试次数，以求更快的平均识别速度，但是有概率影响到识别率，但从目前实验结果看影响不大； 2.客户希望自己控制识别的重试逻辑，那么就改成一个足够大的数，比如99999，也就是说，只要一个trackId出现在画面里，那么只要失败就一直重试，但是客户可以自己添加某种逻辑（比如超时等）来自行中断重试。
		rotation	int	图像旋转角度值，只有FacePassImageRotation中定义的DEG0, DEG90, DEG180, DEG270四种合法值，代表0/90/180/270三种旋转角度。 旋转角度90表示当前图像相比于正确朝向（人脸朝上）顺时针旋转了90°，配置后，算法底层将会自动按照逆时针旋转90°进行处理。 所以这个参数需要客户确保角度正确，一个错误旋转的图片可能会检测不到人脸。
		fileRootPath	String	用于存储底库的文件根目录路径，需要用户传入一个可用的文件目录路径。
模型参数	算法会提供不同模块的算法模型，通过初始化成FacePassModel，作为参数指定给FacePassConfig。	detectRectModel	FacePassModel	用于人脸框检测的模型，通过FacePassModel初始化成实例
		poseModel	FacePassModel	用于人脸角度质量判断的模型，通过FacePassModel初始化成实例
		blurModel	FacePassModel	用于人脸模糊度判断的模型，通过FacePassModel初始化成实例
		searchModel	FacePassModel	用于人脸识别的模型，通过FacePassModel初始化成实例
		livenessModel	FacePassModel	用于人脸活体检测的模型，通过FacePassModel初始化成实例
		rgbIrLivenessModel	FacePassModel	用于人脸红外活体检测的模型，通过FacePassModel初始化成实例
		detectModel	FacePassModel	用于人脸检测的模型，通过FacePassModel初始化成实例
		landmarkModel	FacePassModel	用于获取人脸关键点的模型，通过FacePassModel初始化成实例
		ageGenderModel	FacePassModel	用于年龄性别的模型
		smileModel	FacePassModel	用于微笑指数的模型
		livenessGPUCache	FacePassModel	用于GPU加速的模型，当使用CPU活体模型时，请传null，当使用GPU活体模型时，必须传入加速cache模型

成员函数

FacePassConfig

构造函数一，提供了上述参数的完整的初始化接口，需要new一个FacePassPose实例，和6个FacePassModel实例。

```
public FacePassConfig(float searchThreshold,
                                    float livenessThreshold,
                                    boolean livenessEnabled,
                      int faceMinThreshold,
                                  FacePassPose poseThreshold,
                                    float blurThreshold,
                      float lowBrightnessThreshold,
                                    float highBrightnessThreshold,
                                    float brightnessSTDThreshold,
                      int retryCount,
                                    int rotation,
                                    String fileRootPath,
                                    FacePassModel poseModel,
                                    FacePassModel blurModel,
                                    FacePassModel livenessModel,
                                    FacePassModel searchModel,
                                    FacePassModel detectModel,
                      FacePassModel detectRectModel,
                      FacePassModel landmarkModel)
```

参数说明

参数类型	参数类型	是否必选	参数说明
searchThreshold	float	必选	参见变量说明
livenessThreshold	float	必选	参见变量说明
livenessEnabled	bool	必选	参见变量说明
faceMinThreshold	int	必选	参见变量说明
poseThreshold	FacePassPose	必选	参见变量说明
blurThreshold	float	必选	参见变量说明
lowBrightnessThreshold	float	必选	参见变量说明
highBrightnessThreshold	float	必选	参见变量说明
brightnessSTDThreshold	float	必选	参见变量说明
retryCount	int	必选	参见变量说明
rotation	int	必选	参见变量说明
fileRootPath	String	必选	参见变量说明
poseModel	FacePassModel	必选	参见变量说明
blurModel	FacePassModel	必选	参见变量说明
searchModel	FacePassModel	必选	参见变量说明
livenessModel	FacePassModel	必选	参见变量说明
detectModel	FacePassModel	必选	参见变量说明
detectRectModel	FacePassModel	必选	参见变量说明
landmarkModel	FacePassModel	必选	参见变量说明

返回值

FacePassConfig实例。

构造函数二，提供了上述参数的完整的初始化接口，需要new一个FacePassPose实例，和6个FacePassModel实例。

```
public FacePassConfig(float searchThreshold,
                                        float livenessThreshold,
                                        boolean livenessEnabled,
                      int faceMinThreshold,
                                      FacePassPose poseThreshold,
                                        float blurThreshold,
                      float lowBrightnessThreshold,
                                        float highBrightnessThreshold,
                                        float brightnessSTDThreshold,
                      int retryCount,
                                        int rotation,
                                        String fileRootPath,
                                        FacePassModel poseModel,
                                        FacePassModel blurModel,
                                        FacePassModel livenessModel,
                                        FacePassModel searchModel,
                                        FacePassModel detectModel,
                      FacePassModel detectRectModel,
                      FacePassModel landmarkModel)
```

参数说明

参数类型	参数类型	是否必选	参数说明
searchThreshold	float	必选	参见变量说明
livenessThreshold	float	必选	参见变量说明
livenessEnabled	bool	必选	参见变量说明
faceMinThreshold	int	必选	参见变量说明
poseThreshold	FacePassPose	必选	参见变量说明
blurThreshold	float	必选	参见变量说明
lowBrightnessThreshold	float	必选	参见变量说明
highBrightnessThreshold	float	必选	参见变量说明
brightnessSTDThreshold	float	必选	参见变量说明
retryCount	int	必选	参见变量说明
rotation	int	必选	参见变量说明
fileRootPath	String	必选	参见变量说明
poseModel	FacePassModel	必选	参见变量说明
blurModel	FacePassModel	必选	参见变量说明
searchModel	FacePassModel	必选	参见变量说明
livenessModel	FacePassModel	必选	参见变量说明
detectModel	FacePassModel	必选	参见变量说明
detectRectModel	FacePassModel	必选	参见变量说明
landmarkModel	FacePassModel	必选	参见变量说明

返回值

FacePassConfig实例。

构造函数三，适合快速上手，用于Config的创建，上述参数全部使用旷视提供的默认值，仅传入6个FacePassModel的实例，用于模型指定。其中识别底库默认为1000人。

```
public FacePassConfig(String fileRootPath,
                                        FacePassModel poseModel,
                                        FacePassModel blurModel,
                                        FacePassModel livenessModel,
                                        FacePassModel searchModel,
                                        FacePassModel detectModel,
                      FacePassModel detectRectModel,
                      FacePassModel landmarkModel)
```

参数说明

参数类型	参数类型	是否必选	参数说明
fileRootPath	String	必选	参见变量说明
poseModel	FacePassModel	必选	参见变量说明
blurModel	FacePassModel	必选	参见变量说明
searchModel	FacePassModel	必选	参见变量说明
livenessModel	FacePassModel	必选	参见变量说明
detectModel	FacePassModel	必选	参见变量说明
detectRectModel	FacePassModel	必选	参见变量说明
landmarkModel	FacePassModel	必选	参见变量说明

返回值

FacePassConfig实例。

构造函数四，Config的创建，将创建好的config作为setAddFaceConfig的参数，仅传入入库的阈值，用于指定入库阈值。

```
public FacePassConfig(int faceMinThreshold,
                                        float roll,
                                        float pitch,
                                        float yaw,
                                        float blurThreshold,
                                        float lowBrightnessThreshold,
                                        float highBrightnessThreshold,
                      float brightnessSTDThreshold)
```

参数说明

参数类型	参数类型	是否必选	参数说明
faceMinThreshold	int	必选	参见变量说明
roll	float	必选	参见变量说明
pitch	float	必选	参见变量说明
yaw	float	必选	参见变量说明
blurThreshold	float	必选	参见变量说明
lowBrightnessThreshold	float	必选	参见变量说明
highBrightnessThreshold	float	必选	参见变量说明
brightnessSTDThreshold	float	必选	参见变量说明

返回值

FacePassConfig实例。

FacePassDetectionResult

概述

图片的检测结果的类。

主要应用在FacePassHandler的feedFrame接口的返回结果。

本质上说，这个数据结构代表的，是“一帧检测对应的人脸图片和信息”，分为两个部分：

1.faceList：一个FacePassFace的数组（具体定义可以参见对应的类），这个数组是给客户用的，告知每张人脸的人脸框等检测信息，用于GUI显示等上层操作；

2.message：message是一个加密字符串。在离线模式下，交给FacePassHandler的recognize接口使用。在抓拍模式，message和images是配套使用的，message包含了各种识别所需的信息。

3.images: images是一个FacePassImage的数组，是经过裁剪后的人脸图片列表；与message两个字段是为了上层http协议时封装成必要的数据，提供给/service/recognize 接口。具体的协议收发请参见FacePassHandler的feedFrame接口说明。

关于detect相关的逻辑，也会在feedFrame接口一并说明。

成员变量

变量名	变量类型	变量说明
message	byte[]	用于人脸识别的字符串，由算法基于检测结果加密生成，包含了人脸识别所需的一些信息字段。 长度不定，理论上人脸越多信息越复杂则长，所以如果有外部调用请保证字符串长度空间。 如果没有检测到人脸，这个字段为NULL，但是不缺省。
faceList	FacePassFace[]	FacePassFace的数组，表示提供给客户用的人脸检测结果，具体参见FacePassFace的类定义。 简单讲就是包含了track_id（标识同一个人的id）和rect（人脸框的区域）。客户可以用这些信息做GUI显示。 如果没有检测到人脸，这个数组为空，不缺省、不为NULL。
images	FacePassImage[]	FacePassImage的数组，表示检测到的人脸通过区域裁剪后的小图的数组，具体参见FacePassImage的类定义。 简单讲，就是根据每个人脸裁剪出来的小图的数据，包括了原始数据流、高宽等信息、旋转等控制字段、trackId等。 如果没有检测到人脸，这个数组为空，不缺省、不为NULL。

成员函数

无，由于算法内部生成，并不提供public的构造函数。

FacePassFace

概述

人脸信息类。

提供给客户、用于实时展示人脸检测结果，仅在feedFrame的结果FacePassDetectionResult中出现。

成员变量

变量名	变量类型	变量说明
trackId	long	一个long型的id，作为一个track的标识，以标记一个连续运动中的人。 解释一下，一个人会有连续的很多帧，但并不是每一帧都会送到后端进行比对，不然太占资源。 实际上算法会智能的判断哪些人脸应该是同一个人，并且赋予相同的track_id。 虽然每一帧是单独是别的，但是通过track_id，算法可以综合多帧的结果得出人脸识别的结论，比如某一个track_id的某一帧如果已经通过了识别，门禁将会放行，后续这个track_id下的所有帧将没有必要再次识别等等。
rect	FacePassRect	人脸区域，参见FacePassRect类定义。 通过左上、右下两个点的坐标值，刻画出人脸框的位置，一般用于GUI显示。
pose	FacePassPose	人脸朝向，分为roll、pitch、yaw三个方向的角度值，提供[-90°，90°]的float，具体参见FacePassPose类定义。 人脸方向偏移过大时可以通过GUI提示用户正视摄像头。
blur	float	人脸模糊程度，详细参见FacePassConfig中的定义。[0,1]之间的float，越大越模糊。
smile	float	微笑指数，[0, 1]之间，数值越大表情是微笑的概率越高，只有当传入smileModel时才会开启微笑检测功能，否则该功能关闭。
ifIRPassed	boolean	红外IRFIter接口过滤后的track，如果没有通过，该项则为false。

成员函数

无，由于算法内部生成，并不提供public的构造函数。

FacePassImage

概述

人脸图像类。

SDK中针对图像原始内容的一个类，最主要内容自然是raw数据，但是同时也提供了高宽、旋转等控制字段。

客户通过相机返回数据进行初始化，生成FacePassImage的对象作为参数传给handler的feedFrame函数（以避免过于繁琐的传参）。

成员变量

变量名	变量类型	变量说明
image	byte[]	图像的raw数据，长度依赖于图片的实际大小。
width	int	图像宽度像素值。“相机帧处理”场景下需要客户确保数值正确，否则会引起算法处理的问题。
height	int	图像高度像素值。“相机帧处理”场景下需要客户确保数值正确，否则会引起算法处理的问题。
facePassImageRotation	int	图像逆时针旋转角度值，当前版本此参数不起效，为了向后兼容。 实际rotation的逻辑在FacePassConifg里头配置，可以具体参见对应的类定义。
facePassImageType	int	图像格式类型，目前支持：0: "NV21"，1: "GRAY"，2: "RGB"。 要客户确保参数正确，否则算法处理会有问题。

trackId	long	一个long型的id，作为一个track的标识，以标记一个连续运动中的人。 进出现在“检测帧返回”场景下，由算法根据检测结果对应赋值到每个对象中。

成员函数

FacePassImage

构造函数。

```
public FacePassImage(byte[] image, int width, int height, int facePassImageRotation, int facePassImageType)
```

参数说明

参数类型	参数类型	是否必选	参数说明
image	byte[]	必选	参见变量说明
width	int	必选	参见变量说明
height	int	必选	参见变量说明
facePassImageRotation	int	必选	参见变量说明
facePassImageType	int	必选	参见变量说明

返回值

FacePassImage实例。

FacePassPose

概述

人脸角度类。

对应3D世界的三个轴，分三个参数，详见下表。

成员变量

变量名	变量类型	变量说明
roll	float	以鼻子为中心的旋转角度值，取值[-90, +90]之间，0表示正脸，顺时针为正，逆时针为负，角度过大时算法将无法检测到人脸，通常需要控制在45°之内。
pitch	float	上下俯仰角度值，取值[-90, +90]之间，0表示正脸，抬头为正，低头为负，角度过大时算法将无法检测到人脸，通常需要控制在45°之内。
yaw	float	左右偏角角度值，取值[-90, +90]之间，0表示正脸，左转为正，右转为负，角度过大时算法将无法检测到人脸，通常需要控制在45°之内。

成员函数

FacePassPose

构造函数。

```
FacePassPose FacePassPose(float roll, float pitch, float yaw);
```

参数说明

参数类型	参数类型	是否必选	参数说明
roll	float	必选	参见变量说明
pitch	float	必选	参见变量说明
yaw	float	必选	参见变量说明

返回值

FacePassPose实例。

FacePassRect

概述

人脸位置对应的矩形区域类。

成员变量

变量名	变量类型	变量说明
left	Int	人脸左边界。
top	Int	人脸上边界。
right	Int	人脸右边界。
bottom	Int	人脸下边界。

成员函数

构造函数：

```
FacePassRect FacePassRect(int left, int top, int right, int bottom);
```

参数说明

参数类型	参数类型	是否必选	参数说明
left	int	必选	参见变量说明
top	int	必选	参见变量说明
right	int	必选	参见变量说明
bottom	int	必选	参见变量说明

返回值

FacePassRect实例。

FacePassRecognitionResult

概述

单张人脸识别结果的数据类，用于客户记录日志和做GUI显示。

抓拍模式下，客户调用 /service/recognize 上传抓拍图进行识别，结果通过FacePassHandler的成员数decodeResponse解析，并以这个类来呈现；

离线模式下，客户调用FacePassHandler的成员函数recognize（也就是通常说的人脸识别接口），并返回这个类的结果。

成员变量

变量名	变量类型	变量说明
faceToken	byte[]	24位字符数组，人脸新增后生成的唯一标识，用于人脸增删查以及底库绑定等核心操作。 通常意义上，我们用faceToken来代表底库中的一张人脸。 人脸识别过程分为人脸识别(Recognition)和活体检测(Liveness)两部分： 1.人脸识别：是在某个Group中查找和目标人脸最像的一张脸，而这张脸就通过faceToken来表示； 2.活体检测：跟Group无关，判定是不是真人而非纸质/电子屏等攻击。 返回faceToken的目的，是为了让客户端生成完整的人脸识别记录，反馈给客户以及记录日志。 如果没有检测到人脸，返回空字符串，不缺省，不为NULL。
facePassRecognitionResultType	int	人脸识别的结果。 一共有四种值： 0："OK"，人脸识别通过，也就是说search和liveness都通过了阈值； 1： "SEARCH_LOW_SCORE"，识别失败，原因是人脸检测过低，代表着目标人脸和目标底库中所有人都很不像，就不再进行该track_id的识别了； 2："TRACK_RETRY_EXPIRED"，识别失败，原因是同一个track_id检测了多次依旧没有成功（当前版本是3次），就不再进行该track_id的识别了； 3："TRACK_MISSING"，识别失败，原因是track已经从画面中消失； 4："AWAIT"，识别等待，意味着，目前为止的识别帧尚无法决策（不满足0123中任意一种），还需要识别该track_id的下一帧。 用户可以根据返回值给进行后续的逻辑操作，也可以进行GUI显示。
trackId	long	一个long型的id，作为一个track的标识，以标记一个连续运动中的人。
detail	FacePassRecognitionDetail	具体参见FacePassRecognitionDetail的类定义。 代表一次人脸识别的实际细节，包括比search和liveness的分值和阈值。

成员函数

无，由于算法内部生成，并不提供public的构造函数。

FacePassRecognitionDetail

概述

人脸比对详情的类。

代表一次人脸识别的实际细节，包括比search和liveness的分值和阈值。

成员变量

变量名	变量类型	变量说明
searchScore	float	人脸识别的比分，严格讲是group中最像的那个人的比对分值。 [0,100]之间的float值，越高表示越相似。
searchThreshold	float	人脸识别的阈值，searchScore高于阈值则算通过，反之则拒绝。 [0,100]之间的float值。
livenessScore	float	活体检测的比分。 [0,100]之间的float值，越高表示越是真人，越低表示越是活体攻击。
livenessThreshold	float	活体检测的阈值，livenessScore高于阈值则算通过，反之则拒绝。 [0,100]之间的float值。

成员函数

无，由于算法内部生成，并不提供public的构造函数。

FacePassAddFaceDetectionResult

概述

表示用于人脸注册的人脸检测的结果。

这个类仅用于FacePassHandler的addFaceDetect的返回值，可以配合阅读。

成员变量

变量名	变量类型	变量说明
image	FacePassImage	可以用于入库的人脸图片。 如果没有人脸则返回NULL， 如果有人脸但是人脸质量判断未通过，也返回NULL， 如果有多张人脸时，只会判断最大的一张人脸质量。

faceList	FacePassFace[]	FacePassFace的数组，表示提供给客户用的人脸检测结果，具体参见FacePassFace的类定义。 包含了track_id（标识同一个人的id）和rect（人脸框的区域）。客户可以用这些信息做GUI显示。 如果没有检测到人脸，这个数组为空，不缺省、不为NULL。

成员函数

无，由于算法内部生成，并不提供public的构造函数。

FacePassAddFaceResult

概述

离线模式下，注册人脸后返回结果。

参见FacePassHandler的addFace函数的返回值描述。

成员变量

变量名	变量类型	变量说明
faceToken	byte[]	24位字符串，入库成功后由系统生成，人脸的唯一标识，可以理解为faceID，用户可以通过faceToken绑定/解绑底库，也可以用于face的删除和查询操作。
facePassRect	FacePassRect	添加人脸成功后的人脸位置
result	int	0：成功 1：没有检测到人脸 2：检测到人脸，但是没有通过质量判断

成员函数

无，由于算法内部生成，并不提供public的构造函数。

FacePassAgeGenderResult

概述

年龄性别结果。

成员变量

变量名	变量类型	变量说明
trackId	long	track ID。

age	float	年龄
gender	int	0：男 1：女

成员函数

无，由于算法内部生成，并不提供public的构造函数。

FacePassCompareResult

概述

调用FacePassHandler的compare接口的返回值。

compare接口传入两张图，进行1:1比对。

给入的两张图会进行人脸检测，同时如果开启了活体判断参数，则会同时进行活体检测。

人脸检测失败，则会报错（不符合标准的照片进行compare没有意义，我们也无法保障结果具有实际意义）。

如果以上环节均正确，则会返回一个compre的分值，用于判断是否通过。

分值在(0，100]之间，越高表示越相似（同一个人），越低表示越不相似（不同的两个人）。

通常做法是分值会和一个阈值进行比较，这里一般的建议阈值是70。

说明，和一般的图片使用规则不同，compare接口目前不对图片质量进行严格的校验，原因是这是一个很底层的API，客户会用于各种复杂场景，很难给出有效的质量体系。

所以我们的策略是，API返回值中给出各种细节的参数，如果用户可以自行限制质量体系（并且无视我们的比对分值）。

成员变量

变量名	变量类型	变量说明
result	Int	人脸比对的结果值，为Int型枚举量： 0：比对成功，意指两张图片都成功通过人脸检测，如果活体开启的话则也都通过活体判断，并得出最终的score； 1：人脸检测失败，只要两张图中，有任意一张检测失败即视为结果失败，如果需要知道详情，可以参见detectionResult1和detectionResult2（为FacePassFace的数据结构）中的rect参数，如果为NULL则为检测人脸失败； 2：活体检测失败，只有在FacePassHandler的compare接口的传参livenessEnabled置为true的时候才有可能发生，只要两张图中的任意一张没有通过人脸检测，就会返回result=2，不再进行质量分的比对；
score	Float	1:1 compare的分值，（0,100]。 只有在result = 0 的时候，这个值才会被赋予 > 0 的数值，否则为缺省值 0。 分值越高表示越相似（同一个人），越低表示越不相似（不同的两个人）。 通常做法是分值会和一个阈值进行比较，这里一般的建议阈值是70。

compareThreshold	Float	1:1 compare的建议阈值，(0, 100]的Float。 如果客户不希望采用SDK的建议阈值，也可以自行设定，一般的原则是： 通过率 = 本人比本人正确的次数 / 本人比本人的总次数； 误识率 = 本人比他人正确的次数 / 本人比他人的总次数； 阈值提高，通过率下降，误识率也下降，直观上说，就是自己比自己更容易比不过了，但是自己比他人也更不容易出错了。 客户理论上可以根据实际需要调整自己的使用阈值，如有困难，可以咨询支持人员。
detectionResult1	FacePassFace	FacePassHandler的compare的前两个参数image1和image2的人脸检测结果，这是一个FacePassFace的数据结构，详见对应的定义。 对于compare而言，有意义的是rect和pose和blur。 “rect”：表示image1/image2中的人脸框位置，如果是检测失败（即result = 1的情形），则置为NULL，否则为FacePassRect的数据结构； “pose”/"blur"：分别表示角度和模糊度；这里需要说明的是，这两个质量判断并不影响返回值。原因如前文描述，这个API的应用场景过于宽泛，且很多场景有人值守、可做时候校验，所以普遍希望API能够尽可能放宽要求。一个典型的实际例子是身份证读卡器的输出照片，质量其实比较差（blur可能很不理想），但是我们依旧希望能过通过质量判断体系进行最终的比对。所以说，我们默认不卡角度和模糊度，如果用户真的需要卡，可以二次开发，针对返回值中的这两个参数进行更上层的业务判断。
detectionResult2	FacePassFace	
livenessThreshold	Float	活体阈值，Float型，初始化设置，详见FacePassConfig的配置项说明。
livenessScore1	Float	FacePassHandler的compare的前两个参数image1和image2的活体检测结果，Float型，越大表示越是真人，越小表示越像活体攻击。 livenessScore1和livenessScore2的任意一个低于livenessThreshold，就意味着活体检测失败，对应的 result = 2 。
livenessScore2	Float	

成员函数

无，由于算法内部生成，并不提供public的构造函数。

FacePassHandler

概述

FacePassHandler是整个客户端SDK的核心引擎，不提供public成员变量，但是提供了几乎所有主要的核心功能函数。

以下用较大篇幅逐一介绍Handler的成员函数，并解释背后的产品技术逻辑。

成员函数

FacePassHandler

概述

构造函数，需要传入FacePassConfig进行配置，具体参见FacePassConfig的参数。

FacePassHandler的初始化，需要调用initSDK并确保isAvailable函数返回true之后才能进行。

```
public FacePassHandler(FacePassConfig config) throws FacePassException;
```

参数说明

参数名	参数类型	是否必选	参数说明
config	FacePassConfig	必选	配置信息，包括模型版本、阈值信息等。 详细参见FacePassConfig的类定义。

返回值

FacePassHandler实例对象。

initSDK

概述

静态函数。

FacePassHandler的全局初始化函数，必需首先调用，初始化完成后才能创建实例。
注意：该函数会在程序后台创建异步线程进行初始化操作，调用方需要自行通过 isAvailable 函数判断初始化是否成功并进行等待，具体参见demo。

```
public static void initSDK(final Context context);
```

参数说明

参数名	参数类型	是否必选	参数说明
context	Context	必选	Android程序的Context。

返回值

无

isAvailable

概述

静态函数。

该函数返回 sdk 是否已经完成了初始化设置，在调用 initSDK 之后可以通过此函数判断进度，在没有得到true返回值之前，请勿进行其他操作。

```
public static boolean isAvailable();
```

参数说明

无

返回值

boolean类型，true则表示完成初始化

getAuth

概述

静态函数。

获取授权。需要在设备出厂时调用一次，连接旷视的服务器对设备进行授权，授权对于设备是永久的，成功之后可以无需再次授权。

注意：该函数返回并不说明授权成功，授权成功与否的状态需要通过调用isAuthorized接口进行查看，如果不成功需要再次调用getAuth接口，如果授权一直不成功，请检查网络连接。如果已有授权，调用getAuth接口不会发生任何事。

```
public static void getAuth(final String authURL, final String apiKey, final String apiSecret);
```

参数说明

参数名	参数类型	是否必选	参数说明
authURL	String	必选	授权服务器的地址，如www.megvii-inc.com或者IP地址如192.168.1.1。
apiKey	String	必选	客户申请到的apiKey，用于验证身份获取授权。
apiSecret	String	必选	客户申请到的apiSecret，用于验证身份获取授权。

返回值

无

isAuthorized

概述

静态函数。

查询设备是否授权。在调用getAuth接口后，需要调用这个接口来查询是否成功授权，如果没有成功，则需要等待一段时间再次查询，getAuth接口会一直重复尝试连接授权服务器。

```
public static boolean isAuthorized();
```

参数说明

无

返回值

boolean类型，true表示成功获得了授权。

getVersion

概述

静态函数。

获取当前客户端SDK的版本号。

```
String getVersion();
```

参数说明

无

返回值

字符串，格式如"1.1.0"，表示当前的版本号。

feedFrame

概述

向SDK传输来自于相机的一帧图片以进行人脸检测和识别，同步返回这一帧的检测结果，交识别接口。

```
public FacePassDetectionResult feedFrame(FacePassImage image) throws FacePassException;
```

参数说明

参数名称	参数类型	是否必选	参数说明
image	FacePassImage	是	从相机帧处理成FacePassImage对象，类型参数定义详见FacePassImage类。 主要包括raw流、高端、旋转、编码等，这个场景下没有trackId。 客户需要自行处理相机输入帧，并且通过以上参数初始化Image对象，具体参见示例。

返回值

FacePassDetectionResult对象，具体参见该类的定义。

feedFrame会返回一个FacePassDetectionResult的对象，主要包含：

- message/imageList：这俩是配套的，message是检测信息和控制信息的消息（加密态）；
- faceList：faceList则是提供给客户显示的人脸检测信息（FacePassFace）数组（明文态）。

真正发送给识别的是message而不是faceList，所以客户那个result首先要看message是否为空，为空则表示没有检测到人脸，不为空则继续；

调用识别接口：

调用FacePassHandler的recognize接口，传入groupName, message，进行人脸识别。

返回值是FacePassRecognitionResult对象，把必要的信息通知客户，同时handler决策是否重发等算法策略，直到结束。

也说明一下track和识别的关系：

1.整个识别过程是feedFrame驱动的，只要检测到有效的人脸，就会建议客户发起识别请求；

2.feedFrame返回的是一帧image中的人脸，所以可能有多个，他们通过trackId来区分；

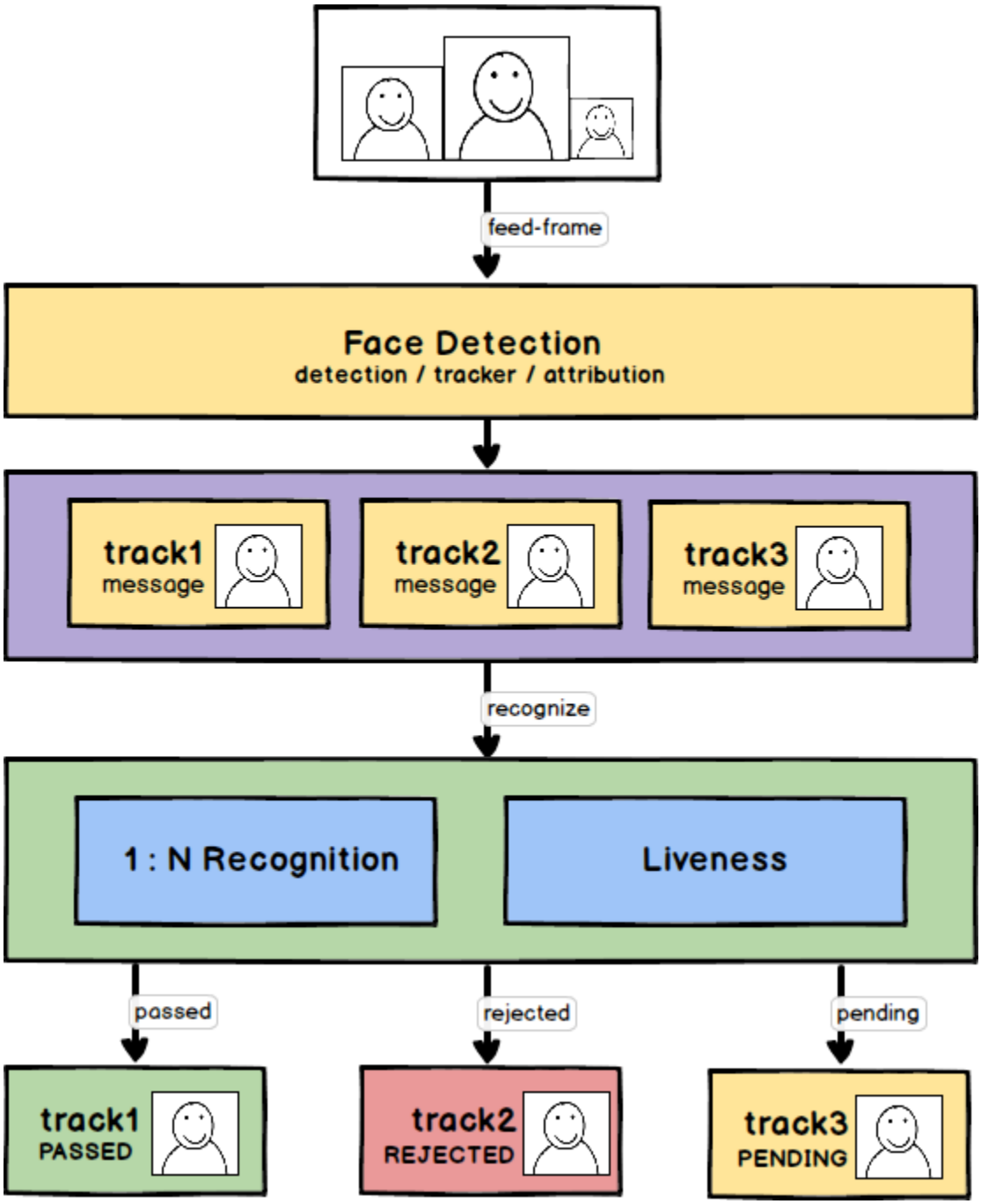
3.识别结果也是以track的维度来展示，给出的是每个track的状态（而非单帧）；

4.不同的track之间是独立的，互不影响；

4.如果是同一个track的同一张图片（这里指的是抠脸的小图，不是原始大图），被重复发送识别，则仅使用第一个返回的结果，后续结果将直接无视；

6.如果是同一个track的不同图片发送识别，则根据返回的不同图片的结果来决策track的状态（具体参见FacePassRecognitionResult中的定义）；如果track的状态已经ending（除了AWAIT这种状态），则后续的识别结果不再影响track状态。

以下是整个检测识别流程的图解：



recognize

概述

根据feedFrame得到的检测结果(FacePassDetectionResult)，进行人脸识别，包括人脸识别和活体检测，结果以FacePassRecognitionResult数组的形式返回，一次检测可以包含多个人脸，数组一项表示一个人脸。

人脸识别：从N个人找到和目标图片最像的人脸，如果过阈值则通过、反之则拒绝。模型和阈值取决于FacePassConfig初始化时候的searchModel, searchThreshold。人脸识别的准确率依赖于底库大小、底库质量、现场光照环境等。

活体检测：判断一张人脸是否真人，和底库无关，比较依赖于现场的环境、摄像头的成像质量等。

```
FacePassRecognitionResult[] recognize(String groupName, String message) throws FacePassException;
```

参数说明

参数名	参数类型	是否必选	参数说明
groupName	String	必选	人脸搜索的group名字。
message	String	必选	用于人脸识别的字符串，由算法基于检测结果加密生成，包含了人脸识别所需的一些信息字段，用于识别协议的表单字段。 长度不定，由feedFrame生成，理论上人脸越多信息越复杂则长，所以如果有外部调用请保证字符串长度空间。 如果没有检测到人脸，这个字段也不为空。

返回值

返回识别结果（FacePassRecognitionResult）的数组，每一项对应一张图的识别结果。

其成员变量如下：

1.faceToken：人脸id；

2.facePassRecognitionResultType：识别的结果，供客户决策上层逻辑和GUI；

3.facePassRecognitionResultDetail：具体的别分和阈值，提供更细致的信息；

4.trackId：trackId，用于关联不同人脸。

具体请参见FacePassRecognitionResult的类定义。

识别相关的具体逻辑请参见feedFrame函数。

IRfilter

概述

红外活体过滤。

将feedFrame返回的结果与对应的红外图片输入到此接口中，输出的新message是经过红外活体过滤的结果。详细请查看红外电子屏过滤的专门文档。

```
public FacePassDetectionResult IRfilter(FacePassImage image, FacePassDetectionResult result) throws
FacePassException;
```

setIRConfig

概述

配置红外电子屏过滤参数。红外电子屏过滤需要考虑双摄像头的位置偏移，该函数即用于配置这个偏移量。SDK生命周期中只需配置一次，在调用ir_filte前设置。偏移量系数需由客户根据自己的设备测试得到，红外电子屏过滤的相关知识请看专门文档。

参数说明

x_k 左右偏移量系数；

x_b 左右偏移量系数；

y_k 上下偏移量系数；

y_b 上下偏移量系数，以上所有系数均应由客户根据设备测试获得，具体方法见红外电子屏过滤文档。

```
public boolean setIRConfig(double left_right_k, double left_right_b, double top_bottom_k, double top_bottom_b,
double overlap_rate)
```

getAgeGender

概述

获取年龄性别，需要将feedFrame返回的message传入，用法类似于recognize。只有在初始化handler时传入年龄性别模型开启了年龄性别功能时才可以使用。

```
public FacePassAgeGenderResult[] getAgeGender(byte[] message) throws FacePassException
```

resetMessage

概述

此接口在一般情况下不要使用！只有极特殊情况下才需要使用，请在技术支持指导下使用。

重置track状态。track在送识别状态下（即feedFrame返回了message的情况下），在将message调用recognize或者请求服务端识别返回调用decodeResponse之前，均不会再次产生message，此方法会将track的“识别中”状态重置，使得不送识别，也可以再次获得该track的message。

```
public void resetMessage(long trackId) throws FacePassException
```

createLocalGroup

概述

新增底库。

底库是一个容器的概念，本产品的人脸识别都是基于“一张图”和“一个底库”的比对，而不是暴露直接的1:1的比对接口供用户轮询，原因是算法底层对1:N检索进行了大量的优化，目前的实现方式是综合性价比最高的。

人脸注册只是在系统中加入了一张人脸并抽取特征值（对应的用faceToken进行索引），但真正要用于比对，则必须让faceToken绑定某个groupName，然后让客户端抓拍图和groupName对应的group进行比较，从而实现1:N的人脸识别检索。

```
public bool createLocalGroup(String groupName) throws FacePassException;
```

底库管理API包括：

- 新建底库
- 删除底库
- 查询底库
- 人脸注册
- 绑定人脸
- 解绑人脸
- 查询底库中人脸
- 下载人脸图片

详细请参见FacePassHandler的相关API接口。

参数说明

参数名称	参数类型	是否必选	参数说明
groupName	String	是	新增的localGroup的groupName，由用户作为参数指定，用于group的增删查操作，也是绑定/解绑faceToken的group唯一标识。 参数限制（如果不满足，则报参数错误的Exception）： • 支持4-16位长度 • 支持英文小写字母、英文大写字母、数字、半角下划线 • 建议正则：/^[a-zA-Z0-9_-]{4,16} groupName可以自由指定；

返回值

boolean类型，是否完成create操作。

如果groupName已经存在，则返回失败。

deleteLocalGroup

概述

删除底库group，对应的也删除了group和faceToken的绑定关系。

```
public bool deleteLocalGroups(String groupName) throws FacePassException;
```

参数说明

参数名称	参数类型	是否必选	参数说明
groupName	String	是	要删除的本地group的名称。

返回值

bool返回值，如果group不存在或者其他问题内部错误，则给false。

getLocalGroups

概述

查询全部底库group，列出对应的groupName。

我们默认group操作、face操作均是SDK底层操作，而客户会基于SDK在上层提供业务层的face和group列表。

SDK的group和user操作，是为了完成底层核心的数据结构，以完成快速的算法操作，并提供有效的结果给上层。

而上层业务层则更多时候是为了实际的业务逻辑操作，以及GUI交互。

所以，在SDK层面，不建议直接使用getLocalGroups进行GUI操作，也不提供诸如游标、翻页等功能。

```
public String[] getLocalGroups();
```

参数说明

无

返回值

参数名	参数类型	出现情况	参数说明
groups	Array	必有，如果没有查询到group，则给空数组	返回数组列表，每一项为一个groupName字符串，规则如下： • 支持4-16位长度 • 支持英文小写字母、英文大写字母、数字、下划线 • 建议正则：/^[a-zA-Z0-9_-]{4,16} 如果整个系统中一个group都没有，则返回空数组(不返回NULL，也不缺省groups字段）。

String数组，返回本地的group的groupName列表，但是不包含id、version、facetoken等内部信息。

getLocalGroupInfo

概述

通过参数groupName，查询group的详细信息。

```
public byte[][] getLocalGroupInfo(String groupName) throws FacePassException;
```

参数说明

参数名	是否必选	参数类型	参数说明
groupName	必选	String	group的名称，由用户作为参数指定，用于group的增删查操作，也是绑定/解绑faceToken的group唯一标识。 参数限制（如果不满足，则报参数错误的通用错误码）： • 支持4-16位长度 • 支持英文小写字母、英文大写字母、数字、下划线 • 建议正则：/^[a-zA-Z0-9_-]{4,16}

返回参数

参数名	参数类型	出现情况	参数说明
faceToken	byte[ ][ ]	必有	24位字符串的字符串数组，入库成功后由系统生成，人脸的唯一标识，可以理解为faceID，用户可以通过faceToken绑定/解绑底库，也可以用于face的删除和查询操作， 返回参数结果为faceToken数组。

addFace

概述

下面花一定的篇幅介绍一下addFace和底库的逻辑关系：

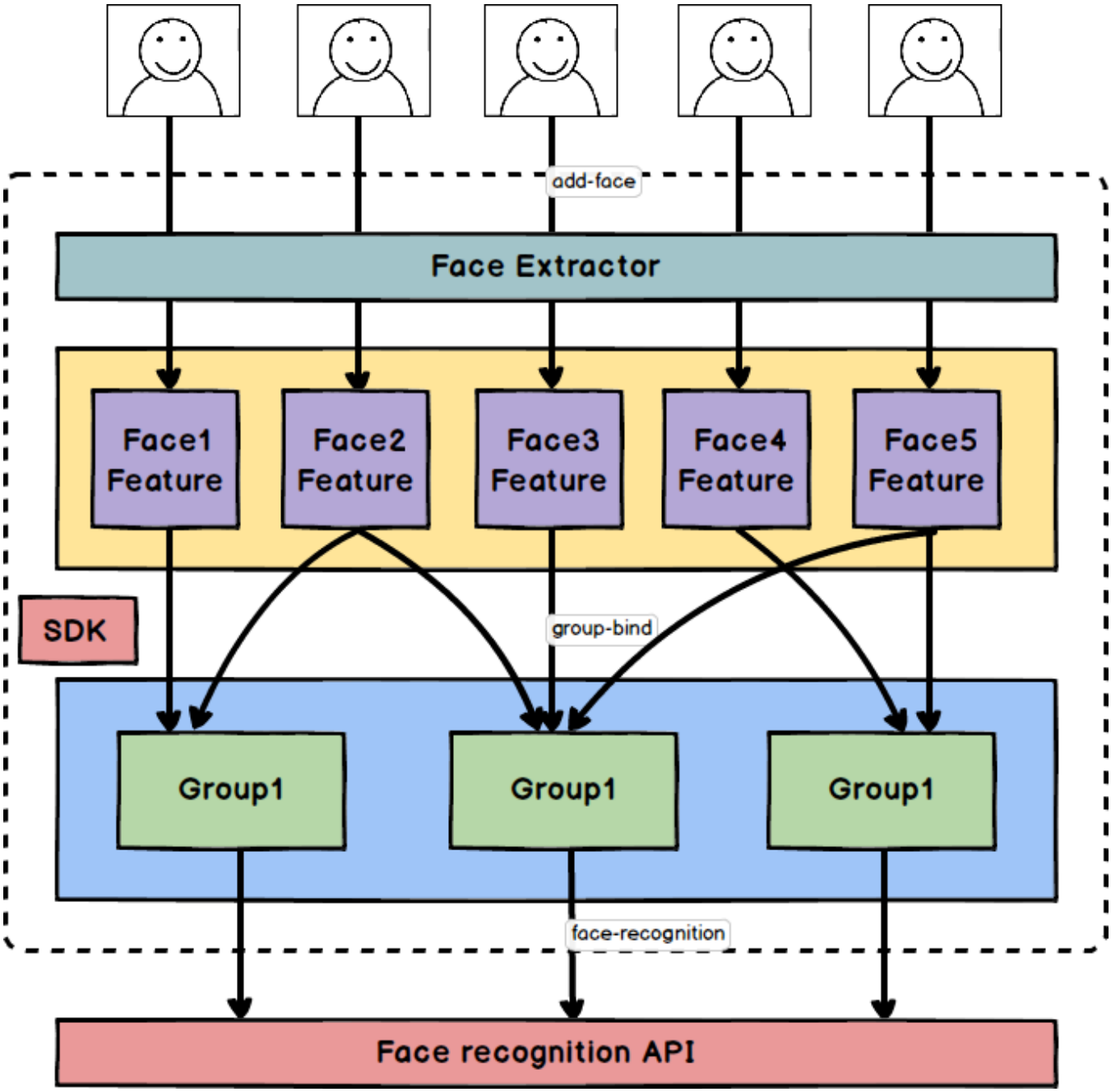
底库是一个人脸特征值的集合，包含了客户希望用于“被识别”的人脸。

FacePass的人脸识别都是基于底库的，一次人脸识别的输入参数是一张目标人脸和一个底库名称。

人脸是通过addFace的API进行注册的，SDK内部是通过一个extractor把一张人脸降维成一个可用于识别算法的特征值。

人脸特征值库是独立于底库的，人脸库是以池子的形式存在，底库和人脸是绑定关系。

也就是说，人脸注册完之后，都是进了人脸大池子，然后每个底库都可以从这个池子里选择需要的目标人员进行绑定。



人脸注册的实际流程为：输入图片字节流，检测其中的最大一张人脸，进行质量判断，如果以上过程都正确，则输出对应faceToken，反之则报错。

1.输入格式为Bitmap，建议约束分辨率大小、过大的图片虽然可以使用，但是可能因为运算缓慢；

2.如果图片中不包含人脸，则无法新增；

3.约定检测的最小人脸为 100*100 px，这是Bitmap符合识别要求的最小尺寸；

4.如果图片中包含多张人脸，则检测最大的人脸（前提是满足脸最小尺寸要求）；

5.如果人脸检测通过，则进行抽特征操作，将图片降解为特征值（这也是实际人脸识别比对用的数据结构）并存储，通过faceToken索引；

6.成功时，返回FaceToken、result、Rect人脸框位置；

7.失败时，返回相应的错误码。

8.注册人脸和人脸绑定底库是两个独立的操作，一张人脸首先要新增到系统中，给出faceToken，然后faceToken绑定到group（可以绑定多个group）。

9.因为添加人脸是较为耗时的操作，建议在子线程中调用此方法。如果在主线程中调用此方法，不建议添加过大图片，否则可能会ANR。

```
public FacePassAddFaceResult addFace(Bitmap faceImage) throws FacePassException;
```

参数说明

参数名	是否必选	参数类型	参数说明

<table>
<tr>
<td>faceImage</td>
<td>必选</td>
<td>Bitmap</td>
<td>
图片属性限制：

1.Bitmap分辨率限制：

• 建议尺寸720p：1280*720
• 支持最小人脸尺寸：必须大于FaceMin，所以必须大于等于100*100。
• 目前并未对最大尺寸做限制，建议图片不要太大，宽高建议最好不要超过2560*2560.
• 分辨率约束的是“面积”，所以长宽比很奇怪的图有可能通过，但前提是其中的人脸满足最小脸像素要求。
• 如果越界，则抛出FacePassException 信息 "INVALID_IMAGE_RESULUTION”。

2.图片控制符限制：

• 我们会读取图片控制符中的如下字段：
  ◦ 旋转：如果图片本身设置了旋转，我们会忠实的按照旋转后的图片来检测人脸，如果参数有误，则有可能导致人脸翻转或者角度过大导致无法检测入库，所以请务必保证输入图片源的该字段表意正确。

3.其他：

• 底图的选择总体目标是“清晰且和使用者本人一致”
• 不建议黑白图
• 不建议使用强烈ps或者艺术化的图
• 建议尽量使用近期生活照，不建议使用跨年龄太久的照片
• 是否化妆、是否戴眼镜，尽量和日常使用习惯一致

人脸属性限制：

检测的综合流程为：

• 检测图片上的人脸；
• 定位最大人脸，并进行质量判断（包括FaceMin）；
• 通过质量判断则入库成功，进行特征值抽取；
• 任意步骤失败则退出。

所以对应的会有如下限制和说明：

1.同框人脸数目：

• 不管原图上有多少人脸，入库环节仅支持单张人脸；
• 如果有多张人脸，仅选择最大的一张；
• 如果最大脸失败，”不会“看第二大的脸，而是直接判失败，所以请确保原图中最大脸就是用户的目标人脸。

2.入库质量判断：

<table>
<tr><th>大类</th><th>子类</th><th>数值</th><th>备注</th></tr>
<tr><td rowspan="3">Pose</td><td>Yaw</td><td><= 20°</td><td>水平角度</td></tr>
<tr><td>Pitch</td><td><= 20°</td><td>垂直角度</td></tr>
<tr><td>Roll</td><td><= 20°</td><td>旋转角度</td></tr>
<tr><td>Blur</td><td>Blurriness</td><td><= 0.2</td><td>清晰度</td></tr>
<tr><td rowspan="2">Illumination</td><td>Brightness</td><td>[65, 210]</td><td>人脸平均亮度</td></tr>
<tr><td>Std_Deviation</td><td><= 80</td><td>人脸亮度标准差</td></tr>
<tr><td>FaceMin</td><td>FaceMin</td><td>>= 100</td><td>最小人脸</td></tr>
</table>

• 最小人脸尺寸（FaceMin或者min-face）：

人脸识别对于比对图片的人脸最小尺寸有要求，太小则损失过多精度、影响识别效果。

  ◦ 默认人脸最小尺寸为：长 >= 100px， 宽 >= 100px，注：等号是成立的，也就是100*100的图可以被使用；
  ◦ 理论上没有最大人脸限制，但是人脸过大会让抽特征的过程略慢于往常，但是是一次性操作，默认可以接受；
  ◦ 如果出现大小完全一样的人脸，则算法会选择先检测到的哪一张。
• 角度pose：大角度会导致人脸信息缺失，我们分三个轴给出限制，越界则报错
  ◦ roll <= 20°：以鼻子为轴、脸部平面旋转；
  ◦ yaw <= 20°：左右摇头；
  ◦ pitch <= 20°：上下点头；
• 模糊blur：照片模糊会导致识别效果变差，所以入库阶段尽量保证图片不模糊，包括
  ◦ 运动模糊：因为高速运动造成的图片内容“拖尾”；
  ◦ 高斯模糊：因为相机失焦造成像素点和周围融合；
  ◦ 其他模糊：比如照片翻拍等其他因素；
• 光照illuminatin：人脸识别对于光照整体上还比较敏感，建议光照柔和清晰、无强烈反差
  ◦ 当前版本没有特别优化逆光、背光；
  ◦ 当前版本没有特别优化暗光；
  ◦ 我们建议平均亮度在 [65-210] lux之间，标准差 <= 80 lux；
</td>
</tr>
</table>

返回参数

具体请参见FacePassAddFaceResult的类定义。

deleteFace

概述

删除人脸。

1.删除对应的人脸faceToken；

2.从group中删除绑定的faceToken关系。

```
public boolean deleteFace(byte[] faceToken) throws FacePassException;
```

参数说明

参数名	是否必选	参数类型	参数说明
faceToken	必选	byte[ ]	24位字符串，入库成功后由系统生成（参见addFace），人脸的唯一标识，可以理解为faceID。 faceToken的编码规则是： • 长度24位 • 支持英文小写字母、英文大写字母、英文半角的等号/下划线/中划线 但是实际API处理中，并不会严格校验编码格式，而是直接解密成所需的明文数据，如果失败则抛出FacePassException。 用户可以通过faceToken绑定/解绑底库，也可以用于face的删除和查询操作。 人脸faceToken会和各种group拥有绑定关系，删除时候需要跟所有的group解绑、然后再删除。 faceToken和group的绑定操作具体参见： FacePassHandler的bindGroup/unbindGroup

返回参数

boolean量，删除成功则为true，否则为false。

getFaceImage

概述

图片内容下载。

目前SDK层面仅存储了底库人脸，一个faceToken对应一张图。

注意这里存储的不是入库时候的原图，而是根据人脸位置抠出来的小图，一般用于GUI层面展示用，

查询方式为传入faceToken，则返回原图中人脸抠图，目前的抠图规则是：

- 抠图的高度等于2倍rect的高，宽度等于2倍rect的宽；
- 宽度扩大，保持中心点不变；
- 高度扩大，上部扩大3/5，下部扩大2/5（换算一下，中心点向上扩展了1/10）。

```
public Bitmap getFaceImage(byte[] faceToken) throws FacePassException;
```

参数说明

参数名	是否必选	参数类型	参数说明
faceToken	必选	byte[ ]	指定faceToken用于返回对应的人脸抠图。 24位字符串，人脸注册成功后由系统生成（参见addFace），人脸的唯一标识。

返回参数

Bitmap，用于显示操作。

bindGroup

概述

绑定一张人脸到底库，以进行识别。

这里汇总一下之前几个核心API的信息到一起，解释一下这块的核心逻辑：

1.“人脸注册”，主要目的是“人脸准入”，确保一张图片符合人脸质量判断需要，可以用于检索。但是这张脸并无法直接被比对，因为所有搜索都是基于groupName的；

2.所有搜索基于groupName，是因为算法底层对这块进行了各种优化（主要是并行计算方面），所以没有暴露直接的faceToken操作给用户，而是由算法黑盒比对；

3.人脸的唯一标识是faceToken，底库的唯一标识是groupName；

4.为了真正让比对生效，绑定faceToken到groupName，再对应的在客户端配置目标为groupName即可；

5.重复绑定不报错，正常再次绑定即可。

```
public boolean bindGroup(String groupName, byte[] faceToken)throws FacePassException;
```

参数说明

参数名	是否必选	参数类型	参数说明
groupName	必选	String	group的名称，由用户作为参数指定，用于group的增删查操作，也是绑定/解绑faceToken的group唯一标识。 参数限制（如果不满足，则报参数错误的通用错误码）： • 支持4-16位长度 • 支持英文小写字母、英文大写字母、数字、下划线 • 建议正则：/^[a-zA-Z0-9_-]{4,16}
faceToken	必选	byte[ ]	24位字符串，入库成功后由系统生成（参见addFace），人脸的唯一标识，可以理解为faceID。 用户可以通过faceToken绑定/解绑底库，也可以用于face的删除和查询操作。

返回参数

boolean量，绑定成功为true，反之为false。

错误码

无

unbindGroup

概述

这是bindGroup的对立操作，解绑一张人脸和底库的绑定关系，更严格的说，是faceToken和groupName的绑定关系。

解绑之后，对目标groupName的搜索就不再包含此faceToken，但是目前支持门禁机配置多个group，所以其他groupName会继续识别（因为没有解绑）。

所以如果目的是为了让门禁机不再识别某个人脸，则请务必确保该门禁机上所有配置的group都完成了解绑操作。

如果对应的绑定关系不存在，则需要报错，解绑失败。

```
public boolean unbindGroup(String groupName, byte[] faceToken) throws FacePassException ;
```

参数说明

参数名	是否必选	参数类型	参数说明
groupName	必选	String	group的名称，由用户作为参数指定，用于group的增删查操作，也是绑定/解绑faceToken的group唯一标识。 参数限制（如果不满足，则报参数错误的通用错误码） • 支持4-16位长度 • 支持英文小写字母、英文大写字母、数字、下划线 • 建议正则：/^[a-zA-Z0-9_-]{4,16}
faceToken	必选	byte[ ]	24位字符串，入库成功后由系统生成（参见addFace），人脸的唯一标识，可以理解为faceID。 用户可以通过faceToken绑定/解绑底库，也可以用于face的删除和查询操作。

返回参数

boolean量，绑定成功为true，反之为false。

setAddFaceConfig

概述

用于设置入库阈值参数，传入FacePassConfig的一个实例，见FacePassConfig构造函数四。

注意，用户自定义入库配置后，识别准确率可能会由于底库质量不足而受到影响，一般情况下请不要使用此方法自定义入库配置。

```
public boolean setAddFaceConfig(FacePassConfig config) throws FacePassException
```

参数说明

参数名称	参数类型	是否必选	参数说明
config	FacePassConfig	是	传入FacepassConfig的一个实例，包含各种入库参数阈值

返回值

boolean

getAddFaceConfig

概述

用于获取当前入库的配置，注意这个方法返回的config中只有阈值相关的项是有意义的，其余模型等项都是未初始化的值。

```
public FacePassConfig getAddFaceConfig()
```

返回值

FacePassConfig的实例

addFaceDetect

概述

向SDK传输来自于相机的一帧图片以进行人脸检测和识别，同步返回这一帧的检测结果，如果人脸质量符合入库要求，会返回image交给客户、用于人脸注册。

这个接口的使用方式类似于feedFrame，建议客户输入连续的相机帧，以便于SDK发挥tracker和质量判断的作用。

但是与feedFrame不同的是：

- 返回值是当前每一帧的结果，而非feedFrame的每个track的结果；
- 仅返回当前最大的人脸信息，并不返回多人列表，原因是人脸注册逻辑上只能针对一张人脸；

在实际使用过程中请不要混用。

```
public FacePassAddFaceDetectionResult addFaceDetect(FacePassImage image) throws FacePassException
```

参数说明

参数名称	参数类型	是否必选	参数说明

image	FacePassImage	是	从相机帧处理成FacePassImage对象，类型参数定义详见FacePassImage类。 主要包括raw流、高端、旋转、编码等。 客户需要自行处理相机输入帧，并且通过以上参数初始化Image对象，具体参见示例。

返回值

FacePassAddFaceDetectionResult对象，具体参见该类的定义。

compare

概述

传入两张图，进行1:1比对。

给入的两张图会进行人脸检测，同时如果开启了活体判断参数，则会同时进行活体检测。

人脸检测失败，则会报错（不符合标准的照片进行compare没有意义，我们也无法保障结果具有实际意义）。

如果以上环节均正确，则会返回一个compre的分值，用于判断是否通过。

分值在[0，100]之间，越高表示越相似（同一个人），越低表示越不相似（不同的两个人）。

通常做法是分值会和一个阈值进行比较，这里一般的建议阈值是70。

说明，和一般的图片使用规则不同，compare接口目前不对图片质量进行严格的校验，原因是这是一个很底层的API，客户会用于各种复杂场景，很难给出有效的质量体系。

所以我们的策略是，API返回值中给出各种细节的参数，如果用户可以自行限制质量体系（并且无视我们的比对分值）。

```
FacePassCompareResult compare(Bitmap image1, Bitmap image2, boolean enableLiveness) throw FacePassException;
```

参数说明

参数名	参数类型	是否必选	参数说明
image1	Bitmap	必选	用于比对的两张图，Bitmap格式。 **Bitmap图片属性限制：** 1.分辨率限制： • 建议尺寸720p：1280*720 • 支持最小人脸尺寸：必须大于FaceMin，所以必须大于等于100*100。 • 目前并未对最大尺寸做限制，建议图片不要太大，宽高建议最好不要超过2560*2560，否则注册会比较缓慢。 • 分辨率约束的是“面积”，所以长宽比很奇怪的图有可能通过，但前提是其中的人脸满足最小脸像素要求。 • 如果越界，则抛出FacePassException 信息 "INVALID_IMAGE_RESULUTION”。 2.图片控制符限制： • 我们会读取图片控制符中的如下字段： • 旋转：如果图片本身设置了旋转，我们会忠实的按照旋转后的图片来检测人脸，如果参数有误，则有可能导致人脸翻转或者角度过大导致无法检测入库，所以请务必保证输入图片源的该字段表意正确。 3.其他： • 底图的选择总体目标是“清晰且和使用者本人一致” • 不建议黑白图 • 不建议使用强烈ps或者艺术化的图 • 建议尽量使用近期生活照，不建议使用跨年龄太久的照片 • 是否化妆、是否戴眼镜，尽量和日常使用习惯一致

image2			**人脸属性限制：** 1:1场景下图片输入来源复杂，所以这个场景下不进行质量判断。 返回值中会给出详细的质量分数（3D-Pose和Blurness），如果确实需要限制，客户可以自行判断。
enableLiveness	boolean	必选	是否判断活体。 说明：因为活体检测比较占用客户端资源，所以在高频使用的时候建议置为false。 另外，如果Bitmap本身没有被检测到人脸，会直接报错，不再浪费资源进行活体判断。

返回值

返回识别结果FacePassCompareResult。

getConfig

概述

获取FacePassHandler当前的config配置

```
public FacePassConfig getConfig();
```

参数说明

无

返回值

FacePassConfig实例。

getAddFaceConfig

概述

获取当前的入库配置

```
public FacePassConfig getAddFaceConfig();
```

参数说明

无

返回值

FacePassConfig实例。

setConfig

概述

设置FacePassHandler的config。

一个典型的场景是：

- 通过getConfig获取当前config；
- 修改config中某个public成员变量；
- 通过setConfig生效新的config。

需要说明一下的是，config的生效是以一次feedFrame为周期的，也就是说，setConfig生效后，下一帧feedFrame才真正生效的，之前已经发出去的协议并不受改变。

```
public int setConfig(FacePassConfig config) throws FacePassException;
```

参数说明

参数名称	参数类型	是否必选	参数说明
config	FacePassConfig	是	参见FacePassConfig类的参数说明

返回值

boolean类型，是否完成配置更新

setAddFaceConfig

概述

设置入库参数。

入库使用的参数与setConfig中的参数是不同的，如果不调用setAddFaceConfig，则会使用默认入库参数。当需要改变入库参数时（如改变光照限制阈值、最小人脸阈值等），需要调用此接口。

```
public boolean setAddFaceConfig(FacePassConfig config) throws FacePassException
```

参数说明

参数名称	参数类型	是否必选	参数说明
config	FacePassConfig	是	参见FacePassConfig类的参数说明

返回值

boolean类型，是否完成配置更新

reset

概述

重置当前handler的设置至初始状态。

reset主要是清楚track和frame的信息，可以理解为handler初始化完成那一刻的状态。

```
public void reset();
```

参数说明

无

返回值

无

release

概述

释放当前SDK handler。

```
public void release();
```

参数说明

无

返回值

无