ALIBABA CLOUD

阿里云

金融级实人认证
开发指南

文档版本：20220712

阿里云

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
危险	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。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会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，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	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。	单击**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。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	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。	`switch {active|stand}`

目录

1.金融级实人认证方案 09
1.1. App（SDK）接入 09
1.1.1. 接入流程 09
1.1.2. 服务端接入 11
1.1.2.1. InitFaceVerify 11
1.1.2.2. DescribeFaceVerify 15
1.1.2.3. Java SDK调用示例 19
1.1.2.4. PHP SDK调用示例 24
1.1.2.5. Python SDK调用示例 28
1.1.2.6. Go SDK调用示例 32
1.1.2.7. .NET SDK调用示例 37
1.1.2.8. Node.js SDK调用示例 42
1.1.3. App（SDK）接入 46
1.1.3.1. Android接入 46
1.1.3.2. iOS接入 51
1.1.3.3. UniApp接入 58
1.1.3.4. APICloud接入 63
1.2. PC或移动端H5网页接入 66
1.2.1. 接入流程 66
1.2.2. 服务端接入 66
1.2.3. PC或移动端H5网页接入 74
1.3. 支付宝H5页面接入 77
1.3.1. 接入流程 77
1.3.2. 服务端接入 78
1.3.2.1. HTTPS原生调用 78
1.3.2.2. SDK调用接入 85

1.3.3. 支付宝H5页面接入 96
1.3.3.1. 端内接入支付宝H5页面 96
1.3.3.2. 端外唤起支付宝H5页面 98
1.4. 支付宝小程序接入 99
1.4.1. 接入流程 99
1.4.2. 服务端接入 101
1.4.2.1. HTTPS原生调用 101
1.4.2.2. SDK调用接入 108
1.4.3. 支付宝小程序刷脸接入 118
1.5. 纯服务端（API）接入 120
1.5.1. 接入流程 120
1.5.2. 照片实人认证 120
1.5.3. 视频实人认证 127
1.5.4. Java SDK调用示例 133
1.5.5. PHP SDK调用示例 139
1.5.6. Python SDK调用示例 147
1.5.7. Go SDK调用示例 153
1.5.8. .NET SDK调用示例 161
1.5.9. Node.js SDK调用示例 169
1.6. 参数加密说明 175
2.金融级多因子意愿认证方案 177
2.1. 接入流程 177
2.2. 服务端接入 179
2.2.1. InitFaceVerify 179
2.2.2. DescribeFaceVerify 182
2.2.3. Java SDK调用示例 188
2.2.4. PHP SDK调用示例 192
2.2.5. Python SDK调用示例 197

2.2.6. Go SDK调用示例 200
2.2.7. .NET SDK调用示例 205
2.2.8. Node.js SDK调用示例 210
2.3. App（SDK）接入 214
2.3.1. Android意愿认证接入 214
2.3.2. iOS意愿认证接入 218
3.金融级活体人脸验证方案 224
3.1. 接入流程 224
3.2. 服务端接入 226
3.2.1. InitFaceVerify 226
3.2.2. DescribeFaceVerify 233
3.2.3. Java SDK调用示例 237
3.2.4. PHP SDK调用示例 241
3.2.5. Python SDK调用示例 246
3.2.6. Go SDK调用示例 250
3.2.7. .NET SDK调用示例 255
3.2.8. Node.js SDK调用示例 260
3.3. App（SDK）接入 264
3.3.1. Android接入 264
3.3.2. iOS接入 269
3.3.3. UniApp接入 275
3.3.4. APICloud接入 279
4.金融级活体检测方案 283
4.1. 服务端接入 283
4.1.1. InitFaceVerify 283
4.1.2. DescribeFaceVerify 286
4.1.3. Java SDK调用示例 290
4.1.4. PHP SDK调用示例 294

4.1.5. Python SDK调用示例 299
4.1.6. Go SDK调用示例 302
4.1.7. .NET SDK调用示例 307
4.1.8. Node.js SDK调用示例 312
4.2. App（SDK）接入 316
4.2.1. 接入流程 316
4.2.2. Android接入 318
4.2.3. iOS接入 323
4.2.4. UniApp接入 330
4.2.5. APICloud接入 335
4.3. PC或移动端H5网页接入 338
4.3.1. 接入流程 338
4.3.2. 服务端接入 338
4.3.3. PC或移动端H5网页接入 344
4.4. 纯服务端（API）接入 346
4.4.1. LivenessFaceVerify 346
4.4.2. Java SDK调用示例 351
4.4.3. PHP SDK调用示例 354
4.4.4. Python SDK调用示例 356
4.4.5. Go SDK调用示例 359
4.4.6. .NET SDK调用示例 362
4.4.7. Node.js SDK调用示例 365
5.金融级人脸比对方案 368
5.1. 接入流程 368
5.2. 纯服务端（API）接入 368
5.3. Java SDK调用示例 373
5.4. PHP SDK调用示例 375
5.5. Python SDK调用示例 378

5.6. Go SDK调用示例 381
5.7. .NET SDK调用示例 384
5.8. Node.js SDK调用示例 387

1.金融级实人认证方案

1.1. App（SDK）接入

1.1.1. 接入流程

无线SDK和服务端接入适用于您自身已有手机App应用，且希望通过该App对用户进行线上认证的场景。

前置条件

您必须了解通过HTTP、HTTPS原生调用金融级实人认证服务接口时，构造请求签名的方法。更多签名机制的信息，请参见API签名机制。

操作步骤

1. 开通金融级实人认证服务。
2. 添加认证场景。

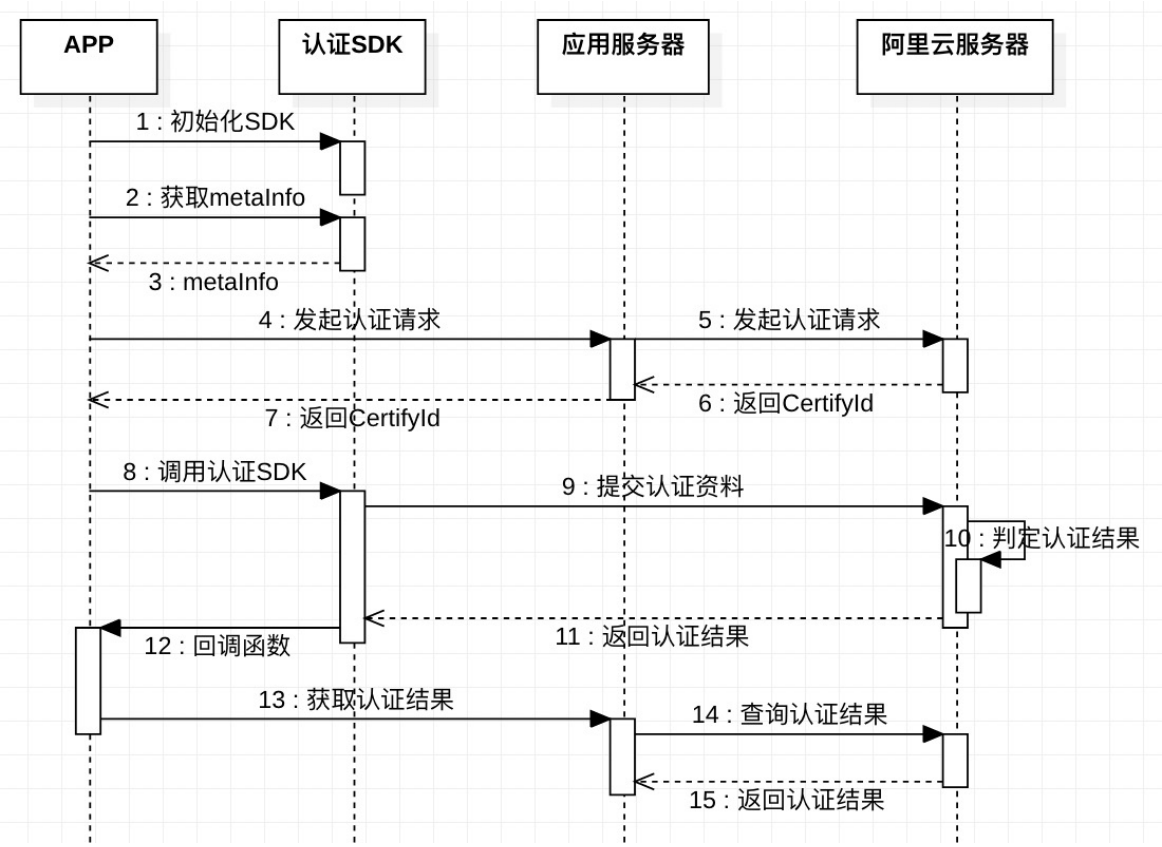
   > **说明** 基于不同业务场景，您可以在添加认证场景时，手动配置是否授权阿里云访问指定OSS存储空间用于存储认证的资料图片。

3. 如果需要将认证过程中的人脸采集照片存储到OSS空间，您需要授权金融级实人认证访问OSS存储空间。
4. 接入App认证方案。

   在您完成上述步骤后，您可以进行无线SDK+服务端接入，接入步骤如下：

   i. 调用服务端初始化接口。
   ii. 调用接入客户端SDK。
   iii. 调用服务端查询接口。

   您可以查看下方调用时序图了解详细接入步骤。



系统调用时序说明表

序号	说明
1	初始化认证SDK。
2	App调用认证SDK获取MetaInfo。
3	认证SDK返回MetaInfo。
4	App向应用服务端发起认证请求并传递MetaInfo。
5	应用服务端向阿里云服务端调用发起认证请求接口。
6	阿里云服务端返回CertifyId给应用服务端。
7	应用服务端将CertifyId传递给App。
8	App调用认证SDK。
9	认证SDK将用户提交的认证资料传递给阿里云服务端。
10	阿里云服务端根据资料判定认证结果。
11	阿里云服务端将认证结果返回给SDK。
12	SDK通过回调函数指引App获取认证状态。

序号	说明
13	App向应用服务端查询认证状态。
14	应用服务端向阿里云服务端调用查询认证结果接口。
15	阿里云服务端将认证结果返回给应用服务端。

在您了解接入步骤后，您可以通过以下方式接入App认证方案：

- 进行服务端接入，请参见服务端接入。
- 在应用客户端集成SDK。具体集成步骤，请根据平台系统，参见以下文档：
  - Android接入
  - iOS接入
  - UniApp接入
  - APICloud接入

1.1.2. 服务端接入

1.1.2.1. InitFaceVerify

本文介绍了App认证方案的接入流程。

发起认证请求

接口名：InitFaceVerify。

服务地址：`cloudauth.aliyuncs.com`。

请求方法：HTTPS POST和GET。

接口说明：每次开始认证前通过本接口获取CertifyId，用来串联认证请求中的各个接口。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	要接入的认证场景ID。该ID在控制台创建认证场景后自动生成。	100000****
OuterOrderNo	String	是	您自定义的业务唯一标识，用于后续定位排查问题使用。 支持长度为32位的字母和数字的组合，请确保唯一。	e0c34a77f5ac40a5aa5e6ed20c35****

名称	类型	是否必选	描述	示例值
ProductCode	String	是	要接入的认证方案。取值： • ID_PRO。使用ID_PRO认证方案，您的用户仅需要做活体认证。 • ID_PLUS。使用ID_PLUS认证方案，您的用户需要拍摄身份证和活体认证。 说明 如果您的业务需要同步采集身份证信息，可选择ID_PLUS方案。	ID_PRO
Model	String	否	要进行活体检测的类型。取值： • LIVENESS（默认）：眨眼动作活体检测。 • PHOTINUS_LIVENESS：眨眼动作活体+炫彩活体双重检测。 • MULTI_ACTION：多动作活体检测。当前为眨眼+任意摇头检测。 说明 活体检测模式仅支持运行在Android SDK 1.2.6及以上版本和iOS SDK 1.2.4及以上版本；多动作活体检测模式仅支持运行在Android SDK 1.6.4及以上版本和iOS SDK 1.5.1及以上版本。	LIVENESS
CertType	String	是	您的终端用户传入的证件类型。 当前仅支持身份证，取值固定为IDENTITY_CARD。 说明 当ProductCode为ID_PLUS时，CertType为非必填字段。	IDENTITY_CARD
CertName	String	是	您的终端用户的真实姓名。 说明 当ProductCode为ID_PLUS时，CertName为非必填字段。	张三

名称	类型	是否必选	描述	示例值
CertNo	String	是	您的终端用户的证件号码。 说明 当ProductCode为ID_PLUS时，CertNo为非必填字段。	330103**********
MetaInfo	String	是	MetaInfo环境参数，需要通过客户端SDK获取。	{"zimVer":"3.0.0","appVersion": "1","bioMetaInfo": "4.1.0:11501568,0","appName": "com.aliyun.antcloudauth","deviceType": "ios","osVersion": "iOS 10.3.2","apdidToken": "","deviceModel": "iPhone9,1"}
Mobile	String	否	您的终端用户的手机号码。	130********
Ip	String	否	您的终端用户的IP。	114.100.XX.XX
UserId	String	否	您自定义的用户ID，请保持唯一。	123456789
CallbackUrl	String	否	认证结果的回调通知地址，必须以 `https` 开头。 平台在完成认证后会回调该地址，并自动添加 `certifyId` 、 `passed` 和 `subcode` 字段，示例： `https://www.aliyun.com?certifyId=xxxx&passed=T&subcode=200` 。	https://www.aliyun.com
CallbackToken	String	否	安全Token，由您自行生成，用于防重复、防篡改校验。 如果设置了该值会在回调地址显示**CallbackToken**字段。	NMjvQanQgplBSaEI0sL86WnQplB
AuthId	String	否	用户授权ID，最大长度为64位字符。	92d46b9e9e2d703f2897f350d5bd4149

名称	类型	是否必选	描述	示例值
EncryptType	String	否	加密类型。为空表示不加密。 如开启加密传输，需传入加密算法。目前仅支持SM2国密算法。 如需传入加密算法，需对CertName和CertNo进行加密，并传入加密后的密文。有关参数加密的更多信息，请参见参数加密说明。	SM2

返回数据

名称	类型	描述	示例值
RequestId	String	请求ID。	130A2C10-B9EE-4D84-88E3-5384FF039795
Message	String	请求消息的响应信息。	success
Code	String	返回码。关于返回码的具体信息，请参见返回Code和Message说明。	200
ResultObject.CertifyId	String	实人认证唯一标识。	91707dc296d469ad38e4c5efa6a0****

返回Code和Message说明

Code	Message	描述
200	success	成功。
400	参数不能为空	参数不能为空。
401	参数非法	非法参数。传入的姓名、身份证号码长度必须符合国家标准且不得包含英文字母等特殊字符。
402	应用配置不存在	应用配置不存在。
404	认证场景配置不存在	认证场景配置不存在，请先在控制台上创建认证场景。
410	未开通服务	未开通OSS产品或未完成OSS读写授权，请登录控制台完成授权。具体操作，请参见授权金融级实人认证访问OSS存储空间
411	RAM无权限	需要给RAM用户授予**AliyunAntCloudAuthFullAccess**的操作权限。
412	欠费中	金融级实人认证或OSS存在欠费，请充值后操作。
414	设备类型不支持	当前移动设备不支持刷脸认证，请更换设备后操作。

Code	Message	描述
415	SDK版本不支持	当前认证SDK版本不支持刷脸认证，请升级SDK后操作。
416	系统版本不支持	当前操作系统版本不支持刷脸认证，请升级系统或更换设备操作。
417	无法使用刷脸服务	当前身份信息无可用比对源，导致此错误的原因请参考认证结果错误码ResultObject.SubCode）201、202和203的描述说明。更多内容，请参见DescribeFaceVerify。
418	刷脸失败次数过多	当天刷脸认证次数过多，请明天再试。 说明 一个身份信息在一天内最多可以认证失败20次，无法调整更多的认证失败次数。
500	系统错误	系统内部错误，请反馈工程师排查。

1.1.2.2. DescribeFaceVerify

本文介绍如何通过DescribeFaceVerify接口查询App实人认证方案的认证结果。

接口说明

接口名：**DescribeFaceVerify**。

服务地址：`cloudauth.aliyuncs.com`。

请求方法：HTTPS POST和GET。

接口说明：当您收到回调通知之后，可以在服务端通过该接口获取相应的认证状态和认证资料。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	认证场景ID。	1000000006
CertifyId	String	是	实人认证唯一标识。该值需要调用InitFaceVerify获取。	91707dc296d469ad38e4c5efa6a0****

返回数据

名称	类型	描述	示例值
RequestId	String	请求ID。	130A2C10-B9EE-4D84-88E3-5384FF039795
Message	String	请求信息的响应消息。	success

名称	类型	描述	示例值
Code	String	返回码，200指接口响应成功。详细认证结果判断请参见ResultObject.SubCode。	200
ResultObject.Passed	String	认证结果。取值： ● T：通过。 ● F：不通过。 说明 判断认证结果请以ResultObject.Passed字段为准。	T
ResultObject.SubCode	String	认证结果描述。更多信息，请参见ResultObject.SubCode错误码说明。 说明 判断认证结果请以ResultObject.Passed字段为准。	200
ResultObject.IdentityInfo	String	认证的主体信息，一般的认证场景返回为空。	null
ResultObject.DeviceToken	String	设备Token。	McozS1ZWRcRZStlERcZZo_QOytx5jcgZoZJEoRLOxxxxxxx
ResultObject.MaterialInfo	String	认证主体附件信息，主要为图片类材料。关于JSON格式，请参见下文中的返回示例。	{"faceAttack":"F","faceOcclusion":"F","facialPictureFront":{"faceAttackScore":0.00008597839769208804,"qualityScore":88.3615493774414,"verifyScore":50.28594166529785,"pictureUrl":"https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg","ossBucketName":"cn-shanghai-aliyun-cloudauth-1260051251634779","ossObjectName":"verify/1260051251634779/6ba7bcfccf33f56cdb44ed086f36ce3e0.jpeg"}}

返回示例

ResultObject.MaterialInfo的JSON格式示例

```
{
    //是否为攻击：攻击为T，非攻击为F。
    "faceAttack": "T",
    // 是否有脸部遮挡：有脸部遮挡为T，否则为F。
    "faceOcclusion": "F",
```

```
    // 认证的照片信息。
    "facialPictureFront": {
        // 人脸攻击分。
         "faceAttackScore": 0.00008597839769208804,
        // 人脸和身份证照片比对分（仅当product code=ID_PLUS时，认证完成返回）。
        "idCardVerifyScore": 50.99797734769861,
        // 照片存放客户上海区域的OSS里，此为照片的bucket名。
        "ossBucketName": "cn-shanghai-aliyun-cloudauth-1260051251634779",
        // 照片存放客户上海区域的OSS里，此为照片的文件名。
        "ossObjectName": "verify/1260051251634779/03a081bd96328aedf83f635f39a50c57_0.jpeg",
        // 照片https地址，有效期15分钟，从查询时开始计时。
        "pictureUrl": "http://cn-shanghai-aliyun-cloudauth-1260051251634779.oss-cn-shanghai
.aliyuncs.com/verify/1260051251634779/03a081bd96328aedf83f635f39a50c57_0.jpeg?Expires=16253
71140&OSSAccessKeyId=STS.NTX1ngfr6Acg2Pmnn2RYM****&Signature=Hw5BF9WxJs6wI68IxKs41cxCU8****
&security-token=CAISjgJ1q6Ft5B2yfSjIr5ftetTTi60X9qGMMHbcim5nXtZhu7GT1Dz2IH1PdXFgA%2Bgds%2Fs
wmm5U7vgalrkqEcEdHRGdN5YpsM8LrlzwO1h2TGRsq%2B5qsoasPETOITyZtZagToeUZdfZfejXGDKgvyRvwLz8WCy%
2FVli%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGfLC1dysQcO4gEWq4bHm5fCskKD1Qenk7F
O%2B9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWwIMvErYbbaMrIV1WwZ9Z7knHaVAq%2BXw
nOBkuuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAF%2FHZbelNLYWtipAu5X5oG1t22SqIr1p3TMK5FrjDIVeWOpp
cxeRXtU%2BjR7hGcwd25scGhiraoxTXV3sxw%2F6dhFSswQ37O4j%2B%2FrCPtiGauqn2ZMdMj%2FvYwKb6KmpQqa%2
BtR%2F9cxhS6hoIQRq0SUIfyXl5ZUiZbTjW22iwuX%2BPwzVCw%3D%3D",
        // 视频存放客户上海区域的OSS里，此为视频的文件名（仅当客户端设置视频参数时返回）。
        "ossVerifyVideoObjectName": "verify/1260051251634779/03a081bd96328aedf83f635****_ve
rifyvideo_dcb7.mov",
        // 视频HTTPS地址，有效期15分钟，从查询时开始计时（仅当客户端设置视频参数时返回）。
        "verifyVideoUrl": "http://cn-shanghai-aliyun-cloudauth-1260051251634779.oss-cn-shan
ghai.aliyuncs.com/verify/1260051251634779/03a081bd96328aedf83f635f39a50c57__verifyvideo_dcb
7.mov?Expires=1625371140&OSSAccessKeyId=STS.NTX1ngfr6Acg2Pmnn2RYM****&Signature=Hw5BF9WxJs6
wI68IxKs41cxCU8****&security-token=CAISjgJ1q6Ft5B2yfSjIr5ftetTTi60X9qGMMHbcim5nXtZhu7GT1Dz2
IH1PdXFgA%2Bgds%2Fswmm5U7vgalrkqEcEdHRGdN5YpsM8LrlzwO1h2TGRsq%2B5qsoasPETOITyZtZagToeUZdfZf
ejXGDKgvyRvwLz8WCy%2FVli%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGfLC1dysQcO4gEW
q4bHm5fCskKD1Qenk7FO%2B9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWwIMvErYbbaMrIV
1WwZ9Z7knHaVAq%2BXwnOBkuuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAF%2FHZbelNLYWtipAu5X5oG1t22SqI
r1p3TMK5FrjDIVeWOppcxeRXtU%2BjR7hGcwd25scGhiraoxTXV3sxw%2F6dhFSswQ37O4j%2B%2FrCPtiGauqn2ZMd
Mj%2FvYwKb6KmpQqa%2BtR%2F9cxhS6hoIQRq0SUIfyXl5ZUiZbTjW22iwuX%2BPwzVCw%3D%3D",
        // 活体人脸质量分数。
        "qualityScore": 99.93476867675781,
        // 人脸和公安比对分数，阈值可参考下表详细说明。
        "verifyScore": 57.678396649466755
    },
    // 身份证OCR信息（仅当product code=ID_PLUS时，认证完成返回OCR识别的信息和图片）。
    "ocrIdCardInfo": {
        // 身份证字段信息。
    },
    "ocrPictureFront": {
        // 身份证OCR照片存放客户上海区域的OSS里，此为身份证OCR照片的Bucket名。
        "ossBucketName": "cn-shanghai-aliyun-cloudauth-1234567890",
        // 身份证OCR照片存放客户上海区域的OSS里，此为身份证OCR照片人脸面的文件名。
        "ossIdFaceObjectName": "verify/1234567890/f7ed1ef80ad1234fdsd95c66d83340cd_ocridfac
e_b749.jpeg",
        // 身份证OCR照片人脸面的https地址，有效期15分钟，从查询时开始计时。
        "ossIdFaceUrl": "http://cn-shanghai-aliyun-cloudauth-1234567890.oss-cn-shanghai.ali
yuncs.com/verify/1234567890/f7ed1ef80ad1234fdsd95c66d83340cd_ocridface_b749.jpeg?Expires=16
25371165&OSSAccessKeyId=STS.NTGterbRtcVY7ykxixaQ6****&Signature=SqZoqBch%2BsdJefkabdaIJrPJW
```

```
qtD%2FlmF****&security-token=CAISjdfgeJ1q6Ft5B2yfSjIr5fyP9%2FGj41V1JSyNV%2FanGktbd4aqInAmjz
2IH1PdXFgA%2Bgds%2Fswmm5U7vgalrkqEcEdHRGdN5YpsM8LrlzwO30McmRsq%2B5qsoasPETOITyZtZagToeUZdfZ
fejXGDKgvyRvwLz8WCy%2FVli%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGfLC1dysQcO4gE
Wq4bHm5fCskKD1Qenk7FO%2B9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWwIMvErYbbaMrI
V1WwZ9Z7knHaVAq%2BXwnOBkuuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAGe2a1NFO2XabhICqmUeN%2FLV25Gu
haexEjjGL4sDC2VWjasWBmKzlXTQ6mHZoI9IhswopdGLLlWPC5ONnLNASWQEObAM28FAViS76clUdE8YVCMyU%2BuyF
qLGeRNx4NjcPa%2FlOa9wsxjg4LwlfHKeXA3g0X%2BQQcRAqFc2j093K2r6A%3D%3D",
        // 身份证OCR照片存放客户上海区域的OSS里，此为身份证OCR照片国徽面的文件名。
        "ossIdNationalEmblemObjectName": "verify/1234567890/f7ed1ef80ad1234fdsd95c66d83340c
d_ocridnationalemblem_a3hf.jpeg",
        // 身份证OCR照片国徽面的https地址，有效期15分钟，从查询时开始计时。
        "ossIdNationalEmblemUrl": "http://cn-shanghai-aliyun-cloudauth-1234567890.oss-cn-sh
anghai.aliyuncs.com/verify/1234567890/f7ed1ef80ad1234fdsd95c66d83340cd_ocridnationalemblem_
a3hf.jpeg?Expires=1625371365&OSSAccessKeyId=STS.NTEz8oi35qJpxxX7no5gpGd****&Signature=wnHjN
u3333ddW3zNMup1r4ovgrw6xJ0****&security-token=CAISjgJ1q6Ft5B2yfSjIr5fwMYLbhOwUxoibel7p0246O
ehcqLL%2BpTz2IH1PdXFgA%2Bgds%2Fswmm5U7vgalrkqEcEdHRGdN5YpsM8LrlzwO34PcmRsq%2B5qsoasPETOITyZ
tZagToeUZdfZfejXGDKgvyRvwLz8WCy%2FVli%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGf
LC1dysQcO4gEWq4bHm5fCskKD1Qenk7FO%2B9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWw
IMvErYbbaMrIV1WwZ9Z7knHaVAq%2BXwnOBkuuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAFThVhIcXH8zQiqbYV
kTCHpZVRmDHH0mZ%2FMNuGikqSWdK3%2FhaSEvxmO1wyLH9E7mFzoKJErtxTCs453H6PYHO8H3pKOpKgZ0PfWTpJ8OE
IJm26fKph%2F%2F3usSUBx5h0Ob17e2XB4BffkOJKQQiU3vXqoOZ6Vlyf7QpEJSvUMXXSKSg%3D%3D"
    }
}
```


Result Object.SubCode错误码说明

错误码	认证描述文案	是否计费	可能原因和运营建议
200	认证通过	是	调用成功。
201	姓名和身份证不一致	是	可能是用户的信息有误或用户的信息为假信息，建议用户确认后重新操作。若同一身份信息重新发起认证，服务端初始化将会返回417错误。
202	查询不到身份信息	是	可能是用户户口迁移、军人等特殊状态导致，建议预留人工审核入口，进行人工审核。若同一身份信息重新发起认证，服务端初始化将会返回417错误。
203	查询不到照片或照片不可用	是	可能是公安库数据问题导致，建议预留人工审核入口，进行人工审核。若同一身份信息重新发起认证，服务端初始化将会返回417错误。
204	人脸比对不一致	是	可能不是同一人或活体照片质量较低，建议根据业务情况分层处理，若为同一人可重新操作。
205	活体检测存在风险	是	可能存在攻击风险，建议人工审核分层处理，若为真人可重新操作。
206	业务策略限制	是	个性化业务安全策略限制，如有疑问可咨询客服。
207	人脸与身份证人脸比对不一致	是	仅在使用ID_PLUS方案时返回，可能人证不符或人脸照片质量较低，建议根据业务情况分层处理，若为同一人可重新操作。
209	权威比对源异常。	是	可能存在权威比对源异常，无法比对。

错误码	认证描述文案	是否计费	可能原因和运营建议
210	认证通过	是	通过产品安全策略综合判断，建议认证通过。

verifyScore阈值说明

千分之一误识率	万分之五误识率	万分之一误识率	十万分之五误识率	十万分之一误识率
70	71.5	75	76.5	80

说明 如果您有个性化需求，您可以根据业务情况，参考返回的比对分和阈值，自定义认证结果。

返回Code和Message说明

Code	Message	描述
200	success	成功。
400	参数不能为空	参数不能为空。
401	参数非法	非法参数。传入的姓名、身份证号码长度必须符合国家标准且不得包含英文字母等特殊字符。
403	认证未完成	认证未完成。
404	认证场景配置不存在	认证场景配置不存在，请先在控制台上创建认证场景。
406	无效的CertifyId	无效的**CertifyId**。
410	未开通服务	未开通OSS产品或未完成OSS读写授权。
411	RAM无权限	需要给RAM用户授予**AliyunAntCloudAuthFullAccess**的操作权限。
424	身份认证记录不存在	当前CertifyId还未完成认证，建议您客户端收到认证回调为1000或2006时，再请求查询接口获取详情。 **说明** 如果出现CertifyId未认证、认证过程中放弃、认证处理中的情况，均会导致查询接口返回此错误码。
427	业务策略限制	仅在使用**ID_PLUS**方案时出现，指该身份因触发内部风控策略未完成认证。
500	系统错误	系统内部错误。

1.1.2.3. Java SDK调用示例

本文介绍了金融级实人认证服务端接入的Java SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI 开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

本章节提供的是最新版本的Java SDK调用示例，如果您需要查看之前版本，请参见Java SDK调用示例（V1版本）。

前提条件

需要使用的Java版本在1.8版本及以上。

使用Maven引入SDK

```
<dependency>
       <groupId>com.aliyun</groupId>
       <artifactId>cloudauth20190307</artifactId>
        <version>2.0.3</version>
</dependency>
```

调用示例

InitFaceVerify接口调用示例

```
import java.util.Arrays;
import java.util.List;
import com.aliyun.cloudauth20190307.Client;
import com.aliyun.cloudauth20190307.models.*;
import com.aliyun.tea.TeaException;
import com.aliyun.tea.TeaUnretryableException;
import com.aliyun.tearpc.models.Config;
import com.aliyun.teautil.models.RuntimeOptions;
public class InitFaceVerify {
    public static void main(String[] args) throws Exception {
        InitFaceVerifyRequest request = new InitFaceVerifyRequest();
        // 请输入场景ID+L。
        request.setSceneId(xxxxL);
        // 设置商户请求的唯一标识。
        request.setOuterOrderNo("xxxx");
        // 认证方案。
        request.setProductCode("ID_PRO");
        // 模式。
        request.setModel("LIVENESS");
        request.setCertType("IDENTITY_CARD");
        request.setCertName("张三");
        request.setCertNo("xxxx");
        // MetaInfo环境参数。
        request.setMetaInfo("xxxxxxxx");
        //request.setMobile("130xxxxxxxx");
        //request.setIp("114.xxx.xxx.xxx");
        //request.setUserId("12345xxxx");
        //request.setCallbackUrl("https://www.aliyundoc.com");
        //request.setCallbackToken("xxxxx");
        // 如需开启个人信息加密传输。
```

```
        //request.setEncryptType("SM2");
        //request.setCertName("BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==");
        //request.setCertNo("BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg==");
        // 推荐，支持服务路由。
        InitFaceVerifyResponse response = initFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        //InitFaceVerifyResponse response = initFaceVerify("cloudauth.cn-shanghai.aliyuncs.
com", request);
        System.out.println(response.getRequestId());
        System.out.println(response.getCode());
        System.out.println(response.getMessage());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getCertifyId());
    }
    private static InitFaceVerifyResponse initFaceVerifyAutoRoute(InitFaceVerifyRequest req
uest) {
        // 第一个为主区域Endpoint，第二个为备区域Endpoint。
        List<String> endpoints = Arrays.asList("cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com");
        InitFaceVerifyResponse lastResponse = null;
        for (String endpoint : endpoints) {
            try {
                InitFaceVerifyResponse response = initFaceVerify(endpoint, request);
                lastResponse = response;
                // 服务端错误，切换到下个区域调用。
                if (response != null && "500".equals(response.getCode())) {
                    continue;
                }
                return response;
            } catch (Exception e) {
                // 网络异常，切换到下个区域调用。
                if (e.getCause() instanceof TeaException) {
                    TeaException teaException = ((TeaException)e.getCause());
                    if (teaException.getData() != null && "ServiceUnavailable".equals(
                        teaException.getData().get("Code"))) {
                        continue;
                    }
                }
                if (e.getCause() instanceof TeaUnretryableException) {
                    continue;
                }
            }
        }
        return lastResponse;
    }
    private static InitFaceVerifyResponse initFaceVerify(String endpoint, InitFaceVerifyReq
uest request)
        throws Exception {
        Config config = new Config();
        config.setAccessKeyId("您的AccessKey ID");
        config.setAccessKeySecret("您的AccessKey Secret");
        config.setEndpoint(endpoint);
        // 设置http代理。
        //config.setHttpProxy("http://xx.xx.xx.xx:xxxx");
```

```
        // 设置https代理。
        //config.setHttpsProxy("https://xx.xx.xx.xx:xxxx");
        Client client = new Client(config);
        // 创建RuntimeObject实例并设置运行参数。
        RuntimeOptions runtime = new RuntimeOptions();
        runtime.readTimeout = 10000;
        runtime.connectTimeout = 10000;
        return client.initFaceVerify(request, runtime);
    }
}
```

DescribeFaceVerify接口调用示例

```
import java.util.Arrays;
import java.util.List;
import com.aliyun.cloudauth20190307.Client;
import com.aliyun.cloudauth20190307.models.DescribeFaceVerifyRequest;
import com.aliyun.cloudauth20190307.models.DescribeFaceVerifyResponse;
import com.aliyun.tea.TeaException;
import com.aliyun.tea.TeaUnretryableException;
import com.aliyun.tearpc.models.Config;
import com.aliyun.teautil.models.RuntimeOptions;
public class DescribeFaceVerify {
    public static void main(String[] args) throws Exception {
        // 通过以下代码创建API请求并设置参数。
        DescribeFaceVerifyRequest request = new DescribeFaceVerifyRequest();
        // 请输入场景ID+L。
        request.setSceneId(xxxxL);
        request.setCertifyId("xxxx");
        // 推荐，支持服务路由。
        DescribeFaceVerifyResponse response = describeFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        //DescribeFaceVerifyResponse response = describeFaceVerify("cloudauth.cn-shanghai.a
liyuncs.com", request);
        System.out.println(response.getRequestId());
        System.out.println(response.getCode());
        System.out.println(response.getMessage());
        System.out.println(
            response.getResultObject() == null ? null : response.getResultObject().getPasse
d());
        System.out.println(
            response.getResultObject() == null ? null : response.getResultObject().getSubCo
de());
        System.out.println(
            response.getResultObject() == null ? null
                : response.getResultObject().getIdentityInfo());
        System.out.println(
            response.getResultObject() == null ? null
                : response.getResultObject().getDeviceToken());
        System.out.println(
            response.getResultObject() == null ? null
                : response.getResultObject().getMaterialInfo());
    }
    private static DescribeFaceVerifyResponse describeFaceVerifyAutoRoute(DescribeFaceVerif
```

```
yRequest request) {
        // 第一个为主区域Endpoint，第二个为备区域Endpoint。
        List<String> endpoints = Arrays.asList("cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com");
        DescribeFaceVerifyResponse lastResponse = null;
        for (String endpoint : endpoints) {
            try {
                DescribeFaceVerifyResponse response = describeFaceVerify(endpoint, request)
;
                lastResponse = response;
                // 服务端错误，切换到下个区域调用。
                if (response != null && "500".equals(response.getCode())) {
                    continue;
                }
                return response;
            } catch (Exception e) {
                // 网络异常，切换到下个区域调用。
                if (e.getCause() instanceof TeaException) {
                    TeaException teaException = ((TeaException)e.getCause());
                    if (teaException.getData() != null && "ServiceUnavailable".equals(
                        teaException.getData().get("Code"))) {
                        continue;
                    }
                }
                if (e.getCause() instanceof TeaUnretryableException) {
                    continue;
                }
            }
        }
        return lastResponse;
    }
    private static DescribeFaceVerifyResponse describeFaceVerify(String endpoint, DescribeF
aceVerifyRequest request)
        throws Exception {
        Config config = new Config();
        config.setAccessKeyId("您的AccessKey ID");
        config.setAccessKeySecret("您的AccessKey Secret");
        config.setEndpoint(endpoint);
        // 设置http代理。
        //config.setHttpProxy("http://xx.xx.xx.xx:xxxx");
        // 设置https代理。
        //config.setHttpsProxy("http://xx.xx.xx.xx:xxxx");
        Client client = new Client(config);
        // 创建RuntimeObject实例并设置运行参数。
        RuntimeOptions runtime = new RuntimeOptions();
        runtime.readTimeout = 10000;
        runtime.connectTimeout = 10000;
        return client.describeFaceVerify(request, runtime);
    }
}
```

1.1.2.4. PHP SDK调用示例

本文介绍了金融级实人认证方案的PHP SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的PHP版本在5.6版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于PHP SDK源代码，请参见PHP SDK包。

- 通过Composer来管理项目依赖（推荐）

```
composer require alibabacloud/cloudauth-20190307 2.0.3
```

- 使用composer.json文件

```
{
   "require": {
       "alibabacloud/cloudauth-20190307": "2.0.3"
    }
}
```

如果因网络问题导致无法按照上述操作安装SDK，您可以使用阿里云Composer全量镜像。

调用示例

InitFaceVerify接口调用示例

```
<?php
namespace AlibabaCloud\SDK\Sample {
use AlibabaCloud\SDK\Cloudauth\V20190307\Cloudauth;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\InitFaceVerifyRequest;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\InitFaceVerifyResponse;
use AlibabaCloud\Tea\Utils\Utils;
use \Exception;
use AlibabaCloud\Tea\Exception\TeaError;
use AlibabaCloud\Tea\Rpc\Rpc\Config;
use AlibabaCloud\Tea\Utils\Utils\RuntimeOptions;
class InitFaceVerify {
    /**
     * @param string $endpoint
     * @return Cloudauth
     */
    public static function createClient($endpoint){
        $config = new Config([
```

```
            // 您的AccessKey ID。
            "accessKeyId" => "<您的AccessKey ID>",
            // 您的AccessKey Secret。
            "accessKeySecret" => "<您的AccessKey Secret>",
            // 设置HTTP代理。
            // "httpProxy" => "<http://xx.xx.xx.xx:xxxx>",
            // 设置HTTPS代理
            // "httpsProxy" => "<https://xx.xx.xx.xx:xxxx>",
            "endpoint" => $endpoint
        ]);
        return new Cloudauth($config);
    }
    /**
     * @return void
     */
    public static function main(){
        $request = new InitFaceVerifyRequest([
            // 请输入场景ID。
            "sceneId" => <xxx>,
            "outerOrderNo" => "<商户请求的唯一标识>",
            // 要接入的认证方案。
            "productCode" => "<ID_PRO>",
            // 要进行活体检测的类型。
            "model" => "<LIVENESS>",
            "certType" => "<IDENTITY_CARD>",
            "certName" => "<xxx>",
            "certNo" => "<xxx>",
            // MetaInfo环境参数，需要通过客户端SDK获取。
            "metaInfo" => '{"zimVer":"3.0.0","appVersion": "1","bioMetaInfo": "4.1.0:115015
68,0",
        "appName": "com.aliyun.antcloudauth","deviceType": "ios","osVersion": "iOS 10.3.2",
"apdidToken": "","deviceModel": "iPhone9,1"}',
            // "mobile" => "<xxx>",
            // "ip" => "<114.xxx.xxx.xxx>",
            // "userId" => "<12345xxxx>",
            // "callbackUrl" => "<https://www.aliyundoc.com>",
            // "callbackToken" => "<xxxxxxx>",
            // 如需开启个人信息加密传输。
            // "encryptType" => "SM2",
            // "certName" => "BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==",
            // "certNo" => "BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg=="
        ]);
        // 推荐，支持服务路由。
        $response = self::initFaceVerifyAutoRoute($request);
        // 不支持服务自动路由。
        // $response = self::initFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", $request)
;
        var_dump($response->requestId);
        var_dump($response->code);
        var_dump($response->message);
        var_dump($response->resultObject->certifyId);
    }
    /**
     * @param InitFaceVerifyRequest $request
```

```
     * @return InitFaceVerifyResponse
     */
    public static function initFaceVerifyAutoRoute(InitFaceVerifyRequest $request){
        $endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        foreach($endpoints as $endpoint){
            try {
                $response = self::initFaceVerify($endpoint, $request);
                if (Utils::equalString("500", $response->code)) {
                    continue;
                }
                return $response;
            }
            catch (Exception $err) {
                if (!($err instanceof TeaError)) {
                    $err = new TeaError([], $err->getMessage(), $err->getCode(), $err);
                }
                // 网络异常，切换到下个区域调用。
                if (Utils::equalString("ServiceUnavailable", $err->code)) {
                    continue;
                }
            }
        }
        return null;
    }
    /**
     * @param string $endpoint
     * @param InitFaceVerifyRequest $request
     * @return InitFaceVerifyResponse
     */
    public static function initFaceVerify($endpoint, $request){
        $client = self::createClient($endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        $runtime = new RuntimeOptions([]);
        $runtime->readTimeout = 10000;
        $runtime->connectTimeout = 10000;
        return $client->initFaceVerify($request, $runtime);
    }
}
$path = __DIR__ . \DIRECTORY_SEPARATOR . \DIRECTORY_SEPARATOR . 'vendor' . \DIRECTORY_SEPAR
ATOR . 'autoload.php';
if (file_exists($path)) {
    require_once $path;
}
InitFaceVerify::main();
}
```

DescribeFaceVerify接口调用示例

```
<?php
namespace AlibabaCloud\SDK\Sample {
use AlibabaCloud\SDK\Cloudauth\V20190307\Cloudauth;
```

```
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\DescribeFaceVerifyRequest;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\DescribeFaceVerifyResponse;
use AlibabaCloud\Tea\Utils\Utils;
use \Exception;
use AlibabaCloud\Tea\Exception\TeaError;
use AlibabaCloud\Tea\Rpc\Rpc\Config;
use AlibabaCloud\Tea\Utils\Utils\RuntimeOptions;
class DescribeFaceVerify {
    /**
     * @param string $endpoint
     * @return Cloudauth
     */
    public static function createClient($endpoint){
        $config = new Config([
            // 您的AccessKey ID。
            "accessKeyId" => "<您的AccessKey ID>",
            // 您的AccessKey Secret。
            "accessKeySecret" => "<您的AccessKey Secret>",
            // 设置HTTP代理。
            // "httpProxy" => "<http://xx.xx.xx.xx:xxxx>",
            // 设置HTTPS代理。
            // "httpsProxy" => "<https://xx.xx.xx.xx:xxxx>",
            "endpoint" => $endpoint
        ]);
        return new Cloudauth($config);
    }
    /**
     * @return void
     */
    public static function main(){
        $request = new DescribeFaceVerifyRequest();
        // 场景ID。
        $request->sceneId=<xxx>;
        // 认证ID。
        $request->certifyId="<xxx>";
        // 推荐，支持服务路由。
        $response = self::describeFaceVerifyAutoRoute($request);
        // 不支持服务自动路由。
        // $response = self::describeFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", $requ
est);
        var_dump($response->requestId);
        var_dump($response->code);
        var_dump($response->message);
        var_dump($response->resultObject->passed);
        var_dump($response->resultObject->subCode);
        var_dump($response->resultObject->materialInfo);
        var_dump($response->resultObject->identityInfo);
        var_dump($response->resultObject->deviceToken);
    }
    /**
     * @param DescribeFaceVerifyRequest $request
     * @return DescribeFaceVerifyResponse
     */
    public static function describeFaceVerifyAutoRoute($request){
```

```
        $endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        foreach($endpoints as $endpoint){
            try {
                $response = self::describeFaceVerify($endpoint, $request);
                if (Utils::equalString("500", $response->code)) {
                    continue;
                }
                return $response;
            }
            catch (Exception $err) {
                if (!($err instanceof TeaError)) {
                    $err = new TeaError([], $err->getMessage(), $err->getCode(), $err);
                }
                // 网络异常，切换到下个区域调用。
                if (Utils::equalString("ServiceUnavailable", $err->code)) {
                    continue;
                }
            }
        }
        return null;
    }
    /**
     * @param string $endpoint
     * @param DescribeFaceVerifyRequest $request
     * @return DescribeFaceVerifyResponse
     */
    public static function describeFaceVerify($endpoint, $request){
        $client = self::createClient($endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        $runtime = new RuntimeOptions([]);
        $runtime->readTimeout = 10000;
        $runtime->connectTimeout = 10000;
        return $client->describeFaceVerify($request, $runtime);
    }
}
$path = __DIR__ . \DIRECTORY_SEPARATOR . \DIRECTORY_SEPARATOR . 'vendor' . \DIRECTORY_SEPAR
ATOR . 'autoload.php';
if (file_exists($path)) {
    require_once $path;
}
DescribeFaceVerify::main();
}
```

1.1.2.5. Python SDK调用示例

本文介绍了金融级实人认证方案的Python SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的Python版本在3.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

引入依赖包

```
pip install alibabacloud_tea_openapi==0.2.5
```

安装SDK

关于Python SDK源代码，请参见Python SDK包。

```
pip install alibabacloud_cloudauth20190307==2.0.3
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
from alibabacloud_cloudauth20190307.client import Client as CloudauthClient
from alibabacloud_tea_openapi import models as open_api_models
from alibabacloud_cloudauth20190307 import models as cloudauth_models
from alibabacloud_tea_util.client import Client as UtilClient
from alibabacloud_tea_util import models as util_models
class InitFaceVerify:
    def __init__(self):
        pass
    @staticmethod
    def create_client(
            endpoint: str,
    ) -> CloudauthClient:
        config = open_api_models.Config(
            # 您的AccessKey ID。
            access_key_id='您的AccessKey ID',
            # 您的AccessKey Secret。
            access_key_secret='您的AccessKey Secret',
            # 设置HTTP代理。
            # http_proxy='http://xx.xx.xx.xx:xxxx',
            # 设置HTTPS代理。
            # https_proxy='https://username:password@xxx.xxx.xxx.xxx:9999',
            endpoint=endpoint
        )
        return CloudauthClient(config)
    @staticmethod
    def main() -> None:
        request = cloudauth_models.InitFaceVerifyRequest(
```

```
            # 请输入场景ID。
            scene_id=xxx,
            outer_order_no='商户请求的唯一标识',
            # 要接入的认证方案。
            product_code='ID_PRO',
            model='LIVENESS',
            # 固定值IDENTITY_CARD。
            cert_type='IDENTITY_CARD',
            cert_name='xxx',
            cert_no='xxx',
            # MetaInfo环境参数，需要通过客户端SDK获取。
            meta_info='{"zimVer":"3.0.0","appVersion": "1","bioMetaInfo": "4.1.0:11501568,0
","appName": "com.aliyun.antcloudauth","deviceType": "ios","osVersion": "iOS 10.3.2","apdid
Token": "","deviceModel": "iPhone9,1"}',
            # mobile='130xxxxxxxx',
            # ip='114.xxx.xxx.xxx',
            # user_id='12345xxxx',
            # callback_url='https://www.aliyundoc.com',
            # callback_token='xxxxxxx',
            # 如需开启个人信息加密传输。
            # encrypt_type='SM2',
            # cert_name='BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==',
            # cert_no='BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg=='
        )
        response = InitFaceVerify.init_face_verify_auto_route(request)
        # 不支持服务自动路由。
        # response = InitFaceVerify.init_face_verify('cloudauth.cn-shanghai.aliyuncs.com',
request)
        print(response.request_id)
        print(response.code)
        print(response.message)
        print(response.result_object.certify_id)
    @staticmethod
    def init_face_verify_auto_route(
            request: cloudauth_models.InitFaceVerifyRequest,
    ) -> cloudauth_models.InitFaceVerifyResponse:
        endpoints = [
            'cloudauth.cn-shanghai.aliyuncs.com',
            'cloudauth.cn-beijing.aliyuncs.com'
        ]
        for endpoint in endpoints:
            try:
                response = InitFaceVerify.init_face_verify(endpoint, request)
                if UtilClient.equal_string('500', response.code):
                    continue
                return response
            except Exception as err:
                # 网络异常，切换到下个区域调用。
                continue
        return None
    @staticmethod
    def init_face_verify(
            endpoint: str,
            request: cloudauth_models.InitFaceVerifyRequest,
```

```
    ) -> cloudauth_models.InitFaceVerifyResponse:
        client = InitFaceVerify.create_client(endpoint)
        # 创建RuntimeObject实例并设置运行参数。
        runtime = util_models.RuntimeOptions()
        runtime.read_timeout = 10000
        runtime.connect_timeout = 10000
        return client.init_face_verify(request, runtime)
if __name__ == '__main__':
    InitFaceVerify.main()
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
from alibabacloud_cloudauth20190307.client import Client as CloudauthClient
from alibabacloud_tea_openapi import models as open_api_models
from alibabacloud_cloudauth20190307 import models as cloudauth_models
from alibabacloud_tea_util.client import Client as UtilClient
from alibabacloud_tea_util import models as util_models
class DescribeFaceVerify:
    def __init__(self):
        pass
    @staticmethod
    def create_client(
        endpoint: str,
    ) -> CloudauthClient:
        config = open_api_models.Config(
            # 您的AccessKey ID。
            access_key_id='您的AccessKey ID',
            # 您的AccessKey Secret。
            access_key_secret='您的AccessKey Secret',
            # 设置HTTP代理。
            # http_proxy='http://xx.xx.xx.xx:xxxx',
            # 设置HTTPS代理。
            # https_proxy='https://xx.xx.xx.xx:xxxx',
            endpoint=endpoint
        )
        return CloudauthClient(config)
    @staticmethod
    def main() -> None:
        request = cloudauth_models.DescribeFaceVerifyRequest(
            # 请输入场景ID。
            scene_id=xxx,
            certify_id='xxx'
        )
        response = DescribeFaceVerify.describe_face_verify_auto_route(request)
        # 不支持服务自动路由。
        # response = DescribeFaceVerify.describe_face_verify('cloudauth.cn-shanghai.aliyunc
s.com', request)
        print(response.request_id)
        print(response.code)
        print(response.message)
        print(response.result_object.passed)
        print(response.result_object.sub_code)
        print(response.result_object.material_info)
```

```
        print(response.result_object.material_info)
    @staticmethod
    def describe_face_verify_auto_route(
        request: cloudauth_models.DescribeFaceVerifyRequest,
    ) -> cloudauth_models.DescribeFaceVerifyResponse:
        endpoints = [
            'cloudauth.cn-shanghai.aliyuncs.com',
            'cloudauth.cn-beijing.aliyuncs.com'
        ]
        for endpoint in endpoints:
            try:
                response = DescribeFaceVerify.describe_face_verify(endpoint, request)
                if UtilClient.equal_string('500', response.code):
                    continue
                return response
            except Exception as err:
                    continue
        return None
    @staticmethod
    def describe_face_verify(
        endpoint: str,
        request: cloudauth_models.DescribeFaceVerifyRequest,
    ) -> cloudauth_models.DescribeFaceVerifyResponse:
        client = DescribeFaceVerify.create_client(endpoint)
        # 创建RuntimeObject实例并设置运行参数。
        runtime = util_models.RuntimeOptions()
        runtime.read_timeout = 10000
        runtime.connect_timeout = 10000
        return client.describe_face_verify(request, runtime)
if __name__ == '__main__':
    DescribeFaceVerify.main()
```

1.1.2.6. Go SDK调用示例

本文介绍了金融级实人认证方案的Go SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 仅支持Go 1.12.0及以上版本。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

引入依赖包

```
go get github.com/alibabacloud-go/tea-rpc
go get github.com/alibabacloud-go/tea-utils
go get github.com/alibabacloud-go/tea
```

安装SDK

关于Go SDK源代码，请参见Go SDK包。

通过Go mod来管理项目依赖：

```
go get github.com/alibabacloud-go/cloudauth-20190307@v2.0.3
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
package main
import (
  "fmt"
  cloudauth "github.com/alibabacloud-go/cloudauth-20190307/client"
  rpc "github.com/alibabacloud-go/tea-rpc/client"
  util "github.com/alibabacloud-go/tea-utils/service"
  "github.com/alibabacloud-go/tea/tea"
)
func CreateClient (endpoint *string) (_result *cloudauth.Client, _err error) {
  config := &rpc.Config{
    // 您的AccessKey ID。
    AccessKeyId: tea.String("您的AccessKey ID"),
    // 您的AccessKey Secret。
    AccessKeySecret: tea.String("您的AccessKey Secret"),
    // 设置HTTP代理。
    // HttpProxy: tea.String("http://xx.xx.xx.xx:xxxx"),
    // 设置HTTPS代理。
    // HttpsProxy: tea.String("https://username:password@xxx.xxx.xxx.xxx:9999"),
    Endpoint: endpoint,
  }
  _result = &cloudauth.Client{}
  _result, _err = cloudauth.NewClient(config)
  return _result, _err
}
func _main () (_err error) {
  request := &cloudauth.InitFaceVerifyRequest{
    // 固定值。
    ProductCode: tea.String("ID_PRO"),
    // 请输入场景ID。
    SceneId: tea.Int64(xxx),
    OuterOrderNo: tea.String("商户请求的唯一标识"),
    Model: tea.String("LIVENESS"),
    // 固定值IDENTITY_CARD。
    CertType: tea.String("IDENTITY_CARD"),
    CertName: tea.String("xxx"),
    CertNo: tea.String("xxx"),
```

```
    MetaInfo: tea.String("{\"zimVer\":\"3.0.0\",\"appVersion\": \"1\",\"bioMetaInfo\": \"4.
1.0:11501568,0\",\"appName\": \"com.aliyun.antcloudauth\",\"deviceType\": \"ios\",\"osVersi
on\": \"iOS 10.3.2\",\"apdidToken\": \"\",\"deviceModel\": \"iPhone9,1\"}"),
    // Ip: tea.String("用户IP地址"),
    // UserId: tea.String("用户ID"),
    // Mobile: tea.String("130xxxxxxxx"),
    // CallbackUrl: tea.String("https://www.aliyundoc.com"),
    // CallbackToken: tea.String("xxxxxxx"),
    // 如需开启个人信息加密传输。
    // EncryptType: tea.String("SM2"),
    // CertName: tea.String("BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w=="),
    // CertNo: tea.String("BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg==")
  }
  response, _err := InitFaceVerifyAutoRoute(request)
  // 不支持服务自动路由。
  // response, _err := InitFaceVerify(tea.String("cloudauth.cn-shanghai.aliyuncs.com"), req
uest)
  if _err != nil {
    return _err
  }
  fmt.Println(*response.RequestId)
  fmt.Println(*response.Code)
  fmt.Println(*response.Message)
  fmt.Println(*response.RequestId)
  fmt.Println(*response.ResultObject.CertifyId)
  return _err
}
func InitFaceVerifyAutoRoute (request *cloudauth.InitFaceVerifyRequest) (_result *cloudauth
.InitFaceVerifyResponse, _err error) {
  endpoints := []*string{tea.String("cloudauth.cn-shanghai.aliyuncs.com"), tea.String("clou
dauth.cn-beijing.aliyuncs.com")}
  var lastResponse *cloudauth.InitFaceVerifyResponse
  for _, endpoint := range endpoints {
    response, _err := InitFaceVerify(endpoint, request)
    if _err != nil {
      var err = &tea.SDKError{}
      if _t, ok := _err.(*tea.SDKError); ok {
        err = _t
        // 网络异常，切换到下个区域调用。
        if tea.BoolValue(util.EqualString(tea.String("ServiceUnavailable"), err.Code)) {
          continue
        }
      }
      return _result, _err
    }
    if tea.BoolValue(util.EqualString(tea.String("500"), response.Code)) {
      continue
    }
    _result = response
    return _result, _err
  }
  _result = lastResponse
  return _result, _err
}
```

```
func InitFaceVerify (endpoint *string, request *cloudauth.InitFaceVerifyRequest) (_result *
cloudauth.InitFaceVerifyResponse, _err error) {
  client, _err := CreateClient(endpoint)
  if _err != nil {
    return _result, _err
  }
  // 创建RuntimeObject实例并设置运行参数。
  runtime := &util.RuntimeOptions{}
  runtime.ReadTimeout = tea.Int(10000)
  runtime.ConnectTimeout = tea.Int(10000)
  _result = &cloudauth.InitFaceVerifyResponse{}
  _body, _err := client.InitFaceVerify(request, runtime)
  if _err != nil {
    return _result, _err
  }
  _result = _body
  return _result, _err
}
func main() {
  err := _main()
  if err != nil {
    panic(err)
  }
}
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
package main
import (
  "fmt"
  cloudauth "github.com/alibabacloud-go/cloudauth-20190307/client"
  rpc "github.com/alibabacloud-go/tea-rpc/client"
  util "github.com/alibabacloud-go/tea-utils/service"
  "github.com/alibabacloud-go/tea/tea"
)
func CreateClient(endpoint *string) (_result *cloudauth.Client, _err error) {
  config := &rpc.Config{
    // 您的AccessKey ID。
    AccessKeyId: tea.String("您的AccessKey ID"),
    // 您的AccessKey Secret。
    AccessKeySecret: tea.String("您的AccessKey Secret"),
    Endpoint:        endpoint,
    // 设置HTTP代理。
    // HttpProxy: tea.String("http://xx.xx.xx.xx:xxxx"),
    // 设置HTTPS代理。
    // HttpsProxy: tea.String("https://xx.xx.xx.xx:xxxx"),
  }
  _result = &cloudauth.Client{}
  _result, _err = cloudauth.NewClient(config)
  return _result, _err
}
func _main() (_err error) {
  request := &cloudauth.DescribeFaceVerifyRequest{
```

```
  request := &cloudauth.DescribeFaceVerifyRequest{
    // 请输入场景ID。
    SceneId:   tea.Int64(xxx),
    CertifyId: tea.String("91707dc296d469ad38e4c5efa6a0xxxx"),
  }
  response, _err := DescribeFaceVerifyAutoRoute(request)
  // 不支持服务自动路由。
  // response, _err := DescribeFaceVerify(tea.String("cloudauth.cn-shanghai.aliyuncs.com"),
request)
  if _err != nil {
    return _err
  }
  if response != nil {
    fmt.Println(*response.RequestId)
    fmt.Println(*response.Code)
    fmt.Println(*response.Message)
    fmt.Println(*response.RequestId)
    fmt.Println(*response.ResultObject.Passed)
    fmt.Println(*response.ResultObject.SubCode)
    fmt.Println(*response.ResultObject.MaterialInfo)
    fmt.Println(*response.ResultObject.IdentityInfo)
    fmt.Println(*response.ResultObject.DeviceToken)
  }
  return _err
}
func DescribeFaceVerifyAutoRoute(request *cloudauth.DescribeFaceVerifyRequest) (_result *cl
oudauth.DescribeFaceVerifyResponse, _err error) {
  endpoints := []*string{tea.String("cloudauth.cn-shanghai.aliyuncs.com"), tea.String("clou
dauth.cn-beijing.aliyuncs.com")}
  var lastResponse *cloudauth.DescribeFaceVerifyResponse
  for _, endpoint := range endpoints {
    response, _err := DescribeFaceVerify(endpoint, request)
    if _err != nil {
      var err = &tea.SDKError{}
      if _t, ok := _err.(*tea.SDKError); ok {
        err = _t
        // 网络异常，切换到下个区域调用。
        if tea.BoolValue(util.EqualString(tea.String("ServiceUnavailable"), err.Code)) {
          continue
        }
      }
      return _result, _err
    }
    if tea.BoolValue(util.EqualString(tea.String("500"), response.Code)) {
      continue
    }
    _result = response
    return _result, _err
  }
  _result = lastResponse
  return _result, _err
}
func DescribeFaceVerify(endpoint *string, request *cloudauth.DescribeFaceVerifyRequest) (_r
esult *cloudauth.DescribeFaceVerifyResponse, _err error) {
  client, _err := CreateClient(endpoint)
```

```
  if _err != nil {
    return _result, _err
  }
  // 创建RuntimeObject实例并设置运行参数。
  runtime := &util.RuntimeOptions{}
  runtime.ReadTimeout = tea.Int(10000)
  runtime.ConnectTimeout = tea.Int(10000)
  _body, _err := client.DescribeFaceVerify(request, runtime)
  if _err != nil {
    return _result, _err
  }
  _result = _body
  return _result, _err
}
func main() {
  err := _main()
  if err != nil {
    panic(err)
  }
}
```

1.1.2.7. .NET SDK调用示例

本文介绍了金融级实人认证方案的.NET SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的.NET Framework版本在4.5.0版本及以上或者.NET Standard版本在2.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于.NET SDK源代码，请参见.NET SDK包。

SDK有以下两种安装方式，请选择合适的方式安装：

- 通过.NET命令行安装。

```
dotnet add package AlibabaCloud.SDK.Cloudauth20190307 --version 2.0.3
```

- 使用包管理安装。

```
Install-Package AlibabaCloud.SDK.Cloudauth20190307 -Version 2.0.3
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
using System;
using System.Collections.Generic;
using Tea;
namespace InitFaceVerify
{
    public class Program
    {
        public static AlibabaCloud.SDK.Cloudauth20190307.Client CreateClient(string endpoin
t)
        {
            AlibabaCloud.RPCClient.Models.Config config = new AlibabaCloud.RPCClient.Models
.Config
            {
                // 您的AccessKey ID。
                AccessKeyId = "您的AccessKey ID",
                // 您的AccessKey Secret。
                AccessKeySecret = "您的AccessKey Secret",
                // 设置HTTP代理。
                // HttpProxy = "http://xx.xx.xx.xx:xxxx",
                // 设置HTTPS代理。
                // HttpsProxy = "https://username:password@xxx.xxx.xxx.xxx:9999",
                Endpoint = endpoint,
            };
            return new AlibabaCloud.SDK.Cloudauth20190307.Client(config);
        }
        public static void Main(string[] args)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest request = new A
libabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest
            {
                // 固定值。
                ProductCode = "ID_PRO",
                // 请输入场景ID。
                SceneId = xxx,
                // 设置商户请求的唯一标识。
                OuterOrderNo = "xxxxx",
                // 模式。
                Model = "LIVENESS",
                // 固定值IDENTITY_CARD。
                CertType = "IDENTITY_CARD",
                CertName = "xxx",
                CertNo = "xxx",
                MetaInfo = "{\"zimVer\":\"3.0.0\",\"appVersion\": \"1\",\"bioMetaInfo\": \"
4.1.0:11501568,0\",\"appName\": \"com.aliyun.antcloudauth\",\"deviceType\": \"ios\",\"osVer
sion\": \"iOS 10.3.2\",\"apdidToken\": \"\",\"deviceModel\": \"iPhone9,1\"}",
                //Mobile = "130xxxxxxxx",
                //Ip = "114.xxx.xxx.xxx",
                //UserId = "用户ID",
                //CallbackUrl = "https://www.aliyundoc.com",
                //CallbackToken = "xxxxxxx",
                // 如需开启个人信息加密传输。
```

```
                // EncryptType = "SM2",
                // CertName = "BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==",
                // CertNo = "BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg=="
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse response = Ini
tFaceVerifyAutoRoute(request);
            // 不支持服务自动路由。
            // AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse response =
InitFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", request);
            Console.WriteLine(response.RequestId);
            Console.WriteLine(response.Code);
            Console.WriteLine(response.Message);
            Console.WriteLine(response.ResultObject.CertifyId);
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse Init
FaceVerifyAutoRoute(AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest request
)
        {
            List<string> endpoints = new List<string>
            {
                "cloudauth.cn-shanghai.aliyuncs.com",
                "cloudauth.cn-beijing.aliyuncs.com"
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse lastResponse =
null;
            foreach (var endpoint in endpoints)
            {
                try
                {
                    AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse respon
se = InitFaceVerify(endpoint, request);
                    if (AlibabaCloud.TeaUtil.Common.EqualString("500", response.Code))
                    {
                        continue;
                    }
                    return response;
                }
                catch (TeaException err)
                {
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
                catch (Exception _err)
                {
                    TeaException err = new TeaException(new Dictionary<string, object>
                    {
                        { "message", _err.Message }
                    });
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
```

```
ode))
                    {
                        continue;
                    }
                }
            }
            return lastResponse;
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse Init
FaceVerify(string endpoint, AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest
request)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Client client = CreateClient(endpoint);
            // 创建RuntimeObject实例并设置运行参数。
            AlibabaCloud.TeaUtil.Models.RuntimeOptions runtime = new AlibabaCloud.TeaUtil.M
odels.RuntimeOptions();
            runtime.ReadTimeout = 10000;
            runtime.ConnectTimeout = 10000;
            return client.InitFaceVerify(request, runtime);
        }
    }
}
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
using System;
using System.Collections.Generic;
using Tea;
namespace DescribeFaceVerify
{
    public class Program
    {
        public static AlibabaCloud.SDK.Cloudauth20190307.Client CreateClient(string endpoin
t)
        {
            AlibabaCloud.RPCClient.Models.Config config = new AlibabaCloud.RPCClient.Models
.Config
            {
                // 您的AccessKey ID。
                AccessKeyId = "您的AccessKey ID",
                // 您的AccessKey Secret。
                AccessKeySecret = "您的AccessKey Secret",
                // 设置HTTP代理。
                // HttpProxy = "http://xx.xx.xx.xx:xxxx",
                // 设置HTTPS代理。
                // HttpsProxy = "https://xx.xx.xx.xx:xxxx",
                Endpoint = endpoint,
            };
            return new AlibabaCloud.SDK.Cloudauth20190307.Client(config);
        }
        public static void Main(string[] args)
        {
```

```
            AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyRequest request = n
ew AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyRequest
            {
                // 请输入场景ID。
                SceneId = xxx,
                CertifyId = "91707dc296d469ad38e4c5efa6a0xxxx",
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse response =
DescribeFaceVerifyAutoRoute(request);
            // 不支持服务自动路由。
            // AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse respons
e = DescribeFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", request);
            Console.WriteLine(response.RequestId);
            Console.WriteLine(response.Code);
            Console.WriteLine(response.Message);
            Console.WriteLine(response.ResultObject.Passed);
            Console.WriteLine(response.ResultObject.SubCode);
            Console.WriteLine(response.ResultObject.MaterialInfo);
            Console.WriteLine(response.ResultObject.IdentityInfo);
            Console.WriteLine(response.ResultObject.DeviceToken);
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse
DescribeFaceVerifyAutoRoute(AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyReq
uest request)
        {
            List<string> endpoints = new List<string>
            {
                "cloudauth.cn-shanghai.aliyuncs.com",
                "cloudauth.cn-beijing.aliyuncs.com"
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse lastRespon
se = null;
            foreach (var endpoint in endpoints)
            {
                try
                {
                    AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse re
sponse = DescribeFaceVerify(endpoint, request);
                    if (AlibabaCloud.TeaUtil.Common.EqualString("500", response.Code))
                    {
                        continue;
                    }
                    return response;
                }
                catch (TeaException err)
                {
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
                catch (Exception _err)
```

```
                {
                    TeaException err = new TeaException(new Dictionary<string, object>
                    {
                        { "message", _err.Message }
                    });
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
            }
            return lastResponse;
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse
DescribeFaceVerify(string endpoint, AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceV
erifyRequest request)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Client client = CreateClient(endpoint);
            // 创建RuntimeObject实例并设置运行参数。
            AlibabaCloud.TeaUtil.Models.RuntimeOptions runtime = new AlibabaCloud.TeaUtil.M
odels.RuntimeOptions();
            runtime.ReadTimeout = 10000;
            runtime.ConnectTimeout = 10000;
            return client.DescribeFaceVerify(request, runtime);
        }
    }
}
```

1.1.2.8. Node.js SDK调用示例

本文介绍了金融级实人认证方案的Node.js SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的Node.js版本在8.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于Node.js SDK源代码，请参见Node.js SDK。

建议您使用npm管理依赖包：

```
npm install @alicloud/tea-util
npm install @alicloud/rpc-client
npm install @alicloud/cloudauth20190307@2.0.3
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
const Config = require("@alicloud/rpc-client");
const CloudAuth = require("@alicloud/cloudauth20190307");
const CloudAuthClient = CloudAuth.default;
const {RuntimeOptions} = require("@alicloud/tea-util");
     function createClient(endpoint) {
        // 通过以下代码创建Config实例并初始化。
        const config = new Config.Config({
            // 您的AccessKey ID。
            accessKeyId: "您的AccessKey ID",
            // 您的AccessKey Secret。
            accessKeySecret: "您的AccessKey Secret",
            // 设置Endpoint。
            endpoint: endpoint,
            // 设置HTTP代理。
            // httpProxy: "http://xxx.xxx.xxx.xxx:9999",
            // 设置HTTPS代理。
            // httpsProxy: "https://username:password@xxx.xxx.xxx.xxx:9999",
        });
        return new CloudAuthClient(config)
    }
    async function initFaceVerify(endpoint, request) {
        const client = createClient(endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        const runtime = new RuntimeOptions([]);
        runtime.readTimeout=10000;
        runtime.connectTimeout=10000;
        return await client.initFaceVerify(request, runtime);
    }
    async function initFaceVerifyAutoRoute(request) {
        const endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        for (const endpoint of endpoints) {
            try {
                const response = await initFaceVerify(endpoint, request);
                if (response.code === '500') {
                    continue;
                }
                return response;
            } catch (err) {
                continue;
            }
        }
```

```
        return null;
    }
    async function main(){
        // 通过以下代码创建API请求并设置参数。
        const request = new CloudAuth.InitFaceVerifyRequest({
        // 请输入场景ID。
        sceneId: xxx,
        // 设置商户请求的唯一标识。
        outerOrderNo: "xxx",
        // 要接入的认证方案。
        productCode: "ID_PRO",
        // 模式。
        model: "LIVENESS",
        // 固定值。
        certType: "IDENTITY_CARD",
        certName: "张三",
        certNo: "xxx",
        metaInfo: "{\"zimVer\":\"3.0.0\",\"appVersion\": \"1\",\"bioMetaInfo\": \"4.1.0:115
01568,0\",\n" +
            "        \"appName\": \"com.aliyun.antcloudauth\",\"deviceType\": \"ios\",\"osV
ersion\": \"iOS 10.3.2\",\"apdidToken\": \"\",\"deviceModel\": \"iPhone9,1\"}",
        // mobile: "130xxxxxxxx",
        // ip: "114.xxx.xxx.xxx",
        // userId: "12345xxxx",
        // callbackUrl: "https://www.aliyundoc.com",
        // callbackToken: "xxxxxxx",
        // 如需开启个人信息加密传输。
        // encryptType: "SM2",
        // certName: "BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==",
        // certNo: "BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg=="
        });
        // 推荐，支持服务路由。
        const response = await initFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        // const response = await initFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", requ
est);
        console.log(response.requestId);
        console.log(response.code);
        console.log(response.message);
        console.log(response.resultObject.certifyId);
        }
main().then(function (response) {});
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
const Config = require("@alicloud/rpc-client");
const CloudAuth = require("@alicloud/cloudauth20190307");
const CloudAuthClient = CloudAuth.default;
const {RuntimeOptions} = require("@alicloud/tea-util");
     function createClient(endpoint) {
        // 通过以下代码创建Config实例并初始化。
        const config = new Config.Config({
```

```
            // 您的AccessKey ID。
            accessKeyId: "您的AccessKey ID",
            // 您的AccessKey Secret。
            accessKeySecret: "您的AccessKey Secret",
            // 设置endpoint
            endpoint: endpoint,
            // 设置http代理。
            // httpProxy: "http://xxx.xxx.xxx.xxx:9999",
            // 设置https代理。
            // httpsProxy: "https://username:password@xxx.xxx.xxx.xxx:9999",
        });
        return new CloudAuthClient(config)
    }
    async function describeFaceVerify(endpoint, request) {
        const client = createClient(endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        const runtime = new RuntimeOptions([]);
        runtime.readTimeout=10000;
        runtime.connectTimeout=10000;
        return await client.describeFaceVerify(request, runtime);
    }
    async function describeFaceVerifyAutoRoute(request) {
        const endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        for (const endpoint of endpoints) {
            try {
                const response = await describeFaceVerify(endpoint, request);
                if (response.code === '500') {
                    continue;
                }
                return response;
            } catch (err) {
                continue;
            }
        }
        return null;
    }
    async function main(){
        // 通过以下代码创建API请求并设置参数。
        const request = new CloudAuth.DescribeFaceVerifyRequest({
            sceneId: xxx,
            certifyId: "xxx"
        });
        // 推荐，支持服务路由。
        const response = await describeFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        //  const response = await describeFaceVerify("cloudauth.cn-shanghai.aliyuncs.com",
request);
        console.log(response.requestId);
        console.log(response.code);
        console.log(response.message);
        console.log(response.resultObject.passed);
```

```
        console.log(response.resultObject.identityInfo);
        console.log(response.resultObject.materialInfo);
        console.log(response.resultObject.deviceToken);
        console.log(response.resultObject.subCode);
        }
main().then(function (response) {});
```

1.1.3. App（SDK）接入

1.1.3.1. Android接入

金融级实人认证服务提供Android客户端SDK，帮助您在业务应用（App）中实现刷脸认证功能。您可以通过应用服务端认证初始化接口，向阿里云服务器调用发起认证请求接口并获取刷脸认证的唯一标识CertifyId，然后使用CertifyId唤起刷脸认证客户端SDK。本文结合示例代码介绍Android客户端的接入流程。

使用限制

金融级实人认证服务Android客户端SDK具有以下使用限制。

- 不支持模拟器模式调试。
- 仅支持Android 4.4及以上系统版本的移动智能设备（手机或Pad）接入。

依赖配置

下载金融级实人认证Android SDK，该SDK为Android标准AAR包，下载之后在*Android Studio*中进行AAR包引用。例如下载完AAR包之后，放入工程目录的*libs*目录下，并添加以下依赖：

```
# 添加阿里云实人认证SDK依赖包。
implementation files('libs/android-aliyunface-sdk-release-xxx.aar')
implementation files('libs/android-aliyunbasicstl-sdk-release-xxx.aar')
implementation files('libs/android-aliyuncomm-sdk-release-xxx.aar')
implementation files('libs/android-aliyunocr-sdk-release-xxx.aar')
implementation files('libs/photinus-1.0.1.211130153658.aar')
implementation files('libs/Android-AliyunDevice-xxx.aar')
implementation files('libs/deviceid-release-xxx.aar')
implementation files('libs/tygerservice-xxx.aar')
implementation 'com.squareup.okhttp3:okhttp:3.11.0'
implementation 'com.squareup.okio:okio:1.14.0'
implementation 'com.alibaba:fastjson:1.2.62'
implementation 'com.aliyun.dpa:oss-android-sdk:+'
```

说明 以上代码示例中xxx表示版本号，使用过程中，请替换为SDK包中文件的实际版本号。

接口定义

```
/**
 * 刷脸认证接口。
 */
public class ZIMFacade {
    /**
     * 初始化接口，建议接入方在调用其它接口前，先调用此接口。
     *
     * @context Context
     */
    public static void install(Context context);
    /**
     * 获取环境参数接口。
     *
     * @context Context
     */
    public static String getMetaInfos(Context context);
    /**
     * identify interface for android
     * 需在UI线程上调用，不会阻塞调用线程。
     *
     * @param certifyId：刷脸认证唯一标识，请从刷脸认证服务端认证初始化接口获取。
     * @param useMsgBox：当刷脸认证过程中出现异常情况，是否使用SDK内部的弹框提示。
     *                    如果取值为true，则SDK先弹框提示，确定之后，返回错误code。
     *                    如果取值为false，则不弹提示框，直接返回错误code，客户业务应用App自行决定
如何提示。
     * @param callback：认证结果的回调接口。
     */
    public void verify(String certifyId, boolean useMsgBox,, ZIMCallback callback);
    /**
     * 开始进行实名认证，带自定义参数的版本。
     * @param certifyId：刷脸认证唯一标识，请从刷脸认证服务端认证初始化接口获取。
     * @param useMsgBox：表示是否在SDK内部使用弹框提示错误信息。
     * @param extParams：扩展参数。
     * @param callback：identify result
     */
    public void verify(final String zimId, boolean useMsgBox, HashMap<String, String> extPa
rams, ZIMCallback callback);
    /**
     * 获取设备指纹DeviceToken。
     * @return
     */
    public ZIMSession getSession();
}
```

接口调用

1. 初始化SDK。

   为了提高身份核验的用户体验，并为刷脸认证准备必要的数据，Android客户端需要进行SDK初始化。初始化的代码示例如下：

   ```
   ZIMFacade.install(context);
   ```

2. 获取MetaInfo数据。

您在调用刷脸认证服务端发起认证请求时，需要传入该MetaInfo值。代码示例如下：

```
String metaInfos = ZIMFacade.getMetaInfos(context);
```

3. 开始认证。

   Android客户端接入支持接入手机端和Pad端两种场景，请您根据自身业务需要选择接入的场景。两种接入场景的示例代码如下：

   - 手机端接入场景（自定义参数设置请参考Pad端设置）

```
ZIMFacade zimFacade = ZIMFacadeBuilder.create(MainActivity.this);
zimFacade.verify(certifyId, true, new ZIMCallback() {
 @Override
 public boolean response(ZIMResponse response) {
     // TODO: 根据实人认证回调结果处理自身的业务。
     if (null != response && 1000 == response.code) {
         // 认证成功。
     } else {
         // 认证失败。
     }
     return true;
 }
}
```

   - 手机端+Pad端接入场景（支持自定义参数）

> **说明** 金融级实人认证支持设置返回活体检测的视频，默认设置为不返回。如果设置为返回，在您使用金融级实人认证SDK完成认证（不区分认证结果为F或T），并在SDK回调时和查询接口返回视频路径，请您及时处理。

```
ZIMFacade zimFacade = ZIMFacadeBuilder.create(MainActivity.this);
HashMap<String, String> extParams = new HashMap<>();
// 如需指定活体检测UI界面方向（横屏+竖屏），请指定这一项。
extParams.put(ZIMFacade.ZIM_EXT_PARAMS_KEY_SCREEN_ORIENTATION, ZIMFacade.ZIM_EXT_PARA
MS_VAL_SCREEN_LAND);
// 如需支持活体视频返回，请指定这一项，并在response.videoFilePath或在服务端查询接口获取视频本
地路径。
extParams.put(ZIMFacade.ZIM_EXT_PARAMS_KEY_USE_VIDEO, ZIMFacade.ZIM_EXT_PARAMS_VAL_US
E_VIDEO_TRUE);
// 如需设置OCR的"下一步"按钮颜色（默认可不设置），请设置此项，如红色 #FF0000。
extParams.put(ZIMFacade.ZIM_EXT_PARAMS_KEY_OCR_BOTTOM_BUTTON_COLOR, "#FF0000");
// 如需自定义活体检测页面的进度条颜色（默认可不设置），请设置此项，如红色 #FF0000。
extParams.put(ZIMFacade.ZIM_EXT_PARAMS_KEY_FACE_PROGRESS_COLOR, "#FF0000");
zimFacade.verify(certifyId, true, extParams, new ZIMCallback() {
    @Override
    public boolean response(final ZIMResponse response) {
        if (1000 == response.code) {
            showMessageBox("认证通过");
        } else {
            showMessageBox("认证失败([" + response.code + "]" + response.reason + ")")
;
        }
        return true;
    }
});
```

说明

- certifyId：刷脸认证唯一标识，可从刷脸认证服务端发起认证请求获取。
- ZIMCallback的response回调需要有返回值，默认为true。

4. 获取设备指纹。

   获取设备指纹（deviceToken），然后通过后端接口查询设备风险。代码示例如下：

```
ZIMSession deviceSession = zimFacade.getSession();
 if(deviceSession.code == 0){
            // 获取session成功，可以通过后端接口查询风险标签。
 }else{
            // 获取session失败，不能通过后端接口查询到有效风险标签。
 }
```

5. 返回刷脸认证回调结果。

   刷脸认SDK回调结果中的ZIMResponse类，该类中定义了对应的结果编码和原因，代码示例如下：

```
/**
 * 刷脸认证回调结果。
 */
public class ZIMResponse {
    /**
     * 返回结果编码。
     */
    public int code;
    /**
     * 返回结果原因信息。
     */
    public String reason;
}
```

返回结果说明

错误码	错误码文案	错误码描述
1000	刷脸成功	代表刷脸成功，该结果仅供参考，可通过服务端查询认证结果获取最终认证结果。
1001	系统错误	表示系统错误。
1003	验证中断	表示验证中断。
2002	网络错误	表示网络错误。
2003	客户端设备时间错误	表示客户端设备时间错误。
2006	刷脸失败	表示刷脸失败，如需获取更详细的失败原因，需调用服务端查询认证结果。

说明 更多错误码信息，请参见Android客户端错误码详情。

接口混淆配置

为避免接口混淆，您可以参考以下代码来保留类名（一般是在封装release包的时候保留类名）：

```
-verbose
-keep class com.aliyun.aliyunface.network.model.** {*;}
-keep class com.aliyun.aliyunface.api.ZIMCallback {*;}
-keep class com.aliyun.aliyunface.api.ZIMFacade {*;}
-keep class com.aliyun.aliyunface.api.ZIMFacadeBuilder {*;}
-keep class com.aliyun.aliyunface.api.ZIMMetaInfo {*;}
-keep class com.aliyun.aliyunface.api.ZIMResponse {*;}
-keep class com.aliyun.aliyunface.api.ZIMSession {*;}
-keep class com.aliyun.aliyunface.config.**{*;}
-keep class com.aliyun.aliyunface.log.RecordBase {*;}
-keep class com.aliyun.aliyunface.ui.ToygerWebView {*;}
-keep class com.alipay.zoloz.toyger.**{*;}
-keep class com.alipay.zoloz.image.** {*;}
-keep class com.alipay.android.** {*;}
-keep class net.security.device.api.** {*;}
-keep class com.alipay.deviceid.** { *; }
-keep class com.alibaba.fastjson.** {*;}
-keep class com.alibaba.sdk.android.oss.** { *; }
-dontwarn okio.**
-dontwarn org.apache.commons.codec.binary.**
-keepclassmembers,allowobfuscation class * {
     @com.alibaba.fastjson.annotation.JSONField <fields>;
}
```

权限说明

金融级实人认证为您提供安全的全链路身份核验服务。为了达到更好的设备风险防范效果，建议您在使用业务应用App调用刷脸SDK之前，提前向您的用户获取如下权限。

```
// 授权获取手机状态的权限。
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
// 授权获取接入网络的权限。开发人员需注意，6.0以上系统属于动态权限。
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Demo代码包

您可以下载Android Demo代码包（aliyunfaceDemo.zip）进行体验。

SDK包大小裁剪说明

如果您不需要使用OCR功能，可以在SDK包中删除*android-aliyunocr-sdk-release-xxxx_version.aar*，并重新编译。

1.1.3.2. iOS接入

刷脸认证提供iOS客户端SDK，帮助您在您的业务应用（App）中实现刷脸认证功能。您可通过刷脸认证的服务端认证初始化接口，获取刷脸认证唯一标识CertifyId，并使用CertifyId唤起刷脸认证客户端SDK。本文将结合示例代码进行iOS客户端接入的详细说明。

前提条件

- 刷脸认证应用必须运行在iOS 9.0及以上版本的系统上。

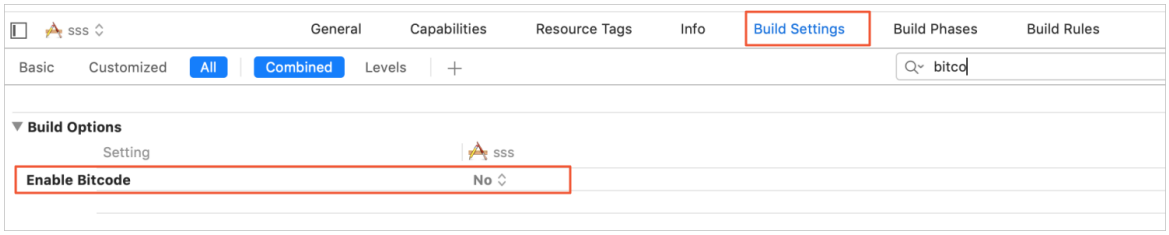
- 您的业务应用必须采用Objective C++或Swift集成金融级实人认证服务。
- 您的业务应用中未集成支付宝mPaaS SDK。

> **说明** iOS接入不支持模拟器模式调试。

配置开发环境

1. 在*info.plist*中配置摄像头权限请求。

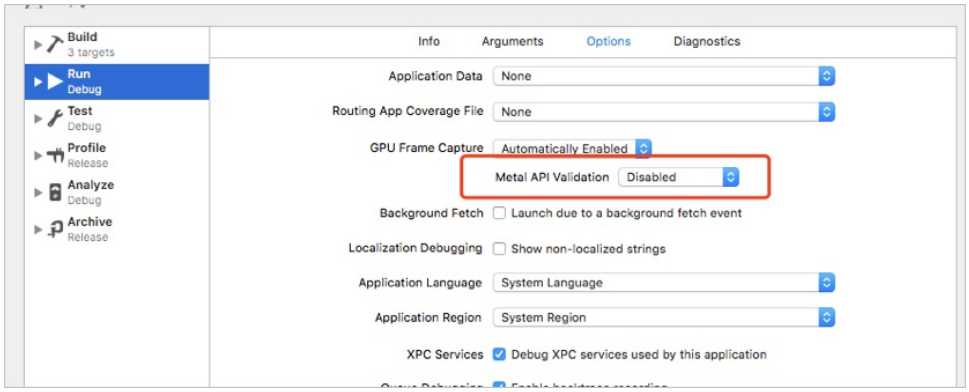
2. 在Xcode的编译设置中关闭Enable Bitcode选项。



3. 在Xcode编译设置的Linking > Other Linker Flags中，添加设置-ObjC -framework "BioAuthAPI" -lxml2。

   如果您的工程已设置了-force_load选项，则需要加入-force_load <framework path>/AliyunOSSiOS。

4. 在Xcode调试设置的Edit Scheme > Run > Options中禁用Metal API Validation。



配置依赖

1. 下载AliyunFace iOS SDK。

   该SDK为framework包。您需要在Xcode添加Link Binary With Libraries、SDK中的15个包和额外系统库依赖。

2. 解压下载的SDK包，并在解压的文件夹中执行以下命令提取所有的framework到Products目录。

```
for i in $(ls *.tgz);do tar xvf $i;done
```

- SDK中的包：

```
APPSecuritySDK
MPRemoteLogging
AliyunOSSiOS
BioAuthEngine
ZolozUtility
AliyunIdentityManager
APBToygerFacade
BioAuthAPI
ToygerService
ZolozSensorServices
ZolozOpenPlatformBuild
deviceiOS
ZolozMobileRPC
ZolozIdentityManager
OCRDetectSDKForTech
ToygerNative
```

- 系统库依赖：

```
CoreGraphics.framework
Accelerate.framework
SystemConfiguration.framework
AssetsLibrary.framework
CoreTelephony.framework
QuartzCore.framework
CoreFoundation.framework
CoreLocation.framework
ImageIO.framework
CoreMedia.framework
CoreMotion.framework
AVFoundation.framework
WebKit.framework
libresolv.tbd
libz.tbd
libc++.tbd
libc++.1.tbd
libc++abi.tbd
AudioToolbox.framework
CFNetwork.framework
MobileCoreServices.framework
libz.1.2.8.tbd
AdSupport.framework
```

拷贝资源文件

选择TARGETS，单击Build Phases页签，在Copy Bundle Resources中添加以下4个bundle：

- **APBToygerFacade.bundle**：位于*APBToygerFacade.framework*中。
- **ToygerNative.bundle**：位于*ToygerNative.framework*中。
- **OCRXMedia.bundle**：位于*OCRDetectSDKForTech.framework*中。
- **BioAuthEngine.bundle**：位于*BioAuthEngine.framework*中。

调用SDK

1. 引入头文件。

   命令如下：

```
#import <AliyunIdentityManager/AliyunIdentityPublicApi.h>
```

2. 初始化SDK。

   > **说明** 为了提升用户体验并为刷脸认证准备一些必备的数据，iOS客户端需要初始化SDK。您可以在调用 `appDelegate` 函数时（如下例代码所示）放置初始化SDK代码。您也可以在获取metaInfo数据前（如步骤3所示）放置初始化SDK代码。

   `appDelegate` 函数调用，示例代码：

```
(BOOL)application:(UIApplication)application didFinishLaunchingWithOptions:(NSDictionary)launchOptions {
[AliyunSdk init];// 初始化SDK。
}
```

3. 获取metaInfo数据。

   命令如下：

```
[AliyunSdk init];// 初始化SDK，如果已经在appDelegate函数中调用过，此处无需再重复调用。
[AliyunIdentityManager getMetaInfo];
```

   > **说明** 返回值类型是NSDictionary，您的服务端需要将其转换为JSON字符串。调用服务端发起认证请求接口时，在入参metaInfo中传入该值。关于发起认证请求的具体操作，请参见发起认证请求。

4. （可选）获取设备指纹。

   命令如下：

```
/*获取设备指纹token。
1，SDK初始化后需要经过一段时间计算才能获取到deviceToken，建议间隔3秒以上。
2，deviceToken短期有效，不能长期缓存使用，不能作为设备唯一标识。
*/
NSString * astring = [AliyunSdk getMobileSession];
```

   > **说明** 调用 `[AliyunSdk init]` 后，才能获取到该结果。

5. 调用SDK开始认证。

   命令如下：

```
[[AliyunIdentityManager sharedInstance] verifyWith:certifyId extParams:extParams onCompletion:^(ZIMResponse *response) {}];
```

参数说明如下表所示。

名称	类型	描述
certifyId	NSString	您可以通过服务端发起申请请求接口获取该参数。关于发起认证请求的具体操作，请参见发起认证请求。

名称	类型	描述

extParams	NSDictionary	通过以下代码传入当前viewController，用于展现网络加载和presentViewController。
		`[extParams setValue:self forKey:@"currentCtr"];`
		○ 如需设置OCR的**下一步**按钮的颜色，在extParams中添加按钮的正常颜色和按钮按下时的颜色。代码示例如下：
		`[extParams setValue:@"00FF00" forKey:ZIM_EXT_PARAMS_KEY_OCR_BOTTOM_BUTTON_COLOR];`
		`[extParams setValue:@"0000FF" forKey:PARAMS_KEY_OCR_BOTTOM_BUTTON_CLICKED_COLZIM_EXT_OR];`
		○ 如需设置扫脸时的圆圈颜色，在extParams中添加圆圈颜色。代码示例如下：
		`[extParams setValue:@"FF0000" forKey:ZIM_EXT_PARAMS_KEY_OCR_FACE_CIRCLE_COLOR];`
		○ 如需返回本地视频，在extParams中添加 `returnVideo` ，并且设置值为 `true` 。代码示例如下：
		`[extParams setValue:@"true" forKey:@"returnVideo"];`
		○ 如需服务端查询接口返回视频，需要额外在extParams中添加 `ZIM_EXT_PARAMS_KEY_USE_VIDEO_UPLOAD` ，并且设置值为 `true` 。代码示例如下：
		`[extParams setValue:@"true" forKey:ZIM_EXT_PARAMS_KEY_USE_VIDEO_UPLOAD]`
		说明 金融级实人认证支持设置返回活体检测的视频，默认设置为不返回。如果设置为返回，在您使用金融级实人认证SDK完成认证（不区分认证结果为F或T），并在SDK回调和查询接口时返回视频路径，请您及时处理。
		○ 如需自定义X号（页面关闭按钮），您可以参照以下步骤：
		■ 如需自定义X号的图片，需要额外在extParams中添加cancelIImage，并设置值为UIImage。代码示例如下：
		`[extParams setValue:UIImage forKey:@"cancelImage"];`
		■ 如需自定义X号的图片位置，需要额外在extParams中添加imageOrientation，并设置值为right或left 。默认为left。代码示例如下：
		`[extParams setValue:@"right" forKey:@"imageOrientation"];`

返回结果说明：

```
[[AliyunIdentityManager sharedInstance] verifyWith:certifyId extParams:extParams onCompletion:^(ZIMResponse *response) {}];
```

`response.code` 包含以下返回参数，具体说明如下表所示。

错误码	错误码文案	错误码描述
1000（ZIMResponseSuccess）	刷脸成功	表示刷脸成功。 说明 该结果仅供参考，可通过服务端查询认证结果接口获取最终认证结果。关于发起认证请求的具体操作，请参见发起认证请求。
1001（ZIMInternalError）	系统错误	表示系统错误。
1003（ZIMInterrupt）	验证中断	表示验证中断。
2002（ZIMNetworkfail）	网络错误	表示网络错误。
2003（ZIMTIMEError）	客户端设备时间错误	表示客户端设备时间错误。
2006（ZIMResponseFail）	刷脸失败	表示刷脸失败。 说明 如需获取失败原因的详细信息，需调用服务端查询认证结果接口。关于发起认证请求的具体操作，请参见发起认证请求。

说明 关于错误码的更多信息，请参见iOS客户端错误码详情。

代码示例：

```
[[AliyunIdentityManager sharedInstance] verifyWith:certifyId extParams:extParams onComp
letion:^(ZIMResponse *response) {
        dispatch_async(dispatch_get_main_queue(), ^{
            NSString *title = @"刷脸成功";
            switch (response.code) {
                case ZIMResponseSuccess: //1000。
                    break;
                case ZIMInterrupt:   //1003。
                    title = @"用户退出";
                    break;
                case ZIMNetworkfail: //2002。
                    title = @"网络错误";
                    break;
                case ZIMTIMEError: //2003。
                    title = @"设备时间设置不对";
                    break;
                case ZIMResponseFail: //2006。
                    title = @"刷脸失败";
                    break;
                case ZIMInternalError:   //1001。
                    title = @"初始化失败";
                    break;
                default:
                    break;
            }
```

Demo代码包

您可以下载以下iOS Demo代码包进行体验。在使用过程中，请替换iOS Demo代码包中的文件的版本为最新的版本：

- Objective-C Demo
- Swift Demo

SDK包大小裁剪说明

当您对SDK包大小有限制时，可以根据如下情况进行裁剪：

- 如果您之前没有使用过支付宝开放平台版，或开放平台版代码已经下线，可以在SDK包中删除ZolozIdentityManager、ZolozMobileRPC内容。
- 如果您不需要使用OCR功能，可以在SDK包中删除OCRDetectSDKForTech内容，并删除对应的bundle（OCRXMedia.bundle）。

1.1.3.3. UniApp接入

金融级实人认证服务提供UniApp插件，帮助您在您的业务应用（App）中实现刷脸认证功能。您可以通过您的应用服务端认证初始化接口，向阿里云服务器调用发起认证请求接口并获取刷脸认证的唯一标识CertifyId，然后使用CertifyId唤起刷脸认证客户端SDK。本文介绍了UniApp客户端接入的操作方法。

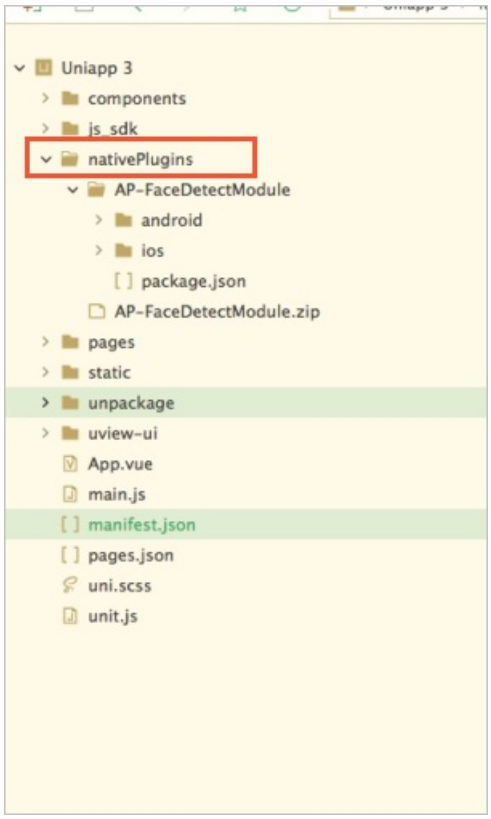
接入前必读

近期发现有个人开发者封装阿里云实人认证SDK后开发了第三方插件，并在某平台上提供付费下载服务。阿里云提醒您，如果您使用了此类非阿里云官方提供的插件或服务，阿里云概不对由此造成的一切后果负责，亦不承担任何法律责任。建议您使用阿里云官方SDK服务。

配置依赖

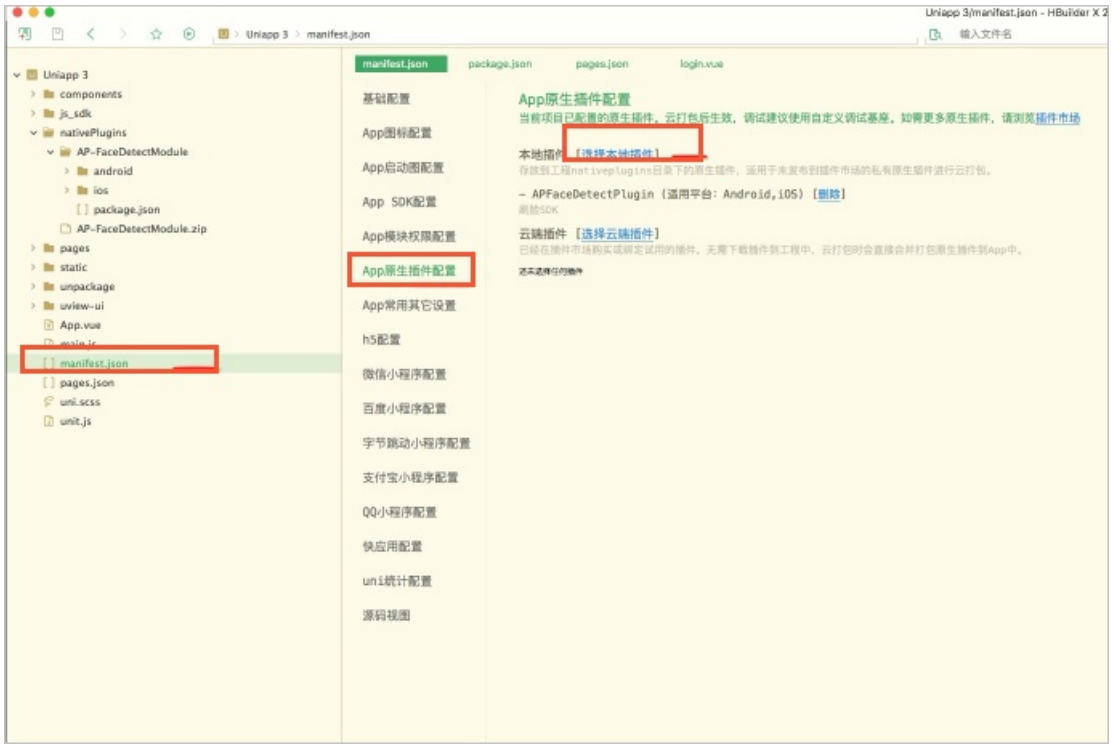
您可以参见DCloud官网文档接入UniApp插件，也可以参考以下步骤接入UniApp插件。

1. 下载金融级实人认证UniApp SDK，解压到项目中的*nativePlugins*目录下。



2. 导入本地插件。
   i. 在*manifest.json*文件下，单击**App原生插件配置**。

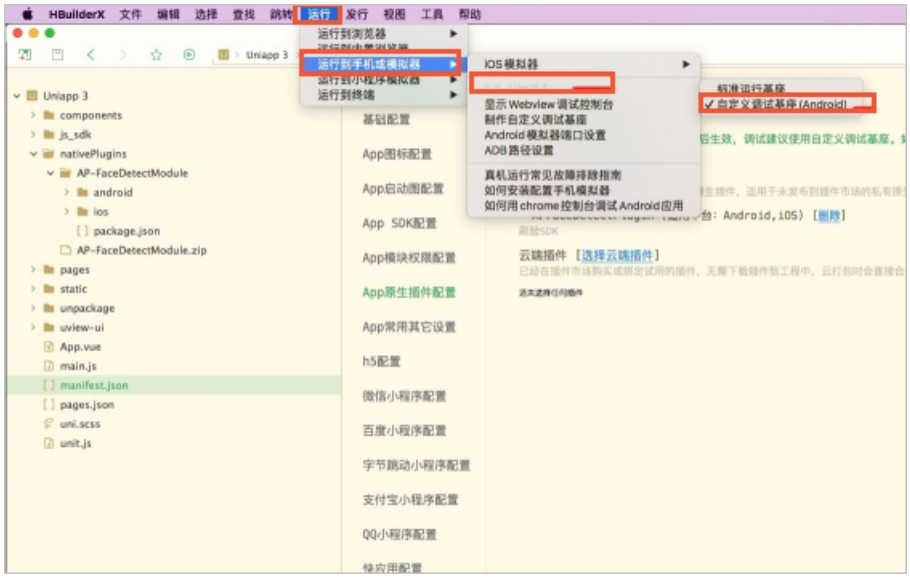
ii. 在**App原生插件配置**区域，单击**选择本地插件**。



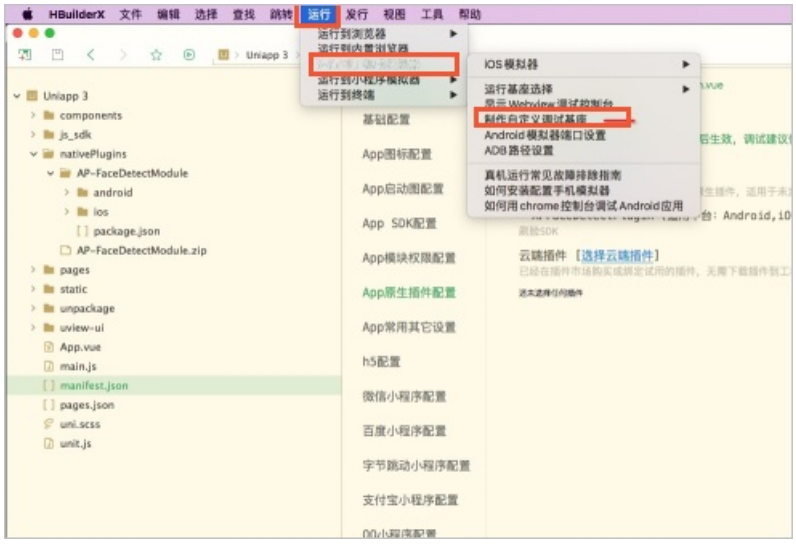
iii. 在**本地插件选择**对话框，选择需要打包生效的插件。

3. 配置基座包。

i. 设置自定义调试基座。在*manifest.json*文件，选择**运行 > 运行到手机或模拟器 > 运行基座选择 > 自定义调试基座(Android)**。



ii. 制作自定义调试基座。在*manifest.json*文件，单击**运行 > 运行到手机或模拟器**，选择**制作自定义调试基座**。



4. 调用实人认证服务。

   i. 获取metaInfo数据。

   您在调用刷脸认证服务端发起认证请求时需要传入该值。具体操作，请参见发起认证请求。

   ii. 调用**verify**接口，传入certifyId进行认证。

   参数说明如下表所示。

名称	类型	描述
certifyId	String	认证的唯一标识。
returnVideo	String	是否返回视频（iOS）。取值： ■ **true**：返回视频。 ■ **false**（默认）：不返回视频。
ext_params_key_use_video	String	是否返回视频（Android）。取值： ■ **true**：返回视频。 ■ **false**（默认）：不返回视频。
ext_params_key_ocr_bottom_button_color	String	OCR底部按钮的默认颜色。
ext_params_key_ocr_bottom_button_clicked_color	String	OCR底部按钮被操作时的颜色。
ext_params_key_ocr_face_circle_color	String	实人认证进度条颜色。

接入示例：

```
<template>
</template>
<script>
const n = uni.requireNativePlugin('AP-FaceDetectModule');
export default {
    methods: {
        verifyC: function() {
            n.verify(
                {
                    certifyId: this.certify
                },
                function(t) {
                    uni.showToast({
                        title: '返回的内容' + t,
                        icon: 'none'
                    });
                }
            );
        },
        //获取环境参数接口。
        getMetaInfo: function() {
            var t = n.getMetaInfo();
        },
    }
};
</script>
```

错误码说明

错误码	错误码文案	错误码描述
1000	刷脸成功	表示刷脸成功。 **说明** 该结果仅供参考。您可通过服务端查询认证结果接口获取最终认证结果。具体操作，请参见InitFaceVerify。
1001	系统错误	表示系统错误。
1003	验证中断	表示验证中断。
2002	网络错误	表示网络错误。
2003	客户端设备时间错误	表示客户端设备时间错误。

错误码	错误码文案	错误码描述
2006	刷脸失败	表示刷脸失败。 **说明** 如需获取失败原因的详细信息，需调用服务端查询认证结果接口获取。具体操作，请参见InitFaceVerify。

1.1.3.4. APICloud接入

金融级实人认证服务提供APICloud插件，帮助您在业务应用（App）中实现刷脸认证功能。您可以通过应用服务端认证初始化接口，向阿里云服务器调用发起认证请求接口并获取刷脸认证的唯一标识CertifyId，然后使用CertifyId唤起刷脸认证客户端SDK。本文介绍了APICloud客户端接入的操作方法。

背景信息

低代码平台APICloud的模块库提供多种模块。您可以使用已有模块，在应用中快速部署集成，缩短开发流程。当已有模块无法满足特殊需求时，您也可以自定义相关模块。

接入前必读

近期发现有个人开发者封装阿里云实人认证SDK后开发了第三方插件，并在某平台上提供付费下载服务。阿里云提醒您，如果您使用了此类非阿里云官方提供的插件或服务，阿里云概不对由此造成的一切后果负责，亦不承担任何法律责任。建议您使用阿里云官方SDK服务。

前提条件

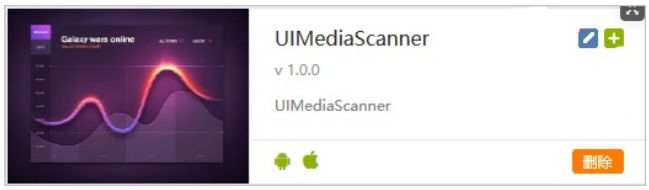
已在APICloud官网中创建应用。

配置依赖

1. 下载金融级实人认证APICloud SDK。
2. 在APICloud控制台上传刷脸认证APICloud模块。
   i. 登录APICloud控制台。
   ii. 在左侧导航栏，选择**端开发 > 模块**。
   iii. 在**模块**页签，选择**自定义模块**。
   iv. 在**自定义模块**页签，输入有效的**模块名称**、**模块概要**及版本号，选择并上传已下载的APICloud模块。

   v. 单击**保存**。
3. 将APICloud模块添加到应用中。
   i. 在**模块**页签，选择**模块库**。

ii. 在**模块库**页签，单击已上传APICloud模块右上角的图标，将APICloud模块添加到应用中。



> **说明** 自定义模块仅对自己的账号可见。如果需要对外开放，您可以在**模块管理**页面提交该模块，官方审核通过后将会发布到模块库。

当您需要更新已上传的自定义模块时，可以通过已上传自定义模块右上角的图标，重新上传自定义模块包，然后单击**保存**。

函数说明

APICloud插件的接入流程包含获取MetaInfo和开始认证函数。具体说明如下。

- **获取MetaInfo数据**

  调用getMetaInfo函数获取当前移动设备端的环境信息，并将这些信息发送到业务服务器端。业务服务器端会将这些信息作为参数之一（MetaInfo）调用服务端初始化认证接口（InitSmartVerify），从而获取CertifyId，用于后续验证环节。

  示例代码：

```
var aliyunFace = api.require('aliyunFaceVerifyModule');
aliyunFace.getMetaInfo(function(ret, err) {
    alert(ret);
});
```

- **开始认证**

  调用Verify函数发起金融级实人认证。

  参数说明：

名称	类型	说明
certifyId	String	从服务端初始化认证接口（InitSmartVerify）获取的CertifyId。 **说明** 每个CertifyId只能调用一次**Verify**函数，每次调用**Verify**函数之前都必须重新获取CertifyId。
extParams	Map<String, String>	用户自定义参数，可以为空。有关目前支持自定义的字段，请参见用户自定义参数extParams支持的字段说明。

名称	类型	说明

callback	IdentityCallback	认证结果回调。回调格式如下：
		`public class IdentityResponse {`
		`// 结果Code，具体含义请参见本文档“返回值说明”章节。`
		`public int code;`
		`// 结果Code描述。`
		`public String message;`
		`}`
		`public interface IdentityCallback {`
		`boolean response(IdentityResponse response);`
		`}`

用户自定义参数extParams支持的字段说明

名称	描述	示例值
ext_params_key_use_video	是否需要返回视频留证。取值： ○ **false（默认）**：不返回视频留证。 ○ **true**：返回视频留证。	false
kIdentityParamKeyNextButtonColor	OCR识别结果确认页底部按钮的底色。	#FF0000
ext_params_key_face_progress_color	定制实人认证进度条的颜色。	#FF0000

示例代码：

```
var aliyunFace = api.require('aliyunFaceVerifyModule');
aliyunFace.verify({
    certifyId: '1adffa'
}, function(ret, err) {
    if (ret.code == 1000) {
        alert('认证成功');
    } else {
        alert(err.code);
    }
});
```

返回值说明

服务器端的验证结果通过 `IdentityResponse.code` 返回，取值范围如下。

错误码	说明
1000	刷脸成功。
1001	系统错误。

错误码	说明
1003	验证中断。
2002	网络错误。
2003	客户端设备时间错误。
2006	刷脸失败。

1.2. PC或移动端H5网页接入

1.2.1. 接入流程

实人认证提供Web SDK帮助您在浏览器或内嵌webview中实现实人认证功能。

前置条件

您必须了解通过HTTP、HTTPS原生调用金融级实人认证服务接口时，构造请求签名的方法。更多签名机制的信息，请参见API签名机制。

操作步骤

1. 开通金融级实人认证服务。
2. 添加认证场景。

   > **说明** 基于不同业务场景，您可以在添加认证场景时，手动配置是否授权阿里云访问指定OSS存储空间用于存储认证的资料图片。

3. 如果需要将认证过程中的人脸采集照片存储到OSS空间，您需要授权金融级实人认证访问OSS存储空间。
4. 接入PC或移动端H5网页认证方案。
   i. 进行服务端接入，获取认证URL。

      具体步骤，请参见服务端接入。

   ii. 在应用客户端接入认证URL。

      具体步骤，请参见PC或移动端H5网页接入。

1.2.2. 服务端接入

本文介绍了PC或移动端H5网页认证方案的接入流程。

发起认证请求

接口名：InitFaceVerify。

服务地址：cloudauth.aliyuncs.com。

请求方法：HTTPS POST和GET。

接口说明：每次开始认证前通过本接口获取CertifyId，用来串联认证请求中的各个接口。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	要接入的认证场景ID。	1000000006
OuterOrderNo	String	是	您自定义的业务唯一标识，用于后续定位排查问题使用。 支持长度为32位的字母和数字组合，请确保唯一。	e0c34a77f5ac40a5aa5e6ed20c353888
ProductCode	String	是	要接入的认证方案。 唯一取值：**ID_PRO**。	ID_PRO
CertType	String	是	您的终端用户的证件类型。 当前仅支持身份证。 唯一取值：IDENTITY_CARD。	IDENTITY_CARD
CertName	String	是	您的终端用户的真实姓名。	张三
CertNo	String	是	您的终端用户的证件号码。	330103xxxxxxxxxxxx
ReturnUrl	String	是	您业务页面回跳的目标地址。	http://www.aliyun.com
MetaInfo	String	是	MetaInfo环境参数，需要通过JS获取。	{"zimVer":"3.0.0","appVersion": "1","bioMetaInfo": "4.1.0:11501568,0","appName": "com.aliyun.antcloudauth","deviceType": "ios","osVersion": "iOS 10.3.2","apdidToken": "","deviceModel": "iPhone9,1"}
Mobile	String	否	您终端用户的手机号码。	130xxxxxxxx
Ip	String	否	您终端用户的IP。	114.xxx.xxx.xxx
UserId	String	否	您自定义的用户ID，请保持唯一。	123456789

名称	类型	是否必选	描述	示例值
CallbackUrl	String	否	认证结果的回调通知地址，必须以 `https` 开头。 平台在完成认证后会回调该地址，并自动添加 `certifyId` 、 `passed` 和 `subcode` 字段，示例： `https://www.aliyun.com?certifyId=xxxx&passed=T&subcode=200` 。	https://www.aliyun.com
CallbackToken	Callback Token	否	安全Token，由您自行生成，用于防重复、防篡改校验。 如果设置了该值会在回调地址显示**CallbackToken**字段。	NMjvQanQgplBSaEI0sL86WnQplB
CertifyUrlType	String	否	Web SDK设备类型。取值**WEB**或者**H5**。 **说明** 只支持Web SDK设备类型。	WEB
AuthId	String	否	用户授权ID，最大长度为64位字符。	92d46b9e9e2d703f2897f350d5bd4149

返回数据

名称	类型	描述	示例值
RequestId	String	请求ID。	130A2C10-B9EE-4D84-88E3-5384FF039795
Message	String	返回信息。	success
Code	String	返回码：200为成功，其他为失败。	200
ResultObject.CertifyId	String	实人认证唯一标识。	91707dc296d469ad38e4c5efa6a0f24b
ResultObject.CertifyUrl	String	Web认证URL，认证结束后根据入参ReturnUrl进行跳转。	ResultObject.CertifyUrl

返回Code和Message说明

Code	Message	描述
200	success	成功。

Code	Message	描述
400	参数不能为空	参数不能为空。
401	参数非法	非法参数。传入的姓名、身份证号码长度必须符合国家标准且不得包含英文字母等特殊字符。
402	应用配置不存在	应用配置不存在。
404	认证场景配置不存在	认证场景配置不存在，请先在控制台上创建认证场景。
410	未开通服务	未开通OSS产品或未完成OSS读写授权，请登录控制台完成授权。
411	RAM无权限	需要给RAM用户授予AliyunAntCloudAuthFullAccess的操作权限。
412	欠费中	金融级实人认证或OSS存在欠费，请充值后操作。
414	设备类型不支持	当前移动设备不支持刷脸认证，请更换设备后操作。
415	SDK版本不支持	当前认证SDK版本不支持刷脸认证，请升级SDK后操作。
416	系统版本不支持	当前操作系统版本不支持刷脸认证，请升级系统或更换设备操作。
417	无法使用刷脸服务	当前身份信息比对源不可用。若信息正确，建议人工审核。
418	刷脸失败次数过多	当天刷脸认证次数过多，请明天再试。
500	系统错误	系统内部错误，请反馈工程师排查。

查询认证结果

接口名：**DescribeFaceVerify**。

服务地址：cloudauth.aliyuncs.com。

请求方法：HTTPS POST和GET。

接口说明：当接入方移动端收到回调之后，其服务端可以调用此接口，来获取相应的认证状态和认证资料。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	认证场景ID。	1000000006
CertifyId	String	是	实人认证唯一标识。	91707dc296d469ad38e4c5efa6a0f24b

返回数据

名称	类型	描述	示例值
RequestId	String	请求ID。	130A2C10-B9EE-4D84-88E3-5384FF039795
Message	String	请求消息的响应信息。	success
Code	String	返回码。关于返回码的详细说明，请参见下文**ResultObject.SubCode错误码说明**。	200
ResultObject.Passed	String	认证结果。取值： • **T**：通过。 • **F**：未通过。 **说明** 判断认证结果请以ResultObject.Passed字段为准。	T
ResultObject.SubCode	String	认证结果描述。具体请参见下文**ResultObject.SubCode错误码说明**。 **说明** 判断认证结果请以ResultObject.Passed字段为准。	200
ResultObject.IdentityInfo	String	认证的主体信息，一般的认证场景返回为空。	null
ResultObject.MaterialInfo	String	认证主体附件信息，主要为图片类材料。 本参数取值中涉及的字段解释，请参见下文的返回示例。	{"faceAttack": "F","faceOcclusion":"F","facialPictureFront": {"faceAttackScore": 0.00008597839769208804,"qualityScore": 88.3615493774414,"verifyScore": 50.28594166529785,"pictureUrl": "https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg","ossBucketName": "cn-shanghai-aliyun-cloudauth-1260051251634779","ossObjectName": "verify/1260051251634779/6ba7bcfccf33f56cdb44ed086f36ce3e0.jpeg"}}

> **说明** 判断认证结果请以ResultObject.Passed字段为准。

返回示例

ResultObject.MaterialInfo的JSON格式示例：

```
{
    // 是否为攻击：攻击为T，非攻击为F。
    "faceAttack": "T",
    // 是否有脸部遮挡：有脸部遮挡为T，否则为F。
    "faceOcclusion": "F",
    // 认证的照片信息。
    "facialPictureFront": {
         //人脸攻击分。
         "faceAttackScore": 0.00008597839769208804,
        // 照片存放客户上海区域的OSS里，此为照片的bucket名。
        "ossBucketName": "cn-shanghai-aliyun-cloudauth-1260051251634779",
        // 照片存放客户上海区域的OSS里，此为照片的文件名。
        "ossObjectName": "verify/1260051251634779/03a081bd96328aedf83f635f39a50c57_0.jpeg",
        // 照片https地址，有效期15分钟，从查询时开始计时。
        "pictureUrl": "http://cn-shanghai-aliyun-cloudauth-1260051251634779.oss-cn-shanghai
.aliyuncs.com/verify/1260051251634779/03a081bd96328aedf83f635f39a50c57_0.jpeg?Expires=16253
71140&OSSAccessKeyId=STS.NTX1ngfr6Acg2Pmnn2RYM****&Signature=Hw5BF9WxJs6wI68IxKs41cxCU8****
&security-token=CAISjgJ1q6Ft5B2yfSjIr5ftetTTi60X9qGMMHbcim5nXtZhu7GT1Dz2IH1PdXFgA%2Bgds%2Fs
wmm5U7vgalrkqEcEdHRGdN5YpsM8LrlzwO1h2TGRsq%2B5qsoasPETOITyZtZagToeUZdfZfejXGDKgvyRvwLz8WCy%
2FVli%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGfLC1dysQcO4gEWq4bHm5fCskKD1Qenk7F
O%2B9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWwIMvErYbbaMrIV1WwZ9Z7knHaVAq%2BXw
nOBkuuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAF%2FHZbelNLYWtipAu5X5oG1t22SqIr1p3TMK5FrjDIVeWOpp
cxeRXtU%2BjR7hGcwd25scGhiraoxTXV3sxw%2F6dhFSswQ37O4j%2B%2FrCPtiGauqn2ZMdMj%2FvYwKb6KmpQqa%2
BtR%2F9cxhS6hoIQRq0SUIfyXl5ZUiZbTjW22iwuX%2BPwzVCw%3D%3D",
        // 活体人脸质量分数。
        "qualityScore": 99.93476867675781,
        // 人脸和公安比对分数，阈值可参考下表详细说明。
        "verifyScore": 57.678396649466755
    },
}
```

ResultObject.SubCode错误码说明

错误码	认证描述文案	是否计费	可能原因和处理建议
200	认证通过	是	成功。
201	姓名和身份证不一致	是	可能是用户的信息有误或用户的信息为假信息，建议用户确认后重新操作。若同一身份信息重新发起认证，服务端初始化将会返回417错误。
202	查询不到身份信息	是	可能是用户户口迁移、用户是军人等特殊状态导致，建议预留人工审核入口，进行人工审核。若同一身份信息重新发起认证，服务端初始化将会返回417错误。

错误码	认证描述文案	是否计费	可能原因和处理建议
203	查询不到照片或照片不可用	是	可能是公安库数据问题导致，建议预留人工审核入口，进行人工审核。若同一身份信息重新发起认证，服务端初始化将会返回417错误。
204	人脸比对不一致	是	可能不是同一人或活体照片质量较低，建议根据业务情况分层处理，若为同一人可重新操作。
205	活体检测存在风险	是	可能存在攻击风险，建议人工审核分层处理，若为真人可重新操作。
206	业务策略限制	是	个性化业务安全策略限制，如有疑问可咨询客服。
210	认证通过	是	通过产品安全策略综合判断，建议认证通过。

verifyScore阈值说明

千分之一误识率	万分之五误识率	万分之一误识率	十万分之五误识率	十万分之一误识率
70	71.5	75	76.5	80

说明 如果您有个性化需求，您可以根据业务情况，参考返回的比对分和阈值，自定义认证结果。

返回Code和Message说明

Code	Message	描述
200	success	成功。
400	参数不能为空	参数不能为空。
401	参数非法	非法参数。传入的姓名、身份证号码长度必须符合国家标准且不得包含英文字母等特殊字符。
403	认证未完成	认证未完成。
404	认证场景配置不存在	认证场景配置不存在，请先在控制台上创建认证场景。
406	无效的CertifyId	无效的CertifyId。
410	未开通服务	未开通OSS产品或未完成OSS读写授权。
411	RAM无权限	需要给RAM用户授予**AliyunAntCloudAuthFullAccess**的操作权限。
424	身份认证记录不存在	当前CertifyId还未完成认证，建议您客户端收到认证回调为1000或2006时，再请求查询接口获取详情。 **说明** 如果出现CertifyId未认证、认证过程中放弃、认证处理中的情况，均会导致查询接口返回此错误码。

Code	Message	描述
500	系统错误	系统内部错误。

SDK使用说明

> **说明** 如需其他语言的SDK示例代码，您可以通过OpenAPI在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。更多信息，请参见：
>
> - 调试InitFaceVerify接口
> - 调试DescribeFaceVerify接口

Java SDK使用示例

- Maven依赖：

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
    <version>4.4.3</version>
</dependency>
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-cloudauth</artifactId>
<version>2.0.32</version>
</dependency>
```

- 使用示例：

```
DefaultProfile profile = DefaultProfile.getProfile(
            "cn-hangzhou",     // 固定为cn-hangzhou。
            "AccessKeyID",       // 您的AccessKey ID。
            "AccessKeySecret");  // 您的AccessKey Secret。
IAcsClient client = new DefaultAcsClient(profile);
InitFaceVerifyRequest request = new InitFaceVerifyRequest();
request.setMetaInfo("MetaInfo环境参数");
request.setCertName("张三");
request.setCertNo("18位身份证号");
// 固定值。
request.setCertType("IDENTITY_CARD");
request.setSceneId(100000xxxxL);
request.setOuterOrderNo("客户请求的唯一标识");
// 固定值。
request.setProductCode("ID_PRO");
// Web SDK请求时为必填。
request.setReturnUrl("http://www.aliyun.com");
InitFaceVerifyResponse response = client.getAcsResponse(request);
System.out.println(response.getCode());
System.out.println(response.getMessage());
System.out.println(response.getRequestId());
System.out.println(response.getResultObject().getCertifyId());
// 等App客户端提交刷脸认证后，在客户端SDK的回调函数中，由客户端通知服务端运行以下代码查询认证结果。
DescribeFaceVerifyRequest request2 = new DescribeFaceVerifyRequest();
request2.setCertifyId("InitFaceVerify接口返回的CertifyId");
request2.setSceneId(100000xxxxL);
DescribeFaceVerifyResponse response2 = client.getAcsResponse(request2);
System.out.println(response2.getCode());
System.out.println(response2.getMessage());
System.out.println(response2.getRequestId());
System.out.println(response2.getResultObject().getPassed());
System.out.println(response2.getResultObject().getSubCode());
System.out.println(response2.getResultObject().getMaterialInfo());
System.out.println(response2.getResultObject().getIdentityInfo());
```

1.2.3. PC或移动端H5网页接入

本文介绍PC或移动端H5网页SDK的接入流程。

手机浏览器支持列表

浏览器名称	支持情况
Edge	支持
FireFox	支持
Chrome	支持
Safari	支持（iOS 11及以上）
Opera	支持

浏览器名称	支持情况
Android Browser 5.0+	支持
UC	支持
QQ	支持
厂商内置浏览器	小米、三星等部分机型支持

说明 受制于浏览器兼容性碎片化问题，建议您在流程设计上引导用户使用推荐的浏览器完成认证。若您在手机应用App内集成该方案，可能会因为内嵌浏览器原因无法兼容，推荐您使用SDK模式接入。若您是通过微信公众号或者小程序集成，由于微信运营审核规则的限制，可能出现无法避免的兼容性问题，建议您使用纯服务端（API）接入方式。

扫脸认证

为了落实用户个人信息保护相关法律法规要求，提升用户刷脸认证体验，URL默认自带扫脸认证引导页，用户勾选同意认证服务协议，同意授权后，可以开始使用扫脸认证服务。微信公众号场景下，扫脸认证引导页是必须存在的，否则会导致摄像头无法唤起，引发黑屏现象。

说明 目前，新增的用户授权页面处于灰度中。如果需要在您的应用客户端增加该页面，您可以提交工单，获取技术支持。

启动刷脸

```
<script type="text/javascript"src="https://cn-shanghai-aliyun-cloudauth.oss-cn-shanghai.aliyuncs.com/web_sdk_js/jsvm_all.js" ></script>
```

1. 引入上述JS文件，调用函数getMetaInfo()获取MetaInfo。
2. 您在调用实人认证服务端发起认证请求时需要传入该MetaInfo值。
3. 获取服务端初始化返回的CertifyUrl，在浏览器中加载该链接进行实人认证。

结果解析

CertifyUrl中ReturnUrl参数为商户自定义参数，认证结果会在扫脸认证完成后拼接在ReturnUrl后作为参数返回给商户。格式如下：

```
response=%7B"code"%3A1001%2C"subCode"%3A"Z5051"%2C"reason"%3A"success"%2C"extInfo"%3A%7B%7D
%7D
```

从URL中获取response参数，URL解码后格式如下：

```
{
  "code":"",
  "subCode":"",
  "reason":"",
  "extInfo" : {}
}
```

说明 前端路由模式为history，只需对response解码一次即可。若您使用的是hash路由模式，则需要对response进行二次解码（例如 `decodeURIComponent(decodeURIComponent('responseValue'))` ）。

结果说明

错误码	描述
1000	代表刷脸成功。该结果仅供参考，可通过服务端查询认证结果获取最终认证结果。
1001	表示系统错误。
1003	表示验证中断。
2002	表示网络错误。
2003	表示客户端设备时间错误。
2006	表示刷脸失败。如需获取更详细的失败原因，需调用服务端查询认证结果。

subCode说明

错误码	描述
Z5051	上传刷脸图片失败。
Z5052	数据错误或程序异常。
Z5053	网络错误。
Z5054	摄像头无权限或者无法获取摄像头数据出错。
Z5055	用户退出。

错误码	描述
Z5056	重试次数过多。

1.3. 支付宝H5页面接入

1.3.1. 接入流程

支付宝H5页面接入适用于业务使用方希望在自己开发的支付宝H5网页中对用户完成线上实人认证的场景。

前置条件

您必须了解通过HTTP、HTTPS原生调用金融级实人认证服务接口时，构造请求签名的方法。关于API签名机制的信息，请参见API签名机制。

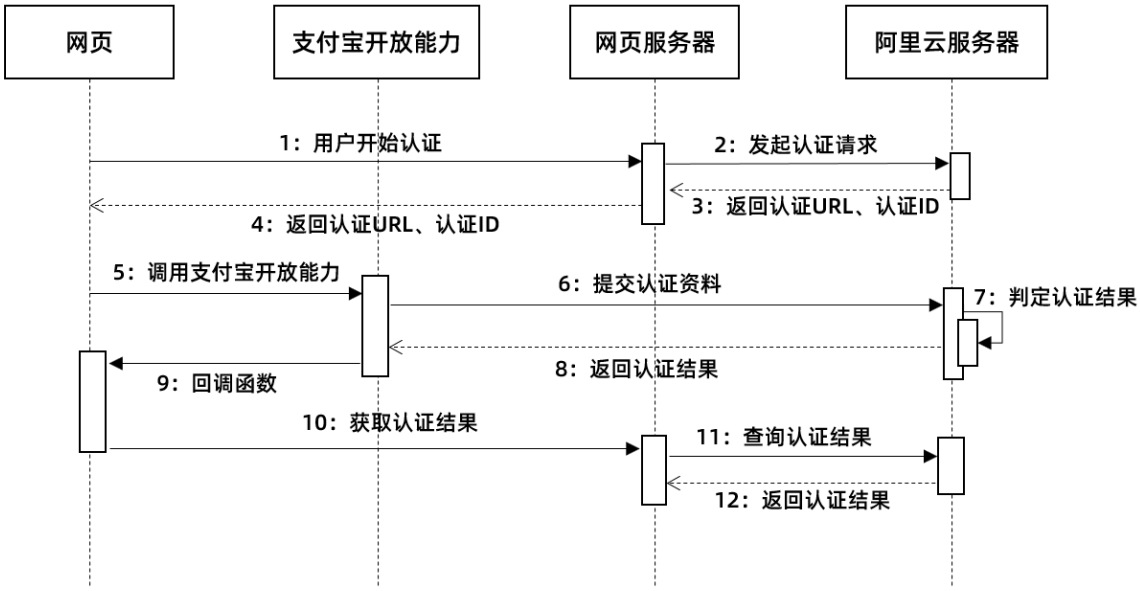
操作步骤

1. 开通金融级实人认证服务。
2. 添加认证场景。
3. 接入支付宝H5页面。

   在您完成上述步骤后，您可以进行支付宝H5页面接入，接入步骤如下：

   i. 调用服务端初始化接口。
   ii. 调用接入支付宝H5页面。
   iii. 调用服务端查询接口。

   您可以查看下方调用时序图了解详细接入步骤。



系统调用时序说明：

序号	说明
1	用户开始认证。

序号	说明
2	网页服务端向阿里云服务端发起认证请求。
3	阿里云服务端返回认证ID和认证URL给网页服务端。
4	网页服务端将认证ID和认证URL传递给网页。
5	网页调用支付宝开放能力。
6	支付宝开放能力将用户提交的认证资料传递给阿里云服务端。
7	阿里云服务端根据资料判定认证结果。
8	阿里云服务端将认证结果返回给支付宝开放能力。
9	支付宝开放能力通过回调函数指引网页获取认证状态。
10	网页向阿里云服务端查询认证状态。
11	阿里云服务端向网页服务端向发起查询。
12	阿里云服务端将认证结果返回给网页服务端。

在您了解接入步骤后，可以通过以下方式接入支付宝H5页面。

- 进行服务端接入，请参见HTTPS原生调用和SDK调用接入。
- 进行支付宝H5页面接入，请参见以下文档：
  - 端内接入支付宝H5页面
  - 端外唤起支付宝H5页面

1.3.2. 服务端接入

1.3.2.1. HTTPS原生调用

本文主要介绍通过HTTPS原生调用方式在服务端集成金融级实人认证的方法。

调用方式

服务请求地址：`https://saf.cn-shanghai.aliyuncs.com`

协议：HTTPS

方式：POST

发起认证请求

发起认证请求时，传入通用参数，并在ServiceParameters的JSON字符串里传入以下字段：

名称	类型	是否必选	描述	示例值

名称	类型	是否必选	描述	示例值
method	String	是	发起认证请求的操作。 取值：*init*。	init
sceneId	String	是	认证场景ID，该ID在控制台创建认证场景后自动生成。	100000****
outerOrderNo	String	是	客户服务端自定义的业务唯一标识，用于后续定位排查问题时使用。值最长为32位长度的字母和数字组合，请确保唯一。	e0c34a77f5ac40a5aa5e6ed20c35****
bizCode	String	是	认证场景码和商户发起认证的接入端有关： • 当商户在iOS或安卓平台发起认证时，认证场景码为 `FACE_SDK` 。 • 当商户在小程序中H5页面中发起认证时，认证场景码为 `FACE` 。	FACE_SDK
identityType	String	是	身份信息的参数类型，必须传入CERT_INFO。	CERT_INFO
certType	String	是	证件类型。当前支持身份证，必须传入IDENTITY_CARD。	IDENTITY_CARD
certNo	String	是	用户身份证件号码。	1**********9
certName	String	是	用户姓名。	张先生
returnUrl	String	是	商户业务页面回调的目标地址。 • 如您不需要回调商户业务页面，您可以在此处传入空字符串。 • 当您采用端外唤起支付宝H5页面，且希望您的用户唤起支付宝完成认证后，能够跳回您的应用页面，您需要在此参数下传入您应用的Scheme。详细内容请参见回跳原应用。	https://www.aliyun.com
encryptType	String	否	加密类型。为空表示不加密。 如开启加密传输，需传入加密算法。目前仅支持SM2国密算法。 如需传入加密算法，需对certName和certNo进行加密，并传入加密后的密文。有关参数加密的更多信息，请参见参数加密说明。	SM2

发起认证请求后，您会收到返回信息。返回参数列表如下：

名称	类型	是否必选	描述	示例值
certifyId	String	是	认证ID，刷脸认证的唯一标识。	*7eff3ad26a9c7b68c511b9f35eb1******
certifyUrl	String	是	认证流程入口Url。	*https://picker.antcloudauth.aliyuncs.com/gateway.do?...*

返回示例如下：

```
{
    "code": 200,
    "requestId": "5CD69C04-8A6B-4CC3-A0C9-D18D5FEFA788",
    "data": {
        "certifyUrl": "https://picker.antcloudauth.aliyuncs.com/gateway.do?...",
        "certifyId": "7eff3ad26a9c7b68c511b9f35eb1****"
    },
    "message": "OK"
}
```

查询认证结果

查询认证结果时，传入通用参数，并在ServiceParameters的JSON字符串里传入以下字段：

名称	类型	是否必选	描述	示例值
method	String	是	查询人脸比对结果的操作。 取值：*query*。	*query*
certifyId	String	是	认证ID，需与发起认证请求时返回的certifyId保持一致。	*7eff3ad26a9c7b68c511b9f35eb1******
sceneId	String	是	认证场景ID，需与发起认证请求时的sceneId保持一致。	*100000*******

发起查询请求后，您会收到返回信息。返回参数列表如下：

名称	类型	是否必选	描述	示例值
passed	String	是	是否通过认证。取值： • *T*：认证通过。 • *F*：未通过认证。	T

名称	类型	是否必选	描述	示例值
identityInfo	String	否	认证的主体信息，一般的认证场景返回为空。	"IdentityInfo": "{"cert_type":"IDENTITY_CARD","cert_no":"1**********9","cert_name":"张先生"}"
materialInfo	String	否	认证主体附件信息，主要为图片类材料，一般的认证场景都是返回空。	"MaterialInfo": "{\"facial_picture_front\": {\"verify_score\":\"80.107902181204\",\"quality_score\":\"82\",\"FEATURE_FACE\":\"/9j/4AAQSkZJRgABAQAASABIAAD/4QBMRX\"}}"

通用参数

使用HTTPS方式在服务端调用金融级实人认证时，您必须传入以下通用参数：

名称	类型	是否必选	描述
Format	String	是	返回值的类型。取值： • *JSON*（默认值） • *XML*
Version	String	是	API版本号，使用日期格式：YYYY-MM-DD。取值：*2017-03-31*。
AccessKeyId	String	是	阿里云颁发给用户的访问服务所用的密钥ID，请参见获取AccessKey。
Signature	String	是	签名结果串，关于签名的计算方法，请参见API签名机制。
SignatureMethod	String	是	签名方式，取值：*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。格式：YYYY-MM-DDThh:mm:ssZ。 例如，*2014-7-29T12:00:00Z*（为北京时间2014年7月29日的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。
Action	String	是	取值必须是*ExecuteRequest*。

名称	类型	是否必选	描述
Service	String	是	具体服务名，本服务该参数固定为*fin_face_verify*。
ServiceParameters	String	是	具体服务参数的JSON格式串。具体定义见下文各接口参数定义。

说明 通过调用不同服务实现不同功能时，实际是通过传入对应服务的ServiceParameters来实现的。

调用服务后，您会收到返回码。以下是返回码列表：

Code	Message
200	成功。
401	参数非法。
402	应用配置不存在。
403	主账号*******，无权调用fin_face_verify服务，或服务已到期、已到最大流量、未购买服务。
404	认证场景配置不存在。
406	无效的certifyId。
407	认证已失效。
408	开放认证单据已失效。
501	系统错误。
502	系统繁忙。
503	系统错误。

完整代码示例（Java语言）

添加POM依赖：

```
<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>19.0</version>
</dependency>
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.2</version>
</dependency>
```

调用接口发起认证和查询认证结果：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.SimpleTimeZone;
import java.util.UUID;
import com.google.common.io.BaseEncoding;
import org.apache.http.NameValuePair;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClientBuilder;
import org.apache.http.message.BasicNameValuePair;
public class SafTest {
    public static void main(String[] args) throws Throwable {
        //阿里云账号的appKey和appSecret。
        String appKey = "<accessKey>";
        String appSecret = "<accessSecret>";
        String safUrl = "https://saf.cn-shanghai.aliyuncs.com";
        //参数。
        Map<String, String> serviceParameters = new HashMap<String, String>();
        //发起认证请求。
        serviceParams.put("method",  "init");
        serviceParams.put("sceneId",  "100000****");
        serviceParams.put("outerOrderNo",  "e0c34a77f5ac40a5aa5e6ed20c35****");
        serviceParams.put("bizCode", "FACE");
        serviceParams.put("identityType", "CERT_INFO");
        serviceParams.put("certType", "IDENTITY_CARD");
        serviceParams.put("certNo", "1******************9");
        serviceParams.put("certName", "张先生");
        serviceParams.put("returnUrl", "https://www.aliyun.com");
        /*
        // 如需开启个人信息加密传输。
        serviceParameters.put("encryptType", "SM2");
        serviceParameters.put("certNo", "BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg==");
        serviceParameters.put("certName", "BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==");
        */
        /*
        // 查询认证结果。
        serviceParams.put("method",  "query");
        serviceParams.put("certifyId", "7eff3ad26a9c7b68c511b9f35eb1****");
        serviceParams.put("sceneId", "100000****");
```

```
        */
        Map<String, String> params = new HashMap<>();
        // 接口业务参数-固定为fin_face_verify。
        params.put("Service", "fin_face_verify");
        // 附加上业务详细参数，服务详细参数，具体见文档里的业务参数部分，业务参数整体转换为json格式。
        params.put("ServiceParameters", com.alibaba.fastjson.JSON.toJSONString(serviceParam
eters));
        doPOPRequest(appKey, appSecret, safUrl, "ExecuteRequest", "2017-03-31", params);
    }
    public static String doPOPRequest(String appKey, String appSecret, String url, String a
ction, String version, Map<String, String> parameters) throws Throwable {
        String encoding = "UTF-8";
        Map<String, String> paramsForPOP = new HashMap<>();
        // 公共参数-固定。
        paramsForPOP.put("AccessKeyId", appKey);
        paramsForPOP.put("Format", "JSON");// 固定。
        paramsForPOP.put("SignatureMethod", "HMAC-SHA1");// 固定。
        paramsForPOP.put("Timestamp", formatIso8601Date(new Date()));// 注意格式是：YYYY-MM-D
DThh:mm:ssZ。
        paramsForPOP.put("SignatureVersion", "1.0");// 固定。
        paramsForPOP.put("SignatureNonce", UUID.randomUUID().toString());// 自行生成一个随机
串，每次请求不可重复。
        paramsForPOP.put("Version", version);// 固定。
        paramsForPOP.put("Action", action);
        if (parameters != null) {
            parameters.remove("Signature");
            paramsForPOP.putAll(parameters);
        }
        String signature = computeSignature(paramsForPOP, appSecret, encoding);
        paramsForPOP.put("Signature", signature);// 把签名附加到post参数里。
        String result = httpPost(url, paramsForPOP, encoding);
        // 返回值。
        System.out.println(result);
        return result;
    }
    private static String httpPost(String url, Map<String, String> paramsForPOP, String enc
oding) throws IOException {
        CloseableHttpClient httpClient = HttpClientBuilder.create().build();
        HttpPost httpPost = new HttpPost(url);
        List<NameValuePair> urlParameters = new ArrayList<NameValuePair>();
        for (Entry<String, String> entry: paramsForPOP.entrySet()) {
            urlParameters.add(new BasicNameValuePair(entry.getKey(), entry.getValue()));
        }
        httpPost.setEntity(new UrlEncodedFormEntity(urlParameters, Consts.UTF_8));
        CloseableHttpResponse response = httpClient.execute(httpPost);
        BufferedReader rd = new BufferedReader(new InputStreamReader(response.getEntity().g
etContent()));
        StringBuffer result = new StringBuffer();
        String line = "";
        while ((line = rd.readLine()) != null) {
            result.append(line);
        }
        return result.toString();
    }
    private static String formatIso8601Date(Date date) {
```

```
    private static String formatISO8601Date(Date date) {
        SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
        df.setTimeZone(new SimpleTimeZone(0, "GMT"));// 注意使用GMT时间。
        return df.format(date);
    }
    public static String computeSignature(Map<String, String> parameters, String secret, St
ring encoding)
        throws Exception {
        // 将参数Key按字典顺序排序。
        String[] sortedKeys = parameters.keySet().toArray(new String[] {});
        Arrays.sort(sortedKeys);
        String separator = "&";
        boolean first = true;
        // 生成规范化请求字符串。
        StringBuilder sb = new StringBuilder();
        for (String key : sortedKeys) {
            if (first) {
                first = false;
            } else {
                sb.append("&");
            }
            sb.append(encode(key, encoding)).append("=").append(encode(parameters.get(key),
encoding));
        }
        // 生成用于计算签名的字符串toSign。
        StringBuilder toSign = new StringBuilder();
        toSign.append("POST").append(separator);
        toSign.append(encode("/", encoding)).append(separator);
        toSign.append(encode(sb.toString(), encoding));
        return getSignature("HmacSHA1", (secret + separator).getBytes(encoding), toSign.toS
tring().getBytes(encoding));
    }
    private static String encode(String value, String encoding) throws UnsupportedEncodingE
xception {
        if (value == null) { return null; }
        // 使用URLEncoder.encode编码后，将"+","*","%7E"做替换即满足 API规定的编码规范。
        return URLEncoder.encode(value, encoding).replace("+", "%20").replace("*", "%2A").r
eplace("%7E", "~");
    }
    public static String getSignature(String algorithm, byte[] secret, byte[] data) {
        try {
            javax.crypto.Mac mac = javax.crypto.Mac.getInstance(algorithm);
            mac.init(new javax.crypto.spec.SecretKeySpec(secret, algorithm));
            return BaseEncoding.base64().encode(mac.doFinal(data));
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

1.3.2.2. SDK调用接入

当您使用SDK进行开发时，无需拼接HTTPS请求或实现签名算法，开发更方便。因此我们强烈建议您通过SDK方式在服务端集成金融级实人认证。金融级实人认证提供了Java、Python、PHP、C#、Golang、Node.js和Ruby语言的SDK。

发起认证请求参数说明

发起认证请求时，传入以下参数：

名称	类型	是否必选	描述	示例值
method	String	是	发起认证请求的操作。 取值：*init*。	init
sceneId	String	是	认证场景ID，该ID在控制台创建认证场景后自动生成。	1000000006
outerOrderNo	String	是	客户服务端自定义的业务唯一标识，用于后续定位排查问题时使用。值最长为32位长度的字母和数字组合，请确保唯一。	e0c34a77f5ac40a5aa5e6ed20c353888
bizCode	String	是	认证场景码和商户发起认证的接入端有关： • 当商户在iOS或安卓平台发起认证时，认证场景码为 `FACE_SDK` 。 • 当商户在小程序中H5页面中发起认证时，认证场景码为 `FACE` 。	FACE_SDK
identityType	String	是	身份信息的参数类型，必须传入CERT_INFO。	CERT_INFO
certType	String	是	证件类型。当前支持身份证，必须传入IDENTITY_CARD。	IDENTITY_CARD
certNo	String	是	用户身份证件号码。	330103xxxxxxxxxxxx
certName	String	是	用户姓名。	张三
returnUrl	String	是	商户业务页面回调的目标地址。 • 如您不需要回调商户业务页面，您可以在此处传入空字符串。 • 当您采用端外唤起支付宝H5方式，且希望您的用户唤起支付宝完成认证后，能够跳回您的应用页面，您需要在此参数下传入您应用的Scheme。更多内容，请参见端外唤起支付宝H5页面。	https://www.aliyun.com

名称	类型	是否必选	描述	示例值
callbackUrl	String	否	认证结果的回调通知地址，必须以HTTPS开头。平台完成认证后会回调该地址，并自动添加**certifyId**、**passed**。	https://www.aliyun.com?certifyId=xxxx&passed=T
callbackToken	String	否	安全Token，由您自行生成，用于防重复、防篡改校验。如果设置了该值会在回调地址显示CallbackToken字段。	NMjvQanQgplBSaEI0sL86WnQplB
encryptType	String	否	加密类型。为空表示不加密。 如开启加密传输，需传入加密算法。目前仅支持SM2国密算法。 如需传入加密算法，需对certName和certNo进行加密，并传入加密后的密文。有关参数加密的更多信息，请参见参数加密说明。	SM2

说明 目前，仅Java语言提供有关开启加密传输的示例代码。更多信息，请参见Java SDK使用说明。

发起认证请求后，您会收到返回信息。返回参数列表如下：

名称	类型	是否必选	描述	示例值
certifyId	String	是	认证ID，刷脸认证的唯一标识。	*7eff3ad26a9c7b68c511b9f35eb1******
certifyUrl	String	是	认证流程入口Url。	*https://picker.antcloudauth.aliyuncs.com/gateway.do?...*

查询认证结果

查询认证结果时，传入以下参数：

名称	类型	是否必选	描述	示例值
method	String	是	查询人脸比对结果的操作。 取值：*query*。	*query*
certifyId	String	是	认证ID，需与发起认证请求时返回的certifyId保持一致。	*7eff3ad26a9c7b68c511b9f35eb1******
sceneId	String	是	认证场景ID，需与发起认证请求时的sceneId保持一致。	*100000*******

发起查询请求后，您会收到返回信息。返回参数列表如下：

名称	类型	是否必选	描述	示例值
passed	String	是	是否通过认证。取值： • *T*：认证通过。 • *F*：未通过认证。	T
identityInfo	String	否	认证的主体信息，一般的认证场景返回为空。	"IdentityInfo": "{"cert_type":"IDENTITY_CARD","cert_no":"1**********9","cert_name":"张先生"}"
materialInfo	String	否	认证主体附件信息，主要为图片类材料，一般的认证场景都是返回空。	"MaterialInfo": "{\"facial_picture_front\":{\"verify_score\":\"80.107902181204\",\"quality_score\":\"82\",\"FEATURE_FACE\":\"/9j/4AAQSkZJRgABAQAASABIAAD/4QBMRX\"}}"

Java SDK使用说明

1. 下载Java SDK源码：aliyun-openapijava-sdk。
2. 配置Maven依赖。

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-saf</artifactId>
    <version>1.0.2</version>
</dependency>
```

推荐依赖仲裁：

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
    <optional>true</optional>
    <version>4.5.0</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.60</version>
</dependency>
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.2</version>
</dependency>
```

说明

- 配置Java SDK环境的具体操作请参见阿里云SDK开发指南。
- 您可以通过站点1和站点2获得对应的Maven依赖。

代码示例如下：

```
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import com.alibaba.fastjson.JSON;
import com.aliyuncs.CommonRequest;
import com.aliyuncs.CommonResponse;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.exceptions.ServerException;
import com.aliyuncs.http.MethodType;
import com.aliyuncs.http.ProtocolType;
import com.aliyuncs.profile.DefaultProfile;
public class FaceVerifyTest {
    public static void main(String[] args) throws IOException {
        DefaultProfile profile = DefaultProfile.getProfile("cn-shanghai", "<your-access-key
-id>", "<your-access-key-secret>");
        IAcsClient client = new DefaultAcsClient(profile);
        CommonRequest request = new CommonRequest();
        request.setSysMethod(MethodType.POST);
        request.setSysDomain("saf.cn-shanghai.aliyuncs.com");
        request.setSysVersion("2017-03-31");
        request.setSysAction("ExecuteRequest");
        request.setSysProtocol(ProtocolType.HTTPS);
        // 业务详细参数。
        Map<String, Object> serviceParams = new HashMap<String, Object>();
        // 发起认证请求。
        serviceParams.put("method",  "init");
```

```
        serviceParams.put("sceneId",  "1000000006");
        serviceParams.put("outerOrderNo",  "e0c34a77f5ac40a5aa5e6ed20c353888");
        serviceParams.put("bizCode", "FACE");  // 当用户在iOS或安卓平台发起认证时，认证场景码是F
ACE_SDK；在小程序中，认证场景码则为FACE。
        serviceParams.put("identityType", "CERT_INFO");
        serviceParams.put("certType", "IDENTITY_CARD");
        serviceParams.put("certNo", "330103xxxxxxxxxxxx");
        serviceParams.put("certName", "张三");
        serviceParams.put("returnUrl", "https://www.aliyun.com");
        serviceParams.put("callbackUrl", "https://www.aliyun.com");
        serviceParams.put("callbackToken", "NMjvQanQgplBSaEI0sL86WnQplB");
        /*
        // 如需开启个人信息加密传输。
        serviceParams.put("encryptType", "SM2");
        serviceParams.put("certNo", "BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg==");
        serviceParams.put("certName", "BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==");
        */
        /*
        // 查询认证结果。
        serviceParams.put("method",  "query");
        serviceParams.put("certifyId", "7eff3ad26a9c7b68c511b9f35eb1a354");
        serviceParams.put("sceneId", "100000000");
        */
        request.putBodyParameter("ServiceParameters", JSON.toJSONString(serviceParams));
        // 固定值，Service = fin_face_verify。
        request.putBodyParameter("Service", "fin_face_verify");
        try {
            CommonResponse response = client.getCommonResponse(request);
            System.out.println(response.getData());
        } catch (ServerException e) {
            e.printStackTrace();
        } catch (ClientException e) {
            e.printStackTrace();
        }
    }
}
```

Python SDK使用说明

1. 下载Python SDK 源码。
2. 安装SDK核心库。
   - 如果您使用的是Python 2.x，执行以下命令，安装阿里云SDK核心库。

     ```
     pip install aliyun-python-sdk-core
     ```

   - 如果您使用的是Python 3.x，执行以下命令，安装阿里云SDK核心库。

     ```
     pip install aliyun-python-sdk-core-v3
     ```

3. 安装云产品SAF SDK。

   ```
   pip install aliyun-python-sdk-saf
   ```

示例代码如下：

```
from aliyunsdkcore import client
from aliyunsdksaf.request.v20170331 import ExecuteRequestRequest
clt = client.AcsClient('your-access-key-id','your-access-key-secret','cn-shanghai')
# 设置参数。
request = ExecuteRequestRequest.ExecuteRequestRequest()
request.set_accept_format('json')
# 产品Service请参考[公共参数]文档中的Service字段描述。
request.add_query_param('Service', '购买的产品Service')
request.add_query_param('ServiceParameters', '入参JSON字符串')
# 发起请求。
response = clt.do_action_with_exception(request)
print(response)
```

> **说明** Python SDK的环境准备、安装和使用，请参见阿里云SDK开发指南。

PHP SDK使用说明

下载PHP SDK。

代码示例：

```
<?php
require __DIR__ . '/vendor/autoload.php';
use AlibabaCloud\Client\AlibabaCloud;
use AlibabaCloud\Client\Exception\ClientException;
use AlibabaCloud\Client\Exception\ServerException;
// 设置一个全局客户端。
// 需要替换AccessKey ID和AccessKey Secret。
AlibabaCloud::accessKeyClient('access-key-id', 'access-key-secret')
            ->regionId('cn-shanghai')
            ->asDefaultClient();
// 下述参数依照实际使用的接口做调整，本演示仅针对认证初始化接口。
$serviceParams = array(
    "method"=>       "init",
    "sceneId"=>      "1000000006",
    "outerOrderNo"=> "e0c34a77f5ac40a5aa5e6ed20c353903",
    "bizCode"=>      "FACE",
    "identityType"=> "CERT_INFO",
    "certType"=>     "IDENTITY_CARD",
    "certNo"=>       "330103xxxxxxxxxxxx",
    "certName"=>     "张三",
    "returnUrl"=>    "https://www.aliyun.com",
    "callbackUrl"=>  "https://www.aliyun.com",
    "callbackToken"=>"NMjvQanQgplBSaEI0sL86WnQplB"
   );
try {
    // 下述参数无需调整。
    $result = AlibabaCloud::rpcRequest()
                            ->product('saf')
                            ->scheme('https')
                            ->version('2017-03-31')
                            ->action('ExecuteRequest')
                            ->method('POST')
                            ->host('saf.cn-shanghai.aliyuncs.com')
                            ->options([
                                          'query' => [
                                              'Service' => 'fin_face_verify',
                                              'ServiceParameters' => json_encode($servicePar
ams)
                                          ],
                                      ])
                            ->request();
    echo($result);
} catch (ClientException $exception) {
    echo $exception->getMessage() . PHP_EOL;
} catch (ServerException $exception) {
    echo $exception->getMessage() . PHP_EOL;
    echo $exception->getErrorCode() . PHP_EOL;
    echo $exception->getRequestId() . PHP_EOL;
    echo $exception->getErrorMessage() . PHP_EOL;
}
?>
```

说明 PHP SDK的环境准备、安装和使用请参见阿里云SDK开发指南。

C# SDK使用说明

下载 C# SDK 源码。

C# SDK的环境准备、安装和使用请参见阿里云SDK开发指南。

Golang SDK使用说明

下载Golang SDK。

示例代码：

```
package main
import (
    "encoding/json"
    "fmt"
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk"
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk/requests"
)
// ResponseContent返回体。
type ResponseContent struct {
    Data      map[string]string `json:"data"`
    RequestID string            `json:"requestId"`
    Code      string            `json:"code"`
    Message   string            `json:"message"`
}
func main() {
    // 需要替换AccessKey ID和AccessKey Secret。
    client, err := sdk.NewClientWithAccessKey(
        "cn-shanghai", "access-key-id", "access-key-secret")
    request := requests.NewCommonRequest()
    // 下述参数无需调整。
    request.Domain = "saf.cn-shanghai.aliyuncs.com"
    request.Version = "2017-03-31"
    request.ApiName = "ExecuteRequest"
    request.Scheme = "https"
    request.QueryParams["Service"] = "fin_face_verify"
    // 下述参数依照实际使用的接口做调整，本演示仅针对认证初始化接口。
    serviceParam := map[string]string{
        "method":       "init",
        "sceneId":      "1000000006",
        "outerOrderNo": "e0c34a77f5ac40a5aa5e6ed20c353900",
        "bizCode":      "FACE",
        "identityType": "CERT_INFO",
        "certType":     "IDENTITY_CARD",
        "certNo":       "330103xxxxxxxxxxxx",
        "certName":     "张三",
        "returnUrl":    "https://www.aliyun.com",
        "callbackUrl":  "https://www.aliyun.com",
        "callbackToken":"NMjvQanQgplBSaEI0sL86WnQplB"
    }
    serviceParamByte, err := json.Marshal(serviceParam)
    if err != nil {
        panic(err)
    }
    request.QueryParams["ServiceParameters"] = string(serviceParamByte)
    response, err := client.ProcessCommonRequest(request)
    if err != nil {
        panic(err)
    }
    respContent := ResponseContent{}
    err = json.Unmarshal(response.GetHttpContentBytes(), &respContent)
    fmt.Printf("response data is %#v\n", respContent.Data)
    fmt.Println("end")
}
```

说明 Golang SDK的环境准备、安装使用请参见阿里云SDK开发指南。

Node.js SDK使用说明

下载Node.js SDK 源码。

执行以下命令安装@alicloud/pop-core模块。命令中的--save会将模块写入应用的package.json文件中，作为依赖模块。

```
$ npm install @alicloud/pop-core --save
```

Node.js 代码示例：

```
const Core = require('@alicloud/pop-core');
// 需要替换AccessKey ID和AccessKey Secret。
var client = new Core({
  accessKeyId: 'access-key-id',
  accessKeySecret: 'access-key-secret',
  endpoint: 'https://saf.cn-shanghai.aliyuncs.com',
  apiVersion: '2017-03-31'
});
// 下述参数依照实际使用的接口做调整，本演示仅针对认证初始化接口。
var serviceParams = {
    "method":       "init",
    "sceneId":      "1000000006",
    "outerOrderNo": "e0c34a77f5ac40a5aa5e6ed20c353901",
    "bizCode":      "FACE",
    "identityType": "CERT_INFO",
    "certType":     "IDENTITY_CARD",
    "certNo":       "330103xxxxxxxxxxxx",
    "certName":     "张三",
    "returnUrl":    "https://www.aliyun.com",
    "callbackUrl":  "https://www.aliyun.com",
    "callbackToken":"NMjvQanQgplBSaEI0sL86WnQplB"
}
var params = {
  "RegionId": "cn-shanghai",
  "Service": "fin_face_verify",
  "ServiceParameters": JSON.stringify(serviceParams)
}
var requestOption = {
  method: 'POST'
};
client.request('ExecuteRequest', params, requestOption).then((result) => {
  console.log(JSON.stringify(result));
}, (ex) => {
  console.log(ex);
})
```

说明 Node.js SDK的环境准备、安装使用请参见阿里云SDK开发指南。

Ruby SDK使用说明

下载Ruby SDK 源码。

执行以下命令安装Alibaba Cloud Core SDK for Ruby。

```
$ gem install aliyunsdkcore
```

Ruby代码示例：

```
require 'aliyunsdkcore'
client = RPCClient.new(
  access_key_id:     'your-access-key-id',
  access_key_secret: 'your-access-key-secret',
  endpoint: 'https://saf.cn-shanghai.aliyuncs.com',
  api_version: '2017-03-31'
)
response = client.request(
  action: 'ExecuteRequest',
  params: {
    "RegionId": "cn-shanghai",
    # 产品Service请参考[公共参数]文档中的Service字段描述。
    "Service": "购买的产品Service",
    "ServiceParameters": "入参JSON字符串"
  },
  opts: {
    method: 'POST'
  }
)
print response
```

说明 Ruby SDK的环境准备、安装使用请参见阿里云SDK开发指南。

1.3.3. 支付宝H5页面接入

1.3.3.1. 端内接入支付宝H5页面

本文指导您在支付宝端内H5页面中集成金融级实人认证服务。

前提条件

客户端安装的支付宝版本必须是10.1.32+。

说明 您可以参考以下代码获取支付宝版本号，并做好startBizService版本的兼容处理。

```
const matchResult = window.navigator.userAgent.match(/AliApp\(AP\/([\d\.]+)\)/i);
const apVersion = (matchResult && matchResult[1]) || ''; // 如:10.1.58.00000170。
```

操作步骤

1. 调用API发起认证。
   i. 注入js库。

```
function ready(callback) {
    // 如果jsbridge已经注入则直接调用。
    if (window.AlipayJSBridge) {
        callback && callback();
    } else {
    // 如果没有注入则监听注入的事件。
        document.addEventListener('AlipayJSBridgeReady', callback, false);
    }
}
```

   ii. 发起认证服务。

   调用startBizService接口发起请求认证。

```
function startAPVerify(options, callback) {
    AlipayJSBridge.call('startBizService', {
        name: 'open-certify',
        param: JSON.stringify(options),
    }, callback);
}
```

   iii. 唤起认证流程。

   发起认证的信息包括certifyId和网关url。两者都通过服务端的相关接口取得。

```
ready(function() {
    // 需要确保在AlipayJSBridge ready之后才调。
    startAPVerify({
        certifyId: certifyId,
        url: url
        }, function(verifyResult) {
        // 认证结果回调触发，以下处理逻辑为示例代码，开发者可根据自身业务特性来自行处理。
        if (verifyResult.resultStatus === '9000') {
            // 验证成功，接入方在此处处理后续的业务逻辑。
            // ...
            return;
        }
    // 用户主动取消认证。
    if (verifyResult.resultStatus === '6001') {
        // 可做下toast弱提示。
        return;
    }
    const errorCode = verifyResult.result && verifyResult.result.errorCode;
    // 其他结果状态码判断和处理。
    });
});
```

2. 验证认证结果。

   回调函数带入的参数verifyResult：{ resultStatus：‘xx’, result： { } }说明如下：

参数名称	类型	描述
resultStatus	string	认证流程结果状态码，具体请参见下文ResultStatus定义。
result.certifyId	string	本次认证流水号certifyId。
result.errorCode	string	业务异常错误码。

说明 result的对象可能为null，API接入者代码逻辑需要做防御性处理，避免NPE异常。

resultStatus枚举取值如下：

状态码	描述
9000	认证通过。
6002	网络异常。
6001	用户取消了业务流程，主动退出。
4000	业务异常。

说明

- resultStatus为6001、6002时，result对象数据为空，接入者不需要获取result对象数据。
- resultStatus为9000时，由于前端数据是可以修改的，业务方需要去查询认证结果接口的最终状态。

当resultStatus为4000时，包含的部分错误码如下表格所示：

错误码	描述
UNKNOWN_ERROR	未知异常。
SYSTEM_ERROR	系统异常。
USER_IS_NOT_CERTIFY	用户未认证。

1.3.3.2. 端外唤起支付宝H5页面

本文指导您在支付宝端外H5页面中集成金融级实人认证服务。

前提条件

客户端安装的支付宝版本必须是10.1.32+。

说明 您可以参考以下代码获取支付宝版本号，并做好startBizService版本的兼容处理：

```
const matchResult = window.navigator.userAgent.match(/AliApp\(AP\/([\d\.]+)\)/i);
const apVersion = (matchResult && matchResult[1]) || ''; //例如10.1.58.00000170
```

操作步骤

1. 唤起支付宝客户端，在支付宝客户端中进行认证。
   i. 在接入流程的基础上，使用以下命令拼接支付宝唤端逻辑。

   ```
   const certifyUrl = alipays://platformapi/startapp?appId=20000067&url=encodeURIComponent(url);
   ```

   ii. 在H5页面中通过 `location.href=certifyUrl` 唤起支付宝并进入实名认证流程。认证结束后，支付宝认证结果页会回调前置条件入参中指定的回调地址。
2. 回跳原应用。
   i. 准备您的应用的Scheme。
      - 对于iOS平台，在*info.plist*文件中可以设置URL Schemes（URL types -> URL Schemes）。

        如果设置正确，在Safari浏览器中输入 `<您设置的 Scheme 名>://` ，打开App。
      - 对于Android平台，您可以在配置清单文件*AndroidManifest.xml*中，设置URL Scheme。

        如果设置正确，您可以在*.html*文件中写入超链接 `href="<您设置的 Scheme 名>://index"` 。然后，在Android手机上使用浏览器打开*.html*文件，单击链接，即可打开App。

   ii. 在服务端发起请求时，在returnUrl中传入您的应用的Scheme。

1.4. 支付宝小程序接入

1.4.1. 接入流程

支付宝小程序接入适用于业务使用方希望在自己开发的支付宝小程序中对用户进行线上认证的场景。

背景信息

如果您希望在支付宝小程序中认证用户身份，您可以接入金融级实人认证服务。

前置条件

您必须了解通过HTTP、HTTPS原生调用金融级实人认证服务接口时，构造请求签名的方法。更多签名机制的信息，请参见API签名机制。

操作步骤

1. 开通金融级实人认证服务。
2. 添加认证场景。
3. 接入支付宝小程序。

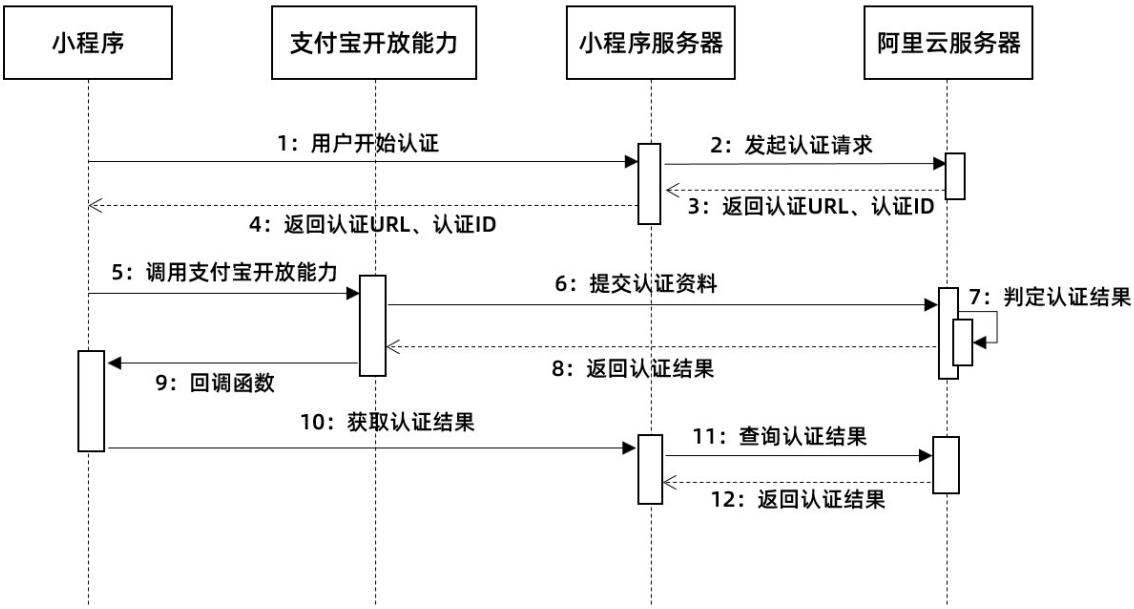
   在您完成上述步骤后，您可以进行支付宝小程序接入，接入步骤如下：

   i. 调用服务端初始化接口。

ii. 调用接入支付宝小程序。

iii. 调用服务端查询接口。

您可以查看下方调用时序图了解详细接入步骤。



系统调用时序说明：

序号	说明
1	用户开始认证。
2	小程序服务端向阿里云服务端发起认证请求。
3	阿里云服务端返回认证ID和认证URL给小程序服务端。
4	小程序服务端将认证ID和认证URL传递给支付宝小程序。
5	支付宝小程序调用支付宝开放能力。
6	支付宝开放能力将用户提交的认证资料传递给阿里云服务端。
7	阿里云服务端根据资料判定认证结果。
8	阿里云服务端将认证结果返回给支付宝开放能力。
9	支付宝开放能力通过回调函数指引小程序获取认证状态。
10	小程序向阿里云服务端查询认证状态。
11	阿里云服务端向小程序服务端向发起查询。
12	阿里云服务端将认证结果返回给小程序服务端。

在您了解接入步骤后，您可以通过以下方式接入支付宝小程序。

- 进行服务端接入，请参见HTTPS原生调用和SDK调用接入。

- 进行支付宝小程序刷脸接入，请参见支付宝小程序刷脸接入。

1.4.2. 服务端接入

1.4.2.1. HTTPS原生调用

本文主要介绍通过HTTPS原生调用方式在服务端集成金融级实人认证的方法。

调用方式

服务请求地址：`https://saf.cn-shanghai.aliyuncs.com`

协议：HTTPS

方式：POST

发起认证请求

发起认证请求时，传入通用参数，并在ServiceParameters的JSON字符串里传入以下字段：

名称	类型	是否必选	描述	示例值
method	String	是	发起认证请求的操作。 取值：*init*。	init
sceneId	String	是	认证场景ID，该ID在控制台创建认证场景后自动生成。	100000****
outerOrderNo	String	是	客户服务端自定义的业务唯一标识，用于后续定位排查问题时使用。值最长为32位长度的字母和数字组合，请确保唯一。	e0c34a77f5ac40a5aa5e6ed20c35****
bizCode	String	是	认证场景码和商户发起认证的接入端有关： • 当商户在iOS或安卓平台发起认证时，认证场景码为 `FACE_SDK` 。 • 当商户在小程序中H5页面中发起认证时，认证场景码为 `FACE` 。	FACE_SDK
identityType	String	是	身份信息的参数类型，必须传入CERT_INFO。	CERT_INFO
certType	String	是	证件类型。当前支持身份证，必须传入IDENTITY_CARD。	IDENTITY_CARD
certNo	String	是	用户身份证件号码。	1**********9
certName	String	是	用户姓名。	张先生

名称	类型	是否必选	描述	示例值
returnUrl	String	是	商户业务页面回调的目标地址。 ● 如您不需要回调商户业务页面，您可以在此处传入空字符串。 ● 当您采用端外唤起支付宝H5页面，且希望您的用户唤起支付宝完成认证后，能够跳回您的应用页面，您需要在此参数下传入您应用的Scheme。详细内容请参见回跳原应用。	https://www.aliyun.com
encryptType	String	否	加密类型。为空表示不加密。 如开启加密传输，需传入加密算法。目前仅支持SM2国密算法。 如需传入加密算法，需对certName和certNo进行加密，并传入加密后的密文。有关参数加密的更多信息，请参见参数加密说明。	SM2

发起认证请求后，您会收到返回信息。返回参数列表如下：

名称	类型	是否必选	描述	示例值
certifyId	String	是	认证ID，刷脸认证的唯一标识。	*7eff3ad26a9c7b68c511b9f35eb1******
certifyUrl	String	是	认证流程入口Url。	*https://picker.antcloudauth.aliyuncs.com/gateway.do?...*

返回示例如下：

```
{
    "code": 200,
    "requestId": "5CD69C04-8A6B-4CC3-A0C9-D18D5FEFA788",
    "data": {
        "certifyUrl": "https://picker.antcloudauth.aliyuncs.com/gateway.do?...",
        "certifyId": "7eff3ad26a9c7b68c511b9f35eb1****"
    },
    "message": "OK"
}
```

查询认证结果

查询认证结果时，传入通用参数，并在ServiceParameters的JSON字符串里传入以下字段：

名称	类型	是否必选	描述	示例值
method	String	是	查询人脸比对结果的操作。 取值：*query*。	*query*
certifyId	String	是	认证ID，需与发起认证请求时返回的certifyId保持一致。	*7eff3ad26a9c7b68c511b9f35eb1*** *
sceneId	String	是	认证场景ID，需与发起认证请求时的sceneId保持一致。	*100000**** *

发起查询请求后，您会收到返回信息。返回参数列表如下：

名称	类型	是否必选	描述	示例值
passed	String	是	是否通过认证。取值： ● *T*：认证通过。 ● *F*：未通过认证。	T
identityInfo	String	否	认证的主体信息，一般的认证场景返回为空。	"IdentityInfo": "{"cert_type":"IDENTITY_CARD","cert_no":"1**********9","cert_name":"张先生"}"
materialInfo	String	否	认证主体附件信息，主要为图片类材料，一般的认证场景都是返回空。	"MaterialInfo": "{\"facial_picture_front\": {\"verify_score\":\"80.107902181204\",\"quality_score\":\"82\",\"FEATURE_FACE\":\"/9j/4AAQSkZJRgABAQAASABIAAD/4QBMRX\"}}"

通用参数

使用HTTPS方式在服务端调用金融级实人认证时，您必须传入以下通用参数：

名称	类型	是否必选	描述
Format	String	是	返回值的类型。取值： ● *JSON*（默认值） ● *XML*
Version	String	是	API版本号，使用日期格式：YYYY-MM-DD。取值：*2017-03-31*。

名称	类型	是否必选	描述
AccessKeyId	String	是	阿里云颁发给用户的访问服务所用的密钥ID，请参见获取AccessKey。
Signature	String	是	签名结果串，关于签名的计算方法，请参见API签名机制。
SignatureMethod	String	是	签名方式，取值：*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。格式：YYYY-MM-DDThh:mm:ssZ。 例如，*2014-7-29T12:00:00Z*（为北京时间2014年7月29日的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。
Action	String	是	取值必须是*ExecuteRequest*。
Service	String	是	具体服务名，本服务该参数固定为*fin_face_verify*。
ServiceParameters	String	是	具体服务参数的JSON格式串。具体定义见下文各接口参数定义。

说明 通过调用不同服务实现不同功能时，实际是通过传入对应服务的ServiceParameters来实现的。

调用服务后，您会收到返回码。以下是返回码列表：

Code	Message
200	成功。
401	参数非法。
402	应用配置不存在。
403	主账号*******，无权调用fin_face_verify服务，或服务已到期、已到最大流量、未购买服务。
404	认证场景配置不存在。
406	无效的certifyId。
407	认证已失效。
408	开放认证单据已失效。
501	系统错误。

Code	Message
502	系统繁忙。
503	系统错误。

完整代码示例（Java语言）

添加POM依赖：

```
<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>19.0</version>
</dependency>
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.2</version>
</dependency>
```

调用接口发起认证和查询认证结果：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.SimpleTimeZone;
import java.util.UUID;
import com.google.common.io.BaseEncoding;
import org.apache.http.NameValuePair;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClientBuilder;
import org.apache.http.message.BasicNameValuePair;
public class SafTest {
    public static void main(String[] args) throws Throwable {
        //阿里云账号的appKey和appSecret。
        String appKey = "<accessKey>";
        String appSecret = "<accessSecret>";
        String safUrl = "https://saf.cn-shanghai.aliyuncs.com";
        //参数。
```

```
        Map<String, String> serviceParameters = new HashMap<String, String>();
        //发起认证请求。
        serviceParams.put("method",  "init");
        serviceParams.put("sceneId",  "100000****");
        serviceParams.put("outerOrderNo",  "e0c34a77f5ac40a5aa5e6ed20c35****");
        serviceParams.put("bizCode", "FACE");
        serviceParams.put("identityType", "CERT_INFO");
        serviceParams.put("certType", "IDENTITY_CARD");
        serviceParams.put("certNo", "1******************9");
        serviceParams.put("certName", "张先生");
        serviceParams.put("returnUrl", "https://www.aliyun.com");
        /*
        // 如需开启个人信息加密传输。
        serviceParameters.put("encryptType", "SM2");
        serviceParameters.put("certNo", "BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg==");
        serviceParameters.put("certName", "BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==");
        */
        /*
        // 查询认证结果。
        serviceParams.put("method",  "query");
        serviceParams.put("certifyId", "7eff3ad26a9c7b68c511b9f35eb1****");
        serviceParams.put("sceneId", "100000****");
        */
        Map<String, String> params = new HashMap<>();
        // 接口业务参数-固定为fin_face_verify。
        params.put("Service", "fin_face_verify");
        // 附加上业务详细参数，服务详细参数，具体见文档里的业务参数部分，业务参数整体转换为json格式。
        params.put("ServiceParameters", com.alibaba.fastjson.JSON.toJSONString(serviceParam
eters));
        doPOPRequest(appKey, appSecret, safUrl, "ExecuteRequest", "2017-03-31", params);
    }
    public static String doPOPRequest(String appKey, String appSecret, String url, String a
ction, String version, Map<String, String> parameters) throws Throwable {
        String encoding = "UTF-8";
        Map<String, String> paramsForPOP = new HashMap<>();
        // 公共参数-固定。
        paramsForPOP.put("AccessKeyId", appKey);
        paramsForPOP.put("Format", "JSON");// 固定。
        paramsForPOP.put("SignatureMethod", "HMAC-SHA1");// 固定。
        paramsForPOP.put("Timestamp", formatIso8601Date(new Date()));// 注意格式是：YYYY-MM-D
DThh:mm:ssZ。
        paramsForPOP.put("SignatureVersion", "1.0");// 固定。
        paramsForPOP.put("SignatureNonce", UUID.randomUUID().toString());// 自行生成一个随机
串，每次请求不可重复。
        paramsForPOP.put("Version", version);// 固定。
        paramsForPOP.put("Action", action);
        if (parameters != null) {
            parameters.remove("Signature");
            paramsForPOP.putAll(parameters);
        }
        String signature = computeSignature(paramsForPOP, appSecret, encoding);
        paramsForPOP.put("Signature", signature);// 把签名附加到post参数里。
        String result = httpPost(url, paramsForPOP, encoding);
        // 返回值。
        System.out.println(result);
```

```
        System.out.println(result);
        return result;
    }
    private static String httpPost(String url, Map<String, String> paramsForPOP, String enc
oding) throws IOException {
        CloseableHttpClient httpClient = HttpClientBuilder.create().build();
        HttpPost httpPost = new HttpPost(url);
        List<NameValuePair> urlParameters = new ArrayList<NameValuePair>();
        for (Entry<String, String> entry: paramsForPOP.entrySet()) {
            urlParameters.add(new BasicNameValuePair(entry.getKey(), entry.getValue()));
        }
        httpPost.setEntity(new UrlEncodedFormEntity(urlParameters, Consts.UTF_8));
        CloseableHttpResponse response = httpClient.execute(httpPost);
        BufferedReader rd = new BufferedReader(new InputStreamReader(response.getEntity().g
etContent()));
        StringBuffer result = new StringBuffer();
        String line = "";
        while ((line = rd.readLine()) != null) {
            result.append(line);
        }
        return result.toString();
    }
    private static String formatIso8601Date(Date date) {
        SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'");
        df.setTimeZone(new SimpleTimeZone(0, "GMT"));// 注意使用GMT时间。
        return df.format(date);
    }
    public static String computeSignature(Map<String, String> parameters, String secret, St
ring encoding)
        throws Exception {
        // 将参数Key按字典顺序排序。
        String[] sortedKeys = parameters.keySet().toArray(new String[] {});
        Arrays.sort(sortedKeys);
        String separator = "&";
        boolean first = true;
        // 生成规范化请求字符串。
        StringBuilder sb = new StringBuilder();
        for (String key : sortedKeys) {
            if (first) {
                first = false;
            } else {
                sb.append("&");
            }
            sb.append(encode(key, encoding)).append("=").append(encode(parameters.get(key),
encoding));
        }
        // 生成用于计算签名的字符串toSign。
        StringBuilder toSign = new StringBuilder();
        toSign.append("POST").append(separator);
        toSign.append(encode("/", encoding)).append(separator);
        toSign.append(encode(sb.toString(), encoding));
        return getSignature("HmacSHA1", (secret + separator).getBytes(encoding), toSign.toS
tring().getBytes(encoding));
    }
    private static String encode(String value, String encoding) throws UnsupportedEncodingE
```

```
xception {
        if (value == null) { return null; }
        // 使用URLEncoder.encode编码后，将"+","*","%7E"做替换即满足 API规定的编码规范。
        return URLEncoder.encode(value, encoding).replace("+", "%20").replace("*", "%2A").r
eplace("%7E", "~");
    }
    public static String getSignature(String algorithm, byte[] secret, byte[] data) {
        try {
            javax.crypto.Mac mac = javax.crypto.Mac.getInstance(algorithm);
            mac.init(new javax.crypto.spec.SecretKeySpec(secret, algorithm));
            return BaseEncoding.base64().encode(mac.doFinal(data));
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

1.4.2.2. SDK调用接入

当您使用SDK进行开发时，无需拼接HTTPS请求或实现签名算法，开发更方便。因此我们强烈建议您通过SDK方式在服务端集成金融级实人认证。金融级实人认证提供了Java、Python、PHP、C#、Golang、Node.js和Ruby语言的SDK。

发起认证请求参数说明

发起认证请求时，传入以下参数：

名称	类型	是否必选	描述	示例值
method	String	是	发起认证请求的操作。 取值：*init*。	init
sceneId	String	是	认证场景ID，该ID在控制台创建认证场景后自动生成。	1000000006
outerOrderNo	String	是	客户服务端自定义的业务唯一标识，用于后续定位排查问题时使用。值最长为32位长度的字母和数字组合，请确保唯一。	e0c34a77f5ac40a5aa5e6ed20c353888
bizCode	String	是	认证场景码和商户发起认证的接入端有关： • 当商户在iOS或安卓平台发起认证时，认证场景码为 `FACE_SDK` 。 • 当商户在小程序中H5页面中发起认证时，认证场景码为 `FACE` 。	FACE_SDK
identityType	String	是	身份信息的参数类型，必须传入CERT_INFO。	CERT_INFO

名称	类型	是否必选	描述	示例值
certType	String	是	证件类型。当前支持身份证，必须传入IDENTITY_CARD。	IDENTITY_CARD
certNo	String	是	用户身份证件号码。	330103xxxxxxxxxxxx
certName	String	是	用户姓名。	张三
returnUrl	String	是	商户业务页面回调的目标地址。 ● 如您不需要回调商户业务页面，您可以在此处传入空字符串。 ● 当您采用端外唤起支付宝H5方式，且希望您的用户唤起支付宝完成认证后，能够跳回您的应用页面，您需要在此参数下传入您应用的Scheme。更多内容，请参见端外唤起支付宝H5页面。	https://www.aliyun.com
callbackUrl	String	否	认证结果的回调通知地址，必须以HTTPS开头。平台完成认证后会回调该地址，并自动添加**certifyId**、**passed**。	https://www.aliyun.com?certifyId=xxxx&passed=T
callbackToken	String	否	安全Token，由您自行生成，用于防重复、防篡改校验。如果设置了该值会在回调地址显示CallbackToken字段。	NMjvQanQgplBSaEI0sL86WnQplB
encryptType	String	否	加密类型。为空表示不加密。 如开启加密传输，需传入加密算法。目前仅支持SM2国密算法。 如需传入加密算法，需对certName和certNo进行加密，并传入加密后的密文。有关参数加密的更多信息，请参见参数加密说明。	SM2

说明 目前仅Java语言提供有关开启加密传输的示例代码。更多信息，请参见Java SDK使用说明。

发起认证请求后，您会收到返回信息。返回参数列表如下：

名称	类型	是否必选	描述	示例值
certifyId	String	是	认证ID，刷脸认证的唯一标识。	*7eff3ad26a9c7b68c511b9f35eb1******
certifyUrl	String	是	认证流程入口Url。	*https://picker.antcloudauth.aliyuncs.com/gateway.do?...*

查询认证结果

查询认证结果时，传入以下参数：

名称	类型	是否必选	描述	示例值
method	String	是	查询人脸比对结果的操作。 取值：*query*。	*query*
certifyId	String	是	认证ID，需与发起认证请求时返回的certifyId保持一致。	*7eff3ad26a9c7b68c511b9f35eb1*** *
sceneId	String	是	认证场景ID，需与发起认证请求时的sceneId保持一致。	*100000*****

发起查询请求后，您会收到返回信息。返回参数列表如下：

名称	类型	是否必选	描述	示例值
passed	String	是	是否通过认证。取值： • *T*：认证通过。 • *F*：未通过认证。	T
identityInfo	String	否	认证的主体信息，一般的认证场景返回为空。	"IdentityInfo": "{"cert_type":"IDENTITY_CARD","cert_no":"1**********9","cert_name":"张先生"}"
materialInfo	String	否	认证主体附件信息，主要为图片类材料，一般的认证场景都是返回空。	"MaterialInfo": "{\"facial_picture_front\":{\"verify_score\":\"80.107902181204\",\"quality_score\":\"82\",\"FEATURE_FACE\":\"/9j/4AAQSkZJRgABAQAASABIAAD/4QBMRX\"}}"

Java SDK使用说明

1. 下载Java SDK源码：aliyun-openapijava-sdk。
2. 配置Maven依赖。

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-saf</artifactId>
    <version>1.0.2</version>
</dependency>
```

推荐依赖仲裁：

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
    <optional>true</optional>
    <version>4.5.0</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.60</version>
</dependency>
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.2</version>
</dependency>
```

说明

- 配置Java SDK环境的具体操作请参见阿里云SDK开发指南。
- 您可以通过站点1和站点2获得对应的Maven依赖。

代码示例如下：

```
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import com.alibaba.fastjson.JSON;
import com.aliyuncs.CommonRequest;
import com.aliyuncs.CommonResponse;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.exceptions.ServerException;
import com.aliyuncs.http.MethodType;
import com.aliyuncs.http.ProtocolType;
import com.aliyuncs.profile.DefaultProfile;
public class FaceVerifyTest {
    public static void main(String[] args) throws IOException {
        DefaultProfile profile = DefaultProfile.getProfile("cn-shanghai", "<your-access-key
-id>", "<your-access-key-secret>");
        IAcsClient client = new DefaultAcsClient(profile);
        CommonRequest request = new CommonRequest();
        request.setSysMethod(MethodType.POST);
```

```
        request.setSysDomain("saf.cn-shanghai.aliyuncs.com");
        request.setSysVersion("2017-03-31");
        request.setSysAction("ExecuteRequest");
        request.setSysProtocol(ProtocolType.HTTPS);
        // 业务详细参数。
        Map<String, Object> serviceParams = new HashMap<String, Object>();
        // 发起认证请求。
        serviceParams.put("method",  "init");
        serviceParams.put("sceneId",  "1000000006");
        serviceParams.put("outerOrderNo",  "e0c34a77f5ac40a5aa5e6ed20c353888");
        serviceParams.put("bizCode", "FACE");  // 当用户在iOS或安卓平台发起认证时，认证场景码是F
ACE_SDK；在小程序中，认证场景码则为FACE。
        serviceParams.put("identityType", "CERT_INFO");
        serviceParams.put("certType", "IDENTITY_CARD");
        serviceParams.put("certNo", "330103xxxxxxxxxxxx");
        serviceParams.put("certName", "张三");
        serviceParams.put("returnUrl", "https://www.aliyun.com");
        serviceParams.put("callbackUrl", "https://www.aliyun.com");
        serviceParams.put("callbackToken", "NMjvQanQgplBSaEI0sL86WnQplB");
        /*
        // 如需开启个人信息加密传输。
        serviceParams.put("encryptType",  "SM2");
        serviceParams.put("certNo", "BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg==");
        serviceParams.put("certName", "BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==");
        */
        /*
        // 查询认证结果。
        serviceParams.put("method",  "query");
        serviceParams.put("certifyId", "7eff3ad26a9c7b68c511b9f35eb1a354");
        serviceParams.put("sceneId", "1000000006");
        */
        request.putBodyParameter("ServiceParameters", JSON.toJSONString(serviceParams));
        // 固定值，Service = fin_face_verify。
        request.putBodyParameter("Service", "fin_face_verify");
        try {
            CommonResponse response = client.getCommonResponse(request);
            System.out.println(response.getData());
        } catch (ServerException e) {
            e.printStackTrace();
        } catch (ClientException e) {
            e.printStackTrace();
        }
    }
}
```

Python SDK使用说明

1. 下载Python SDK 源码。
2. 安装SDK核心库。
   - 如果您使用的是Python 2.x，执行以下命令，安装阿里云SDK核心库。

```
pip install aliyun-python-sdk-core
```

- 如果您使用的是Python 3.x，执行以下命令，安装阿里云SDK核心库。

```
pip install aliyun-python-sdk-core-v3
```

3. 安装云产品SAF SDK。

```
pip install aliyun-python-sdk-saf
```

示例代码如下：

```
from aliyunsdkcore import client
from aliyunsdksaf.request.v20170331 import ExecuteRequestRequest
clt = client.AcsClient('your-access-key-id','your-access-key-secret','cn-shanghai')
# 设置参数。
request = ExecuteRequestRequest.ExecuteRequestRequest()
request.set_accept_format('json')
# 产品Service请参考[公共参数]文档中的Service字段描述。
request.add_query_param('Service', '购买的产品Service')
request.add_query_param('ServiceParameters', '入参JSON字符串')
# 发起请求。
response = clt.do_action_with_exception(request)
print(response)
```

> **说明** Python SDK的环境准备、安装和使用，请参见阿里云SDK开发指南。

PHP SDK使用说明

下载PHP SDK。

代码示例：

```
<?php
require __DIR__ . '/vendor/autoload.php';
use AlibabaCloud\Client\AlibabaCloud;
use AlibabaCloud\Client\Exception\ClientException;
use AlibabaCloud\Client\Exception\ServerException;
// 设置一个全局客户端。
// 需要替换AccessKey ID和AccessKey Secret。
AlibabaCloud::accessKeyClient('<access-key-id>', '<access-key-secret>')
            ->regionId('cn-shanghai')
            ->asDefaultClient();
// 下述参数依照实际使用的接口做调整，本演示仅针对认证初始化接口。
$serviceParams = array(
    "method"=>       "init",
    "sceneId"=>      "1000000006",
    "outerOrderNo"=> "e0c34a77f5ac40a5aa5e6ed20c353903",
    "bizCode"=>      "FACE",
    "identityType"=> "CERT_INFO",
    "certType"=>     "IDENTITY_CARD",
    "certNo"=>       "330103xxxxxxxxxxxx",
    "certName"=>     "张三",
    "returnUrl"=>    "https://www.aliyun.com",
    "callbackUrl"=>  "https://www.aliyun.com",
    "callbackToken"=>"NMjvQanQgplBSaEI0sL86WnQplB"
   );
try {
    // 下述参数无需调整。
    $result = AlibabaCloud::rpcRequest()
                            ->product('saf')
                            ->scheme('https')
                            ->version('2017-03-31')
                            ->action('ExecuteRequest')
                            ->method('POST')
                            ->host('saf.cn-shanghai.aliyuncs.com')
                            ->options([
                                          'query' => [
                                              'Service' => 'fin_face_verify',
                                              'ServiceParameters' => json_encode($servicePar
ams)
                                          ],
                                       ])
                            ->request();
    echo($result);
} catch (ClientException $exception) {
    echo $exception->getMessage() . PHP_EOL;
} catch (ServerException $exception) {
    echo $exception->getMessage() . PHP_EOL;
    echo $exception->getErrorCode() . PHP_EOL;
    echo $exception->getRequestId() . PHP_EOL;
    echo $exception->getErrorMessage() . PHP_EOL;
}
?>
```

说明 PHP SDK的环境准备、安装和使用请参见阿里云SDK开发指南。

C# SDK使用说明

下载 C# SDK 源码。

C# SDK的环境准备、安装和使用请参见阿里云SDK开发指南。

Golang SDK使用说明

下载Golang SDK。

示例代码：

```
package main
import (
    "encoding/json"
    "fmt"
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk"
    "github.com/aliyun/alibaba-cloud-sdk-go/sdk/requests"
)
// ResponseContent返回体。
type ResponseContent struct {
    Data      map[string]string `json:"data"`
    RequestID string            `json:"requestId"`
    Code      string            `json:"code"`
    Message   string            `json:"message"`
}
func main() {
    // 需要替换AccessKey ID和AccessKey Secret。
    client, err := sdk.NewClientWithAccessKey(
        "cn-shanghai", "access-key-id", "access-key-secret")
    request := requests.NewCommonRequest()
    // 下述参数无需调整。
    request.Domain = "saf.cn-shanghai.aliyuncs.com"
    request.Version = "2017-03-31"
    request.ApiName = "ExecuteRequest"
    request.Scheme = "https"
    request.QueryParams["Service"] = "fin_face_verify"
    // 下述参数依照实际使用的接口做调整，本演示仅针对认证初始化接口。
    serviceParam := map[string]string{
        "method":       "init",
        "sceneId":      "1000000006",
        "outerOrderNo": "e0c34a77f5ac40a5aa5e6ed20c353900",
        "bizCode":      "FACE",
        "identityType": "CERT_INFO",
        "certType":     "IDENTITY_CARD",
        "certNo":       "330103xxxxxxxxxxxx",
        "certName":     "张三",
        "returnUrl":    "https://www.aliyun.com",
        "callbackUrl":  "https://www.aliyun.com",
        "callbackToken":"NMjvQanQgplBSaEI0sL86WnQplB"
    }
    serviceParamByte, err := json.Marshal(serviceParam)
    if err != nil {
        panic(err)
    }
    request.QueryParams["ServiceParameters"] = string(serviceParamByte)
    response, err := client.ProcessCommonRequest(request)
    if err != nil {
        panic(err)
    }
    respContent := ResponseContent{}
    err = json.Unmarshal(response.GetHttpContentBytes(), &respContent)
    fmt.Printf("response data is %#v\n", respContent.Data)
    fmt.Println("end")
}
```

说明 Golang SDK的环境准备、安装使用请参见阿里云SDK开发指南。

Node.js SDK使用说明

下载Node.js SDK 源码。

执行以下命令安装@alicloud/pop-core模块。命令中的--save会将模块写入应用的package.json文件中，作为依赖模块。

```
$ npm install @alicloud/pop-core --save
```

Node.js 代码示例：

```
const Core = require('@alicloud/pop-core');
// 需要替换AccessKey ID和AccessKey Secret。
var client = new Core({
  accessKeyId: 'access-key-id',
  accessKeySecret: 'access-key-secret',
  endpoint: 'https://saf.cn-shanghai.aliyuncs.com',
  apiVersion: '2017-03-31'
});
// 下述参数依照实际使用的接口做调整，本演示仅针对认证初始化接口。
var serviceParams = {
    "method":       "init",
    "sceneId":      "1000000006",
    "outerOrderNo": "e0c34a77f5ac40a5aa5e6ed20c353901",
    "bizCode":      "FACE",
    "identityType": "CERT_INFO",
    "certType":     "IDENTITY_CARD",
    "certNo":       "330103xxxxxxxxxxxx",
    "certName":     "张三",
    "returnUrl":    "https://www.aliyun.com",
    "callbackUrl":  "https://www.aliyun.com",
    "callbackToken":"NMjvQanQgplBSaEI0sL86WnQplB"
}
var params = {
  "RegionId": "cn-shanghai",
  "Service": "fin_face_verify",
  "ServiceParameters": JSON.stringify(serviceParams)
}
var requestOption = {
  method: 'POST'
};
client.request('ExecuteRequest', params, requestOption).then((result) => {
  console.log(JSON.stringify(result));
}, (ex) => {
  console.log(ex);
})
```

说明 Node.js SDK的环境准备、安装使用请参见阿里云SDK开发指南。

Ruby SDK使用说明

下载Ruby SDK 源码。

执行以下命令安装Alibaba Cloud Core SDK for Ruby：

```
$ gem install aliyunsdkcore
```

Ruby代码示例：

```
require 'aliyunsdkcore'
client = RPCClient.new(
  access_key_id:     'your-access-key-id',
  access_key_secret: 'your-access-key-secret',
  endpoint: 'https://saf.cn-shanghai.aliyuncs.com',
  api_version: '2017-03-31'
)
response = client.request(
  action: 'ExecuteRequest',
  params: {
    "RegionId": "cn-shanghai",
    # 产品Service请参考[公共参数]文档中的Service字段描述。
    "Service": "购买的产品Service",
    "ServiceParameters": "入参JSON字符串"
  },
  opts: {
    method: 'POST'
  }
)
print response
```

> **说明** Ruby SDK的环境准备、安装使用请参见阿里云SDK开发指南。

1.4.3. 支付宝小程序刷脸接入

本文指导您通过集成支付宝小程序唤起实人认证服务。

操作步骤

1. 调用 API 发起认证。
   i. 发起认证服务。

      调用 **startBizService** 接口请求认证。

```
function startAPVerify(options, callback) {
    my.call('startBizService', {
        name: 'open-certify',
        param: JSON.stringify(options),
    }, callback);
}
```

ii. 唤起认证流程。

发起认证包含的信息包括 certifyId 和网关 url。两者都通过服务端的相关接口取得。

```
startAPVerify({
    certifyId: certifyId, url: url }, function (verifyResult) {
    // 认证结果回调触发，以下处理逻辑为示例代码，开发者可根据自身业务特性来自行处理
    if (verifyResult.resultStatus === '9000') {
        // 验证成功，接入方在此处处理后续的业务逻辑
        // ...
        return;
    }
    // 用户主动取消认证
    if (verifyResult.resultStatus === '6001') {
        // 可做下 toast 弱提示
        return;
    }
    const errorCode = verifyResult.result && verifyResult.result.errorCode;
    // 其他结果状态码判断和处理 ...
    }
});
```

2. 验证认证结果。

回调函数带入的参数verifyResult：{ resultStatus：'xx', result： { } }说明如下：

参数名称	类型	描述
resultStatus	string	认证流程结果状态码，具体请参见下文ResultStatus定义。
result.certifyId	string	本次认证流水号certifyId。
result.errorCode	string	业务异常错误码。

说明 result的对象可能为null，API接入者代码逻辑需要做防御性处理，避免NPE异常。

resultStatus枚举取值如下：

状态码	描述
9000	认证通过。
6002	网络异常。
6001	用户取消了业务流程，主动退出。
4000	业务异常。

> 说明
> - resultStatus为6001、6002时，result对象数据为空，接入者不需要获取result对象数据。
> - resultStatus为9000时，由于前端数据是可以修改的，业务方需要去查询认证结果接口的最终状态。

当resultStatus为4000时，包含的部分错误码如下表格所示：

错误码	描述
UNKNOWN_ERROR	未知异常。
SYSTEM_ERROR	系统异常。
USER_IS_NOT_CERTIFY	用户未认证。

1.5. 纯服务端（API）接入

1.5.1. 接入流程

本文介绍了纯服务端接入的详细操作。

前置条件

您必须了解通过HTTP、HTTPS原生调用金融级实人认证服务接口时，构造请求签名的方法。更多签名机制的信息，请参见API签名机制。

操作步骤

1. 开通金融级实人认证服务。
2. 添加认证场景。
3. 授权金融级实人认证访问OSS存储空间。
4. 接入服务端（图片）或者服务端（视频）。
   - 进行服务端接入，请参见如下文档：
     - 照片实人认证
     - 视频实人认证
   - 在应用客户端集成SDK，请参见Java SDK调用示例。

1.5.2. 照片实人认证

调用ContrastFaceVerify接口通过传入图片进行实人认证。

传入图片格式要求

当您在进行实人认证的时候，请传入满足以下所有条件的图片：

- 认证主体的近期照片，人脸需要完整清晰无遮挡、表情自然且正对摄像头。
- 照片清晰、曝光正常，脸部无过黑、过亮、光晕等情况、角度无太大偏差。

- 分辨率不超过1920*1080（高*宽），至少为640*480，推荐短边缩放到720像素，压缩率大于0.9。照片高大于宽，如果传入的照片宽大于高，可能会影响检测效果。
- 照片大小需要小于1 MB。
- 支持人像顺时针旋转90度、180度和270度，多人脸的情况会选取最大人脸。

示例：

发起认证请求

接口名：**ContrastFaceVerify**。

服务地址：cloudauth.aliyuncs.com。

请求方法：HTTPS POST和GET。

接口说明：通过传入图片进行实人认证。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	认证场景ID。	100000****
OuterOrderNo	String	是	客户服务端自定义的业务唯一标识，用于后续定位排查问题使用。值最长为32位长度的字母数字组合，请确保唯一。	e0c34a77f5ac40a5aa5e6ed20c35****
ProductCode	String	是	认证方案。 唯一取值：**ID_MIN**。	ID_MIN

名称	类型	是否必选	描述	示例值
Model	String	否	活体检测类型。取值： • NO_LIVENESS：关闭活体检测。 • FRONT_CAMERA_LIVENESS：移动设备前置摄像头采集的人脸照片活体检测。 • REAR_CAMERA_LIVENESS：其他场景采集的人脸照片活体检测。	FRONT_CAMERA_LIVENESS
CertType	String	是	证件类型。 当前仅支持身份证。 唯一取值：IDENTITY_CARD。	IDENTITY_CARD
CertName	String	是	真实姓名。	张三
CertNo	String	是	证件号码。	1**********9
DeviceToken	String	否	当前用户设备的Token，用于风险识别。	McozS1ZWRcRZStlERcZZo_QOytx5jcgZoZJEoRLOxxxxxxx
Mobile	String	否	用户手机号码。	1390000****
Ip	String	否	用户IP。	114.100.XX.XX
UserId	String	否	客户业务自定义的用户ID。	148562088256****

名称	类型	是否必选	描述	示例值
FaceContrastPicture	String	否 说明 您需要在下述四种传入人脸照片方式中，选择其中一种： • FaceContrastPicture • FaceContrastPictureUrl • CertifyId • OSS 如果您选择OSS方式传入人脸照片，您必须传入OssBucketName和OssObjectName参数。OSS请求参数格式仅支持开通实人认证服务时授权的OSS生成的URL、Bucket和Object。	照片Base64编码。如果您选择FaceContrastPicture（照片Base64编码）方式传入人脸照片，请注意检查照片大小，不要传入过大的照片；您可以通过OSS方式上传较大的人脸照片。	/9j/4AAQSkZJRgABAQAASxxxxxxx
FaceContrastPictureUrl	String		人像地址，公网可访问的HTTP、HTTPS链接。	https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg
CertifyId	String		之前实人认证通过的CertifyId，认证时的照片作为比对照片。您需要确保传入CertifyId关联的图片正常存储在授权的OSS Bucket中。	0bfa7c493f850e5178b9f8613634****
OssBucketName	String		已授权OSS空间的Bucket名称。	cn-shanghai-aliyun-cloudauth-xxxxx
OssObjectName	String		已授权OSS空间的Object名称。	verify/xxxxx/xxxxxx.jpeg
Crop	String	否	是否允许裁剪人脸图片，默认不允许。 • T：表示允许剪裁。 • F：表示不允许剪裁。 说明 若您请求的图片并非来自标准活体检测SDK采集，建议您允许裁剪人脸图片。 开启该功能后，将会对请求的图片先进行人脸裁剪与矫正，完成后会发起请求服务。	T

名称	类型	是否必选	描述	示例值
EncryptType	String	否	加密类型。为空表示不加密。 如开启加密传输，需传入加密算法。目前仅支持SM2国密算法。 如需传入加密算法，需对CertName和CertNo进行加密，并传入加密后的密文。有关参数加密的更多信息，请参见参数加密说明。	SM2

返回参数

名称	类型	是否必选	描述	示例值
RequestId	String	是	请求ID。	130A2C10-B9EE-4D84-88E3-5384FF039795
Message	String	否	请求的响应信息。	success
Code	String	是	返回码，200指接口响应成功。详细认证结果判断，请参见下文**ResultObject.SubCode错误码说明**。	200
ResultObject.Passed	String	是	认证结果。取值： • **T**：认证通过。 • **F**：认证未通过。 **说明** 判断认证结果请以ResultObject.Passed字段为准。	T
ResultObject.SubCode	String	否	认证结果描述，请参见ResultObject.SubCode错误码说明。 **说明** 判断认证结果请以ResultObject.Passed字段为准。	200
ResultObject.IdentityInfo	String	否	认证的主体信息，目前支持的认证场景返回为空。	null

名称	类型	是否必选	描述	示例值
ResultObject.MaterialInfo	String	否	认证主体附件信息，主要为图片类材料，使用JSON结构体转换的字符串表示，请参见ResultObject.MaterialInfo的JSON格式示例。	{"faceAttack":"F","faceOcclusion":"F","facialPictureFront":{"faceAttackScore":0.00008597839769208804,"qualityScore":88.36154937744 14,"verifyScore":50.28594166529785}}

ResultObject.MaterialInfo的JSON格式示例：

```
{
    // 是否为攻击：攻击为T，非攻击为F。
    "faceAttack": "F",
    // 是否有脸部遮挡：有脸部遮挡为T，否则为F。
    "faceOcclusion": "F",
    "facialPictureFront": {
        // 人脸攻击分。
        "faceAttackScore": 0.00008597839769208804,
        // 活体人脸质量分数。
        "qualityScore": 88.3615493774414,
        // 人脸和公安比对分数，阈值可参考下表详细说明。
        "verifyScore": 50.28594166529785
    }
}
```

返回Code和Message说明

Code	Message	描述
200	success	请求响应成功。
400	参数不能为空	参数不能为空。
401	参数非法	非法参数。传入的姓名、身份证号码长度必须符合国家标准且不得包含英文字母等特殊字符。
402	应用配置不存在	应用配置不存在。
404	认证场景配置不存在	认证场景配置不存在，请先在控制台上创建认证场景。具体操作，请参见添加认证场景。

Code	Message	描述
410	未开通服务	未开通OSS产品或未完成OSS读写授权，请登录控制台完成授权。
411	RAM无权限	需要给子账号授予AliyunAntCloudAuthFullAccess的操作权限。
412	欠费中	金融级实人认证或OSS存在欠费，请充值后操作。
417	无法使用刷脸服务	当前身份信息比对源不可用。若信息正确，建议人工审核。
418	刷脸失败次数过多	当天刷脸认证次数过多，请明天再试。
419	传入图片不可用	图片无法下载、图片内容为空、图片分辨率不符合要求或提取不到人脸特征，建议更换图片。
420	数据重复	您通过多种方式传入了认证材料。目前支持以下四种方式中的一种传入认证材料： • FaceContrastPicture • FaceContrastPictureUrl • Oss • CertifyId
421	传入图片过大	图片超过了1 MB，建议压缩图片或更换图片上传方式。
422	下载图片超时	图片下载超过了3秒，请排查网络后重新操作。
423	状态错误	传入的CertifyId认证状态需要为T（认证通过），您也可以更换其他方式传入图片。
500	系统错误	系统内部错误，请提交工单进行反馈，由阿里云工程师协助排查。

ResultObject.SubCode错误码说明

错误码	认证描述文案	是否计费	可能原因和运营建议
200	认证通过	是	无。
201	姓名和身份证不一致	是	可能是用户的信息有误或用户的信息为假信息，建议用户确认后重新操作。若同一身份信息重复发起认证，服务端初始化将会返回417错误。
202	查询不到身份信息	是	可能是用户户口迁移、用户是军人等特殊状态导致，建议预留人工审核入口，进行人工审核。若同一身份信息重复发起认证，服务端初始化将会返回417错误。
203	查询不到照片或照片不可用	是	可能是公安库数据问题导致，建议预留人工审核入口，进行人工审核。若同一身份信息重复发起认证，服务端初始化将会返回417错误。

错误码	认证描述文案	是否计费	可能原因和运营建议
204	人脸比对不一致	是	可能不是同一人或活体照片质量较低，建议根据业务情况分层处理，若为同一人可重复操作。
205	活体检测存在风险	是	可能存在攻击风险，建议人工审核分层处理，若为真人可重复操作。
206	业务策略限制	是	个性化业务安全策略限制，如有疑问可咨询客服。
209	权威比对源异常	是	可能存在权威比对源异常，无法比对。
210	认证通过	是	通过产品安全策略综合判断，建议认证通过。

verifyScore阈值说明

千分之一误识率	万分之五误识率	万分之一误识率	十万分之五误识率	十万分之一误识率
70	71.5	75	76.5	80

说明 如果您有个性化需求，您可以根据业务情况，参考返回的比对分和阈值，自定义认证结果。

1.5.3. 视频实人认证

调用ContrastFaceVerify接口对带眨眼动作的视频进行实人认证。

传入视频要求

当您在进行实人认证的时候，请保证传入视频满足以下所有条件，否则会返回500报错。

- 格式：*FFmpeg*所支持的格式。例如，*MOV*、*MP4*、*M4A*、*3GP*及码率。
- 屏幕方向：目前不支持横屏视频。
- 动作：视频中必须包含眨眼动作。
- 时长：视频时长小于或者等于5秒，建议为2~3秒。
- 大小：小于或者等于10 MB。
- 分辨率：大于或者等于320*480像素，且小于或者等于1080P，即1920*1080像素。

发起认证请求

接口名：**ContrastFaceVerify**。

服务地址：cloudauth.aliyuncs.com（IPv4）或cloudauth-dualstack.aliyuncs.com（IPv6）。

请求方法：HTTPS POST和GET。

接口说明：通过带眨眼动作的视频进行实人认证。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	认证场景ID。	100000****

名称	类型	是否必选	描述	示例值
OuterOrderNo	String	是	客户服务端自定义的业务唯一标识，用于后续定位排查问题使用。值最长为32位长度的字母数字组合，请确保唯一。	e0c34a77f5ac40a5aa5e6ed20c35****
ProductCode	String	是	认证方案。 唯一取值：**ID_MIN_VIDEO**。	ID_MIN_VIDEO
Model	String	否	活体检测类型。取值： • **FRONT_CAMERA_LIVENESS**：移动设备前置摄像头采集的视频活体检测。 • **REAR_CAMERA_LIVENESS**：其他场景采集的视频活体检测。	FRONT_CAMERA_LIVENESS
CertType	String	是	证件类型。 当前仅支持身份证。 唯一取值：**IDENTITY_CARD**。	IDENTITY_CARD
CertName	String	是	真实姓名。	张三
CertNo	String	是	证件号码。	1**********9
DeviceToken	String	否	当前用户设备的Token，用于风险识别。	McozS1ZWRcRZStlERcZZo_QOytx5jcgZoZJEoRLO****
Mobile	String	否	用户手机号码。	1390000****
Ip	String	否	用户IP。	114.100.XX.XX
UserId	String	否	客户业务自定义的用户ID。	148562088256****

名称	类型	是否必选	描述	示例值
OssBucketName	String	是。在以下两种传入视频方式中，选择其中一种： • OSS视频文件 • FaceContrastFileObject，即本地视频文件 说明 如果选择OSS方式传入视频，您必须传入OssBucketName和OssObjectName参数。OSS请求参数格式仅支持开通实人认证服务时授权的OSS生成的URL、Bucket和Object名称。	已授权OSS空间的Bucket名称。	cn-shanghai-aliyun-cloudauth-xxxxx
OssObjectName	String		已授权OSS空间的Object名称。	verify/xxxxx/xxxxxx.jpeg
FaceContrastFileObject	InputStream		本地视频文件。	具体接入方式，请参见Java SDK调用示例。
EncryptType	String	否	加密类型。为空表示不加密。 如开启加密传输，需传入加密算法。目前仅支持SM2国密算法。 如需传入加密算法，需对CertName和CertNo进行加密，并传入加密后的密文。有关参数加密的更多信息，请参见参数加密说明。	SM2

返回数据

名称	类型	描述	示例值
RequestId	String	请求ID。	130A2C10-B9EE-4D84-88E3-5384FF039795
Message	String	请求的响应信息。	success

名称	类型	描述	示例值
Code	String	返回码，200指接口响应成功。详细认证结果判断，请参见错误码。	200
ResultObject.Passed	String	认证结果。取值： ● T：认证通过。 ● F：认证未通过。 说明 判断认证结果请以ResultObject.Passed字段为准。	T
ResultObject.CertifyId	String	08573be80f944d95ac812e019e36****	实人认证请求的唯一标识。
ResultObject.SubCode	String	认证结果描述，请参见ResultObject.SubCode错误码说明。 说明 判断认证结果请以ResultObject.Passed字段为准。	200
ResultObject.IdentityInfo	String	认证的主体信息，目前支持的认证场景返回为空。	xxx
ResultObject.MaterialInfo	String	认证主体附件信息，主要为图片类材料，JSON格式，请参见ResultObject.MaterialInfo的JSON格式示例。	{"faceAttack":"F","faceOcclusion":"F","facialPictureFront":{"faceAttackScore":0.00008597839769208804,"qualityScore":88.3615493774414,"verifyScore":50.28594166529785}}

ResultObject.MaterialInfo的JSON格式示例

```
{
    // 是否为攻击：攻击为T，非攻击为F。
    "faceAttack": "F",
    // 是否有脸部遮挡：有脸部遮挡为T，否则为F。
    "faceOcclusion": "F",
    "facialPictureFront": {
         // 人脸攻击分。
         "faceAttackScore": 0.00008597839769208804,
         // 视频抽帧质量最佳人脸照片Base64编码。
         "pictureBase64": "_9j_4AAQSkZJRgABAQAAAQABAAD_2wX****",
         // 活体人脸质量分数。
        "qualityScore": 88.3615493774414,
        // 人脸和公安比对分数，阈值可参考下表详细说明。
        "verifyScore": 50.28594166529785
    }
}
```

ResultObject.SubCode错误码说明

错误码	认证描述文案	是否计费	可能原因和运营建议
200	认证通过	是	无。
201	姓名和身份证不一致	是	可能是用户的信息有误或者用户信息为虚假信息，建议用户确认后重新操作。如果同一身份信息重复发起认证，服务端初始化将会返回417错误。
202	查询不到身份信息	是	可能是用户户口迁移、用户是军人等特殊状态导致，建议预留人工审核入口，进行人工审核。如果同一身份信息重复发起认证，服务端初始化将会返回417错误。
203	查询不到照片或照片不可用	是	可能是公安库数据问题导致，建议预留人工审核入口，进行人工审核。如果同一身份信息重复发起认证，服务端初始化将会返回417错误。
204	人脸比对不一致	是	可能不是同一人或活体照片质量较低，建议根据业务情况分层处理，如果为同一人可重复操作。
205	活体检测存在风险	是	可能存在攻击风险，建议人工审核分层处理，如果为真人可重复操作。
206	业务策略限制	是	个性化业务安全策略限制，如有疑问可咨询客服。
210	认证通过	是	通过产品安全策略综合判断，建议认证通过。

verifyScore阈值说明

千分之一误识率	万分之五误识率	万分之一误识率	十万分之五误识率	十万分之一误识率
70	71.5	75	76.5	80

说明 如果您有个性化需求，您可以根据业务情况，参考返回的比对分和阈值，自定义认证结果。

错误码

Code	Message	描述
200	success	请求响应成功。
400	参数不能为空	参数不能为空。
401	参数非法	非法参数。传入的姓名、身份证号码长度必须符合国家标准且不得包含英文字母等特殊字符。
402	应用配置不存在	应用配置不存在。
404	认证场景配置不存在	认证场景配置不存在，请先在控制台上添加认证场景。具体操作，请参见添加认证场景。
410	未开通服务	未开通OSS产品或未完成OSS读写授权，请登录控制台完成授权。具体操作，请参见授权金融级实人认证访问OSS存储空间。
411	RAM无权限	需要给子账号授予AliyunAntCloudAuthFullAccess的操作权限。
412	欠费中	金融级实人认证或OSS存在欠费，请充值后操作。
417	无法使用刷脸服务	当前身份信息比对源不可用。若信息正确，建议人工审核。
418	刷脸失败次数过多	当天刷脸认证次数过多，请在明天再试。
419	传入图片不可用	图片无法下载、图片内容为空、图片分辨率不符合要求或提取不到人脸特征，建议更换图片。
420	数据重复	通过多种方式传入了认证材料，目前支持以下两种方式中的一种传入认证材料： • OSS • FaceContrastFile
421	传入图片过大	图片超过了10 MB，建议压缩图片或更换图片上传方式。
422	下载图片超时	图片下载超过了3秒，请排查网络后重新操作。
423	状态错误	传入的CertifyId认证状态需要为T（认证通过），您也可以更换其他方式传入图片。
430	视频读取失败	视频读取失败，建议参考视频要求重新提交。
431	视频过大	视频大小超过限制，建议参考视频要求，压缩视频后重新提交。
432	视频过长	视频长度超过限制，建议将录制视频时长控制在3秒内。
433	视频格式不支持	视频格式不支持，参考视频要求重新提交。

Code	Message	描述
434	视频分辨率不符合要求	视频分辨率不支持，参考视频要求，调整分辨率后提交。
435	视频中无人脸	视频中检测不到人脸，建议参考视频要求重新提交。
500	系统错误	系统内部错误，请排查传入视频是否符合要求，具体信息，请参见传入视频要求。如果视频符合要求但仍然提示报错，请提交工单反馈。

1.5.4. Java SDK调用示例

本文介绍了实人认证纯服务端API的Java SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

调试ContrastFaceVerify接口

本章节提供的是最新版本的Java SDK调用示例，如果您需要查看之前版本，请参见Java SDK调用示例（V1版本）。

前提条件

需要使用的Java版本在1.8版本及以上。

使用Maven引入SDK

```
<dependency>
       <groupId>com.aliyun</groupId>
       <artifactId>cloudauth20190307</artifactId>
        <version>2.0.3</version>
</dependency>
```

照片实人认证调用示例

```
import java.util.Arrays;
import java.util.List;
import com.aliyun.cloudauth20190307.Client;
import com.aliyun.cloudauth20190307.models.ContrastFaceVerifyRequest;
import com.aliyun.cloudauth20190307.models.ContrastFaceVerifyResponse;
import com.aliyun.tea.TeaException;
import com.aliyun.tea.TeaUnretryableException;
import com.aliyun.tearpc.models.Config;
import com.aliyun.teautil.models.RuntimeOptions;
public class ContrastFaceVerify {
    public static void main(String[] args) throws Exception {
        // 通过以下代码创建API请求并设置参数。
        ContrastFaceVerifyRequest request = new ContrastFaceVerifyRequest();
        // 固定值。
        request.setProductCode("ID_MIN");
        // 固定值。
```

```
        // 自定义。
        request.setCertType("IDENTITY_CARD");
        request.setCertName("姓名张三");
        request.setIp("用户IP地址");
        request.setMobile("用户手机号");
        request.setDeviceToken("通过设备指纹获取的deviceToken");
        request.setUserId("用户ID");
        request.setCertNo("18位身份证号");
        request.setOuterOrderNo("商户请求的唯一标识");
        // 请输入场景+L。
        request.setSceneId(xxxxL);
        request.setModel("FRONT_CAMERA_LIVENESS");
        // 请在以下四种方式中选择其一。
        // 1.照片Base64编码。
        // 2.照片OSS bucket名和文件名。
        // 3.照片OSS的URL地址。
        // 4.已经实人认证通过的CertifyId。
        // 方式一：照片Base64。
        request.setFaceContrastPicture("人脸照片Base64编码");
        // 方式二：照片OSS bucket名和文件名。
        //request.setOssBucketName("cn-shanghai-aliyun-cloudauth-xxxxx");
        //request.setOssObjectName("verify/xxxxx/xxxxxx.jpeg");
        // 方式三：照片OSS的URL地址，公网可访问。
        //request.setFaceContrastPictureUrl("https://cn-shanghai-aliyun-cloudauth-xxxxxx.os
s-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg")
        // 方式四：已经实人认证通过的CertifyId。
        //request.setCertifyId("0bfa7c493f850e5178b9f8613634c9xx");
        // 如需开启个人信息加密传输。
        // request.setEncryptType("SM2");
        // request.setCertName("BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==");
        // request.setCertNo("BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg==");
        // 推荐，支持服务路由。
        ContrastFaceVerifyResponse response = contrastFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        //ContrastFaceVerifyResponse response = contrastFaceVerify("cloudauth.cn-shanghai.a
liyuncs.com", request);
        System.out.println(response.getRequestId());
        System.out.println(response.getCode());
        System.out.println(response.getMessage());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getPassed());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getSubCode());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getIdentityInfo());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getMaterialInfo());
    }
    private static ContrastFaceVerifyResponse contrastFaceVerifyAutoRoute(ContrastFaceVerif
yRequest request) {
        // 第一个为主区域Endpoint，第二个为备区域Endpoint。
        List<String> endpoints = Arrays.asList("cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com");
        ContrastFaceVerifyResponse lastResponse = null;
        for (String endpoint : endpoints) {
```

```
            try {
                ContrastFaceVerifyResponse response = contrastFaceVerify(endpoint, request)
;
                lastResponse = response;
                // 服务端错误，切换到下个区域调用。
                if (response != null && "500".equals(response.getCode())) {
                    continue;
                }
                return response;
            } catch (Exception e) {
                // 网络异常，切换到下个区域调用。
                if (e.getCause() instanceof TeaException) {
                    TeaException teaException = ((TeaException)e.getCause());
                    if (teaException.getData() != null && "ServiceUnavailable".equals(
                        teaException.getData().get("Code"))) {
                        continue;
                    }
                }
                if (e.getCause() instanceof TeaUnretryableException) {
                    continue;
                }
            }
        }
        return lastResponse;
    }
    private static ContrastFaceVerifyResponse contrastFaceVerify(String endpoint, ContrastF
aceVerifyRequest request)
        throws Exception {
        Config config = new Config();
        config.setAccessKeyId("您的AccessKey ID");
        config.setAccessKeySecret("您的AccessKey Secret");
        config.setEndpoint(endpoint);
        // 设置http代理。
        //config.setHttpProxy("http://xx.xx.xx.xx:xxxx");
        // 设置https代理。
        //config.setHttpsProxy("http://xx.xx.xx.xx:xxxx");
        Client client = new Client(config);
        // 创建RuntimeObject实例并设置运行参数。
        RuntimeOptions runtime = new RuntimeOptions();
        runtime.readTimeout = 10000;
        runtime.connectTimeout = 10000;
        return client.contrastFaceVerify(request, runtime);
    }
}
```

视频实人认证调用示例

支持OSS对象存储方式和本地文件两种方式。

使用本地文件的方式（推荐）

```
import java.io.FileInputStream;
import java.io.InputStream;
import java.util.Arrays;
```

```
import java.util.List;
import com.aliyun.cloudauth20190307.Client;
import com.aliyun.cloudauth20190307.models.ContrastFaceVerifyAdvanceRequest;
import com.aliyun.cloudauth20190307.models.ContrastFaceVerifyResponse;
import com.aliyun.tea.TeaException;
import com.aliyun.tea.TeaUnretryableException;
import com.aliyun.tearpc.models.Config;
import com.aliyun.teautil.models.RuntimeOptions;
public class ContrastFaceVerifyAdvance {
    public static void main(String[] args) throws Exception {
        // 通过以下代码创建API请求并设置参数。
        ContrastFaceVerifyAdvanceRequest request = new ContrastFaceVerifyAdvanceRequest();
        // 请输入场景ID+L。
        request.setSceneId(1L);
        request.setOuterOrderNo("商户请求的唯一标识");
        request.setProductCode("ID_MIN_VIDEO");
        // 卡证核身类型，固定值。
        request.setModel("FRONT_CAMERA_LIVENESS");
        // 证件类型，固定值。
        request.setCertType("IDENTITY_CARD");
        request.setCertName("张三");
        request.setCertNo("xxxx");
        //request.setDeviceToken("xxxx");
        //request.setMobile("130xxxxxxxx");
        //request.setIp("114.xxx.xxx.xxx");
        //request.setUserId("12345xxxx");
        // 如需开启个人信息加密传输。
        // request.setEncryptType("SM2");
        // request.setCertName("BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==");
        // request.setCertNo("BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg==");
        InputStream inputStream = null;
        try {
            // 设置本地文件路径。
            inputStream = new FileInputStream("test.mp4");
            request.setFaceContrastFileObject(inputStream);
            // 推荐，支持服务路由。
            ContrastFaceVerifyResponse response = contrastFaceVerifyAutoRoute(request);
            // 不支持服务自动路由。
            //ContrastFaceVerifyResponse response = contrastFaceVerify("cloudauth.cn-shangh
ai.aliyuncs.com", request);
            System.out.println(response.getRequestId());
            System.out.println(response.getCode());
            System.out.println(response.getMessage());
            System.out.println(response.getResultObject() == null ? null
                : response.getResultObject().getPassed());
            System.out.println(response.getResultObject() == null ? null
                : response.getResultObject().getSubCode());
            System.out.println(response.getResultObject() == null ? null
                : response.getResultObject().getIdentityInfo());
            System.out.println(response.getResultObject() == null ? null
                : response.getResultObject().getMaterialInfo());
        } finally {
            if(inputStream != null){
                inputStream.close();
```

```
            }
        }
    }
    private static ContrastFaceVerifyResponse contrastFaceVerifyAutoRoute(ContrastFaceVerif
yAdvanceRequest request) {
        // 第一个为主区域Endpoint，第二个为备区域Endpoint。
        List<String> endpoints = Arrays.asList("cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com");
        ContrastFaceVerifyResponse lastResponse = null;
        for (String endpoint : endpoints) {
            try {
                ContrastFaceVerifyResponse response = contrastFaceVerify(endpoint, request)
;
                lastResponse = response;
                // 服务端错误，切换到下个区域调用。
                if (response != null && "500".equals(response.getCode())) {
                    continue;
                }
                return response;
            } catch (Exception e) {
                // 网络异常，切换到下个区域调用。
                if (e.getCause() instanceof TeaException) {
                    TeaException teaException = ((TeaException)e.getCause());
                    if (teaException.getData() != null && "ServiceUnavailable".equals(
                        teaException.getData().get("Code"))) {
                        continue;
                    }
                }
                if (e.getCause() instanceof TeaUnretryableException) {
                    continue;
                }
            }
        }
        return lastResponse;
    }
    private static ContrastFaceVerifyResponse contrastFaceVerify(String endpoint, ContrastF
aceVerifyAdvanceRequest request)
        throws Exception {
        Config config = new Config();
        config.setAccessKeyId("您的AccessKey ID");
        config.setAccessKeySecret("您的AccessKey Secret");
        config.setEndpoint(endpoint);
        // 设置http代理。
        //config.setHttpProxy("http://xx.xx.xx.xx:xxxx");
        // 设置https代理。
        //config.setHttpsProxy("http://xx.xx.xx.xx:xxxx");
        Client client = new Client(config);
        // 创建RuntimeObject实例并设置运行参数。
        RuntimeOptions runtime = new RuntimeOptions();
        runtime.readTimeout = 10000;
        runtime.connectTimeout = 10000;
        return client.contrastFaceVerifyAdvance(request, runtime);
    }
}
```

使用OSS对象存储的方式

```
import java.util.Arrays;
import java.util.List;
import com.aliyun.cloudauth20190307.Client;
import com.aliyun.cloudauth20190307.models.ContrastFaceVerifyRequest;
import com.aliyun.cloudauth20190307.models.ContrastFaceVerifyResponse;
import com.aliyun.tea.TeaException;
import com.aliyun.tea.TeaUnretryableException;
import com.aliyun.tearpc.models.Config;
import com.aliyun.teautil.models.RuntimeOptions;
public class ContrastFaceVerify {
    public static void main(String[] args) throws Exception {
        // 通过以下代码创建API请求并设置参数。
        ContrastFaceVerifyRequest request = new ContrastFaceVerifyRequest();
        // 请输入场景ID+L。
        request.setSceneId(xxxxL);
        request.setOuterOrderNo("商户请求的唯一标识");
        request.setProductCode("ID_MIN_VIDEO");
        // 卡证核身类型，固定值。
        request.setModel("FRONT_CAMERA_LIVENESS");
        // 证件类型，固定值。
        request.setCertType("IDENTITY_CARD");
        request.setCertName("张三");
        request.setCertNo("xxxx");
        //request.setDeviceToken("xxxx");
        //request.setMobile("130xxxxxxxx");
        //request.setIp("114.xxx.xxx.xxx");
        //request.setUserId("12345xxxx");
        request.setOssBucketName("xxxxx");
        request.setOssObjectName("xxxxx");
        // 如需开启个人信息加密传输。
        // request.setEncryptType("SM2");
        // request.setCertName("BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==");
        // request.setCertNo("BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg==");
        // 推荐，支持服务路由。
        ContrastFaceVerifyResponse response = contrastFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        //ContrastFaceVerifyResponse response = contrastFaceVerify("cloudauth.cn-shanghai.a
liyuncs.com", request);
        System.out.println(response.getRequestId());
        System.out.println(response.getCode());
        System.out.println(response.getMessage());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getPassed());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getSubCode());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getIdentityInfo());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getMaterialInfo());
    }
    private static ContrastFaceVerifyResponse contrastFaceVerifyAutoRoute(ContrastFaceVerif
yRequest request) {
```

```
        // 第一个为主区域Endpoint，第二个为备区域Endpoint。
        List<String> endpoints = Arrays.asList("cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com");
        ContrastFaceVerifyResponse lastResponse = null;
        for (String endpoint : endpoints) {
            try {
                ContrastFaceVerifyResponse response = contrastFaceVerify(endpoint, request)
;
                lastResponse = response;
                // 服务端错误，切换到下个区域调用。
                if (response != null && "500".equals(response.getCode())) {
                    continue;
                }
                return response;
            } catch (Exception e) {
                // 网络异常，切换到下个区域调用。
                if (e.getCause() instanceof TeaException) {
                    TeaException teaException = ((TeaException)e.getCause());
                    if (teaException.getData() != null && "ServiceUnavailable".equals(
                        teaException.getData().get("Code"))) {
                        continue;
                    }
                }
                if (e.getCause() instanceof TeaUnretryableException) {
                    continue;
                }
            }
        }
        return lastResponse;
    }
    private static ContrastFaceVerifyResponse contrastFaceVerify(String endpoint, ContrastF
aceVerifyRequest request)
        throws Exception {
        Config config = new Config();
        config.setAccessKeyId("您的AccessKey ID");
        config.setAccessKeySecret("您的AccessKey Secret");
        config.setEndpoint(endpoint);
        // 设置http代理。
        //config.setHttpProxy("http://xx.xx.xx.xx:xxxx");
        // 设置https代理。
        //config.setHttpsProxy("http://xx.xx.xx.xx:xxxx");
        Client client = new Client(config);
        // 创建RuntimeObject实例并设置运行参数。
        RuntimeOptions runtime = new RuntimeOptions();
        runtime.readTimeout = 10000;
        runtime.connectTimeout = 10000;
        return client.contrastFaceVerify(request, runtime);
    }
}
```

1.5.5. PHP SDK调用示例

本文介绍了金融级实人认证纯服务端API的PHP SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

调试ContrastFaceVerify接口

前提条件

- 需要使用的PHP版本在5.6版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于PHP SDK源代码，请参见PHP SDK包。

- 通过Composer来管理项目依赖（推荐）

```
composer require alibabacloud/cloudauth-20190307 2.0.3
```

- 使用composer.json文件

```
{
   "require": {
       "alibabacloud/cloudauth-20190307": "2.0.3"
    }
}
```

如果因网络问题导致无法按照上述操作安装SDK，您可以使用阿里云Composer全量镜像。

照片实人认证调用示例

关于参数的具体描述，请参见照片实人认证。

```
<?php
namespace AlibabaCloud\SDK\Sample {
use AlibabaCloud\SDK\Cloudauth\V20190307\Cloudauth;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\ContrastFaceVerifyRequest;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\ContrastFaceVerifyResponse;
use AlibabaCloud\Tea\Utils\Utils;
use \Exception;
use AlibabaCloud\Tea\Exception\TeaError;
use AlibabaCloud\Tea\Rpc\Rpc\Config;
use AlibabaCloud\Tea\Utils\Utils\RuntimeOptions;
class ContrastFaceVerify {
    /**
     * @param string $endpoint
     * @return Cloudauth
     */
    public static function createClient($endpoint){
        $config = new Config([
            // 您的AccessKey ID。
            "accessKeyId" => "<您的AccessKey ID>",
            // 您的AccessKey Secret。
            "accessKeySecret" => "<您的AccessKey Secret>",
            // 设置HTTP代理。
            // "httpProxy" => "<http://xx.xx.xx.xx:xxxx>",
```

```
            // 设置HTTPS代理。
            // "httpsProxy" => "<https://xx.xx.xx.xx:xxxx>",
            "endpoint" => $endpoint
        ]);
        return new Cloudauth($config);
    }
    /**
     * @return void
     */
    public static function main(){
        $request = new ContrastFaceVerifyRequest([
            // 请输入场景ID。
            "sceneId" => <xxx>,
            "outerOrderNo" => "<商户请求的唯一标识>",
            // 要接入的认证方案。
            "productCode" => "<ID_MIN>",
            // 要进行活体检测的类型。
            "model" => "<NO_LIVENESS>",
            "certType" => "<IDENTITY_CARD>",
            "certName" => "<xxx>",
            "certNo" => "<xxx>",
            // "deviceToken" => "<McozS1ZWRcRZStlERcZZo_QOytx5jcgZoZJEoRLOxxxxxxx>",
            // "mobile" => "<xxx>",
            // "ip" => "<114.xxx.xxx.xxx>",
            // "userId" => "<12345xxxx>",
            // 请在以下四种方式中选择其一。
            // 1.照片Base64编码。
            // 2.照片OSS Bucket名和文件名。
            // 3.照片OSS的URL地址。
            // 4.已经实人认证通过的CertifyId。
            // 方式一：照片Base64。
            // "faceContrastPicture" => "<人脸照片Base64编码>",
            // 方式二：照片OSS bucket名和文件名。
            // "ossBucketName" => "<cn-shanghai-aliyun-cloudauth-xxxxx>",
            // "ossObjectName" => "<verify/xxxxx/xxxxxx.jpeg>",
            // 方式三：照片OSS的URL地址，公网可访问。
            "faceContrastPictureUrl" => "<https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-c
n-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg>",
            // 方式四：已经实人认证通过的CertifyId。
            // "certifyId" => "<0bfa7c493f850e5178b9f8613634c9xx>",
            // 如需开启个人信息加密传输。
            // "encryptType" => "<SM2>",
            // "certName" => "<BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==>",
            // "certNo" => "<BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg==>"
        ]);
        // 推荐，支持服务路由。
        $response = self::contrastFaceVerifyAutoRoute($request);
        // 不支持服务自动路由。
        // $response = self::contrastFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", $requ
est);
        var_dump($response->requestId);
        var_dump($response->code);
        var_dump($response->message);
        var_dump($response->resultObject->materialInfo);
```

```
        var_dump($response->resultObject->subCode);
        var_dump($response->resultObject->identityInfo);
        var_dump($response->resultObject->passed);
    }
    /**
     * @param ContrastFaceVerifyRequest $request
     * @return ContrastFaceVerifyResponse
     */
    public static function contrastFaceVerifyAutoRoute(ContrastFaceVerifyRequest $request){
        $endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        foreach($endpoints as $endpoint){
            try {
                $response = self::contrastFaceVerify($endpoint, $request);
                if (Utils::equalString("500", $response->code)) {
                    continue;
                }
                return $response;
            }
            catch (Exception $err) {
                if (!($err instanceof TeaError)) {
                    $err = new TeaError([], $err->getMessage(), $err->getCode(), $err);
                }
                // 网络异常，切换到下个区域调用。
                if (Utils::equalString("ServiceUnavailable", $err->code)) {
                    continue;
                }
            }
        }
        return null;
    }
    /**
     * @param string $endpoint
     * @param ContrastFaceVerifyRequest $request
     * @return ContrastFaceVerifyResponse
     */
    public static function contrastFaceVerify($endpoint, $request){
        $client = self::createClient($endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        $runtime = new RuntimeOptions([]);
        $runtime->readTimeout = 10000;
        $runtime->connectTimeout = 10000;
        return $client->contrastFaceVerify($request, $runtime);
    }
}
$path = __DIR__ . \DIRECTORY_SEPARATOR . \DIRECTORY_SEPARATOR . 'vendor' . \DIRECTORY_SEPAR
ATOR . 'autoload.php';
if (file_exists($path)) {
    require_once $path;
}
ContrastFaceVerify::main();
}
```

视频实人认证调用示例

支持OSS对象存储方式和本地文件两种方式。关于参数的具体描述，请参见视频实人认证。

使用本地文件的方式（推荐）

```
<?php
namespace AlibabaCloud\SDK\Sample {
use AlibabaCloud\SDK\Cloudauth\V20190307\Cloudauth;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\ContrastFaceVerifyAdvanceRequest;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\ContrastFaceVerifyResponse;
use AlibabaCloud\Tea\Utils\Utils;
use \Exception;
use AlibabaCloud\Tea\Exception\TeaError;
use AlibabaCloud\Tea\Rpc\Rpc\Config;
use AlibabaCloud\Tea\Utils\Utils\RuntimeOptions;
use GuzzleHttp\Psr7\Stream;
class ContrastFaceVerify {
    /**
     * @param string $endpoint
     * @return Cloudauth
     */
    public static function createClient($endpoint){
        $config = new Config([
            // 您的AccessKey ID。
            "accessKeyId" => "<您的AccessKey ID>",
            // 您的AccessKey Secret。
            "accessKeySecret" => "<您的AccessKey Secret>",
            // 设置HTTP代理。
            // "httpProxy" => "<http://xx.xx.xx.xx:xxxx>",
            // 设置HTTPS代理。
            // "httpsProxy" => "<https://xx.xx.xx.xx:xxxx>",
            "endpoint" => $endpoint
        ]);
        return new Cloudauth($config);
    }
    /**
     * @return void
     */
    public static function main(){
        $file = new Stream(fopen('本地文件路径', 'r'));
        $request = new ContrastFaceVerifyAdvanceRequest([
            // 请输入场景ID。
            "sceneId" => <xxx>,
            "outerOrderNo" => "<商户请求的唯一标识>",
            // 要接入的认证方案。
            "productCode" => "<ID_MIN_VIDEO>",
            // 要进行活体检测的类型。
            "model" => "<FRONT_CAMERA_LIVENESS>",
            "certType" => "<IDENTITY_CARD>",
            "certName" => "<xxx>",
            "certNo" => "<xxx>",
            // "deviceToken" => "<McozS1ZWRcRZStlERcZZo_QOytx5jcgZoZJEoRLOxxxxxxx>",
            // "mobile" => "<xxx>",
```

```
            // "ip" => "<114.xxx.xxx.xxx>",
            // "userId" => "<12345xxxx>",
            "faceContrastFileObject" => $file,
            // 如需开启个人信息加密传输。
            // "encryptType" => "<SM2>",
            // "certName" => "<BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==>",
            // "certNo" => "<BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg==>"
        ]);
        try {
            // 推荐，支持服务路由。
            // $response = self::contrastFaceVerifyAutoRoute($request);
            // 不支持服务自动路由。
            $response = self::contrastFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", $req
uest);
            var_dump($response->requestId);
            var_dump($response->code);
            var_dump($response->message);
            var_dump($response->resultObject->materialInfo);
            var_dump($response->resultObject->subCode);
            var_dump($response->resultObject->identityInfo);
            var_dump($response->resultObject->passed);
        } finally {
            fclose($file);
        }
    }
    /**
     * @param ContrastFaceVerifyAdvanceRequest $request
     * @return ContrastFaceVerifyResponse
     */
    public static function contrastFaceVerifyAutoRoute(ContrastFaceVerifyAdvanceRequest $re
quest){
        $endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        foreach($endpoints as $endpoint){
            try {
                $response = self::contrastFaceVerify($endpoint, $request);
                if (Utils::equalString("500", $response->code)) {
                    continue;
                }
                return $response;
            }
            catch (Exception $err) {
                if (!($err instanceof TeaError)) {
                    $err = new TeaError([], $err->getMessage(), $err->getCode(), $err);
                }
                // 网络异常，切换到下个区域调用。
                if (Utils::equalString("ServiceUnavailable", $err->code)) {
                    continue;
                }
            }
        }
        return null;
```

```
    }
    /**
     * @param string $endpoint
     * @param ContrastFaceVerifyAdvanceRequest $request
     * @return ContrastFaceVerifyResponse
     */
    public static function contrastFaceVerify($endpoint, $request){
        $client = self::createClient($endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        $runtime = new RuntimeOptions([]);
        $runtime->readTimeout = 10000;
        $runtime->connectTimeout = 10000;
        return $client->contrastFaceVerifyAdvance($request, $runtime);
    }
}
$path = __DIR__ . \DIRECTORY_SEPARATOR . \DIRECTORY_SEPARATOR . 'vendor' . \DIRECTORY_SEPAR
ATOR . 'autoload.php';
if (file_exists($path)) {
    require_once $path;
}
ContrastFaceVerify::main();
}
```

使用OSS对象存储的方式

```
<?php
namespace AlibabaCloud\SDK\Sample {
use AlibabaCloud\SDK\Cloudauth\V20190307\Cloudauth;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\ContrastFaceVerifyRequest;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\ContrastFaceVerifyResponse;
use AlibabaCloud\Tea\Utils\Utils;
use \Exception;
use AlibabaCloud\Tea\Exception\TeaError;
use AlibabaCloud\Tea\Rpc\Rpc\Config;
use AlibabaCloud\Tea\Utils\Utils\RuntimeOptions;
class ContrastFaceVerify {
    /**
     * @param string $endpoint
     * @return Cloudauth
     */
    public static function createClient($endpoint){
        $config = new Config([
            // 您的AccessKey ID。
            "accessKeyId" => "<您的AccessKey ID>",
            // 您的AccessKey Secret。
            "accessKeySecret" => "<您的AccessKey Secret>",
            // 设置HTTP代理。
            // "httpProxy" => "<http://xx.xx.xx.xx:xxxx>",
            // 设置HTTPS代理。
            // "httpsProxy" => "<https://xx.xx.xx.xx:xxxx>",
            "endpoint" => $endpoint
        ]);
        return new Cloudauth($config);
    }
```

```
    }
    /**
     * @return void
     */
    public static function main(){
        $request = new ContrastFaceVerifyRequest([
            // 请输入场景ID。
            "sceneId" => <xxx>,
            "outerOrderNo" => "<商户请求的唯一标识>",
            // 要接入的认证方案。
            "productCode" => "<ID_MIN_VIDEO>",
            // 要进行活体检测的类型。
            "model" => "<FRONT_CAMERA_LIVENESS>",
            "certType" => "<IDENTITY_CARD>",
            "certName" => "<xxx>",
            "certNo" => "<xxx>",
            // "deviceToken" => "<McozS1ZWRcRZStlERcZZo_QOytx5jcgZoZJEoRLOxxxxxxx>",
            // "mobile" => "<xxx>",
            // "ip" => "<114.xxx.xxx.xxx>",
            // "userId" => "<12345xxxx>",
            "ossBucketName" => "<cn-shanghai-aliyun-cloudauth-xxxxx>",
            "ossObjectName" => "<verify/xxxxx/xxxxxx.mp4>",
            // 如需开启个人信息加密传输。
            // "encryptType" => "<SM2>",
            // "certName" => "<BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==>",
            // "certNo" => "<BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg==>"
        ]);
        // 推荐，支持服务路由。
        $response = self::contrastFaceVerifyAutoRoute($request);
        // 不支持服务自动路由。
        // $response = self::contrastFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", $requ
est);
        var_dump($response->requestId);
        var_dump($response->code);
        var_dump($response->message);
        var_dump($response->resultObject->materialInfo);
        var_dump($response->resultObject->subCode);
        var_dump($response->resultObject->identityInfo);
        var_dump($response->resultObject->passed);
    }
    /**
     * @param ContrastFaceVerifyRequest $request
     * @return ContrastFaceVerifyResponse
     */
    public static function contrastFaceVerifyAutoRoute(ContrastFaceVerifyRequest $request){
        $endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        foreach($endpoints as $endpoint){
            try {
                $response = self::contrastFaceVerify($endpoint, $request);
                if (Utils::equalString("500", $response->code)) {
                    continue;
                }
```

```
                return $response;
            }
            catch (Exception $err) {
                if (!($err instanceof TeaError)) {
                    $err = new TeaError([], $err->getMessage(), $err->getCode(), $err);
                }
                // 网络异常，切换到下个区域调用。
                if (Utils::equalString("ServiceUnavailable", $err->code)) {
                    continue;
                }
            }
        }
        return null;
    }
    /**
     * @param string $endpoint
     * @param ContrastFaceVerifyRequest $request
     * @return ContrastFaceVerifyResponse
     */
    public static function contrastFaceVerify($endpoint, $request){
        $client = self::createClient($endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        $runtime = new RuntimeOptions([]);
        $runtime->readTimeout = 10000;
        $runtime->connectTimeout = 10000;
        return $client->contrastFaceVerify($request, $runtime);
    }
}
$path = __DIR__ . \DIRECTORY_SEPARATOR . \DIRECTORY_SEPARATOR . 'vendor' . \DIRECTORY_SEPAR
ATOR . 'autoload.php';
if (file_exists($path)) {
    require_once $path;
}
ContrastFaceVerify::main();
}
```

1.5.6. Python SDK调用示例

本文介绍了金融级实人认证纯服务端API的Python SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

详细信息，请参见调试ContrastFaceVerify接口。

前提条件

- 需要使用的Python版本在3.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

引入依赖包

```
pip install alibabacloud_tea_openapi==0.2.5
```

安装SDK

关于Python SDK源代码，请参见Python SDK包。

```
pip install alibabacloud_cloudauth20190307==2.0.3
```

照片实人认证调用示例

关于参数的具体描述，请参见照片实人认证。

```
from alibabacloud_cloudauth20190307.client import Client as CloudauthClient
from alibabacloud_tea_openapi import models as open_api_models
from alibabacloud_cloudauth20190307 import models as cloudauth_models
from alibabacloud_tea_util.client import Client as UtilClient
from alibabacloud_tea_util import models as util_models
class ContrastFaceVerify:
    def __init__(self):
        pass
    @staticmethod
    def create_client(
            endpoint: str,
    ) -> CloudauthClient:
        config = open_api_models.Config(
            # 您的AccessKey ID。
            access_key_id='您的AccessKey ID',
            # 您的AccessKey Secret。
            access_key_secret='您的AccessKey Secret',
            # 设置HTTP代理。
            # http_proxy='http://xx.xx.xx.xx:xxxx',
            # 设置HTTPS代理。
            # https_proxy='https://username:password@xxx.xxx.xxx.xxx:9999',
            endpoint=endpoint
        )
        return CloudauthClient(config)
    @staticmethod
    def main() -> None:
        request = cloudauth_models.ContrastFaceVerifyRequest(
            # 请输入场景ID。
            scene_id=xxx,
            outer_order_no='商户请求的唯一标识',
            # 要接入的认证方案。
            product_code='ID_MIN',
            model='NO_LIVENESS',
            # 固定值IDENTITY_CARD。
            cert_type='IDENTITY_CARD',
            cert_name='xxx',
            cert_no='xxx',
            # ip='114.xxx.xxx.xxx',
            # user_id='用户ID',
            # mobile='130xxxxxxxx',
            # device_token='通过设备指纹获取的deviceToken',
```

```
            # 请在以下四种方式中选择其一。
            # 1.照片Base64编码。
            # 2.照片OSS bucket名和文件名。
            # 3.照片OSS的URL地址。
            # 4.已经实人认证通过的CertifyId。
            # 方式一：照片Base64。
            # face_contrast_picture='人脸照片Base64编码'
            # 方式二：照片OSS bucket名和文件名。
            # oss_bucket_name='cn-shanghai-aliyun-cloudauth-xxxxx',
            # oss_object_name='verify/xxxxx/xxxxxx.jpeg'
            # 方式三：照片OSS的URL地址，公网可访问。
            face_contrast_picture_url='https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-s
hanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg'
            # 方式四：已经实人认证通过的CertifyId。
            # certify_id='0bfa7c493f850e5178b9f8613634c9xx',
            # 如需开启个人信息加密传输。
            # encrypt_type='SM2',
            # cert_name='BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==',
            # cert_no='BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg=='
        )
        response = ContrastFaceVerify.contrast_face_verify_auto_route(request)
        # 不支持服务自动路由。
        # response = ContrastFaceVerify.contrast_face_verify('cloudauth.cn-shanghai.aliyunc
s.com', request)
        print(response.request_id)
        print(response.code)
        print(response.message)
        print(response.result_object.passed)
        print(response.result_object.identity_info)
        print(response.result_object.sub_code)
        print(response.result_object.material_info)
        print(response.result_object.certify_id)
    @staticmethod
    def contrast_face_verify_auto_route(
            request: cloudauth_models.ContrastFaceVerifyRequest,
    ) -> cloudauth_models.ContrastFaceVerifyResponse:
        endpoints = [
            'cloudauth.cn-shanghai.aliyuncs.com',
            'cloudauth.cn-beijing.aliyuncs.com'
        ]
        for endpoint in endpoints:
            try:
                response = ContrastFaceVerify.contrast_face_verify(endpoint, request)
                if UtilClient.equal_string('500', response.code):
                    continue
                return response
            except Exception as err:
                    continue
        return None
    @staticmethod
    def contrast_face_verify(
            endpoint: str,
            request: cloudauth_models.ContrastFaceVerifyRequest,
    ) -> cloudauth_models.ContrastFaceVerifyResponse:
```

```
        client = ContrastFaceVerify.create_client(endpoint)
        # 创建RuntimeObject实例并设置运行参数。
        runtime = util_models.RuntimeOptions()
        runtime.read_timeout = 10000
        runtime.connect_timeout = 10000
        return client.contrast_face_verify(request, runtime)
if __name__ == '__main__':
    ContrastFaceVerify.main()
```

视频实人认证调用示例

支持OSS对象存储方式和本地文件两种方式。关于参数的具体描述，请参见视频实人认证。

使用本地文件的方式（推荐）

```
from alibabacloud_cloudauth20190307.client import Client as CloudauthClient
from alibabacloud_tea_openapi import models as open_api_models
from alibabacloud_cloudauth20190307 import models as cloudauth_models
from alibabacloud_tea_util.client import Client as UtilClient
from alibabacloud_tea_util import models as util_models
class ContrastFaceVerify:
    def __init__(self):
        pass
    @staticmethod
    def create_client(
            endpoint: str,
    ) -> CloudauthClient:
        config = open_api_models.Config(
            # 您的AccessKey ID。
            access_key_id='您的AccessKey ID',
            # 您的AccessKey Secret。
            access_key_secret='您的AccessKey Secret',
            # 设置HTTP代理。
            # http_proxy='http://xx.xx.xx.xx:xxxx',
            # 设置HTTPS代理。
            # https_proxy='https://username:password@xxx.xxx.xxx.xxx:9999',
            endpoint=endpoint
        )
        return CloudauthClient(config)
    @staticmethod
    def main() -> None:
        file = open('本地文件路径', 'rb')
        request = cloudauth_models.ContrastFaceVerifyAdvanceRequest(
            # 请输入场景ID。
            scene_id=xxx,
            outer_order_no='商户请求的唯一标识',
            # 要接入的认证方案。
            product_code='ID_MIN_VIDEO',
            model='FRONT_CAMERA_LIVENESS',
            # 固定值IDENTITY_CARD。
            cert_type='IDENTITY_CARD',
            cert_name='xxx',
            cert_no='xxx',
            # ip='114.xxx.xxx.xxx',
            # user_id='用户ID',
```

```
            # mobile='130xxxxxxxx',
            # device_token='通过设备指纹获取的deviceToken',
            face_contrast_file_object=file,
            # 如需开启个人信息加密传输。
            # encrypt_type='SM2',
            # cert_name='BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==',
            # cert_no='BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg=='
        )
        response = ContrastFaceVerify.contrast_face_verify_auto_route(request)
        # 不支持服务自动路由。
        # response = ContrastFaceVerify.contrast_face_verify('cloudauth.cn-shanghai.aliyunc
s.com', request)
        print(response.request_id)
        print(response.code)
        print(response.message)
        print(response.result_object.passed)
        print(response.result_object.identity_info)
        print(response.result_object.sub_code)
        print(response.result_object.material_info)
        print(response.result_object.certify_id)
        file.close()
    @staticmethod
    def contrast_face_verify_auto_route(
            request: cloudauth_models.ContrastFaceVerifyAdvanceRequest,
    ) -> cloudauth_models.ContrastFaceVerifyResponse:
        endpoints = [
            'cloudauth.cn-shanghai.aliyuncs.com',
            'cloudauth.cn-beijing.aliyuncs.com'
        ]
        for endpoint in endpoints:
            try:
                response = ContrastFaceVerify.contrast_face_verify(endpoint, request)
                if UtilClient.equal_string('500', response.code):
                    continue
                return response
            except Exception as err:
                    continue
        return None
    @staticmethod
    def contrast_face_verify(
            endpoint: str,
            request: cloudauth_models.ContrastFaceVerifyAdvanceRequest,
    ) -> cloudauth_models.ContrastFaceVerifyResponse:
        client = ContrastFaceVerify.create_client(endpoint)
        # 创建RuntimeObject实例并设置运行参数。
        runtime = util_models.RuntimeOptions()
        runtime.read_timeout = 10000
        runtime.connect_timeout = 10000
        return client.contrast_face_verify_advance(request, runtime)
if __name__ == '__main__':
    ContrastFaceVerify.main()
```

使用OSS对象存储的方式

```
from alibabacloud_cloudauth20190307.client import Client as CloudauthClient
from alibabacloud_tea_openapi import models as open_api_models
from alibabacloud_cloudauth20190307 import models as cloudauth_models
from alibabacloud_tea_util.client import Client as UtilClient
from alibabacloud_tea_util import models as util_models
class ContrastFaceVerify:
    def __init__(self):
        pass
    @staticmethod
    def create_client(
            endpoint: str,
    ) -> CloudauthClient:
        config = open_api_models.Config(
            # 您的AccessKey ID。
            access_key_id='您的AccessKey ID',
            # 您的AccessKey Secret。
            access_key_secret='您的AccessKey Secret',
            # 设置HTTP代理。
            # http_proxy='http://xx.xx.xx.xx:xxxx',
            # 设置HTTPS代理。
            # https_proxy='https://username:password@xxx.xxx.xxx.xxx:9999',
            endpoint=endpoint
        )
        return CloudauthClient(config)
    @staticmethod
    def main() -> None:
        request = cloudauth_models.ContrastFaceVerifyRequest(
            # 请输入场景ID。
            scene_id=xxx,
            outer_order_no='商户请求的唯一标识',
            # 要接入的认证方案。
            product_code='ID_MIN_VIDEO',
            model='FRONT_CAMERA_LIVENESS',
            # 固定值IDENTITY_CARD。
            cert_type='IDENTITY_CARD',
            cert_name='xxx',
            cert_no='xxx',
            # ip='114.xxx.xxx.xxx',
            # user_id='用户ID',
            # mobile='130xxxxxxxx',
            # device_token='通过设备指纹获取的deviceToken',
            oss_bucket_name='cn-shanghai-aliyun-cloudauth-xxxxx',
            oss_object_name='verify/xxxxx/xxxxxx.mp4',
            # 如需开启个人信息加密传输。
            # encrypt_type='SM2',
            # cert_name='BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==',
            # cert_no='BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg=='
        )
        response = ContrastFaceVerify.contrast_face_verify_auto_route(request)
        # 不支持服务自动路由。
        # response = ContrastFaceVerify.contrast_face_verify('cloudauth.cn-shanghai.aliyunc
s.com', request)
        print(response.request_id)
        print(response.code)
```

```
        print(response.message)
        print(response.result_object.passed)
        print(response.result_object.identity_info)
        print(response.result_object.sub_code)
        print(response.result_object.material_info)
        print(response.result_object.certify_id)
    @staticmethod
    def contrast_face_verify_auto_route(
            request: cloudauth_models.ContrastFaceVerifyRequest,
    ) -> cloudauth_models.ContrastFaceVerifyResponse:
        endpoints = [
            'cloudauth.cn-shanghai.aliyuncs.com',
            'cloudauth.cn-beijing.aliyuncs.com'
        ]
        for endpoint in endpoints:
            try:
                response = ContrastFaceVerify.contrast_face_verify(endpoint, request)
                if UtilClient.equal_string('500', response.code):
                    continue
                return response
            except Exception as err:
                    continue
        return None
    @staticmethod
    def contrast_face_verify(
            endpoint: str,
            request: cloudauth_models.ContrastFaceVerifyRequest,
    ) -> cloudauth_models.ContrastFaceVerifyResponse:
        client = ContrastFaceVerify.create_client(endpoint)
        # 创建RuntimeObject实例并设置运行参数。
        runtime = util_models.RuntimeOptions()
        runtime.read_timeout = 10000
        runtime.connect_timeout = 10000
        return client.contrast_face_verify(request, runtime)
if __name__ == '__main__':
    ContrastFaceVerify.main()
```

1.5.7. Go SDK调用示例

本文介绍了金融级实人认证纯服务端API的Go SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

详细信息，请参见调试ContrastFaceVerify接口。

前提条件

- 仅支持Go 1.12.0及以上版本。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

引入依赖包

```
go get github.com/alibabacloud-go/tea-rpc
go get github.com/alibabacloud-go/tea-utils
go get github.com/alibabacloud-go/tea
```

安装SDK

关于Go SDK源代码，请参见Go SDK包。

通过Go mod来管理项目依赖：

```
go get github.com/alibabacloud-go/cloudauth-20190307@v2.0.3
```

照片实人认证调用示例

关于参数的具体描述，请参见照片实人认证。

```
package main
import (
  "fmt"
  cloudauth "github.com/alibabacloud-go/cloudauth-20190307/client"
  rpc "github.com/alibabacloud-go/tea-rpc/client"
  util "github.com/alibabacloud-go/tea-utils/service"
  "github.com/alibabacloud-go/tea/tea"
)
func CreateClient (endpoint *string) (_result *cloudauth.Client, _err error) {
  config := &rpc.Config{
    // 您的AccessKey ID。
    AccessKeyId: tea.String("您的AccessKey ID"),
    // 您的AccessKey Secret。
    AccessKeySecret: tea.String("您的AccessKey Secret"),
    // 设置HTTP代理。
    // HttpProxy: tea.String("http://xx.xx.xx.xx:xxxx"),
    // 设置HTTPS代理。
    // HttpsProxy: tea.String("https://username:password@xxx.xxx.xxx.xxx:9999"),
    Endpoint: endpoint,
  }
  _result = &cloudauth.Client{}
  _result, _err = cloudauth.NewClient(config)
  return _result, _err
}
func _main () (_err error) {
  request := &cloudauth.ContrastFaceVerifyRequest{
    // 固定值。
    ProductCode: tea.String("ID_MIN"),
    // 请输入场景ID。
    SceneId: tea.Int64(xxx),
    OuterOrderNo: tea.String("商户请求的唯一标识"),
    Model: tea.String("NO_LIVENESS"),
    // 固定值IDENTITY_CARD。
    CertType: tea.String("IDENTITY_CARD"),
    CertName: tea.String("xxx"),
    CertNo: tea.String("xxx"),
```

```
    // Ip: tea.String("用户IP地址"),
    // UserId: tea.String("用户ID"),
    // Mobile: tea.String("130xxxxxxxx"),
    // DeviceToken: tea.String("通过设备指纹获取的deviceToken"),
    // 请在以下四种方式中选择其一。
    // 1.照片Base64编码。
    // 2.照片OSS Bucket名和文件名。
    // 3.照片OSS的URL地址。
    // 4.已经实人认证通过的CertifyId。
    // 方式一：照片Base64。
    // FaceContrastPicture: tea.String("人脸照片Base64编码"),
    // 方式二：照片OSS Bucket名和文件名。
    // OssBucketName: tea.String("cn-shanghai-aliyun-cloudauth-xxxxx"),
    // OssObjectName: tea.String("verify/xxxxx/xxxxxx.jpeg"),
    // 方式三：照片OSS的URL地址，公网可访问。
    FaceContrastPictureUrl: tea.String("https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-
shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg"),
    // 方式四：已经实人认证通过的CertifyId。
    // CertifyId: tea.String("0bfa7c493f850e5178b9f8613634c9xx"),
    // 如需开启个人信息加密传输。
    // EncryptType: tea.String("SM2"),
    // CertName: tea.String("BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w=="),
    // CertNo: tea.String("BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg==")
  }
  response, _err := ContrastFaceVerifyAutoRoute(request)
  // 不支持服务自动路由。
  // response, _err := ContrastFaceVerify(tea.String("cloudauth.cn-shanghai.aliyuncs.com"),
request)
  if _err != nil {
    return _err
  }
  fmt.Println(*response.RequestId)
  fmt.Println(*response.Code)
  fmt.Println(*response.Message)
  fmt.Println(*response.RequestId)
  fmt.Println(*response.ResultObject.Passed)
  fmt.Println(*response.ResultObject.IdentityInfo)
  fmt.Println(*response.ResultObject.SubCode)
  fmt.Println(*response.ResultObject.MaterialInfo)
  return _err
}
func ContrastFaceVerifyAutoRoute (request *cloudauth.ContrastFaceVerifyRequest) (_result *c
loudauth.ContrastFaceVerifyResponse, _err error) {
  endpoints := []*string{tea.String("cloudauth.cn-shanghai.aliyuncs.com"), tea.String("clou
dauth.cn-beijing.aliyuncs.com")}
  var lastResponse *cloudauth.ContrastFaceVerifyResponse
  for _, endpoint := range endpoints {
    response, _err := ContrastFaceVerify(endpoint, request)
    if _err != nil {
      var err = &tea.SDKError{}
      if _t, ok := _err.(*tea.SDKError); ok {
        err = _t
        // 网络异常，切换到下个区域调用。
        if tea.BoolValue(util.EqualString(tea.String("ServiceUnavailable"), err.Code)) {
```

```
            continue
          }
        }
        return _result, _err
      }
      if tea.BoolValue(util.EqualString(tea.String("500"), response.Code)) {
        continue
      }
      _result = response
      return _result, _err
    }
    _result = lastResponse
    return _result, _err
  }
  func ContrastFaceVerify (endpoint *string, request *cloudauth.ContrastFaceVerifyRequest) (_
  result *cloudauth.ContrastFaceVerifyResponse, _err error) {
    client, _err := CreateClient(endpoint)
    if _err != nil {
      return _result, _err
    }
    // 创建RuntimeObject实例并设置运行参数。
    runtime := &util.RuntimeOptions{}
    runtime.ReadTimeout = tea.Int(10000)
    runtime.ConnectTimeout = tea.Int(10000)
    _result = &cloudauth.ContrastFaceVerifyResponse{}
    _body, _err := client.ContrastFaceVerify(request, runtime)
    if _err != nil {
      return _result, _err
    }
    _result = _body
    return _result, _err
  }
  func main() {
    err := _main()
    if err != nil {
      panic(err)
    }
  }
```

视频实人认证调用示例

支持OSS对象存储方式和本地文件两种方式。关于参数的具体描述，请参见视频实人认证。

使用本地文件的方式（推荐）

```
package main
import (
  "fmt"
  cloudauth "github.com/alibabacloud-go/cloudauth-20190307/client"
  rpc "github.com/alibabacloud-go/tea-rpc/client"
  util "github.com/alibabacloud-go/tea-utils/service"
  "github.com/alibabacloud-go/tea/tea"
  "os"
)
```

```
func CreateClient (endpoint *string) (_result *cloudauth.Client, _err error) {
  config := &rpc.Config{
    // 您的AccessKey ID。
    AccessKeyId: tea.String("您的AccessKey ID"),
    // 您的AccessKey Secret。
    AccessKeySecret: tea.String("您的AccessKey Secret"),
    // 设置HTTP代理。
    // HttpProxy: tea.String("http://xx.xx.xx.xx:xxxx"),
    // 设置HTTPS代理。
    // HttpsProxy: tea.String("https://username:password@xxx.xxx.xxx.xxx:9999"),
    Endpoint: endpoint,
  }
  _result = &cloudauth.Client{}
  _result, _err = cloudauth.NewClient(config)
  return _result, _err
}
func _main () (_err error) {
  f, err := os.Open("/xxx/xxx.mp4")
  if err != nil {
    return err
  }
  request := &cloudauth.ContrastFaceVerifyAdvanceRequest{
    // 固定值。
    ProductCode: tea.String("ID_MIN_VIDEO"),
    // 请输入场景ID。
    SceneId: tea.Int64(xxx),
    OuterOrderNo: tea.String("商户请求的唯一标识"),
    Model: tea.String("FRONT_CAMERA_LIVENESS"),
    // 固定值IDENTITY_CARD。
    CertType: tea.String("IDENTITY_CARD"),
    CertName: tea.String("xxx"),
    CertNo: tea.String("xxx"),
    // Ip: tea.String("用户IP地址"),
    // UserId: tea.String("用户ID"),
    // Mobile: tea.String("130xxxxxxxx"),
    // DeviceToken: tea.String("通过设备指纹获取的deviceToken"),
    FaceContrastFileObject: f,
    // 如需开启个人信息加密传输。
    // EncryptType: tea.String("SM2"),
    // CertName: tea.String("BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w=="),
    // CertNo: tea.String("BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg==")
  }
  response, _err := ContrastFaceVerifyAutoRoute(request)
  // 不支持服务自动路由。
  // response, _err := ContrastFaceVerify(tea.String("cloudauth.cn-shanghai.aliyuncs.com"),
request)
  f.Close()
  if _err != nil {
    return _err
  }
  fmt.Println(*response.RequestId)
  fmt.Println(*response.Code)
  fmt.Println(*response.Message)
  fmt.Println(*response.ResultObject.Passed)
  fmt.Println(*response.ResultObject.IdentityInfo)
```

```
  fmt.Println(*response.ResultObject.IdentityInfo)
  fmt.Println(*response.ResultObject.SubCode)
  fmt.Println(*response.ResultObject.MaterialInfo)
  return _err
}
func ContrastFaceVerifyAutoRoute (request *cloudauth.ContrastFaceVerifyAdvanceRequest) (_re
sult *cloudauth.ContrastFaceVerifyResponse, _err error) {
  endpoints := []*string{tea.String("cloudauth.cn-shanghai.aliyuncs.com"), tea.String("clou
dauth.cn-beijing.aliyuncs.com")}
  var lastResponse *cloudauth.ContrastFaceVerifyResponse
  for _, endpoint := range endpoints {
    response, _err := ContrastFaceVerify(endpoint, request)
    if _err != nil {
      var err = &tea.SDKError{}
      if _t, ok := _err.(*tea.SDKError); ok {
        err = _t
        // 网络异常，切换到下个区域调用。
        if tea.BoolValue(util.EqualString(tea.String("ServiceUnavailable"), err.Code)) {
          continue
        }
      }
      return _result, _err
    }
    if tea.BoolValue(util.EqualString(tea.String("500"), response.Code)) {
      continue
    }
    _result = response
    return _result, _err
  }
  _result = lastResponse
  return _result, _err
}
func ContrastFaceVerify (endpoint *string, request *cloudauth.ContrastFaceVerifyAdvanceRequ
est) (_result *cloudauth.ContrastFaceVerifyResponse, _err error) {
  client, _err := CreateClient(endpoint)
  if _err != nil {
    return _result, _err
  }
  // 创建RuntimeObject实例并设置运行参数。
  runtime := &util.RuntimeOptions{}
  runtime.ReadTimeout = tea.Int(10000)
  runtime.ConnectTimeout = tea.Int(10000)
  _result = &cloudauth.ContrastFaceVerifyResponse{}
  _body, _err := client.ContrastFaceVerifyAdvance(request, runtime)
  if _err != nil {
    return _result, _err
  }
  _result = _body
  return _result, _err
}
func main() {
  err := _main()
  if err != nil {
    panic(err)
  }
```

```
}
```

使用OSS对象存储的方式

```
package main
import (
  "fmt"
  cloudauth "github.com/alibabacloud-go/cloudauth-20190307/client"
  rpc "github.com/alibabacloud-go/tea-rpc/client"
  util "github.com/alibabacloud-go/tea-utils/service"
  "github.com/alibabacloud-go/tea/tea"
)
func CreateClient (endpoint *string) (_result *cloudauth.Client, _err error) {
  config := &rpc.Config{
    // 您的AccessKey ID。
    AccessKeyId: tea.String("您的AccessKey ID"),
    // 您的AccessKey Secret。
    AccessKeySecret: tea.String("您的AccessKey Secret"),
    // 设置HTTP代理。
    // HttpProxy: tea.String("http://xx.xx.xx.xx:xxxx"),
    // 设置HTTPS代理。
    // HttpsProxy: tea.String("https://username:password@xxx.xxx.xxx.xxx:9999"),
    Endpoint: endpoint,
  }
  _result = &cloudauth.Client{}
  _result, _err = cloudauth.NewClient(config)
  return _result, _err
}
func _main () (_err error) {
  request := &cloudauth.ContrastFaceVerifyRequest{
    // 固定值。
    ProductCode: tea.String("ID_MIN_VIDEO"),
    // 请输入场景ID。
    SceneId: tea.Int64(xxx),
    OuterOrderNo: tea.String("商户请求的唯一标识"),
    Model: tea.String("FRONT_CAMERA_LIVENESS"),
    // 固定值IDENTITY_CARD。
    CertType: tea.String("IDENTITY_CARD"),
    CertName: tea.String("xxx"),
    CertNo: tea.String("xxx"),
    // Ip: tea.String("用户IP地址"),
    // UserId: tea.String("用户ID"),
    // Mobile: tea.String("130xxxxxxxx"),
    // DeviceToken: tea.String("通过设备指纹获取的deviceToken"),
    OssBucketName: tea.String("cn-shanghai-aliyun-cloudauth-xxxxx"),
    OssObjectName: tea.String("verify/xxxxx/xxxxxx.mp4"),
    // 如需开启个人信息加密传输。
    // EncryptType: tea.String("SM2"),
    // CertName: tea.String("BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w=="),
    // CertNo: tea.String("BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg==")
  }
  // response, _err := ContrastFaceVerifyAutoRoute(request)
  // 不支持服务自动路由。
```

```
  response, _err := ContrastFaceVerify(tea.String("cloudauth.cn-shanghai.aliyuncs.com"), re
quest)
  if _err != nil {
    return _err
  }
  fmt.Println(*response.RequestId)
  fmt.Println(*response.Code)
  fmt.Println(*response.Message)
  fmt.Println(*response.ResultObject.Passed)
  fmt.Println(*response.ResultObject.IdentityInfo)
  fmt.Println(*response.ResultObject.SubCode)
  fmt.Println(*response.ResultObject.MaterialInfo)
  fmt.Println(*response.ResultObject.CertifyId)
  return _err
}
func ContrastFaceVerifyAutoRoute (request *cloudauth.ContrastFaceVerifyRequest) (_result *c
loudauth.ContrastFaceVerifyResponse, _err error) {
  endpoints := []*string{tea.String("cloudauth.cn-shanghai.aliyuncs.com"), tea.String("clou
dauth.cn-beijing.aliyuncs.com")}
  var lastResponse *cloudauth.ContrastFaceVerifyResponse
  for _, endpoint := range endpoints {
    response, _err := ContrastFaceVerify(endpoint, request)
    if _err != nil {
      var err = &tea.SDKError{}
      if _t, ok := _err.(*tea.SDKError); ok {
        err = _t
        // 网络异常，切换到下个区域调用。
        if tea.BoolValue(util.EqualString(tea.String("ServiceUnavailable"), err.Code)) {
          continue
        }
      }
      return _result, _err
    }
    if tea.BoolValue(util.EqualString(tea.String("500"), response.Code)) {
      continue
    }
    _result = response
    return _result, _err
  }
  _result = lastResponse
  return _result, _err
}
func ContrastFaceVerify (endpoint *string, request *cloudauth.ContrastFaceVerifyRequest) (_
result *cloudauth.ContrastFaceVerifyResponse, _err error) {
  client, _err := CreateClient(endpoint)
  if _err != nil {
    return _result, _err
  }
  // 创建RuntimeObject实例并设置运行参数。
  runtime := &util.RuntimeOptions{}
  runtime.ReadTimeout = tea.Int(10000)
  runtime.ConnectTimeout = tea.Int(10000)
  _result = &cloudauth.ContrastFaceVerifyResponse{}
  _body, _err := client.ContrastFaceVerify(request, runtime)
  if _err != nil {
```

```
  if _err != nil {
    return _result, _err
  }
  _result = _body
  return _result, _err
}
func main() {
  err := _main()
  if err != nil {
    panic(err)
  }
}
```

1.5.8. .NET SDK调用示例

本文介绍了金融级实人认证纯服务端API的.NET SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

详细信息，请参见调试ContrastFaceVerify接口。

前提条件

- 需要使用的.NET Framework版本在4.5.0版本及以上或者.NET Standard版本在2.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于.NET SDK源代码，请参见.NET SDK包。

SDK有以下两种安装方式，请选择合适的方式安装：

- 通过.NET命令行安装。

```
dotnet add package AlibabaCloud.SDK.Cloudauth20190307 --version 2.0.3
```

- 使用包管理安装。

```
Install-Package AlibabaCloud.SDK.Cloudauth20190307 -Version 2.0.3
```

照片实人认证调用示例

关于参数的具体描述，请参见照片实人认证。

```
using System;
using System.Collections.Generic;
using Tea;
namespace ContrastFaceVerify
{
    public class Program
    {
        public static AlibabaCloud.SDK.Cloudauth20190307.Client CreateClient(string endpoin
t)
```

```
        {
            AlibabaCloud.RPCClient.Models.Config config = new AlibabaCloud.RPCClient.Models
.Config
            {
                // 您的AccessKey ID。
                AccessKeyId = "您的AccessKey ID",
                // 您的AccessKey Secret。
                AccessKeySecret = "您的AccessKey Secret",
                // 设置HTTP代理。
                // HttpProxy = "http://xx.xx.xx.xx:xxxx",
                // 设置HTTPS代理。
                // HttpsProxy = "https://username:password@xxx.xxx.xxx.xxx:9999",
                Endpoint = endpoint,
            };
            return new AlibabaCloud.SDK.Cloudauth20190307.Client(config);
        }
        public static void Main(string[] args)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyRequest request = n
ew AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyRequest
            {
                // 固定值。
                ProductCode = "ID_MIN",
                // 请输入场景ID。
                SceneId = xxx,
                OuterOrderNo = "商户请求的唯一标识",
                Model = "NO_LIVENESS",
                // 固定值IDENTITY_CARD。
                CertType = "IDENTITY_CARD",
                CertName = "张三",
                CertNo = "xxx",
                //Ip = "用户IP地址",
                //UserId = "用户ID",
                //Mobile = "130xxxxxxxx",
                //DeviceToken = "通过设备指纹获取的deviceToken",
                // 请在以下四种方式中选择其一。
                // 1.照片Base64编码。
                // 2.照片OSS Bucket名和文件名。
                // 3.照片OSS的URL地址。
                // 4.已经实人认证通过的CertifyId。
                // 方式一：照片Base64。
                //FaceContrastPicture = "人脸照片Base64编码",
                // 方式二：照片OSS Bucket名和文件名。
                // OssBucketName = "cn-shanghai-aliyun-cloudauth-xxxxx",
                // OssObjectName = "verify/xxxxx/xxxxxx.jpeg",
                // 方式三：照片OSS的URL地址，公网可访问。
                FaceContrastPictureUrl = "https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-c
n-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg",
                // 方式四：已经实人认证通过的CertifyId。
                // CertifyId = "0bfa7c493f850e5178b9f8613634c9xx",
                // 如需开启个人信息加密传输。
                // EncryptType = "SM2",
                // CertName = "BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==",
                // CertNo = "BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg=="
            };
```

```
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse response =
ContrastFaceVerifyAutoRoute(request);
            // 不支持服务自动路由。
            //AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse response
= ContrastFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", request);
            Console.WriteLine(response.RequestId);
            Console.WriteLine(response.Code);
            Console.WriteLine(response.Message);
            Console.WriteLine(response.ResultObject.Passed);
            Console.WriteLine(response.ResultObject.IdentityInfo);
            Console.WriteLine(response.ResultObject.SubCode);
            Console.WriteLine(response.ResultObject.MaterialInfo);
            Console.WriteLine(response.ResultObject.CertifyId);
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse
ContrastFaceVerifyAutoRoute(AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyReq
uest request)
        {
            List<string> endpoints = new List<string>
            {
                "cloudauth.cn-shanghai.aliyuncs.com",
                "cloudauth.cn-beijing.aliyuncs.com"
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse lastRespon
se = null;
            foreach (var endpoint in endpoints)
            {
                try
                {
                    AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse re
sponse = ContrastFaceVerify(endpoint, request);
                    if (AlibabaCloud.TeaUtil.Common.EqualString("500", response.Code))
                    {
                        continue;
                    }
                    return response;
                }
                catch (TeaException err)
                {
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
                catch (Exception _err)
                {
                    TeaException err = new TeaException(new Dictionary<string, object>
                    {
                        { "message", _err.Message }
                    });
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
```

```
ode))
                    {
                        continue;
                    }
                }
            }
            return lastResponse;
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse
ContrastFaceVerify(string endpoint, AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceV
erifyRequest request)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Client client = CreateClient(endpoint);
            // 创建RuntimeObject实例并设置运行参数。
            AlibabaCloud.TeaUtil.Models.RuntimeOptions runtime = new AlibabaCloud.TeaUtil.M
odels.RuntimeOptions();
            runtime.ReadTimeout = 10000;
            runtime.ConnectTimeout = 10000;
            return client.ContrastFaceVerify(request, runtime);
        }
    }
}
```

视频实人认证调用示例

支持OSS对象存储方式和本地文件两种方式。关于参数的具体描述，请参见视频实人认证。

使用本地文件的方式（推荐）

```
using System;
using System.Collections.Generic;
using Tea;
using System.IO;
namespace ContrastFaceVerify
{
    public class Program
    {
        public static AlibabaCloud.SDK.Cloudauth20190307.Client CreateClient(string endpoin
t)
        {
            AlibabaCloud.RPCClient.Models.Config config = new AlibabaCloud.RPCClient.Models
.Config
            {
                // 您的AccessKey ID。
                AccessKeyId = "您的AccessKey ID",
                // 您的AccessKey Secret。
                AccessKeySecret = "您的AccessKey Secret",
                // 设置HTTP代理。
                // HttpProxy = "http://xx.xx.xx.xx:xxxx",
                // 设置HTTPS代理。
                // HttpsProxy = "https://username:password@xxx.xxx.xxx.xxx:9999",
                Endpoint = endpoint,
            };
            return new AlibabaCloud.SDK.Cloudauth20190307.Client(config);
```

```
        }
        public static void Main(string[] args)
        {
            FileStream fs = new FileStream("本地文件路径", FileMode.Open);
            AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyAdvanceRequest requ
est = new AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyAdvanceRequest
            {
                // 固定值。
                ProductCode = "ID_MIN_VIDEO",
                // 请输入场景ID。
                SceneId = xxx,
                OuterOrderNo = "商户请求的唯一标识",
                Model = "FRONT_CAMERA_LIVENESS",
                // 固定值IDENTITY_CARD。
                CertType = "IDENTITY_CARD",
                CertName = "张三",
                CertNo = "xxx",
                //Ip = "用户IP地址",
                //UserId = "用户ID",
                //Mobile = "130xxxxxxxx",
                //DeviceToken = "通过设备指纹获取的deviceToken",
                FaceContrastFileObject = fs,
                // 如需开启个人信息加密传输。
                // EncryptType = "SM2",
                // CertName = "BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==",
                // CertNo = "BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg=="
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse response =
ContrastFaceVerifyAutoRoute(request);
            // 不支持服务自动路由。
            //AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse response
= ContrastFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", request);
            Console.WriteLine(response.RequestId);
            Console.WriteLine(response.Code);
            Console.WriteLine(response.Message);
            Console.WriteLine(response.ResultObject.Passed);
            Console.WriteLine(response.ResultObject.IdentityInfo);
            Console.WriteLine(response.ResultObject.SubCode);
            Console.WriteLine(response.ResultObject.MaterialInfo);
            Console.WriteLine(response.ResultObject.CertifyId);
            fs.Close();
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse
ContrastFaceVerifyAutoRoute(AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyAdv
anceRequest request)
        {
            List<string> endpoints = new List<string>
            {
                "cloudauth.cn-shanghai.aliyuncs.com",
                "cloudauth.cn-beijing.aliyuncs.com"
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse lastRespon
se = null;
            foreach (var endpoint in endpoints)
```

```
            {
                try
                {
                    AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse re
sponse = ContrastFaceVerify(endpoint, request);
                    if (AlibabaCloud.TeaUtil.Common.EqualString("500", response.Code))
                    {
                        continue;
                    }
                    return response;
                }
                catch (TeaException err)
                {
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
                catch (Exception _err)
                {
                    TeaException err = new TeaException(new Dictionary<string, object>
                    {
                        { "message", _err.Message }
                    });
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
            }
            return lastResponse;
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse
ContrastFaceVerify(string endpoint, AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceV
erifyAdvanceRequest request)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Client client = CreateClient(endpoint);
            // 创建RuntimeObject实例并设置运行参数。
            AlibabaCloud.TeaUtil.Models.RuntimeOptions runtime = new AlibabaCloud.TeaUtil.M
odels.RuntimeOptions();
            runtime.ReadTimeout = 10000;
            runtime.ConnectTimeout = 10000;
            return client.ContrastFaceVerifyAdvance(request, runtime);
        }
    }
}
```

使用OSS对象存储的方式

```
using System;
```

```
using System.Collections.Generic;
using Tea;
namespace ContrastFaceVerify
{
    public class Program
    {
        public static AlibabaCloud.SDK.Cloudauth20190307.Client CreateClient(string endpoin
t)
        {
            AlibabaCloud.RPCClient.Models.Config config = new AlibabaCloud.RPCClient.Models
.Config
            {
                // 您的AccessKey ID。
                AccessKeyId = "您的AccessKey ID",
                // 您的AccessKey Secret。
                AccessKeySecret = "您的AccessKey Secret",
                // 设置HTTP代理。
                // HttpProxy = "http://xx.xx.xx.xx:xxxx",
                // 设置HTTPS代理。
                // HttpsProxy = "https://username:password@xxx.xxx.xxx.xxx:9999",
                Endpoint = endpoint,
            };
            return new AlibabaCloud.SDK.Cloudauth20190307.Client(config);
        }
        public static void Main(string[] args)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyRequest request = n
ew AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyRequest
            {
                // 固定值。
                ProductCode = "ID_MIN_VIDEO",
                // 请输入场景ID。
                SceneId = xxx,
                OuterOrderNo = "商户请求的唯一标识",
                Model = "FRONT_CAMERA_LIVENESS",
                // 固定值IDENTITY_CARD。
                CertType = "IDENTITY_CARD",
                CertName = "张三",
                CertNo = "xxx",
                //Ip = "用户IP地址",
                //UserId = "用户ID",
                //Mobile = "130xxxxxxxx",
                //DeviceToken = "通过设备指纹获取的deviceToken",
                OssBucketName = "cn-shanghai-aliyun-cloudauth-xxxxx",
                OssObjectName = "verify/xxxxx/xxxxxx.mp4",
                // 如需开启个人信息加密传输。
                // EncryptType = "SM2",
                // CertName = "BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==",
                // CertNo = "BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg=="
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse response =
ContrastFaceVerifyAutoRoute(request);
            // 不支持服务自动路由。
            //AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse response
```

```
= ContrastFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", request);
            Console.WriteLine(response.RequestId);
            Console.WriteLine(response.Code);
            Console.WriteLine(response.Message);
            Console.WriteLine(response.ResultObject.Passed);
            Console.WriteLine(response.ResultObject.IdentityInfo);
            Console.WriteLine(response.ResultObject.SubCode);
            Console.WriteLine(response.ResultObject.MaterialInfo);
            Console.WriteLine(response.ResultObject.CertifyId);
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse
ContrastFaceVerifyAutoRoute(AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyReq
uest request)
        {
            List<string> endpoints = new List<string>
            {
                "cloudauth.cn-shanghai.aliyuncs.com",
                "cloudauth.cn-beijing.aliyuncs.com"
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse lastRespon
se = null;
            foreach (var endpoint in endpoints)
            {
                try
                {
                    AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse re
sponse = ContrastFaceVerify(endpoint, request);
                    if (AlibabaCloud.TeaUtil.Common.EqualString("500", response.Code))
                    {
                        continue;
                    }
                    return response;
                }
                catch (TeaException err)
                {
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
                catch (Exception _err)
                {
                    TeaException err = new TeaException(new Dictionary<string, object>
                    {
                        { "message", _err.Message }
                    });
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
```

```
                }
            }
            return lastResponse;
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceVerifyResponse
ContrastFaceVerify(string endpoint, AlibabaCloud.SDK.Cloudauth20190307.Models.ContrastFaceV
erifyRequest request)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Client client = CreateClient(endpoint);
            // 创建RuntimeObject实例并设置运行参数。
            AlibabaCloud.TeaUtil.Models.RuntimeOptions runtime = new AlibabaCloud.TeaUtil.M
odels.RuntimeOptions();
            runtime.ReadTimeout = 10000;
            runtime.ConnectTimeout = 10000;
            return client.ContrastFaceVerify(request, runtime);
        }
    }
}
```

1.5.9. Node.js SDK调用示例

本文介绍了金融级实人认证纯服务端API的Node.js SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

详细信息，请参见调试ContrastFaceVerify接口。

前提条件

- 需要使用的Node.js版本在8.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于Node.js SDK源代码，请参见Node.js SDK。

建议您使用npm管理依赖包：

```
npm install @alicloud/tea-util
npm install @alicloud/rpc-client
npm install @alicloud/cloudauth20190307@2.0.3
```

照片实人认证调用示例

关于参数的具体描述，请参见照片实人认证。

```
const Config = require("@alicloud/rpc-client");
const CloudAuth = require("@alicloud/cloudauth20190307");
const CloudAuthClient = CloudAuth.default;
const {RuntimeOptions} = require("@alicloud/tea-util");
     function createClient(endpoint) {
        // 通过以下代码创建Config实例并初始化。
```

```
        // 通过以下代码创建Config实例并初始化。
        const config = new Config.Config({
            // 您的AccessKey ID。
            accessKeyId: "您的AccessKey ID",
            // 您的AccessKey Secret。
            accessKeySecret: "您的AccessKey Secret",
            // 设置Endpoint
            endpoint: endpoint,
            // 设置HTTP代理。
            // httpProxy: "http://xxx.xxx.xxx.xxx:9999",
            // 设置HTTPS代理。
            // httpsProxy: "https://username:password@xxx.xxx.xxx.xxx:9999",
        });
        return new CloudAuthClient(config)
    }
    async function contrastFaceVerify(endpoint, request) {
        const client = createClient(endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        const runtime = new RuntimeOptions([]);
        runtime.readTimeout=10000;
        runtime.connectTimeout=10000;
        return await client.contrastFaceVerify(request, runtime);
    }
    async function contrastFaceVerifyAutoRoute(request) {
        const endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        for (const endpoint of endpoints) {
            try {
                const response = await contrastFaceVerify(endpoint, request);
                if (response.code === '500') {
                    continue;
                }
                return response;
            } catch (err) {
                continue;
            }
        }
        return null;
    }
    async function main(){
        // 通过以下代码创建API请求并设置参数。
        const request = new CloudAuth.ContrastFaceVerifyRequest({
        // 请输入场景ID。
        sceneId: xxx,
        // 设置商户请求的唯一标识。
        outerOrderNo: "xxx",
        // 要接入的认证方案。
        productCode: "ID_MIN",
        // 模式。
        model: "NO_LIVENESS",
        // 固定值。
        certType: "IDENTITY_CARD",
        certName: "张三",
```

```
        certNo: "xxx",
        // mobile: "130xxxxxxxx",
        // 请在以下四种方式中选择其一。
        // 1.人脸图片的Base64编码。
        // 2.已授权OSS空间的Bucket名和已授权OSS空间的文件名。
        // 3.人脸图片的HTTPS或HTTP地址。
        // 4.已通过智能核身得到的CertifyId，认证时的图片作为比对图片。
        // 方式一：人脸图片的Base64编码。
        // faceContrastPicture: "/9j/4AAQSkZJRgABAQAASxxxxxxx",
        // 方式二：已授权OSS空间的Bucket名和已授权OSS空间的文件名。
        // ossBucketName: "cn-shanghai-aliyun-cloudauth-xxxxx",
        // ossObjectName: "verify/xxxxx/xxxxxx.jpeg",
        //方式三：人脸图片的HTTPS或HTTP地址。
        faceContrastPictureUrl: "https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-shangha
i.aliyuncs.com/smart/xxxxx/xxxxx.jpeg",
        // 方式四：已通过智能核身得到的CertifyId，认证时的图片作为比对图片。
        // certifyId: "0bfa7c493f850e5178b9f8613634c9xx",
        // ip: "114.xxx.xxx.xxx",
        // userId: "12345xxxx",
        // deviceToken: "xxxxxxx",
        // 如需开启个人信息加密传输。
        // encryptType: "SM2",
        // certName: "BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==",
        // certNo: "BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg=="
        });
        // 推荐，支持服务路由。
        const response = await contrastFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        // const response = await contrastFaceVerify("cloudauth.cn-shanghai.aliyuncs.com",
request);
        console.log(response.requestId);
        console.log(response.code);
        console.log(response.message);
        console.log(response.resultObject.certifyId);
        }
main().then(function (response) {});
```

视频实人认证调用示例

支持OSS对象存储方式和本地文件两种方式。关于参数的具体描述，请参见视频实人认证。

使用本地文件的方式（推荐）

```
const Config = require("@alicloud/rpc-client");
const CloudAuth = require("@alicloud/cloudauth20190307");
const CloudAuthClient = CloudAuth.default;
const {RuntimeOptions} = require("@alicloud/tea-util");
const fs = require("fs");
     function createClient(endpoint) {
        // 通过以下代码创建Config实例并初始化。
        const config = new Config.Config({
            // 您的AccessKey ID。
            accessKeyId: "您的AccessKey ID",
            // 您的AccessKey Secret。
```

```
            accessKeySecret: "您的AccessKey Secret",
            // 设置Endpoint。
            endpoint: endpoint,
            // 设置HTTP代理。
            // httpProxy: "http://xxx.xxx.xxx.xxx:9999",
            // 设置HTTPS代理。
            // httpsProxy: "https://username:password@xxx.xxx.xxx.xxx:9999",
        });
        return new CloudAuthClient(config)
    }
    async function contrastFaceVerify(endpoint, request) {
        const client = createClient(endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        const runtime = new RuntimeOptions([]);
        runtime.readTimeout=10000;
        runtime.connectTimeout=10000;
        return await client.contrastFaceVerifyAdvance(request, runtime);
    }
    async function contrastFaceVerifyAutoRoute(request) {
        const endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        for (const endpoint of endpoints) {
            try {
                const response = await contrastFaceVerify(endpoint, request);
                if (response.code === '500') {
                    continue;
                }
                return response;
            } catch (err) {
                continue;
            }
        }
        return null;
    }
    async function main(){
        const fileStream = fs.createReadStream("本地文件路径");
        // 通过以下代码创建API请求并设置参数。
        const request = new CloudAuth.ContrastFaceVerifyAdvanceRequest({
        // 请输入场景ID。
        sceneId: xxx,
        // 设置商户请求的唯一标识。
        outerOrderNo: "xxx",
        // 要接入的认证方案。
        productCode: "ID_MIN_VIDEO",
        // 模式。
        model: "FRONT_CAMERA_LIVENESS",
        // 固定值。
        certType: "IDENTITY_CARD",
        certName: "张三",
        certNo: "xxx",
        faceContrastFileObject: fileStream
        // mobile: "130xxxxxxxx",
```

```
        // ip: "114.xxx.xxx.xxx",
        // userId: "12345xxxx",
        // deviceToken: "xxxxxxx",
        // 如需开启个人信息加密传输。
        // encryptType: "SM2",
        // certName: "BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==",
        // certNo: "BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg=="
        });
        // 推荐，支持服务路由。
        const response = await contrastFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        // const response = await contrastFaceVerify("cloudauth.cn-shanghai.aliyuncs.com",
request);
        console.log(response.requestId);
        console.log(response.code);
        console.log(response.message);
        console.log(response.resultObject.certifyId);
        fileStream.close();
    }
main().then(function (response) {});
```

使用OSS对象存储的方式

```
const Config = require("@alicloud/rpc-client");
const CloudAuth = require("@alicloud/cloudauth20190307");
const CloudAuthClient = CloudAuth.default;
const {RuntimeOptions} = require("@alicloud/tea-util");
     function createClient(endpoint) {
        // 通过以下代码创建Config实例并初始化。
        const config = new Config.Config({
            // 您的AccessKey ID。
            accessKeyId: "您的AccessKey ID",
            // 您的AccessKey Secret。
            accessKeySecret: "您的AccessKey Secret",
            // 设置Endpoint。
            endpoint: endpoint,
            // 设置HTTP代理。
            // httpProxy: "http://xxx.xxx.xxx.xxx:9999",
            // 设置HTTPS代理。
            // httpsProxy: "https://username:password@xxx.xxx.xxx.xxx:9999",
        });
        return new CloudAuthClient(config)
    }
    async function contrastFaceVerify(endpoint, request) {
        const client = createClient(endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        const runtime = new RuntimeOptions([]);
        runtime.readTimeout=10000;
        runtime.connectTimeout=10000;
        return await client.contrastFaceVerify(request, runtime);
    }
    async function contrastFaceVerifyAutoRoute(request) {
        const endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
```

```
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        for (const endpoint of endpoints) {
            try {
                const response = await contrastFaceVerify(endpoint, request);
                if (response.code === '500') {
                    continue;
                }
                return response;
            } catch (err) {
                continue;
            }
        }
        return null;
    }
    async function main(){
        // 通过以下代码创建API请求并设置参数。
        const request = new CloudAuth.ContrastFaceVerifyRequest({
        // 请输入场景ID。
        sceneId: xxx,
        // 设置商户请求的唯一标识。
        outerOrderNo: "xxx",
        // 要接入的认证方案。
        productCode: "ID_MIN_VIDEO",
        // 模式。
        model: "FRONT_CAMERA_LIVENESS",
        // 固定值。
        certType: "IDENTITY_CARD",
        certName: "张三",
        certNo: "xxx",
        ossBucketName: "cn-shanghai-aliyun-cloudauth-xxxxx",
        ossObjectName: "verify/xxxxx/xxxxxx.mp4",
        // mobile: "130xxxxxxxx",
        // ip: "114.xxx.xxx.xxx",
        // userId: "12345xxxx",
        // deviceToken: "xxxxxxx",
        // 如需开启个人信息加密传输。
        // encryptType: "SM2",
        // certName: "BCRD/7ZkNy7Q*****M1BMBezZe8GaYHrLwyJv558w==",
        // certNo: "BMjsstxK3S4b1YH*****Pet8ECObfxmLN92SLsNg=="
        });
        // 推荐，支持服务路由。
        const response = await contrastFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        // const response = await contrastFaceVerify("cloudauth.cn-shanghai.aliyuncs.com",
request);
        console.log(response.requestId);
        console.log(response.code);
        console.log(response.message);
        console.log(response.resultObject.certifyId);
        }
main().then(function (response) {});
```

1.6. 参数加密说明

金融级实人认证方案支持在认证请求中针对个人信息（姓名、身份证号码）进行SM2加密传输。您可以通过公钥对认证请求中的个人信息参数进行加密。本文介绍如何开启SM2参数加密。

公钥

以下为金融级实人认证方案中刷脸认证时使用的SM2公钥。

```
02cd77e007bdc86eeaf9a479ba7a2c22bc0a517ccb3a6975c3f94b4ac93347dea6
```

Java语言示例

添加Maven依赖

```
<dependency>
  <groupId>org.bouncycastle</groupId>
  <artifactId>bcprov-jdk15on</artifactId>
  <version>1.65</version>
</dependency>
```

示例代码

```
import org.bouncycastle.asn1.gm.GMNamedCurves;
import org.bouncycastle.asn1.x9.X9ECParameters;
import org.bouncycastle.jcajce.provider.asymmetric.ec.BCECPublicKey;
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import org.bouncycastle.jce.spec.ECParameterSpec;
import org.bouncycastle.jce.spec.ECPublicKeySpec;
import org.bouncycastle.math.ec.ECPoint;
import org.bouncycastle.util.encoders.Hex;
import javax.crypto.Cipher;
import java.security.KeyFactory;
import java.util.Base64;
public class SM2Utils2 {
    /**
     * SM2加密
     *
     * @param publicKey 公钥
     * @param content   内容
     * @return 密文
     */
    public static String encrypt(String publicKey, String content) {
        String contentEncrypt = "";
        try {
            BouncyCastleProvider provider = new BouncyCastleProvider();
            // 获取SM2相关参数
            X9ECParameters parameters = GMNamedCurves.getByName("sm2p256v1");
            // 椭圆曲线参数规格
            ECParameterSpec ecParameterSpec = new ECParameterSpec(parameters.getCurve(), pa
rameters.getG(), parameters.getN(), parameters.getH());
            // 将公钥HEX字符串转换为椭圆曲线对应的点
            ECPoint ecPoint = parameters.getCurve().decodePoint(Hex.decode(publicKey));
            // 获取椭圆曲线KEY生成器
            KeyFactory keyFactory = KeyFactory.getInstance("EC", provider);
            // 将椭圆曲线点转为公钥KEY对象
            BCECPublicKey bcecPublicKey = (BCECPublicKey) keyFactory.generatePublic(new ECP
ublicKeySpec(ecPoint, ecParameterSpec));
            // 获取SM2加密器
            Cipher cipher = Cipher.getInstance("SM2", provider);
            // 初始化为加密模式
            cipher.init(Cipher.ENCRYPT_MODE, bcecPublicKey);
            // 加密并编码为base64格式
            contentEncrypt = Base64.getEncoder().encodeToString(cipher.doFinal(content.getB
ytes()));
        } catch (Exception e) {
            e.printStackTrace();
        }
        return contentEncrypt;
    }
}
```

2.金融级多因子意愿认证方案

2.1. 接入流程

无线SDK和服务端接入适用于您自身已经开发了手机应用App，且希望通过该App对用户进行线上认证的场景。

前置条件

您必须了解通过HTTP、HTTPS原生调用金融级实人认证服务接口时，构造请求签名的方法。更多签名机制的信息，请参见API签名机制。

操作步骤

1. 开通金融级实人认证服务。
2. 添加认证场景。

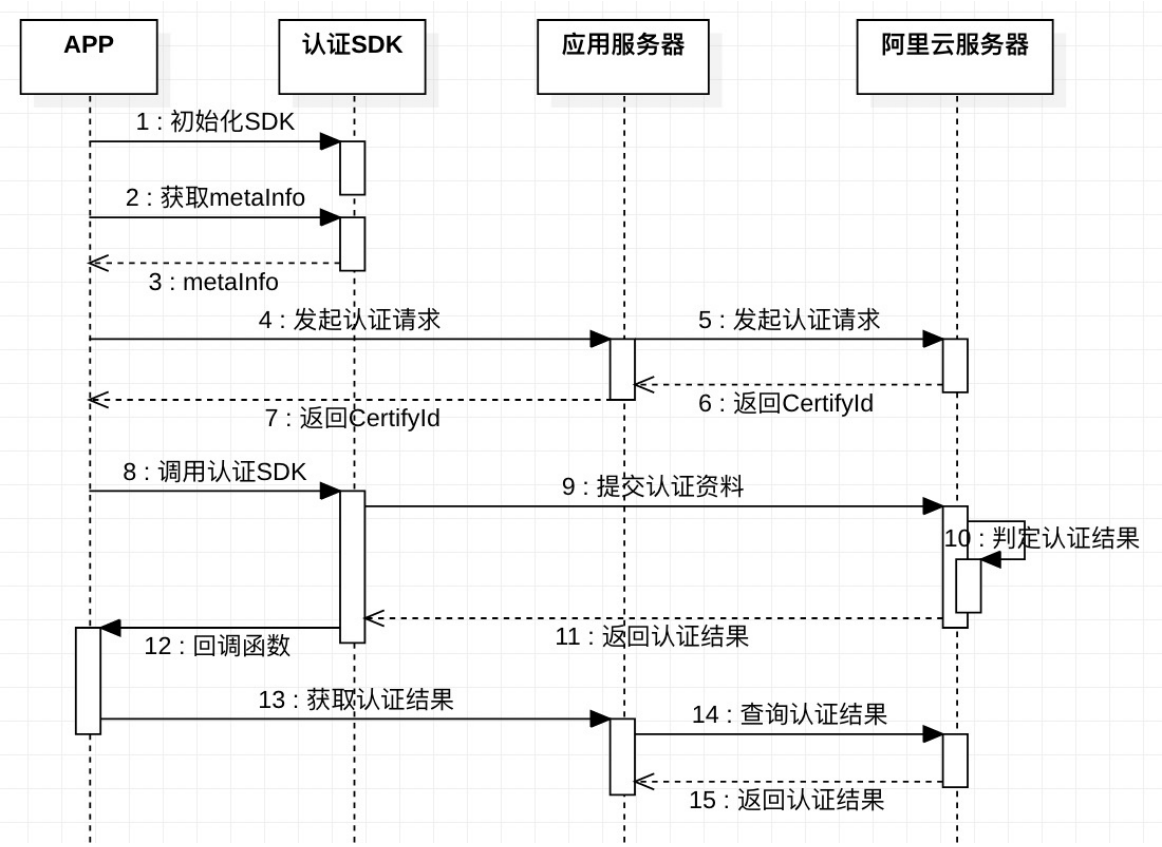
   > **说明** 基于不同业务场景，您可以在添加认证场景时，手动配置是否授权阿里云访问指定OSS存储空间用于存储认证的资料图片。

3. 如果需要将认证过程中的人脸采集照片存储到OSS空间，您需要授权金融级实人认证访问OSS存储空间。
4. 设置多因子意愿认证。具体操作，请参见设置意愿认证。
5. 接入金融级多因子意愿认证方案。

   在您完成上述步骤后，您可以进行无线SDK+服务端接入，接入步骤如下：

   i. 调用服务端初始化接口。
   ii. 调用接入客户端SDK。
   iii. 调用服务端查询接口。

   您可以查看下方调用时序图了解详细接入步骤。



系统调用时序说明表

序号	说明
1	初始化认证SDK。
2	App调用认证SDK获取MetaInfo。
3	认证SDK返回MetaInfo。
4	App向应用服务端发起认证请求并传递MetaInfo。
5	应用服务端向阿里云服务端调用发起认证请求接口。
6	阿里云服务端返回CertifyId给应用服务端。
7	应用服务端将CertifyId传递给App。
8	App调用认证SDK。
9	认证SDK将用户提交的认证资料传递给阿里云服务端。
10	阿里云服务端根据资料判定认证结果。
11	阿里云服务端将认证结果返回给SDK。
12	SDK通过回调函数指引App获取认证状态。

序号	说明
13	App向应用服务端查询认证状态。
14	应用服务端向阿里云服务端调用查询认证结果接口。
15	阿里云服务端将认证结果返回给应用服务端。

在您了解接入步骤后，您可以通过以下方式接入金融级多因子意愿认证方案。

- 进行服务端接入，请参见InitFaceVerify。
- 在应用客户端集成SDK，请参见以下文档：
  - Android意愿认证接入
  - iOS意愿认证接入

2.2. 服务端接入

2.2.1. InitFaceVerify

本文介绍如何通过调用InitFaceVerify接口发起金融级多因子意愿认证请求。

发起认证请求

接口名：**InitFaceVerify**。

服务地址：`cloudauth.aliyuncs.com`。

请求方法：HTTPS POST和GET。

接口说明：每次开始认证前通过本接口获取CertifyId，用来串联认证请求中的各个接口。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	要接入的认证场景ID。该ID在控制台创建认证场景后自动生成。	100000****
OuterOrderNo	String	是	您自定义的业务唯一标识，用于后续定位排查问题使用。 支持长度为32位的字母和数字的组合，请确保唯一。	e0c34a77f5ac40a5aa5e6ed20c35****
ProductCode	String	是	要接入的认证方案。取值**MFVC**。 使用**MFVC**认证方案，您的用户需要进行活体认证和意愿表达。	MFVC

名称	类型	是否必选	描述	示例值
CertType	String	否	您的终端用户传入的证件类型。 当前仅支持身份证，取值固定为IDENTITY_CARD。 说明 在控制台进行意愿认证设置时，**人脸对比源**设置为**权威数据源**时，CertType为必填字段。	IDENTITY_CARD
CertName	String	否	您的终端用户的真实姓名。 说明 在控制台进行意愿认证设置时，**人脸对比源**设置为**权威数据源**时，CertName为必填字段。	张三
CertNo	String	否	您的终端用户的证件号码。 说明 在控制台进行意愿认证设置时，**人脸对比源**设置为**权威数据源**时，CertNo为必填字段。	330103**********
MetaInfo	String	是	MetaInfo环境参数，需要通过客户端SDK获取。	{"zimVer":"3.0.0","appVersion": "1","bioMetaInfo": "4.1.0:11501568,0","appName": "com.aliyun.antcloudauth","deviceType": "ios","osVersion": "iOS 10.3.2","apdidToken": "","deviceModel": "iPhone9,1"}
Mobile	String	否	您的终端用户的手机号码。	130********
Ip	String	否	您的终端用户的IP。	114.100.XX.XX

名称	类型	是否必选	描述	示例值
UserId	String	否	您自定义的用户ID，请保持唯一。 说明 以下两种场景，该参数为必填字段。 • **意愿认证模式**设置为**声纹注册模式**的场景。 • **意愿认证模式**设置为其他模式，但开启**声纹对比**的场景。	123456789
CallbackUrl	String	否	认证结果的回调通知地址，必须以 `https` 开头。 平台在完成认证后会回调该地址，并自动添加 `certifyId` 、 `passed` 和 `subcode` 字段，示例： `https://www.aliyun.com?certifyId=xxxx&passed=T&subcode=200` 。	https://www.aliyun.com
CallbackToken	String	否	安全Token，由您自行生成，用于防重复、防篡改校验。 如果设置了该值会在回调地址显示**CallbackToken**字段。	NMjvQanQgplBSaEI0sL86WnQplB
AuthId	String	否	用户授权ID，最大长度为64位字符。	92d46b9e9e2d703f2897f350d5bd4149

返回数据

名称	类型	描述	示例值
RequestId	String	请求ID。	130A2C10-B9EE-4D84-88E3-5384FF039795
Message	String	请求消息的响应信息。	success
Code	String	返回码。关于返回码的具体信息，请参见返回Code和Message说明。	200
ResultObject.CertifyId	String	实人认证唯一标识。	91707dc296d469ad38e4c5efa6a0****

返回Code和Message说明

Code	Message	描述
200	success	成功。
400	参数不能为空	参数不能为空。
401	参数非法	非法参数。传入的姓名、身份证号码长度必须符合国家标准且不得包含英文字母等特殊字符。
402	应用配置不存在	应用配置不存在。
404	认证场景配置不存在	认证场景配置不存在，请先在控制台上创建认证场景。
410	未开通服务	未开通OSS产品或未完成OSS读写授权，请登录控制台完成授权。具体操作，请参见授权金融级实人认证访问OSS存储空间
411	RAM无权限	需要给RAM用户授予**AliyunAntCloudAuthFullAccess**的操作权限。具体操作，请参见授权RAM用户访问服务。
412	欠费中	金融级实人认证或OSS存在欠费，请充值后操作。
414	设备类型不支持	当前移动设备不支持刷脸认证，请更换设备后操作。
415	SDK版本不支持	当前认证SDK版本不支持刷脸认证，请升级SDK后操作。
416	系统版本不支持	当前操作系统版本不支持刷脸认证，请升级系统或更换设备操作。
417	无法使用刷脸服务	当前身份信息无可用比对源，导致此错误的原因请参考认证结果错误码ResultObject.SubCode）201、202和203的描述说明。更多内容，请参见DescribeFaceVerify。
418	刷脸失败次数过多	当天刷脸认证次数过多，请明天再试。 **说明** 一个身份信息在一天内最多可以认证失败20次，无法调整更多的认证失败次数。
438	暂时无法使用声纹服务	已开启**声纹对比**功能，但未注册过声纹。请先进行声纹注册，具体操作，请参见设置意愿认证。
500	系统错误	系统内部错误，请反馈工程师排查。

2.2.2. DescribeFaceVerify

本文介绍如何调用DescribeFaceVerify接口查询金融级多因子意愿认证的认证结果。

接口说明

接口名：**DescribeFaceVerify**。

服务地址：`cloudauth.aliyuncs.com`。

请求方法：HTTPS POST和GET。

接口说明：当您收到回调通知之后，可以在服务端通过该接口获取相应的认证状态和认证资料。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	认证场景ID。	100000****
CertifyId	String	是	实人认证唯一标识。该值需要调用InitFaceVerify获取。	91707dc296d469ad38e4c5efa6a0****

返回数据

名称	类型	描述	示例值
RequestId	String	请求ID。	130A2C10-B9EE-4D84-88E3-5384FF039795
Message	String	请求信息的响应消息。	success
Code	String	返回码，200指接口响应成功。详细认证结果判断请参见ResultObject.SubCode。	200
ResultObject.Passed	String	认证结果。取值： • **T**：通过。 • **F**：不通过。 **说明** 判断认证结果请以ResultObject.Passed字段为准。	T
ResultObject.SubCode	String	认证结果描述。更多信息，请参见ResultObject.SubCode错误码说明。 **说明** 判断认证结果请以ResultObject.Passed字段为准。	200
ResultObject.IdentityInfo	String	认证的主体信息，一般的认证场景返回为空。	null
ResultObject.DeviceToken	String	设备Token。	McozS1ZWRcRZStlERcZZo_QOytx5jcgZoZJEoRLOxxxxxxx

名称	类型	描述	示例值
ResultObject.MaterialInfo	String	认证主体附件信息，主要为图片类材料。关于JSON格式，请参见下方的ResultObject.MaterialInfo的JSON格式示例。	{"faceAttack":"F","faceOcclusion":"F","facialPictureFront":{"faceAttackScore":0.00008597839769208804,"qualityScore":88.3615493774414,"verifyScore":50.28594166529785,"pictureUrl":"https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg","ossBucketName":"cn-shanghai-aliyun-cloudauth-126005125163xxxx","ossObjectName":"verify/1234567890/6ba7bcfccf33f56cdb44ed086f36ce3e0.jpeg"}}

ResultObject.MaterialInfo的JSON格式示例

```
{
    //是否为攻击：攻击为T，非攻击为F。
    "faceAttack": "T",
    // 是否有脸部遮挡：有脸部遮挡为T，否则为F。
    "faceOcclusion": "F",
    // 认证的照片信息。
    "facialPictureFront": {
        // 人脸攻击分。
         "faceAttackScore": 0.00008597839769208804,
        // 人脸和身份证照片比对分（仅当product code=ID_PLUS时，认证完成返回）。
        "idCardVerifyScore": 50.99797734769861,
        // 照片存放客户上海区域的OSS里，此为照片的Bucket名称。
        "ossBucketName": "cn-shanghai-aliyun-cloudauth-1260051251634779",
        // 照片存放客户上海区域的OSS里，此为照片的文件名称。
        "ossObjectName": "verify/1260051251634779/03a081bd96328aedf83f635f39a50c57_0.jpeg",
        // 照片HTTPS地址，有效期15分钟，从查询时开始计时。
        "pictureUrl": "http://cn-shanghai-aliyun-cloudauth-1260051251634779.oss-cn-shanghai
.aliyuncs.com/verify/1260051251634779/03a081bd96328aedf83f635f39a50c57_0.jpeg?Expires=16253
71140&OSSAccessKeyId=STS.NTX1ngfr6Acg2Pmnn2RYM****&Signature=Hw5BF9WxJs6wI68IxKs41cxCU8****
&security-token=CAISjgJ1q6Ft5B2yfSjIr5ftetTTi60X9qGMMHbcim5nXtZhu7GT1Dz2IH1PdXFgA%2Bgds%2Fs
wmm5U7vgalrkqEcEdHRGdN5YpsM8LrlzwO1h2TGRsq%2B5qsoasPETOITyZtZagToeUZdfZfejXGDKgvyRvwLz8WCy%
2FVli%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGfLC1dysQcO4gEWq4bHm5fCskKD1Qenk7F
O%2B9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWwIMvErYbbaMrIV1WwZ9Z7knHaVAq%2BXw
nOBkuuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAF%2FHZbelNLYWtipAu5X5oG1t22SqIr1p3TMK5FrjDIVeWOpp
cxeRXtU%2BjR7hGcwd25scGhiraoxTXV3sxw%2F6dhFSswQ37O4j%2B%2FrCPtiGauqn2ZMdMj%2FvYwKb6KmpQqa%2
BtR%2F9cxhS6hoIQRq0SUIfyXl5ZUiZbTjW22iwuX%2BPwzVCw%3D%3D",
        // 视频存放客户上海区域的OSS里，此为视频的文件名（仅当客户端设置视频参数时返回）。
        "ossVerifyVideoObjectName": "verify/1260051251634779/03a081bd96328aedf83f635****_ve
rifyvideo_dcb7.mov",
        // 视频HTTPS地址，有效期15分钟，从查询时开始计时（仅当客户端设置视频参数时返回）。
        "verifyVideoUrl": "http://cn-shanghai-aliyun-cloudauth-1260051251634779.oss-cn-shan
ghai.aliyuncs.com/verify/1260051251634779/03a081bd96328aedf83f635f39a50c57__verifyvideo_dcb
```

```
7.mov?Expires=1625371140&OSSAccessKeyId=STS.NTX1ngfr6Acg2Pmnn2RYM****&Signature=Hw5BF9WxJs6
wI68IxKs41cxCU8****&security-token=CAISjgJ1q6Ft5B2yfSjIr5ftetTTi60X9qGMMHbcim5nXtZhu7GT1Dz2
IH1PdXFgA%2Bgds%2Fswmm5U7vgalrkqEcEdHRGdN5YpsM8LrlzwO1h2TGRsq%2B5qsoasPETOITyZtZagToeUZdfZf
ejXGDKgvyRvwLz8WCy%2FVli%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGfLC1dysQcO4gEW
q4bHm5fCskKD1Qenk7FO%2B9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWwIMvErYbbaMrIV
1WwZ9Z7knHaVAq%2BXwnOBkuuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAF%2FHZbelNLYWtipAu5X5oG1t22SqI
r1p3TMK5FrjDIVeWOppcxeRXtU%2BjR7hGcwd25scGhiraoxTXV3sxw%2F6dhFSswQ37O4j%2B%2FrCPtiGauqn2ZMd
Mj%2FvYwKb6KmpQqa%2BtR%2F9cxhS6hoIQRq0SUIfyXl5ZUiZbTjW22iwuX%2BPwzVCw%3D%3D",
        // 活体人脸质量分数。
        "qualityScore": 99.93476867675781,
        // 人脸和公安比对分数，阈值可参考下表详细说明。
        "verifyScore": 57.678396649466755
    },
    // 意愿认证信息。
    "voluntaryCertInfo": {
        // 用户意愿表达语音识别出的文本列表
        "asrTexts":
         [
           "本人同意办理XXX业务",
           "本人同意办理XXX业务",
           "本人同意办理XXX业务"
         ],
        // 意愿认证相关资料存放客户上海区域的OSS里，此为资料的Bucket名称。
        "ossBucketName": "cn-shanghai-aliyun-cloudauth-1260051251634779",
        // 意愿认证音频存放客户上海区域的OSS里，此为音频的文件名列表。
        "voiceOssObjectNames":
         [
           "verify/1234567890/sha8d374f73c461a00e029def92c5c4c_wish_voice_0.wav",
           "verify/1234567890/sha8d374f73c461a00e029def92c5c4c_wish_voice_1.wav",
           "verify/1234567890/sha8d374f73c461a00e029def92c5c4c_wish_voice_2.wav"
         ],
        // 意愿认证音频HTTPS地址，有效期15分钟，从查询时开始计时。
        "voiceUrls":
         [
           "http://cn-shanghai-aliyun-cloudauth-1234567890.oss-cn-shanghai.aliyuncs.com/ver
ify/1234567890/sha8d374f73c461a00e029def92c5c4c_wish_voice_0.wav?Expires=1625371365&OSSAcce
ssKeyId=STS.NTEz8oi35qJpxxX7no5gpGd****&Signature=wnHjNu3333ddW3zNMup1r4ovgrw6xJ0****&secur
ity-token=CAISjgJ1q6Ft5B2yfSjIr5fwMYLbhOwUxoibel7p0246OehcqLL%2BpTz2IH1PdXFgA%2Bgds%2Fswmm5
U7vgalrkqEcEdHRGdN5YpsM8LrlzwO34PcmRsq%2B5qsoasPETOITyZtZagToeUZdfZfejXGDKgvyRvwLz8WCy%2FVl
i%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGfLC1dysQcO4gEWq4bHm5fCskKD1Qenk7FO%2B
9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWwIMvErYbbaMrIV1WwZ9Z7knHaVAq%2BXwnOBk
uuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAFThVhIcXH8zQiqbYVkTCHpZVRmDHH0mZ%2FMNuGikqSWdK3%2FhaS
EvxmO1wyLH9E7mFzoKJErtxTCs453H6PYHO8H3pKOpKgZ0PfWTpJ8OEIJm26fKph%2F%2F3usSUBx5h0Ob17e2XB4Bf
fkOJKQQiU3vXqoOZ6Vlyf7QpEJSvUMXXSKSg%3D%3D",
           "http://cn-shanghai-aliyun-cloudauth-1234567890.oss-cn-shanghai.aliyuncs.com/ver
ify/1234567890/sha8d374f73c461a00e029def92c5c4c_wish_voice_1.wav?Expires=1625371365&OSSAcce
ssKeyId=STS.NTEz8oi35qJpxxX7no5gpGd****&Signature=wnHjNu3333ddW3zNMup1r4ovgrw6xJ0****&secur
ity-token=CAISjgJ1q6Ft5B2yfSjIr5fwMYLbhOwUxoibel7p0246OehcqLL%2BpTz2IH1PdXFgA%2Bgds%2Fswmm5
U7vgalrkqEcEdHRGdN5YpsM8LrlzwO34PcmRsq%2B5qsoasPETOITyZtZagToeUZdfZfejXGDKgvyRvwLz8WCy%2FVl
i%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGfLC1dysQcO4gEWq4bHm5fCskKD1Qenk7FO%2B
9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWwIMvErYbbaMrIV1WwZ9Z7knHaVAq%2BXwnOBk
uuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAFThVhIcXH8zQiqbYVkTCHpZVRmDHH0mZ%2FMNuGikqSWdK3%2FhaS
EvxmO1wyLH9E7mFzoKJErtxTCs453H6PYHO8H3pKOpKgZ0PfWTpJ8OEIJm26fKph%2F%2F3usSUBx5h0Ob17e2XB4Bf
fkOJKQQiU3vXqoOZ6Vlyf7QpEJSvUMXXSKSg%3D%3D",
```

```
            "http://cn-shanghai-aliyun-cloudauth-1234567890.oss-cn-shanghai.aliyuncs.com/ver
ify/1234567890/sha8d374f73c461a00e029def92c5c4c_wish_voice_2.wav?Expires=1625371365&OSSAcce
ssKeyId=STS.NTEz8oi35qJpxxX7no5gpGd****&Signature=wnHjNu3333ddW3zNMup1r4ovgrw6xJ0****&secur
ity-token=CAISjgJ1q6Ft5B2yfSjIr5fwMYLbhOwUxoibel7p0246OehcqLL%2BpTz2IH1PdXFgA%2Bgds%2Fswmm5
U7vgalrkqEcEdHRGdN5YpsM8LrlzwO34PcmRsq%2B5qsoasPETOITyZtZagToeUZdfZfejXGDKgvyRvwLz8WCy%2FVl
i%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGfLC1dysQcO4gEWq4bHm5fCskKD1Qenk7FO%2B
9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWwIMvErYbbaMrIV1WwZ9Z7knHaVAq%2BXwnOBk
uuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAFThVhIcXH8zQiqbYVkTCHpZVRmDHH0mZ%2FMNuGikqSWdK3%2FhaS
EvxmO1wyLH9E7mFzoKJErtxTCs453H6PYHO8H3pKOpKgZ0PfWTpJ8OEIJm26fKph%2F%2F3usSUBx5h0Ob17e2XB4Bf
fkOJKQQiU3vXqoOZ6Vlyf7QpEJSvUMXXSKSg%3D%3D"
          ],
        // 意愿认证录屏视频存放客户上海区域的OSS里，此为录屏视频的文件名列表（仅当用户控制台设置需要留证时返回）。
        "screenVideoOssObjectNames":
          [
            "verify/1234567890/sha8d374f73c461a00e029def92c5c4c_wish_video_0.mov",
            "verify/1234567890/sha8d374f73c461a00e029def92c5c4c_wish_video_1.mov",
            "verify/1234567890/sha8d374f73c461a00e029def92c5c4c_wish_video_2.mov"
          ]
        // 意愿认证录屏视频HTTPS地址，有效期15分钟，从查询时开始计时（仅当用户控制台设置需要留证时返回）。
        "screenVideoUrls":
          [
             "http://cn-shanghai-aliyun-cloudauth-1234567890.oss-cn-shanghai.aliyuncs.com/ve
rify/1234567890/sha8d374f73c461a00e029def92c5c4c_wish_video_0.mov?Expires=1625371365&OSSAcc
essKeyId=STS.NTEz8oi35qJpxxX7no5gpGd****&Signature=wnHjNu3333ddW3zNMup1r4ovgrw6xJ0****&secu
rity-token=CAISjgJ1q6Ft5B2yfSjIr5fwMYLbhOwUxoibel7p0246OehcqLL%2BpTz2IH1PdXFgA%2Bgds%2Fswmm
5U7vgalrkqEcEdHRGdN5YpsM8LrlzwO34PcmRsq%2B5qsoasPETOITyZtZagToeUZdfZfejXGDKgvyRvwLz8WCy%2FV
li%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGfLC1dysQcO4gEWq4bHm5fCskKD1Qenk7FO%2
B9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWwIMvErYbbaMrIV1WwZ9Z7knHaVAq%2BXwnOB
kuuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAFThVhIcXH8zQiqbYVkTCHpZVRmDHH0mZ%2FMNuGikqSWdK3%2Fha
SEvxmO1wyLH9E7mFzoKJErtxTCs453H6PYHO8H3pKOpKgZ0PfWTpJ8OEIJm26fKph%2F%2F3usSUBx5h0Ob17e2XB4B
ffkOJKQQiU3vXqoOZ6Vlyf7QpEJSvUMXXSKSg%3D%3D",
             "http://cn-shanghai-aliyun-cloudauth-1234567890.oss-cn-shanghai.aliyuncs.com/ve
rify/1234567890/sha8d374f73c461a00e029def92c5c4c_wish_video_1.mov?Expires=1625371365&OSSAcc
essKeyId=STS.NTEz8oi35qJpxxX7no5gpGd****&Signature=wnHjNu3333ddW3zNMup1r4ovgrw6xJ0****&secu
rity-token=CAISjgJ1q6Ft5B2yfSjIr5fwMYLbhOwUxoibel7p0246OehcqLL%2BpTz2IH1PdXFgA%2Bgds%2Fswmm
5U7vgalrkqEcEdHRGdN5YpsM8LrlzwO34PcmRsq%2B5qsoasPETOITyZtZagToeUZdfZfejXGDKgvyRvwLz8WCy%2FV
li%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGfLC1dysQcO4gEWq4bHm5fCskKD1Qenk7FO%2
B9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWwIMvErYbbaMrIV1WwZ9Z7knHaVAq%2BXwnOB
kuuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAFThVhIcXH8zQiqbYVkTCHpZVRmDHH0mZ%2FMNuGikqSWdK3%2Fha
SEvxmO1wyLH9E7mFzoKJErtxTCs453H6PYHO8H3pKOpKgZ0PfWTpJ8OEIJm26fKph%2F%2F3usSUBx5h0Ob17e2XB4B
ffkOJKQQiU3vXqoOZ6Vlyf7QpEJSvUMXXSKSg%3D%3D",
             "http://cn-shanghai-aliyun-cloudauth-1234567890.oss-cn-shanghai.aliyuncs.com/ve
rify/1234567890/sha8d374f73c461a00e029def92c5c4c_wish_video_2.mov?Expires=1625371365&OSSAcc
essKeyId=STS.NTEz8oi35qJpxxX7no5gpGd****&Signature=wnHjNu3333ddW3zNMup1r4ovgrw6xJ0****&secu
rity-token=CAISjgJ1q6Ft5B2yfSjIr5fwMYLbhOwUxoibel7p0246OehcqLL%2BpTz2IH1PdXFgA%2Bgds%2Fswmm
5U7vgalrkqEcEdHRGdN5YpsM8LrlzwO34PcmRsq%2B5qsoasPETOITyZtZagToeUZdfZfejXGDKgvyRvwLz8WCy%2FV
li%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGfLC1dysQcO4gEWq4bHm5fCskKD1Qenk7FO%2
B9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWwIMvErYbbaMrIV1WwZ9Z7knHaVAq%2BXwnOB
kuuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAFThVhIcXH8zQiqbYVkTCHpZVRmDHH0mZ%2FMNuGikqSWdK3%2Fha
SEvxmO1wyLH9E7mFzoKJErtxTCs453H6PYHO8H3pKOpKgZ0PfWTpJ8OEIJm26fKph%2F%2F3usSUBx5h0Ob17e2XB4B
ffkOJKQQiU3vXqoOZ6Vlyf7QpEJSvUMXXSKSg%3D%3D"
          ]
```

```
        }
}
```

ResultObject.SubCode错误码说明

错误码	认证描述文案	是否计费	可能原因和运营建议
200	认证通过	是	调用成功。
201	姓名和身份证不一致	是	可能是用户的信息有误或用户的信息为假信息，建议用户确认后重新操作。若同一身份信息重新发起认证，服务端初始化将会返回417错误。
202	查询不到身份信息	是	可能是用户户口迁移、军人等特殊状态导致，建议预留人工审核入口，进行人工审核。若同一身份信息重新发起认证，服务端初始化将会返回417错误。
203	查询不到照片或照片不可用	是	可能是公安库数据问题导致，建议预留人工审核入口，进行人工审核。若同一身份信息重新发起认证，服务端初始化将会返回417错误。
204	人脸比对不一致	是	可能不是同一人或活体照片质量较低，建议根据业务情况分层处理，若为同一人可重新操作。
205	活体检测存在风险	是	可能存在攻击风险，建议人工审核分层处理，若为真人可重新操作。
206	业务策略限制	是	个性化业务安全策略限制，如有疑问可咨询客服。
207	人脸与身份证人脸比对不一致	是	仅在使用ID_PLUS方案时返回，可能人证不符或人脸照片质量较低，建议根据业务情况分层处理，若为同一人可重新操作。
209	权威比对源异常。	是	可能存在权威比对源异常，无法比对。
210	认证通过	是	通过产品安全策略综合判断，建议认证通过。
220	意愿判定不通过	是	客户未表达对应业务意愿或客户表达不清晰。若表达不清晰，可重新操作。
221	声纹比对不一致	是	可能不是同一人或音频质量较低，建议根据业务情况分层处理。若为同一人，可重新操作。

返回Code和Message说明

Code	Message	描述
200	success	成功。
400	参数不能为空	参数不能为空。
401	参数非法	非法参数。传入的姓名、身份证号码长度必须符合国家标准且不得包含英文字母等特殊字符。
403	认证未完成	认证未完成。

Code	Message	描述
404	认证场景配置不存在	认证场景配置不存在，请先在控制台上创建认证场景。
406	无效的CertifyId	无效的**CertifyId**。
410	未开通服务	未开通OSS产品或未完成OSS读写授权。
411	RAM无权限	需要给RAM用户授予**AliyunAntCloudAuthFullAccess**的操作权限。
424	身份认证记录不存在	当前CertifyId还未完成认证，建议您客户端收到认证回调为1000或2006时，再请求查询接口获取详情。 **说明** 如果出现CertifyId未认证、认证过程中放弃、认证处理中的情况，均会导致查询接口返回此错误码。
427	业务策略限制	仅在使用**ID_PLUS**方案时出现，指该身份因触发内部风控策略未完成认证。
500	系统错误	系统内部错误。

2.2.3. Java SDK调用示例

本文介绍金融级多因子意愿认证服务端接入的Java SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI 开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

使用的Java版本在1.8及以上版本。

使用Maven引入SDK

```
<dependency>
        <groupId>com.aliyun</groupId>
        <artifactId>cloudauth20190307</artifactId>
         <version>1.0.7</version>
</dependency>
```

调用示例

InitFaceVerify接口调用示例

```
import java.util.Arrays;
```

```
import java.util.List;
import com.aliyun.cloudauth20190307.Client;
import com.aliyun.cloudauth20190307.models.*;
import com.aliyun.tea.TeaException;
import com.aliyun.tea.TeaUnretryableException;
import com.aliyun.tearpc.models.Config;
import com.aliyun.teautil.models.RuntimeOptions;
public class InitFaceVerify {
    public static void main(String[] args) throws Exception {
        InitFaceVerifyRequest request = new InitFaceVerifyRequest();
        // 请输入场景ID+L。
        request.setSceneId(100000****L);
        // 设置商户请求的唯一标识。
        request.setOuterOrderNo("xxxx");
        // 认证方案。
        request.setProductCode("MFVC");
        request.setCertType("IDENTITY_CARD");
        request.setCertName("张三");
        request.setCertNo("xxxx");
        // MetaInfo环境参数。
        request.setMetaInfo("xxxxxxxx");
        //request.setMobile("1390000****");
        //request.setIp("114.100.XX.XX");
        //request.setUserId("148562088256****");
        //request.setCallbackUrl("https://www.aliyundoc.com");
        //request.setCallbackToken("xxxxx");
        // 推荐，支持服务路由。
        InitFaceVerifyResponse response = initFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        //InitFaceVerifyResponse response = initFaceVerify("cloudauth.cn-shanghai.aliyuncs.
com", request);
        System.out.println(response.getRequestId());
        System.out.println(response.getCode());
        System.out.println(response.getMessage());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getCertifyId());
    }
    private static InitFaceVerifyResponse initFaceVerifyAutoRoute(InitFaceVerifyRequest req
uest) {
        // 第一个为主区域Endpoint，第二个为备区域Endpoint。
        List<String> endpoints = Arrays.asList("cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com");
        InitFaceVerifyResponse lastResponse = null;
        for (String endpoint : endpoints) {
            try {
                InitFaceVerifyResponse response = initFaceVerify(endpoint, request);
                lastResponse = response;
                // 服务端错误，切换到下个区域调用。
                if (response != null && "500".equals(response.getCode())) {
                    continue;
                }
                return response;
            } catch (Exception e) {
                // 网络异常，切换到下个区域调用。
```

```
                if (e.getCause() instanceof TeaException) {
                    TeaException teaException = ((TeaException)e.getCause());
                    if (teaException.getData() != null && "ServiceUnavailable".equals(
                        teaException.getData().get("Code"))) {
                        continue;
                    }
                }
                if (e.getCause() instanceof TeaUnretryableException) {
                    continue;
                }
            }
        }
        return lastResponse;
    }
    private static InitFaceVerifyResponse initFaceVerify(String endpoint, InitFaceVerifyReq
uest request)
        throws Exception {
        Config config = new Config();
        config.setAccessKeyId("您的AccessKey ID");
        config.setAccessKeySecret("您的AccessKey Secret");
        config.setEndpoint(endpoint);
        // 设置HTTP代理。
        //config.setHttpProxy("http://xx.xx.xx.xx:xxxx");
        // 设置HTTPS代理。
        //config.setHttpsProxy("https://xx.xx.xx.xx:xxxx");
        Client client = new Client(config);
        // 创建RuntimeObject实例并设置运行参数。
        RuntimeOptions runtime = new RuntimeOptions();
        runtime.readTimeout = 10000;
        runtime.connectTimeout = 10000;
        return client.initFaceVerify(request, runtime);
    }
}
```

DescribeFaceVerify接口调用示例

```
import java.util.Arrays;
import java.util.List;
import com.aliyun.cloudauth20190307.Client;
import com.aliyun.cloudauth20190307.models.DescribeFaceVerifyRequest;
import com.aliyun.cloudauth20190307.models.DescribeFaceVerifyResponse;
import com.aliyun.tea.TeaException;
import com.aliyun.tea.TeaUnretryableException;
import com.aliyun.tearpc.models.Config;
import com.aliyun.teautil.models.RuntimeOptions;
public class DescribeFaceVerify {
    public static void main(String[] args) throws Exception {
        // 通过以下代码创建API请求并设置参数。
        DescribeFaceVerifyRequest request = new DescribeFaceVerifyRequest();
        // 请输入场景ID+L。
        request.setSceneId(xxxxL);
        request.setCertifyId("xxxx");
        // 推荐，支持服务路由。
        DescribeFaceVerifyResponse response = describeFaceVerifyAutoRoute(request);
```

```
        // 不支持服务自动路由。
        //DescribeFaceVerifyResponse response = describeFaceVerify("cloudauth.cn-shanghai.a
liyuncs.com", request);
        System.out.println(response.getRequestId());
        System.out.println(response.getCode());
        System.out.println(response.getMessage());
        System.out.println(
            response.getResultObject() == null ? null : response.getResultObject().getPasse
d());
        System.out.println(
            response.getResultObject() == null ? null : response.getResultObject().getSubCo
de());
        System.out.println(
            response.getResultObject() == null ? null
                : response.getResultObject().getIdentityInfo());
        System.out.println(
            response.getResultObject() == null ? null
                : response.getResultObject().getDeviceToken());
        System.out.println(
            response.getResultObject() == null ? null
                : response.getResultObject().getMaterialInfo());
    }
    private static DescribeFaceVerifyResponse describeFaceVerifyAutoRoute(DescribeFaceVerif
yRequest request) {
        // 第一个为主区域Endpoint，第二个为备区域Endpoint。
        List<String> endpoints = Arrays.asList("cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com");
        DescribeFaceVerifyResponse lastResponse = null;
        for (String endpoint : endpoints) {
            try {
                DescribeFaceVerifyResponse response = describeFaceVerify(endpoint, request)
;
                lastResponse = response;
                // 服务端错误，切换到下个区域调用。
                if (response != null && "500".equals(response.getCode())) {
                    continue;
                }
                return response;
            } catch (Exception e) {
                // 网络异常，切换到下个区域调用。
                if (e.getCause() instanceof TeaException) {
                    TeaException teaException = ((TeaException)e.getCause());
                    if (teaException.getData() != null && "ServiceUnavailable".equals(
                        teaException.getData().get("Code"))) {
                        continue;
                    }
                }
                if (e.getCause() instanceof TeaUnretryableException) {
                    continue;
                }
            }
        }
        return lastResponse;
    }
```

```
    private static DescribeFaceVerifyResponse describeFaceVerify(String endpoint, DescribeF
aceVerifyRequest request)
        throws Exception {
        Config config = new Config();
        config.setAccessKeyId("您的AccessKey ID");
        config.setAccessKeySecret("您的AccessKey Secret");
        config.setEndpoint(endpoint);
        // 设置http代理。
        //config.setHttpProxy("http://xx.xx.xx.xx:xxxx");
        // 设置https代理。
        //config.setHttpsProxy("http://xx.xx.xx.xx:xxxx");
        Client client = new Client(config);
        // 创建RuntimeObject实例并设置运行参数。
        RuntimeOptions runtime = new RuntimeOptions();
        runtime.readTimeout = 10000;
        runtime.connectTimeout = 10000;
        return client.describeFaceVerify(request, runtime);
    }
}
```

2.2.4. PHP SDK调用示例

本文介绍金融级多因子意愿认证服务端接入的PHP SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的PHP版本在5.6版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于PHP SDK源代码，请参见PHP SDK包。

- 通过Composer来管理项目依赖（推荐）

```
composer require alibabacloud/cloudauth-20190307 1.0.7
```

- 使用composer.json文件

```
{
   "require": {
       "alibabacloud/cloudauth-20190307": "1.0.7"
    }
}
```

如果因网络问题导致无法按照上述操作安装SDK，您可以使用阿里云Composer全量镜像。

调用示例

InitFaceVerify接口调用示例

```
<?php
namespace AlibabaCloud\SDK\Sample {
use AlibabaCloud\SDK\Cloudauth\V20190307\Cloudauth;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\InitFaceVerifyRequest;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\InitFaceVerifyResponse;
use AlibabaCloud\Tea\Utils\Utils;
use \Exception;
use AlibabaCloud\Tea\Exception\TeaError;
use AlibabaCloud\Tea\Rpc\Rpc\Config;
use AlibabaCloud\Tea\Utils\Utils\RuntimeOptions;
class InitFaceVerify {
    /**
     * @param string $endpoint
     * @return Cloudauth
     */
    public static function createClient($endpoint){
        $config = new Config([
            // 您的AccessKey ID。
            "accessKeyId" => "<您的AccessKey ID>",
            // 您的AccessKey Secret。
            "accessKeySecret" => "<您的AccessKey Secret>",
            // 设置HTTP代理。
            // "httpProxy" => "<http://xx.xx.xx.xx:xxxx>",
            // 设置HTTPS代理
            // "httpsProxy" => "<https://xx.xx.xx.xx:xxxx>",
            "endpoint" => $endpoint
        ]);
        return new Cloudauth($config);
    }
    /**
     * @return void
     */
    public static function main(){
        $request = new InitFaceVerifyRequest([
            // 请输入场景ID。
            "sceneId" => <xxx>,
            "outerOrderNo" => "<商户请求的唯一标识>",
            // 要接入的认证方案。
            "productCode" => "<MFVC>",
            "userId" => "<148562088256****>",
            "certType" => "<IDENTITY_CARD>",
            "certName" => "<xxx>",
            "certNo" => "<1******************9>",
            // MetaInfo环境参数，需要通过客户端SDK获取。
            "metaInfo" => '{"zimVer":"3.0.0","appVersion": "1","bioMetaInfo": "4.1.0:115015
68,0",
        "appName": "com.aliyun.antcloudauth","deviceType": "ios","osVersion": "iOS 10.3.2",
"apdidToken": "","deviceModel": "iPhone9,1"}',
            // "mobile" => "<1390000****>",
            // "ip" => "<114.100.XX.XX>",
```

```
            // "callbackUrl" => "<https://www.aliyundoc.com>",
            // "callbackToken" => "<xxxxxxx>"
        ]);
        // 推荐，支持服务路由。
        $response = self::initFaceVerifyAutoRoute($request);
        // 不支持服务自动路由。
        // $response = self::initFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", $request)
;
        var_dump($response->requestId);
        var_dump($response->code);
        var_dump($response->message);
        var_dump($response->resultObject->certifyId);
    }
    /**
     * @param InitFaceVerifyRequest $request
     * @return InitFaceVerifyResponse
     */
    public static function initFaceVerifyAutoRoute(InitFaceVerifyRequest $request){
        $endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        foreach($endpoints as $endpoint){
            try {
                $response = self::initFaceVerify($endpoint, $request);
                if (Utils::equalString("500", $response->code)) {
                    continue;
                }
                return $response;
            }
            catch (Exception $err) {
                if (!($err instanceof TeaError)) {
                    $err = new TeaError([], $err->getMessage(), $err->getCode(), $err);
                }
                // 网络异常，切换到下个区域调用。
                if (Utils::equalString("ServiceUnavailable", $err->code)) {
                    continue;
                }
            }
        }
        return null;
    }
    /**
     * @param string $endpoint
     * @param InitFaceVerifyRequest $request
     * @return InitFaceVerifyResponse
     */
    public static function initFaceVerify($endpoint, $request){
        $client = self::createClient($endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        $runtime = new RuntimeOptions([]);
        $runtime->readTimeout = 10000;
        $runtime->connectTimeout = 10000;
        return $client->initFaceVerify($request, $runtime);
```

```
    }
}
$path = __DIR__ . \DIRECTORY_SEPARATOR . \DIRECTORY_SEPARATOR . 'vendor' . \DIRECTORY_SEPAR
ATOR . 'autoload.php';
if (file_exists($path)) {
    require_once $path;
}
InitFaceVerify::main();
}
```

DescribeFaceVerify接口调用示例

```
<?php
namespace AlibabaCloud\SDK\Sample {
use AlibabaCloud\SDK\Cloudauth\V20190307\Cloudauth;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\DescribeFaceVerifyRequest;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\DescribeFaceVerifyResponse;
use AlibabaCloud\Tea\Utils\Utils;
use \Exception;
use AlibabaCloud\Tea\Exception\TeaError;
use AlibabaCloud\Tea\Rpc\Rpc\Config;
use AlibabaCloud\Tea\Utils\Utils\RuntimeOptions;
class DescribeFaceVerify {
    /**
     * @param string $endpoint
     * @return Cloudauth
     */
    public static function createClient($endpoint){
        $config = new Config([
            // 您的AccessKey ID。
            "accessKeyId" => "<您的AccessKey ID>",
            // 您的AccessKey Secret。
            "accessKeySecret" => "<您的AccessKey Secret>",
            // 设置HTTP代理。
            // "httpProxy" => "<http://xx.xx.xx.xx:xxxx>",
            // 设置HTTPS代理。
            // "httpsProxy" => "<https://xx.xx.xx.xx:xxxx>",
            "endpoint" => $endpoint
        ]);
        return new Cloudauth($config);
    }
    /**
     * @return void
     */
    public static function main(){
        $request = new DescribeFaceVerifyRequest();
        // 场景ID。
        $request->sceneId=<100000****>;
        // 认证ID。
        $request->certifyId="<xxx>";
        // 推荐，支持服务路由。
        $response = self::describeFaceVerifyAutoRoute($request);
        // 不支持服务自动路由。
        // $response = self::describeFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", $requ
```

```
est);
        var_dump($response->requestId);
        var_dump($response->code);
        var_dump($response->message);
        var_dump($response->resultObject->passed);
        var_dump($response->resultObject->subCode);
        var_dump($response->resultObject->materialInfo);
        var_dump($response->resultObject->identityInfo);
        var_dump($response->resultObject->deviceToken);
    }
    /**
     * @param DescribeFaceVerifyRequest $request
     * @return DescribeFaceVerifyResponse
     */
    public static function describeFaceVerifyAutoRoute($request){
        $endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        foreach($endpoints as $endpoint){
            try {
                $response = self::describeFaceVerify($endpoint, $request);
                if (Utils::equalString("500", $response->code)) {
                    continue;
                }
                return $response;
            }
            catch (Exception $err) {
                if (!($err instanceof TeaError)) {
                    $err = new TeaError([], $err->getMessage(), $err->getCode(), $err);
                }
                // 网络异常，切换到下个区域调用。
                if (Utils::equalString("ServiceUnavailable", $err->code)) {
                    continue;
                }
            }
        }
        return null;
    }
    /**
     * @param string $endpoint
     * @param DescribeFaceVerifyRequest $request
     * @return DescribeFaceVerifyResponse
     */
    public static function describeFaceVerify($endpoint, $request){
        $client = self::createClient($endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        $runtime = new RuntimeOptions([]);
        $runtime->readTimeout = 10000;
        $runtime->connectTimeout = 10000;
        return $client->describeFaceVerify($request, $runtime);
    }
}
$path = __DIR__ . \DIRECTORY_SEPARATOR . \DIRECTORY_SEPARATOR . 'vendor' . \DIRECTORY_SEPAR
```

```
ATOR . 'autoload.php';
if (file_exists($path)) {
    require_once $path;
}
DescribeFaceVerify::main();
}
```

2.2.5. Python SDK调用示例

本文介绍金融级多因子意愿认证服务端接入的Python SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的Python版本在3.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

引入依赖包

```
pip install alibabacloud_tea_openapi==0.2.5
```

安装SDK

关于Python SDK源代码，请参见Python SDK包。

```
pip install alibabacloud_cloudauth20190307==1.0.7
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
from alibabacloud_cloudauth20190307.client import Client as CloudauthClient
from alibabacloud_tea_openapi import models as open_api_models
from alibabacloud_cloudauth20190307 import models as cloudauth_models
from alibabacloud_tea_util.client import Client as UtilClient
from alibabacloud_tea_util import models as util_models
class InitFaceVerify:
    def __init__(self):
        pass
    @staticmethod
    def create_client(
            endpoint: str,
    ) -> CloudauthClient:
        config = open_api_models.Config(
```

```
            # 您的AccessKey ID。
            access_key_id='您的AccessKey ID',
            # 您的AccessKey Secret。
            access_key_secret='您的AccessKey Secret',
            # 设置HTTP代理。
            # http_proxy='http://xx.xx.xx.xx:xxxx',
            # 设置HTTPS代理。
            # https_proxy='https://username:password@xxx.xxx.xxx.xxx:9999',
            endpoint=endpoint
        )
        return CloudauthClient(config)
    @staticmethod
    def main() -> None:
        request = cloudauth_models.InitFaceVerifyRequest(
            # 请输入场景ID。
            scene_id=100000****,
            outer_order_no='商户请求的唯一标识',
            # 要接入的认证方案。
            product_code='MFVC',
            # 您自定义的用户ID，请保持唯一。
            user_id='148562088256****',
            # 固定值IDENTITY_CARD。
            cert_type='IDENTITY_CARD',
            cert_name='xxx',
            cert_no='1******************9',
            # MetaInfo环境参数，需要通过客户端SDK获取。
            meta_info='{"zimVer":"3.0.0","appVersion": "1","bioMetaInfo": "4.1.0:11501568,0
","appName": "com.aliyun.antcloudauth","deviceType": "ios","osVersion": "iOS 10.3.2","apdid
Token": "","deviceModel": "iPhone9,1"}',
            # mobile='1390000****',
            # ip='114.100.XX.XX',
            # callback_url='https://www.aliyundoc.com',
            # callback_token='xxxxxxx'
        )
        response = InitFaceVerify.init_face_verify_auto_route(request)
        # 不支持服务自动路由。
        # response = InitFaceVerify.init_face_verify('cloudauth.cn-shanghai.aliyuncs.com',
request)
        print(response.request_id)
        print(response.code)
        print(response.message)
        print(response.result_object.certify_id)
    @staticmethod
    def init_face_verify_auto_route(
            request: cloudauth_models.InitFaceVerifyRequest,
    ) -> cloudauth_models.InitFaceVerifyResponse:
        endpoints = [
            'cloudauth.cn-shanghai.aliyuncs.com',
            'cloudauth.cn-beijing.aliyuncs.com'
        ]
        for endpoint in endpoints:
            try:
                response = InitFaceVerify.init_face_verify(endpoint, request)
                if UtilClient.equal_string('500', response.code):
```

```
                    continue
                return response
            except Exception as err:
                # 网络异常，切换到下个区域调用。
                continue
        return None
    @staticmethod
    def init_face_verify(
            endpoint: str,
            request: cloudauth_models.InitFaceVerifyRequest,
    ) -> cloudauth_models.InitFaceVerifyResponse:
        client = InitFaceVerify.create_client(endpoint)
        # 创建RuntimeObject实例并设置运行参数。
        runtime = util_models.RuntimeOptions()
        runtime.read_timeout = 10000
        runtime.connect_timeout = 10000
        return client.init_face_verify(request, runtime)
if __name__ == '__main__':
    InitFaceVerify.main()
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
from alibabacloud_cloudauth20190307.client import Client as CloudauthClient
from alibabacloud_tea_openapi import models as open_api_models
from alibabacloud_cloudauth20190307 import models as cloudauth_models
from alibabacloud_tea_util.client import Client as UtilClient
from alibabacloud_tea_util import models as util_models
class DescribeFaceVerify:
    def __init__(self):
        pass
    @staticmethod
    def create_client(
        endpoint: str,
    ) -> CloudauthClient:
        config = open_api_models.Config(
            # 您的AccessKey ID。
            access_key_id='您的AccessKey ID',
            # 您的AccessKey Secret。
            access_key_secret='您的AccessKey Secret',
            # 设置HTTP代理。
            # http_proxy='http://xx.xx.xx.xx:xxxx',
            # 设置HTTPS代理。
            # https_proxy='https://xx.xx.xx.xx:xxxx',
            endpoint=endpoint
        )
        return CloudauthClient(config)
    @staticmethod
    def main() -> None:
        request = cloudauth_models.DescribeFaceVerifyRequest(
            # 请输入场景ID。
            scene_id=100000****,
            certify_id='xxx'
```

```
        )
        response = DescribeFaceVerify.describe_face_verify_auto_route(request)
        # 不支持服务自动路由。
        # response = DescribeFaceVerify.describe_face_verify('cloudauth.cn-shanghai.aliyunc
s.com', request)
        print(response.request_id)
        print(response.code)
        print(response.message)
        print(response.result_object.passed)
        print(response.result_object.sub_code)
        print(response.result_object.material_info)
    @staticmethod
    def describe_face_verify_auto_route(
        request: cloudauth_models.DescribeFaceVerifyRequest,
    ) -> cloudauth_models.DescribeFaceVerifyResponse:
        endpoints = [
            'cloudauth.cn-shanghai.aliyuncs.com',
            'cloudauth.cn-beijing.aliyuncs.com'
        ]
        for endpoint in endpoints:
            try:
                response = DescribeFaceVerify.describe_face_verify(endpoint, request)
                if UtilClient.equal_string('500', response.code):
                    continue
                return response
            except Exception as err:
                    continue
        return None
    @staticmethod
    def describe_face_verify(
        endpoint: str,
        request: cloudauth_models.DescribeFaceVerifyRequest,
    ) -> cloudauth_models.DescribeFaceVerifyResponse:
        client = DescribeFaceVerify.create_client(endpoint)
        # 创建RuntimeObject实例并设置运行参数。
        runtime = util_models.RuntimeOptions()
        runtime.read_timeout = 10000
        runtime.connect_timeout = 10000
        return client.describe_face_verify(request, runtime)
if __name__ == '__main__':
    DescribeFaceVerify.main()
```

2.2.6. Go SDK调用示例

本文介绍金融级多因子意愿认证服务端接入的Go SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。详细信息，请参见：

- 调试InitFaceVerify接口

- 调试DescribeFaceVerify接口

前提条件

- 仅支持Go 1.12.0及以上版本。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

引入依赖包

```
go get github.com/alibabacloud-go/tea-rpc
go get github.com/alibabacloud-go/tea-utils
go get github.com/alibabacloud-go/tea
```

安装SDK

关于Go SDK源代码，请参见Go SDK包。

通过Go mod来管理项目依赖：

```
go get github.com/alibabacloud-go/cloudauth-20190307@v1.0.7
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
package main
import (
  "fmt"
  cloudauth "github.com/alibabacloud-go/cloudauth-20190307/client"
  rpc "github.com/alibabacloud-go/tea-rpc/client"
  util "github.com/alibabacloud-go/tea-utils/service"
  "github.com/alibabacloud-go/tea/tea"
)
func CreateClient (endpoint *string) (_result *cloudauth.Client, _err error) {
  config := &rpc.Config{
    // 您的AccessKey ID。
    AccessKeyId: tea.String("您的AccessKey ID"),
    // 您的AccessKey Secret。
    AccessKeySecret: tea.String("您的AccessKey Secret"),
    // 设置HTTP代理。
    // HttpProxy: tea.String("http://xx.xx.xx.xx:xxxx"),
    // 设置HTTPS代理。
    // HttpsProxy: tea.String("https://username:password@xxx.xxx.xxx.xxx:9999"),
    Endpoint: endpoint,
  }
  _result = &cloudauth.Client{}
  _result, _err = cloudauth.NewClient(config)
  return _result, _err
}
func _main () (_err error) {
  request := &cloudauth.InitFaceVerifyRequest{
    // 固定值。
```

```
    ProductCode: tea.String("MFVC"),
    // 请输入场景ID。
    SceneId: tea.Int64(100000****),
    OuterOrderNo: tea.String("商户请求的唯一标识"),
    UserId: tea.String("用户ID"),
    // 固定值IDENTITY_CARD。
    CertType: tea.String("IDENTITY_CARD"),
    CertName: tea.String("xxx"),
    CertNo: tea.String("xxx"),
    MetaInfo: tea.String("{\"zimVer\":\"3.0.0\",\"appVersion\": \"1\",\"bioMetaInfo\": \"4.
1.0:11501568,0\",\"appName\": \"com.aliyun.antcloudauth\",\"deviceType\": \"ios\",\"osVersi
on\": \"iOS 10.3.2\",\"apdidToken\": \"\",\"deviceModel\": \"iPhone9,1\"}"),
    // Ip: tea.String("用户IP地址"),
    // Mobile: tea.String("1390000****"),
    // CallbackUrl: tea.String("https://www.aliyundoc.com"),
    // CallbackToken: tea.String("xxxxxxx")
  }
  response, _err := InitFaceVerifyAutoRoute(request)
  // 不支持服务自动路由。
  // response, _err := InitFaceVerify(tea.String("cloudauth.cn-shanghai.aliyuncs.com"), req
uest)
  if _err != nil {
    return _err
  }
  fmt.Println(*response.RequestId)
  fmt.Println(*response.Code)
  fmt.Println(*response.Message)
  fmt.Println(*response.RequestId)
  fmt.Println(*response.ResultObject.CertifyId)
  return _err
}
func InitFaceVerifyAutoRoute (request *cloudauth.InitFaceVerifyRequest) (_result *cloudauth
.InitFaceVerifyResponse, _err error) {
  endpoints := []*string{tea.String("cloudauth.cn-shanghai.aliyuncs.com"), tea.String("clou
dauth.cn-beijing.aliyuncs.com")}
  var lastResponse *cloudauth.InitFaceVerifyResponse
  for _, endpoint := range endpoints {
    response, _err := InitFaceVerify(endpoint, request)
    if _err != nil {
      var err = &tea.SDKError{}
      if _t, ok := _err.(*tea.SDKError); ok {
        err = _t
        // 网络异常，切换到下个区域调用。
        if tea.BoolValue(util.EqualString(tea.String("ServiceUnavailable"), err.Code)) {
          continue
        }
      }
      return _result, _err
    }
    if tea.BoolValue(util.EqualString(tea.String("500"), response.Code)) {
      continue
    }
    _result = response
    return _result, _err
  }
```

```
  }
  _result = lastResponse
  return _result, _err
}
func InitFaceVerify (endpoint *string, request *cloudauth.InitFaceVerifyRequest) (_result *
cloudauth.InitFaceVerifyResponse, _err error) {
  client, _err := CreateClient(endpoint)
  if _err != nil {
    return _result, _err
  }
  // 创建RuntimeObject实例并设置运行参数。
  runtime := &util.RuntimeOptions{}
  runtime.ReadTimeout = tea.Int(10000)
  runtime.ConnectTimeout = tea.Int(10000)
  _result = &cloudauth.InitFaceVerifyResponse{}
  _body, _err := client.InitFaceVerify(request, runtime)
  if _err != nil {
    return _result, _err
  }
  _result = _body
  return _result, _err
}
func main() {
  err := _main()
  if err != nil {
    panic(err)
  }
}
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
package main
import (
  "fmt"
  cloudauth "github.com/alibabacloud-go/cloudauth-20190307/client"
  rpc "github.com/alibabacloud-go/tea-rpc/client"
  util "github.com/alibabacloud-go/tea-utils/service"
  "github.com/alibabacloud-go/tea/tea"
)
func CreateClient(endpoint *string) (_result *cloudauth.Client, _err error) {
  config := &rpc.Config{
    // 您的AccessKey ID。
    AccessKeyId: tea.String("您的AccessKey ID"),
    // 您的AccessKey Secret。
    AccessKeySecret: tea.String("您的AccessKey Secret"),
    Endpoint:        endpoint,
    // 设置HTTP代理。
    // HttpProxy: tea.String("http://xx.xx.xx.xx:xxxx"),
    // 设置HTTPS代理。
    // HttpsProxy: tea.String("https://xx.xx.xx.xx:xxxx"),
  }
  _result = &cloudauth.Client{}
```

```
  _result, _err = cloudauth.NewClient(config)
  return _result, _err
}
func _main() (_err error) {
  request := &cloudauth.DescribeFaceVerifyRequest{
    // 请输入场景ID。
    SceneId:   tea.Int64(100000****),
    CertifyId: tea.String("91707dc296d469ad38e4c5efa6a0xxxx"),
  }
  response, _err := DescribeFaceVerifyAutoRoute(request)
  // 不支持服务自动路由。
  // response, _err := DescribeFaceVerify(tea.String("cloudauth.cn-shanghai.aliyuncs.com"),
request)
  if _err != nil {
    return _err
  }
  if response != nil {
    fmt.Println(*response.RequestId)
    fmt.Println(*response.Code)
    fmt.Println(*response.Message)
    fmt.Println(*response.RequestId)
    fmt.Println(*response.ResultObject.Passed)
    fmt.Println(*response.ResultObject.SubCode)
    fmt.Println(*response.ResultObject.MaterialInfo)
    fmt.Println(*response.ResultObject.IdentityInfo)
    fmt.Println(*response.ResultObject.DeviceToken)
  }
  return _err
}
func DescribeFaceVerifyAutoRoute(request *cloudauth.DescribeFaceVerifyRequest) (_result *cl
oudauth.DescribeFaceVerifyResponse, _err error) {
  endpoints := []*string{tea.String("cloudauth.cn-shanghai.aliyuncs.com"), tea.String("clou
dauth.cn-beijing.aliyuncs.com")}
  var lastResponse *cloudauth.DescribeFaceVerifyResponse
  for _, endpoint := range endpoints {
    response, _err := DescribeFaceVerify(endpoint, request)
    if _err != nil {
      var err = &tea.SDKError{}
      if _t, ok := _err.(*tea.SDKError); ok {
        err = _t
        // 网络异常，切换到下个区域调用。
        if tea.BoolValue(util.EqualString(tea.String("ServiceUnavailable"), err.Code)) {
          continue
        }
      }
      return _result, _err
    }
    if tea.BoolValue(util.EqualString(tea.String("500"), response.Code)) {
      continue
    }
    _result = response
    return _result, _err
  }
  _result = lastResponse
  return _result, _err
```

```
  return _result, _err
}
func DescribeFaceVerify(endpoint *string, request *cloudauth.DescribeFaceVerifyRequest) (_r
esult *cloudauth.DescribeFaceVerifyResponse, _err error) {
  client, _err := CreateClient(endpoint)
  if _err != nil {
    return _result, _err
  }
  // 创建RuntimeObject实例并设置运行参数。
  runtime := &util.RuntimeOptions{}
  runtime.ReadTimeout = tea.Int(10000)
  runtime.ConnectTimeout = tea.Int(10000)
  _body, _err := client.DescribeFaceVerify(request, runtime)
  if _err != nil {
    return _result, _err
  }
  _result = _body
  return _result, _err
}
func main() {
  err := _main()
  if err != nil {
    panic(err)
  }
}
```

2.2.7. .NET SDK调用示例

本文介绍金融级多因子意愿认证服务端接入的.NET SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的.NET Framework版本在4.5.0版本及以上或者.NET Standard版本在2.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于.NET SDK源代码，请参见.NET SDK包。

SDK有以下两种安装方式，请选择合适的方式安装：

- 通过.NET命令行安装。

```
dotnet add package AlibabaCloud.SDK.Cloudauth20190307 --version 1.0.7
```

- 使用包管理安装。

```
Install-Package AlibabaCloud.SDK.Cloudauth20190307 -Version 1.0.7
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
using System;
using System.Collections.Generic;
using Tea;
namespace InitFaceVerify
{
    public class Program
    {
        public static AlibabaCloud.SDK.Cloudauth20190307.Client CreateClient(string endpoin
t)
        {
            AlibabaCloud.RPCClient.Models.Config config = new AlibabaCloud.RPCClient.Models
.Config
            {
                // 您的AccessKey ID。
                AccessKeyId = "您的AccessKey ID",
                // 您的AccessKey Secret。
                AccessKeySecret = "您的AccessKey Secret",
                // 设置HTTP代理。
                // HttpProxy = "http://xx.xx.xx.xx:xxxx",
                // 设置HTTPS代理。
                // HttpsProxy = "https://username:password@xxx.xxx.xxx.xxx:9999",
                Endpoint = endpoint,
            };
            return new AlibabaCloud.SDK.Cloudauth20190307.Client(config);
        }
        public static void Main(string[] args)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest request = new A
libabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest
            {
                // 固定值。
                ProductCode = "MFVC",
                // 请输入场景ID。
                SceneId = 100000****,
                // 设置商户请求的唯一标识。
                OuterOrderNo = "xxxxx",
                UserId = "用户ID",
                // 固定值IDENTITY_CARD。
                CertType = "IDENTITY_CARD",
                CertName = "xxx",
                CertNo = "1******************9",
                MetaInfo = "{\"zimVer\":\"3.0.0\",\"appVersion\": \"1\",\"bioMetaInfo\": \"
4.1.0:11501568,0\",\"appName\": \"com.aliyun.antcloudauth\",\"deviceType\": \"ios\",\"osVer
sion\": \"iOS 10.3.2\",\"apdidToken\": \"\",\"deviceModel\": \"iPhone9,1\"}",
                //Mobile = "1390000****",
                //Ip = "114.100.XX.XX",
```

```
                //CallbackUrl = "https://www.aliyundoc.com",
                //CallbackToken = "xxxxxxx"
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse response = Ini
tFaceVerifyAutoRoute(request);
            // 不支持服务自动路由。
            // AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse response =
InitFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", request);
            Console.WriteLine(response.RequestId);
            Console.WriteLine(response.Code);
            Console.WriteLine(response.Message);
            Console.WriteLine(response.ResultObject.CertifyId);
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse Init
FaceVerifyAutoRoute(AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest request
)
        {
            List<string> endpoints = new List<string>
            {
                "cloudauth.cn-shanghai.aliyuncs.com",
                "cloudauth.cn-beijing.aliyuncs.com"
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse lastResponse =
null;
            foreach (var endpoint in endpoints)
            {
                try
                {
                    AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse respon
se = InitFaceVerify(endpoint, request);
                    if (AlibabaCloud.TeaUtil.Common.EqualString("500", response.Code))
                    {
                        continue;
                    }
                    return response;
                }
                catch (TeaException err)
                {
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
                catch (Exception _err)
                {
                    TeaException err = new TeaException(new Dictionary<string, object>
                    {
                        { "message", _err.Message }
                    });
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
```

```
                    {
                        continue;
                    }
                }
            }
            return lastResponse;
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse Init
FaceVerify(string endpoint, AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest
request)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Client client = CreateClient(endpoint);
            // 创建RuntimeObject实例并设置运行参数。
            AlibabaCloud.TeaUtil.Models.RuntimeOptions runtime = new AlibabaCloud.TeaUtil.M
odels.RuntimeOptions();
            runtime.ReadTimeout = 10000;
            runtime.ConnectTimeout = 10000;
            return client.InitFaceVerify(request, runtime);
        }
    }
}
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
using System;
using System.Collections.Generic;
using Tea;
namespace DescribeFaceVerify
{
    public class Program
    {
        public static AlibabaCloud.SDK.Cloudauth20190307.Client CreateClient(string endpoin
t)
        {
            AlibabaCloud.RPCClient.Models.Config config = new AlibabaCloud.RPCClient.Models
.Config
            {
                // 您的AccessKey ID。
                AccessKeyId = "您的AccessKey ID",
                // 您的AccessKey Secret。
                AccessKeySecret = "您的AccessKey Secret",
                // 设置HTTP代理。
                // HttpProxy = "http://xx.xx.xx.xx:xxxx",
                // 设置HTTPS代理。
                // HttpsProxy = "https://xx.xx.xx.xx:xxxx",
                Endpoint = endpoint,
            };
            return new AlibabaCloud.SDK.Cloudauth20190307.Client(config);
        }
        public static void Main(string[] args)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyRequest request = n
```

```
ew AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyRequest
            {
                // 请输入场景ID。
                SceneId = 100000****,
                CertifyId = "91707dc296d469ad38e4c5efa6a0xxxx",
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse response =
DescribeFaceVerifyAutoRoute(request);
            // 不支持服务自动路由。
            // AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse respons
e = DescribeFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", request);
            Console.WriteLine(response.RequestId);
            Console.WriteLine(response.Code);
            Console.WriteLine(response.Message);
            Console.WriteLine(response.ResultObject.Passed);
            Console.WriteLine(response.ResultObject.SubCode);
            Console.WriteLine(response.ResultObject.MaterialInfo);
            Console.WriteLine(response.ResultObject.IdentityInfo);
            Console.WriteLine(response.ResultObject.DeviceToken);
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse
DescribeFaceVerifyAutoRoute(AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyReq
uest request)
        {
            List<string> endpoints = new List<string>
            {
                "cloudauth.cn-shanghai.aliyuncs.com",
                "cloudauth.cn-beijing.aliyuncs.com"
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse lastRespon
se = null;
            foreach (var endpoint in endpoints)
            {
                try
                {
                    AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse re
sponse = DescribeFaceVerify(endpoint, request);
                    if (AlibabaCloud.TeaUtil.Common.EqualString("500", response.Code))
                    {
                        continue;
                    }
                    return response;
                }
                catch (TeaException err)
                {
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
                catch (Exception _err)
                {
```

```
                    TeaException err = new TeaException(new Dictionary<string, object>
                    {
                        { "message", _err.Message }
                    });
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
            }
            return lastResponse;
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse
DescribeFaceVerify(string endpoint, AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceV
erifyRequest request)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Client client = CreateClient(endpoint);
            // 创建RuntimeObject实例并设置运行参数。
            AlibabaCloud.TeaUtil.Models.RuntimeOptions runtime = new AlibabaCloud.TeaUtil.M
odels.RuntimeOptions();
            runtime.ReadTimeout = 10000;
            runtime.ConnectTimeout = 10000;
            return client.DescribeFaceVerify(request, runtime);
        }
    }
}
```

2.2.8. Node.js SDK调用示例

本文介绍金融级多因子意愿认证服务端接入的Node.js SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的Node.js版本在8.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于Node.js SDK源代码，请参见Node.js SDK。

建议您使用npm管理依赖包：

```
npm install @alicloud/tea-util
npm install @alicloud/rpc-client
npm install @alicloud/cloudauth20190307@1.0.7
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
const Config = require("@alicloud/rpc-client");
const CloudAuth = require("@alicloud/cloudauth20190307");
const CloudAuthClient = CloudAuth.default;
const {RuntimeOptions} = require("@alicloud/tea-util");
     function createClient(endpoint) {
        // 通过以下代码创建Config实例并初始化。
        const config = new Config.Config({
            // 您的AccessKey ID。
            accessKeyId: "您的AccessKey ID",
            // 您的AccessKey Secret。
            accessKeySecret: "您的AccessKey Secret",
            // 设置Endpoint。
            endpoint: endpoint,
            // 设置HTTP代理。
            // httpProxy: "http://xxx.xxx.xxx.xxx:9999",
            // 设置HTTPS代理。
            // httpsProxy: "https://username:password@xxx.xxx.xxx.xxx:9999",
        });
        return new CloudAuthClient(config)
    }
    async function initFaceVerify(endpoint, request) {
        const client = createClient(endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        const runtime = new RuntimeOptions([]);
        runtime.readTimeout=10000;
        runtime.connectTimeout=10000;
        return await client.initFaceVerify(request, runtime);
    }
    async function initFaceVerifyAutoRoute(request) {
        const endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        for (const endpoint of endpoints) {
            try {
                const response = await initFaceVerify(endpoint, request);
                if (response.code === '500') {
                    continue;
                }
                return response;
            } catch (err) {
                continue;
            }
        }
```

```
        return null;
    }
    async function main(){
        // 通过以下代码创建API请求并设置参数。
        const request = new CloudAuth.InitFaceVerifyRequest({
        // 请输入场景ID。
        sceneId: 100000****,
        // 设置商户请求的唯一标识。
        outerOrderNo: "xxx",
        // 要接入的认证方案。
        productCode: "MFVC",
        userId: "148562088256****",
        // 固定值。
        certType: "IDENTITY_CARD",
        certName: "张三",
        certNo: "xxx",
        metaInfo: "{\"zimVer\":\"3.0.0\",\"appVersion\": \"1\",\"bioMetaInfo\": \"4.1.0:115
01568,0\",\n" +
            "        \"appName\": \"com.aliyun.antcloudauth\",\"deviceType\": \"ios\",\"osV
ersion\": \"iOS 10.3.2\",\"apdidToken\": \"\",\"deviceModel\": \"iPhone9,1\"}",
        // mobile: "1******************9",
        // ip: "114.100.XX.XX",
        // callbackUrl: "https://www.aliyundoc.com",
        // callbackToken: "xxxxxxx"
        });
        // 推荐，支持服务路由。
        const response = await initFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        // const response = await initFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", requ
est);
        console.log(response.requestId);
        console.log(response.code);
        console.log(response.message);
        console.log(response.resultObject.certifyId);
        }
main().then(function (response) {});
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
const Config = require("@alicloud/rpc-client");
const CloudAuth = require("@alicloud/cloudauth20190307");
const CloudAuthClient = CloudAuth.default;
const {RuntimeOptions} = require("@alicloud/tea-util");
     function createClient(endpoint) {
        // 通过以下代码创建Config实例并初始化。
        const config = new Config.Config({
            // 您的AccessKey ID。
            accessKeyId: "您的AccessKey ID",
            // 您的AccessKey Secret。
            accessKeySecret: "您的AccessKey Secret",
            // 设置endpoint
            endpoint: endpoint
```

```
            endpoint: endpoint,
            // 设置http代理。
            // httpProxy: "http://xxx.xxx.xxx.xxx:9999",
            // 设置https代理。
            // httpsProxy: "https://username:password@xxx.xxx.xxx.xxx:9999",
        });
        return new CloudAuthClient(config)
    }
    async function describeFaceVerify(endpoint, request) {
        const client = createClient(endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        const runtime = new RuntimeOptions([]);
        runtime.readTimeout=10000;
        runtime.connectTimeout=10000;
        return await client.describeFaceVerify(request, runtime);
    }
    async function describeFaceVerifyAutoRoute(request) {
        const endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        for (const endpoint of endpoints) {
            try {
                const response = await describeFaceVerify(endpoint, request);
                if (response.code === '500') {
                    continue;
                }
                return response;
            } catch (err) {
                continue;
            }
        }
        return null;
    }
    async function main(){
        // 通过以下代码创建API请求并设置参数。
        const request = new CloudAuth.DescribeFaceVerifyRequest({
            sceneId: 100000****,
            certifyId: "xxx"
        });
        // 推荐，支持服务路由。
        const response = await describeFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        //  const response = await describeFaceVerify("cloudauth.cn-shanghai.aliyuncs.com",
request);
        console.log(response.requestId);
        console.log(response.code);
        console.log(response.message);
        console.log(response.resultObject.passed);
        console.log(response.resultObject.identityInfo);
        console.log(response.resultObject.materialInfo);
        console.log(response.resultObject.deviceToken);
        console.log(response.resultObject.subCode);
        }
main().then(function (response) {});
```

2.3. App（SDK）接入

2.3.1. Android意愿认证接入

金融级实人认证服务提供Android客户端SDK，帮助您在业务应用（App）中实现刷脸认证功能。您可以通过应用服务端认证初始化接口，向阿里云服务器调用发起认证请求接口并获取刷脸认证的唯一标识CertifyId，然后使用CertifyId唤起刷脸认证客户端SDK。本文结合示例代码介绍Android客户端的接入流程。

使用限制

金融级实人认证服务Android客户端SDK具有以下使用限制。

- 不支持模拟器模式调试。
- 仅支持Android 5.0及以上系统版本的移动智能设备（手机或Pad）接入。

依赖配置

下载意愿认证Android SDK，该SDK为Android标准AAR包，下载之后在*Android Studio*中进行AAR包引用。例如下载完AAR包之后，放入工程目录的*libs*目录下，并添加以下依赖：

```
# 添加阿里云意愿认证SDK依赖包。
implementation files('libs/face-xxx.aar')
implementation files('libs/ocr-xxx.aar')
implementation files('libs/wishverify-xxx.aar')
implementation files('libs/photinus-xxx.aar')
implementation files('libs/Android-AliyunDevice-xxx.aar')
implementation files('libs/APSecuritySDK-DeepSec-xxx.aar')
implementation 'com.squareup.okhttp3:okhttp:3.11.0'
implementation 'com.squareup.okio:okio:1.14.0'
implementation 'com.alibaba:fastjson:1.2.62'
implementation 'com.aliyun.dpa:oss-android-sdk:+'
```

说明 以上代码示例中xxx表示版本号，使用过程中，请替换为SDK包中文件的实际版本号。

接口定义

```
/**
* 刷脸认证接口。
*/
public class ZIMFacade {
    /**
    * 初始化接口，建议接入方在调用其它接口前，先调用此接口。
    *
    * @context Context
    */
    public static void install(Context context);
    /**
    * 获取环境参数接口。
    *
    * @context Context
    */
    public static String getMetaInfos(Context context);
    /**
    * identify interface for android
    * 需在UI线程上调用，不会阻塞调用线程。
    *
    * @param certifyId：刷脸认证唯一标识，请从刷脸认证服务端认证初始化接口获取。
    * @param useMsgBox：当刷脸认证过程中出现异常情况，是否使用SDK内部的弹框提示。
    *                    如果取值为true，则SDK先弹框提示，确定之后，返回错误code。
    *                    如果取值为false，则不弹提示框，直接返回错误code，客户业务应用App自行决定
如何提示。
    * @param callback：认证结果的回调接口。
    */
    public void verify(String certifyId, boolean useMsgBox, ZIMCallback callback);
    /**
    * 开始进行实名认证，带自定义参数的版本。
    * @param certifyId：刷脸认证唯一标识，请从刷脸认证服务端认证初始化接口获取。
    * @param useMsgBox：表示是否在SDK内部使用弹框提示错误信息。
    * @param extParams：扩展参数。
    * @param callback：identify result
    */
    public void verify(final String zimId, boolean useMsgBox, HashMap<String, String> extPa
rams, ZIMCallback callback);
}
```

接口调用

1. 初始化SDK。

   为了提高身份核验的用户体验，并为刷脸认证准备必要的数据，Android客户端需要进行SDK初始化。初始化的代码示例如下：

   ```
   ZIMFacade.install(context);
   ```

2. 获取MetaInfo数据。

   您在调用刷脸认证服务端发起认证请求时，需要传入该MetaInfo值。代码示例如下：

   ```
   String metaInfos = ZIMFacade.getMetaInfos(context);
   ```

3. 开始认证。接入的示例代码如下：

```
ZIMFacade zimFacade = ZIMFacadeBuilder.create(MainActivity.this);
zimFacade.verify(certifyId, true, new ZIMCallback() {
 @Override
 public boolean response(ZIMResponse response) {
     // TODO: 根据实人认证回调结果处理自身的业务。
     if (null != response && 1000 == response.code) {
         // 认证成功。
     } else {
         // 认证失败。
     }
     return true;
 }
}
```

说明

- certifyId：刷脸认证唯一标识，可从刷脸认证服务端发起认证请求获取。
- ZIMCallback的response回调需要有返回值，默认为true。

4. 返回刷脸认证回调结果。

   刷脸认SDK回调结果中的ZIMResponse类，该类中定义了对应的结果编码和原因，代码示例如下：

```
/**
 * 刷脸认证回调结果。
 */
public class ZIMResponse {
    /**
     * 返回结果编码。
     */
    public int code;
    /**
     * 返回结果原因信息。
     */
    public String reason;
}
```

返回结果说明

错误码	错误码文案	错误码描述
1000	刷脸成功	代表刷脸成功，该结果仅供参考，可通过服务端查询认证结果获取最终认证结果。
1001	系统错误	表示系统错误。
1003	验证中断	表示验证中断。
2002	网络错误	表示网络错误。
2003	客户端设备时间错误	表示客户端设备时间错误。

错误码	错误码文案	错误码描述
2006	刷脸失败	表示刷脸失败，如需获取更详细的失败原因，需调用服务端查询认证结果。

说明 更多错误码信息，请参见Android客户端错误码详情。

接口混淆配置

为避免接口混淆，您可以参考以下代码来保留类名（一般是在封装release包的时候保留类名）：

```
# SDK编译 Keep
-keep class com.alipay.deviceid.** { *; }
-keep class net.security.device.api.** {*;}
-keep class com.alibaba.fastjson.** {*;}
-keep class com.alibaba.sdk.android.oss.** { *; }
-dontwarn okio.**
-dontwarn org.apache.commons.codec.binary.**
-keepclassmembers,allowobfuscation class * {
     @com.alibaba.fastjson.annotation.JSONField <fields>;
}
```

权限说明

金融级实人认证为您提供安全的全链路身份核验服务。为了达到更好的设备风险防范效果，建议您在使用业务应用App调用刷脸SDK之前，提前向您的用户获取如下权限。

```
// 授权获取手机状态的权限。
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
// 授权获取接入网络的权限。开发人员需注意，6.0以上系统属于动态权限。
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Demo代码包

您可以下载Android Demo代码包进行体验。

SDK包大小裁剪说明

如果您不需要使用OCR功能，可以在SDK包中删除*ocr-xxxx.aar*，并重新编译。

如果您不需要使用意愿认证功能，只使用实人认证功能，可以在SDK包删除wishverify-xxx.aar，并重新编译。

意愿认证升级事项

如果您之前没有接入过金融级实人认证的方案，请忽略此信息。如果您在使用意愿认证之前接入过金融级实人认证的方案，那么您需要删除项目中部分实人认证方案的相关配置以及AAR包文件，需要删除的依赖配置如下：

```
implementation files('libs/android-aliyunface-sdk-release-xxx.aar')
implementation files('libs/android-aliyunbasicstl-sdk-release-xxx.aar')
implementation files('libs/android-aliyuncomm-sdk-release-xxx.aar')
implementation files('libs/android-aliyunocr-sdk-release-xxx.aar')
implementation files('libs/tygerservice-xxx.aar')
```

配置删除之后您原有的认证逻辑会编译报错，会存在相关的类查找不到的错误，请根据Android Studio的提示，先删除原有的import逻辑，重新import相关的类别即可。意愿认证会兼容您之前的所有逻辑，其他无需做任何调整。

2.3.2. iOS意愿认证接入

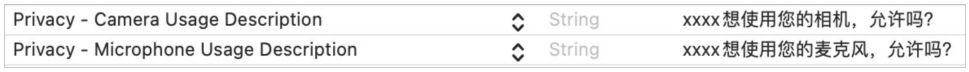
刷脸认证提供iOS客户端SDK，帮助您在您的业务应用（App）中实现刷脸认证功能。您可通过刷脸认证的服务端认证初始化接口，获取刷脸认证唯一标识CertifyId，并使用CertifyId唤起刷脸认证客户端SDK。本文将结合示例代码进行iOS客户端接入的详细说明。

前提条件

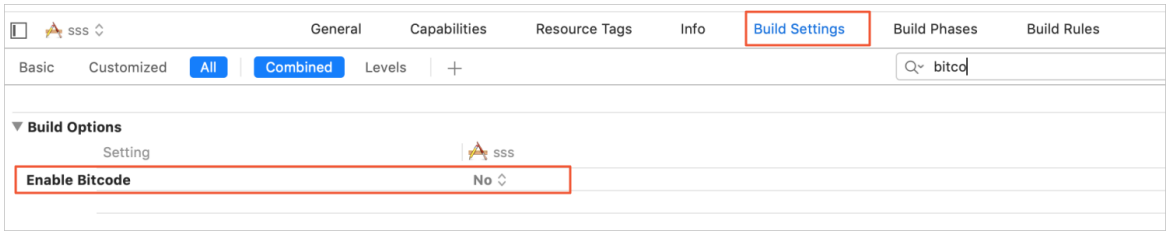
- 意愿认证应用必须运行在iOS 11.0及以上版本的系统上。
- 您的业务应用必须采用Objective C++或Swift集成金融级实人认证服务。
- 您的业务应用中未集成支付宝mPaaS SDK。

配置开发环境

1. 在*info.plist*中配置摄像头与麦克风权限请求。



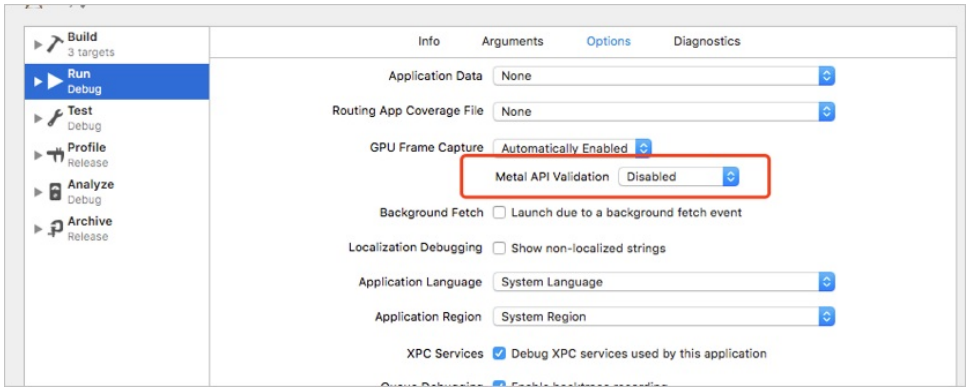
2. 在Xcode的编译设置中关闭Enable Bitcode选项。



3. 在Xcode编译设置的Linking > Other Linker Flags中，添加设置-ObjC -framework "BioAuthAPI" -lxml2。

   如果您的工程已设置了-force_load选项，则需要加入-force_load <framework path>/AliyunOSSiOS。

4. 在Xcode调试设置的Edit Scheme > Run > Options中禁用Metal API Validation。



配置依赖

1. 下载意愿认证iOS SDK。

   该SDK为framework包。您需要在Xcode添加**Link Binary With Libraries**、SDK中的15个包和额外系统库依赖。

2. 解压下载的SDK包，并在解压的文件夹中执行以下命令提取所有的framework到Products目录。

```
for i in $(ls *.tgz);do tar xvf $i;done
```

   - SDK中的包：

```
APPSecuritySDK
MPRemoteLogging
AliyunOSSiOS
BioAuthEngine
ZolozUtility
AliyunIdentityManager
APBToygerFacade
MultiFactorFacade
BioAuthAPI
ToygerService
ZolozSensorServices
ZolozOpenPlatformBuild
deviceiOS
ZolozMobileRPC
ZolozIdentityManager
OCRDetectSDKForTech
ToygerNative
```

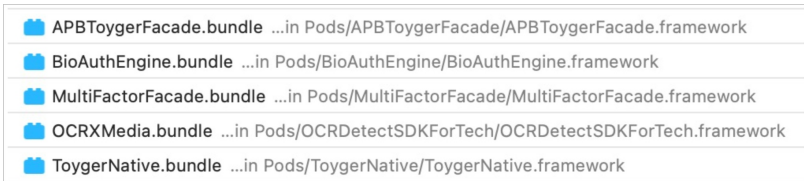
   - 系统库依赖：

```
CoreGraphics.framework
Accelerate.framework
SystemConfiguration.framework
AssetsLibrary.framework
CoreTelephony.framework
QuartzCore.framework
CoreFoundation.framework
CoreLocation.framework
ImageIO.framework
CoreMedia.framework
CoreMotion.framework
AVFoundation.framework
ReplayKit.famework
WebKit.framework
libresolv.tbd
libz.tbd
libc++.tbd
libc++.1.tbd
libc++abi.tbd
AudioToolbox.framework
CFNetwork.framework
MobileCoreServices.framework
libz.1.2.8.tbd
AdSupport.framework
```

拷贝资源文件

选择TARGETS，单击Build Phases页签，在Copy Bundle Resources中添加以下4个bundle：

- **APBToygerFacade.bundle**：位于*APBToygerFacade.framework*。
- **ToygerNative.bundle**：位于*ToygerNative.framework*。
- **OCRXMedia.bundle**：位于*OCRDetectSDKForTech.framework*。
- **BioAuthEngine.bundle**：位于*BioAuthEngine.framework*。
- **MultiFactorFacade.bundle**：位于*MultiFactorFacade.framework*。



调用SDK

1. 引入头文件。

   命令如下：

   ```
   #import <AliyunIdentityManager/AliyunIdentityPublicApi.h>
   ```

2. 初始化SDK。

> **说明** 为了提升用户体验并为刷脸认证准备一些必备的数据，iOS客户端需要初始化SDK。您可以在调用 `appDelegate` 函数时（如下例代码所示）放置初始化SDK代码。您也可以在获取metaInfo数据前（如步骤所示）放置初始化SDK代码。

`appDelegate` 函数调用，示例代码：

```
(BOOL)application:(UIApplication)application didFinishLaunchingWithOptions:(NSDictionary)launchOptions { [AliyunSdk init];// 初始化SDK。 }
```

3. 获取metaInfo数据。

   命令如下：

```
[AliyunSdk init];// 初始化SDK，如果已经在appDelegate函数中调用过，此处无需再重复调用。
[AliyunIdentityManager getMetaInfo];
```

> **说明** 返回值类型是NSDictionary，您的服务端需要将其转换为JSON字符串。调用服务端发起认证请求接口时，在入参metaInfo中传入该值。关于发起认证请求的具体操作，请参见发起认证请求。

4. 调用SDK开始认证。

   命令如下：

```
[[AliyunIdentityManager sharedInstance] verifyWith:certifyId extParams:extParams onCompletion:^(ZIMResponse *response) {}];
```

参数说明如下表所示。

名称	类型	描述
certifyId	NSString	您可以通过服务端发起申请请求接口获取该参数。关于发起认证请求的具体操作，请参见发起认证请求。
extParams	NSDictionary	通过以下代码传入当前viewController，用于展现网络加载和presentViewController。 `[extParams setValue:self forKey:@"currentCtr"];`

返回结果说明：

```
[[AliyunIdentityManager sharedInstance] verifyWith:certifyId extParams:extParams onCompletion:^(ZIMResponse *response) {}];
```

`response.code` 包含以下返回参数，具体说明如下表所示。

错误码	错误码文案	错误码描述

错误码	错误码文案	错误码描述
1000（ZIMResponseSuccess）	刷脸成功	表示刷脸成功。 说明 该结果仅供参考，可通过服务端查询认证结果接口获取最终认证结果。关于发起认证请求的具体操作，请参见发起认证请求。
1001（ZIMInternalError）	系统错误	表示系统错误。
1003（ZIMInterrupt）	验证中断	表示验证中断。
2002（ZIMNetworkfail）	网络错误	表示网络错误。
2003（ZIMTIMEError）	客户端设备时间错误	表示客户端设备时间错误。
2006（ZIMResponseFail）	刷脸失败	表示刷脸失败。 说明 如需获取失败原因的详细信息，需调用服务端查询认证结果接口。关于发起认证请求的具体操作，请参见发起认证请求。

说明 关于错误码的更多信息，请参见iOS客户端错误码详情。

代码示例：

```
[[AliyunIdentityManager sharedInstance] verifyWith:certifyId extParams:extParams onComp
letion:^(ZIMResponse *response) {
        dispatch_async(dispatch_get_main_queue(), ^{
            NSString *title = @"刷脸成功";
            switch (response.code) {
                case ZIMResponseSuccess: //1000。
                    break;
                case ZIMInterrupt:   //1003。
                    title = @"用户退出";
                    break;
                case ZIMNetworkfail: //2002。
                    title = @"网络错误";
                    break;
                case ZIMTIMEError: //2003。
                    title = @"设备时间设置不对";
                    break;
                case ZIMResponseFail: //2006。
                    title = @"刷脸失败";
                    break;
                case ZIMInternalError:   //1001。
                    title = @"初始化失败";
                    break;
                default:
                    break;
            }
```

Demo代码包

您可以下载iOS Demo代码包进行体验。在使用过程中，请替换iOS Demo代码包中的文件版本为最新版本。

SDK包大小裁剪说明

当您对SDK包大小有限制时，可以根据如下情况进行裁剪：

- 如果您之前没有使用过支付宝开放平台版，或开放平台版代码已经下线，可以在SDK包中删除ZolozIdentityManager、ZolozMobileRPC内容。
- 如果您不需要使用OCR功能，可以在SDK包中删除OCRDetectSDKForTech内容，并删除对应的bundle（OCRXMedia.bundle）。
- 如果只需要使用意愿认证，可以不添加APBToygerFacade。

3.金融级活体人脸验证方案

3.1. 接入流程

无线SDK和服务端接入适用于您自身已有手机App应用，且希望通过该App对用户进行线上认证的场景。

前置条件

您必须了解通过HTTP、HTTPS原生调用金融级实人认证服务接口时，构造请求签名的方法。更多签名机制的信息，请参见API签名机制。

操作步骤

1. 开通金融级实人认证服务。
2. 添加认证场景。

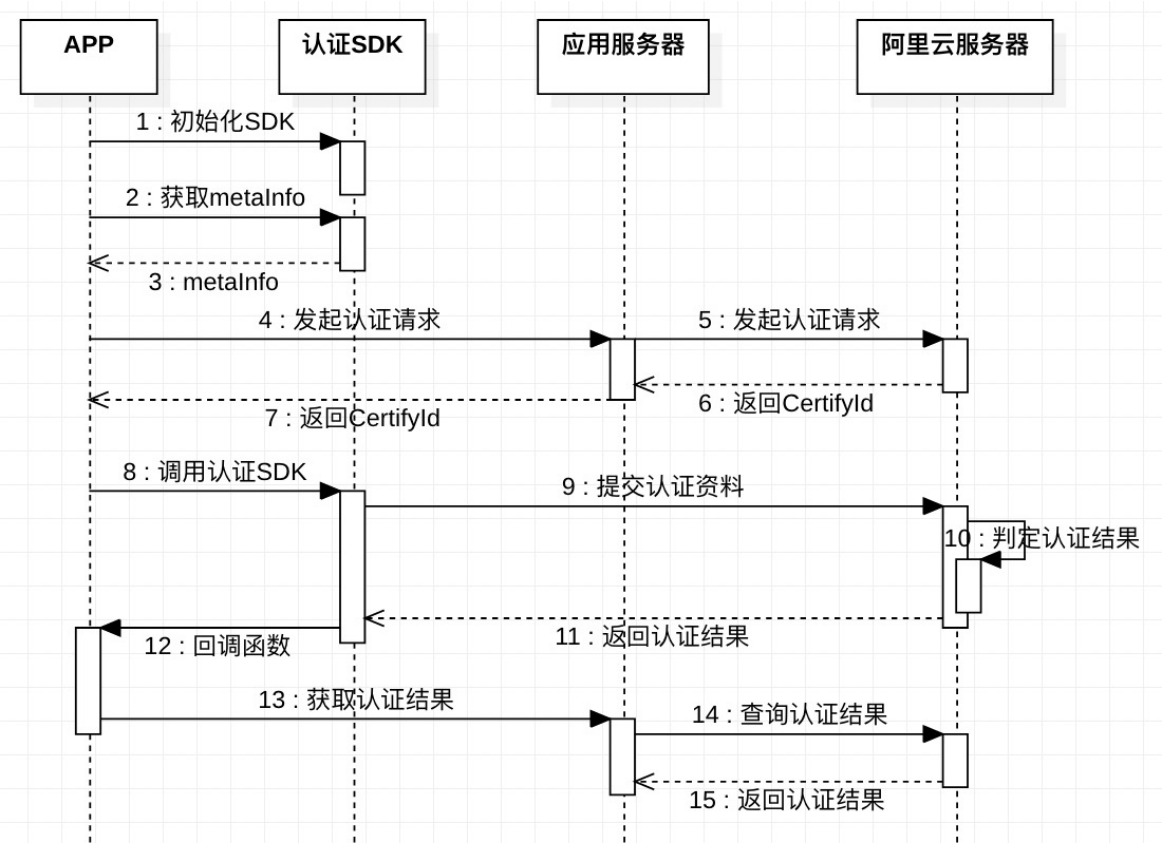
   > **说明** 基于不同业务场景，您可以在添加认证场景时，手动配置是否授权阿里云访问指定OSS存储空间用于存储认证的资料图片。

3. 如果需要将认证过程中的人脸采集照片存储到OSS空间，您需要授权金融级实人认证访问OSS存储空间。
4. 接入活体人脸验证方案。

   在您完成上述步骤后，您可以进行无线SDK+服务端接入，接入步骤如下：

   i. 调用服务端初始化接口。
   ii. 调用接入客户端SDK。
   iii. 调用服务端查询接口。

   您可以查看下方调用时序图了解详细接入步骤。



系统调用时序说明表

序号	说明
1	初始化认证SDK。
2	App调用认证SDK获取MetaInfo。
3	认证SDK返回MetaInfo。
4	App向应用服务端发起认证请求并传递MetaInfo。
5	应用服务端向阿里云服务端调用发起认证请求接口。
6	阿里云服务端返回CertifyId给应用服务端。
7	应用服务端将CertifyId传递给App。
8	App调用认证SDK。
9	认证SDK将用户提交的认证资料传递给阿里云服务端。
10	阿里云服务端根据资料判定认证结果。
11	阿里云服务端将认证结果返回给SDK。
12	SDK通过回调函数指引App获取认证状态。

序号	说明
13	App向应用服务端查询认证状态。
14	应用服务端向阿里云服务端调用查询认证结果接口。
15	阿里云服务端将认证结果返回给应用服务端。

在您了解接入步骤后，您可以通过以下方式接入活体人脸验证方案。

- 进行服务端接入，请参见InitFaceVerify。
- 在应用客户端集成SDK，请参见以下文档：
  - Android接入
  - iOS接入
  - UniApp接入

3.2. 服务端接入

3.2.1. InitFaceVerify

本文介绍了如何通过InitFaceVerify接口进行活体人脸验证。

传入图片格式要求

当您的终端用户在进行实人认证的时候，请确保传入的照片满足以下所有条件：

- 近期照片，人脸完整清晰无遮挡，表情自然，正对摄像头，角度无太大偏差。
- 照片清晰、曝光正常，脸部无过黑、过亮、光晕等情况。
- 照片的分辨率不超过1920*1080（高*宽），不低于640*480（高*宽），且照片的高大于宽。推荐短边缩放到720像素，压缩率大于0.9。
- 照片小于1 MB。
- 照片支持顺时针旋转90°、180°和270°。
- 照片中有且只有一张人脸。

> **说明** 如果一张照片中存在多张人脸，默认会选取面积最大的人脸进行验证。

正确的图片示例：

发起认证请求

接口名：InitFaceVerify。

服务地址：`cloudauth.aliyuncs.com`。

请求方法：HTTPS POST和GET。

接口说明：每次开始认证前调用本接口获取CertifyId，用来串联认证请求中的各个接口。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	认证场景ID。该ID在控制台创建认证场景后自动生成。	100000****
OuterOrderNo	String	是	您自定义的业务唯一标识，用于后续定位排查问题使用。 值最长为32位长度的字母和数字组合，请确保唯一。	e0c34a77f5ac40a5aa5e6ed20c35****
ProductCode	String	是	认证方案。 唯一取值：PV_FV。	PV_FV

名称	类型	是否必选	描述	示例值
Model	String	否	活体检测类型。取值： • LIVENESS（默认）：眨眼动作活体检测。 • PHOTINUS_LIVENESS：眨眼动作活体+炫彩活体双重检测。 • MULTI_ACTION：多动作活体检测。当前为眨眼+任意摇头检测。 说明 活体检测模式仅支持运行在Android SDK 1.2.6及以上版本和iOS SDK 1.2.4及以上版本；多动作活体检测模式仅支持运行在Android SDK 1.6.4及以上版本和iOS SDK 1.5.1及以上版本。	LIVENESS
UserId	String	是	您自定义的用户ID，请保持唯一。	12345****
CertType	String	否	证件类型。 当前仅支持身份证。唯一取值：IDENTITY_CARD。 说明 该方案中，不验证姓名和证件号码的真实性、一致性。如需验证，请选择实人认证方案。	IDENTITY_CARD
CertName	String	否	您的终端用户的真实姓名。	张三
CertNo	String	否	您的终端用户的证件号码。	33010312345678****
MetaInfo	String	是	MetaInfo环境参数。需要通过客户端SDK获取。	{"zimVer":"3.0.0","appVersion": "1","bioMetaInfo": "4.1.0:11501568,0","appName": "com.aliyun.antcloudauth","deviceType": "ios","osVersion": "iOS 10.3.2","apdidToken": "","deviceModel": "iPhone9,1"}

名称	类型	是否必选	描述	示例值
Mobile	String	否	您的终端用户的手机号码。	1304567****
Ip	String	否	您的终端用户的IP。	114.112.xx.xx
FaceContrastPicture	String	否	照片Base64编码。如果您选择FaceContrastPicture（照片Base64编码）方式传入人脸照片，请注意检查照片大小，不要传入过大的照片；您可以通过OSS方式上传较大的人脸照片。 **说明** 您需要在下述四种传入人脸照片方式中，选择其中一种： • FaceContrastPicture • FaceContrastPictureUrl • CertifyId • OSS	/9j/4AAQSkZJRgABAQAASASBC****
FaceContrastPictureUrl	String	否	人像地址，公网可访问的HTTP、HTTPS链接。 **说明** 您需要在下述四种传入人脸照片方式中，选择其中一种： • FaceContrastPicture • FaceContrastPictureUrl • CertifyId • OSS	https://cn-shanghai-aliyun-cloudauth-****.oss-cn-shanghai.aliyuncs.com/verify/****.jpeg

名称	类型	是否必选	描述	示例值
CertifyId	String	否	之前实人认证通过的CertifyId，认证时的照片作为比对照片。您需要确保传入CertifyId关联的图片正常存储在授权的OSS Bucket中。 **说明** 您需要在下述四种传入人脸照片方式中，选择其中一种： • FaceContrastPicture • FaceContrastPictureUrl • CertifyId • OSS	0bfa7c493f850e5178b9f8613634****
OssBucketName	String	否	已授权OSS的存储空间（Bucket）名称。如果您选择OSS方式传入人脸照片，您必须传入OssBucketName和OssObjectName参数。OSS请求参数格式仅支持开通实人认证服务时授权的OSS生成的URL、Bucket和文件名。 **说明** 您需要在下述四种传入人脸照片方式中，选择其中一种： • FaceContrastPicture • FaceContrastPictureUrl • CertifyId • OSS	cn-shanghai-aliyun-cloudauth-****

名称	类型	是否必选	描述	示例值
OssObjectName	String	否	已授权OSS存储空间的文件（Object）名称。如果您选择OSS方式传入人脸照片，您必须传入OssBucketName和OssObjectName参数。OSS请求参数格式仅支持开通实人认证服务时授权的OSS生成的URL、Bucket和文件名。 **说明** 您需要在下述四种传入人脸照片方式中，选择其中一种： • FaceContrastPicture • FaceContrastPictureUrl • CertifyId • OSS	verify/****.jpeg
CallbackUrl	String	否	认证结果的回调通知地址，必须以 `https` 开头。 平台在完成认证后会回调该地址，并自动添加 `certifyId` 、 `passed` 和 `subcode` 字段，示例： `https://www.aliyun.com?certifyId=xxxx&passed=T&subcode=200` 。	https://www.aliyun.com
CallbackToken	String	否	安全Token，由您自行生成，用于防重复、防篡改校验。 如果设置了该值会在回调地址显示**CallbackToken**字段。	NMjvQanQgplBSaEI0sL86WnQplB

名称	类型	是否必选	描述	示例值
Crop	String	否	是否允许裁剪人脸图片，默认不允许。 ● T：表示允许剪裁。 ● F：表示不允许剪裁。 说明 若您请求的图片并非来自标准活体检测SDK采集，建议您允许裁剪人脸图片。 开启该功能后，将会对请求的图片先进行人脸裁剪与矫正，完成后会发起请求服务。	T
AuthId	String	否	用户授权ID，最大长度为64位字符。	92d46b9e9e2d703f2897f350d5bd4149

说明

OSS入参格式仅支持开通实人认证服务时授权的OSS生成的URL、Bucket和文件名 。关于如何查询授权的OSS，请参见接入流程 。

如果您选择FaceContrastPicture（照片Base64编码）方式传入人脸照片，请注意检查照片大小，不要传入过大的照片；您可以通过OSS方式上传较大的人脸照片。

返回数据

名称	类型	是否必选	描述	示例值
RequestId	String	是	本次请求的ID。	130A2C10-B9EE-4D84-88E3-5384FF03****
Message	String	否	请求消息的响应信息。	success
Code	String	是	返回码。关于返回码的具体内容，请参见返回Code和Message说明。	200
ResultObject.CertifyId	String	否	实人认证唯一标识。	91707dc296d469ad38e4c5efa6a0****

返回Code和Message说明

Code	Message	描述
200	success	成功。

Code	Message	描述
400	参数不能为空	参数不能为空。
401	参数非法	非法参数。传入的姓名、身份证号码长度必须符合国家标准且不得包含英文字母等特殊字符。
402	应用配置不存在	应用配置不存在。
404	认证场景配置不存在	认证场景配置不存在，请先在控制台上创建认证场景。
410	未开通服务	未开通OSS产品或未完成OSS读写授权，请登录金融级实人认证控制台完成授权。具体操作，请参见授权金融级实人认证访问OSS存储空间
411	RAM无权限	需要给RAM账号授予**AliyunAntCloudAuthFullAccess**的操作权限。
412	欠费中	金融级实人认证或OSS存在欠费，请充值后操作。
414	设备类型不支持	当前移动设备不支持刷脸认证，请更换设备后操作。
415	SDK版本不支持	当前认证SDK版本不支持刷脸认证，请升级SDK后操作。
416	系统版本不支持	当前操作系统版本不支持刷脸认证，请升级系统或更换设备操作。
417	无法使用刷脸服务	自定义比对源的图片人脸特征提取失败，建议更换图片上传。
418	刷脸失败次数过多	当天刷脸认证次数过多，请明天再试。
419	传入图片不可用	图片无法下载、图片内容为空或图片分辨率不符合要求。
420	数据重复	图片传入方式超过了一种。 支持以下四种传入人脸照片的方式，必须且只能使用其中一种： • FaceContrastPicture • FaceContrastPictureUrl • Oss • CertifyId
421	传入图片过大	图片超过了1 MB，建议压缩图片或更换图片上传方式。
422	下载图片超时	图片下载超过了3秒，请排查网络后重新操作。
423	状态错误	传入的CertifyId认证状态需要为T（认证通过），您也可以更换其他方式传入图片。
500	系统错误	系统内部错误，请反馈工程师排查。

3.2.2. DescribeFaceVerify

本文介绍了如何通过DescribeFaceVerify接口查询活体人脸验证的结果。

查询认证结果

接口名：DescribeFaceVerify。

服务地址：cloudauth.aliyuncs.com。

请求方法：HTTPS POST和GET。

接口说明：当您的终端用户收到回调通知之后，您在服务端通过该接口获取相应的认证状态和认证资料。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	认证场景ID。	100000****
CertifyId	String	是	实人认证唯一标识。该值需要调用InitFaceVerify获取。	91707dc296d469ad38e4c5efa6a0****

返回数据

名称	类型	描述	示例值
RequestId	String	请求ID。	130A2C10-B9EE-4D84-88E3-5384FF0****
Message	String	请求消息的响应信息。	success
Code	String	返回码。关于返回码的详细内容，请参见SubCode错误码说明。	200
ResultObject	JSON	认证结果。该认证结果是一个JSON结构体，关于具体结构描述，请参见ResultObject。	

ResultObject

名称	类型	描述	示例值
Passed	String	认证结果。取值： • T：通过。 • F：不通过。 说明 判断认证结果请以Passed字段为准。	T
SubCode	String	认证结果描述。更多信息，请参见SubCode错误码说明。 说明 判断认证结果请以Passed字段为准。	200
IdentityInfo	String	认证的主体信息，一般的认证场景返回为空。	null

名称	类型	描述	示例值
DeviceToken	String	设备Token。	McozS1ZWRcRZStlERcZZo_QOytx5jcgZoZJEoRLOABBF****
MaterialInfo	String	认证主体附件信息，主要为图片类材料。JSON格式，更多信息，请参见返回示例。	{"faceAttack": "F","faceOcclusion": "F","facialPictureFront": {"faceAttackScore": 0.00008597839769208804,"qualityScore": 88.3615493774414,"verifyScore": 50.28594166529785,"pictureUrl": "https://cn-shanghai-aliyun-cloudauth-****.oss-cn-shanghai.aliyuncs.com/verify/****.jpeg","ossBucketName": "cn-shanghai-aliyun-cloudauth-1260051251634779","ossObjectName": "verify/1260051251634779/6ba7bcfccf33f56cdb44ed086f36c****.jpeg"}}

返回示例

ResultObject.MaterialInfo的JSON格式示例：

```
{
     // 是否为攻击：攻击为T，非攻击为F。
    "faceAttack": "F",
    // 是否有脸部遮挡：有脸部遮挡为T，否则为F。
    "faceOcclusion": "F",
    // 认证的照片信息。
    "facialPictureFront": {
        // 人脸攻击分。
        "faceAttackScore": 0.00008597839769208804,
        // 活体人脸质量分数。
        "qualityScore": 88.3615493774414,
        // 人脸比对分数。
        "verifyScore": 50.28594166529785,
        // 照片https地址，有效期15分钟，从查询时开始计时。
        "pictureUrl": "https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-shanghai.aliyuncs
.com/verify/xxxxx/xxxxx.jpeg",
        // 照片存放客户上海区域的OSS里，此为照片的bucket名。
        "ossBucketName": "cn-shanghai-aliyun-cloudauth-126005125163****",
        // 照片存放客户上海区域的OSS里，此为照片的文件名。
        "ossObjectName": "verify/1260051251634779/6ba7bcfccf33f56cdb44ed086f36****.jpeg",
        // 视频存放客户上海区域的OSS里，此为视频的文件名（仅当客户端设置视频参数时返回）。
        "ossVerifyVideoObjectName": "verify/1260051251634779/03a081bd96328aedf83f635****_ve
rifyvideo_dcb7.mov",
        // 视频HTTPS地址，有效期15分钟，从查询时开始计时（仅当客户端设置视频参数时返回）。
        "verifyVideoUrl": "http://cn-shanghai-aliyun-cloudauth-1260051251634779.oss-cn-shan
ghai.aliyuncs.com/verify/1260051251634779/03a081bd96328aedf83f635f39a50c57__verifyvideo_dcb
7.mov?Expires=1625371140&OSSAccessKeyId=STS.NTX1ngfr6Acg2Pmnn2RYM****&Signature=Hw5BF9WxJs6
wI68IxKs41cxCU8****&security-token=CAISjgJ1q6Ft5B2yfSjIr5ftetTTi60X9qGMMHbcim5nXtZhu7GT1Dz2
IH1PdXFgA%2Bgds%2Fswmm5U7vgalrkqEcEdHRGdN5YpsM8LrlzwO1h2TGRsq%2B5qsoasPETOITyZtZagToeUZdfZf
ejXGDKgvyRvwLz8WCy%2FVli%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGfLC1dysQcO4gEW
q4bHm5fCskKD1Qenk7FO%2B9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWwIMvErYbbaMrIV
1WwZ9Z7knHaVAq%2BXwnOBkuuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAF%2FHZbelNLYWtipAu5X5oG1t22SqI
r1p3TMK5FrjDIVeWOppcxeRXtU%2BjR7hGcwd25scGhiraoxTXV3sxw%2F6dhFSswQ37O4j%2B%2FrCPtiGauqn2ZMd
Mj%2FvYwKb6KmpQqa%2BtR%2F9cxhS6hoIQRq0SUIfyXl5ZUiZbTjW22iwuX%2BPwzVCw%3D%3D"
    }
}
```


SubCode错误码说明

错误码	认证描述文案	是否计费	可能原因和运营建议
200	认证通过	是	无。
204	人脸比对不一致	是	可能不是同一人或活体照片质量较低。建议根据业务情况分层处理，若为同一人可重新操作。
205	活体检测存在风险	是	可能存在攻击风险。建议人工审核分层处理，若为真人可重新操作。
206	业务策略限制	是	个性化业务安全策略限制，如有疑问可咨询客服。

3.2.3. Java SDK调用示例

本文介绍了金融级活体人脸验证方案Java SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

本章节提供的是最新版本的Java SDK调用示例，如果您需要查看之前版本，请参见Java SDK调用示例（V1版本）。

前提条件

需要使用的Java版本在1.8版本及以上。

使用Maven引入SDK

```
<dependency>
       <groupId>com.aliyun</groupId>
       <artifactId>cloudauth20190307</artifactId>
        <version>1.0.7</version>
</dependency>
```

调用示例

InitFaceVerify接口调用示例

```
import java.util.Arrays;
import java.util.List;
import com.aliyun.cloudauth20190307.Client;
import com.aliyun.cloudauth20190307.models.*;
import com.aliyun.tea.TeaException;
import com.aliyun.tea.TeaUnretryableException;
import com.aliyun.tearpc.models.Config;
import com.aliyun.teautil.models.RuntimeOptions;
public class InitFaceVerify {
    public static void main(String[] args) throws Exception {
        InitFaceVerifyRequest request = new InitFaceVerifyRequest();
         // 请输入场景ID+L。
        request.setSceneId(xxxL);
        // 设置商户请求的唯一标识。
        request.setOuterOrderNo("xxxx");
        // 认证方案。
        request.setProductCode("PV_FV");
        request.setUserId("12345xxxx");
        // 模式。
        request.setModel("LIVENESS");
        request.setCertType("IDENTITY_CARD");
        //request.setCertName("张三");
        //request.setCertNo("xxxx");
```

```
        // MetaInfo环境参数。
        request.setMetaInfo("{\"zimVer\":\"3.0.0\",\"appVersion\": \"1\",\"bioMetaInfo\": \
"4.1.0:11501568,0\","
            + "\"appName\": \"com.aliyun.antcloudauth\",\"deviceType\": \"ios\",\"osVersion
\": \"iOS 10.3.2\","
            + "\"apdidToken\": \"\",\"deviceModel\": \"iPhone9,1\"}");
        //request.setMobile("130xxxxxxxx");
        //request.setIp("114.xxx.xxx.xxx");
        // 请在以下四种方式中选择其一。
        // 1.照片Base64编码。
        // 2.照片OSS bucket名和文件名。
        // 3.照片OSS的URL地址。
        // 4.已经实人认证通过的CertifyId。
        // 方式一：照片Base64。
        request.setFaceContrastPicture("人脸照片Base64编码");
        // 方式二：照片OSS bucket名和文件名。
        //request.setOssBucketName("cn-shanghai-aliyun-cloudauth-xxxxx");
        //request.setOssObjectName("verify/xxxxx/xxxxxx.jpeg");
        // 方式三：照片OSS的URL地址，公网可访问。
        //request.setFaceContrastPictureUrl("https://cn-shanghai-aliyun-cloudauth-xxxxxx.os
s-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg")
        // 方式四：已经实人认证通过的CertifyId。
        //request.setCertifyId("0bfa7c493f850e5178b9f8613634c9xx");
        // 推荐，支持服务路由。
        InitFaceVerifyResponse response = initFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        //InitFaceVerifyResponse response = initFaceVerify("cloudauth.cn-shanghai.aliyuncs.
com", request);
        System.out.println(response.getRequestId());
        System.out.println(response.getCode());
        System.out.println(response.getMessage());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getCertifyId());
    }
    private static InitFaceVerifyResponse initFaceVerifyAutoRoute(InitFaceVerifyRequest req
uest) {
        // 第一个为主区域Endpoint，第二个为备区域Endpoint。
        List<String> endpoints = Arrays.asList("cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com");
        InitFaceVerifyResponse lastResponse = null;
        for (String endpoint : endpoints) {
            try {
                InitFaceVerifyResponse response = initFaceVerify(endpoint, request);
                lastResponse = response;
                // 服务端错误，切换到下个区域调用。
                if (response != null && "500".equals(response.getCode())) {
                    continue;
                }
                return response;
            } catch (Exception e) {
                // 网络异常，切换到下个区域调用。
                if (e.getCause() instanceof TeaException) {
                    TeaException teaException = ((TeaException)e.getCause());
                    if (teaException.getData() != null && "ServiceUnavailable".equals(
                        teaException.getData().get("Code"))) {
```

```
                        teaException.getData().get("code")) {
                        continue;
                    }
                }
                if (e.getCause() instanceof TeaUnretryableException) {
                    continue;
                }
            }
        }
        return lastResponse;
    }
    private static InitFaceVerifyResponse initFaceVerify(String endpoint, InitFaceVerifyReq
uest request)
        throws Exception {
        Config config = new Config();
        config.setAccessKeyId("您的AccessKey ID");
        config.setAccessKeySecret("您的AccessKey Secret");
        config.setEndpoint(endpoint);
        // 设置http代理。
        //config.setHttpProxy("http://xx.xx.xx.xx:xxxx");
        // 设置https代理。
        //config.setHttpsProxy("https://xx.xx.xx.xx:xxxx");
        Client client = new Client(config);
        // 创建RuntimeObject实例并设置运行参数。
        RuntimeOptions runtime = new RuntimeOptions();
        runtime.readTimeout = 10000;
        runtime.connectTimeout = 10000;
        return client.initFaceVerify(request, runtime);
    }
}
```

DescribeFaceVerify接口调用示例

```
import java.util.Arrays;
import java.util.List;
import com.aliyun.cloudauth20190307.Client;
import com.aliyun.cloudauth20190307.models.DescribeFaceVerifyRequest;
import com.aliyun.cloudauth20190307.models.DescribeFaceVerifyResponse;
import com.aliyun.tea.TeaException;
import com.aliyun.tea.TeaUnretryableException;
import com.aliyun.tearpc.models.Config;
import com.aliyun.teautil.models.RuntimeOptions;
public class DescribeFaceVerify {
    public static void main(String[] args) throws Exception {
        // 通过以下代码创建API请求并设置参数。
        DescribeFaceVerifyRequest request = new DescribeFaceVerifyRequest();
        // 请输入场景ID+L。
        request.setSceneId(xxxxL);
        request.setCertifyId("xxxx");
        // 推荐，支持服务路由。
        DescribeFaceVerifyResponse response = describeFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        //DescribeFaceVerifyResponse response = describeFaceVerify("cloudauth.cn-shanghai.a
liyuncs.com", request);
```

```
        System.out.println(response.getRequestId());
        System.out.println(response.getCode());
        System.out.println(response.getMessage());
        System.out.println(
            response.getResultObject() == null ? null : response.getResultObject().getPasse
d());
        System.out.println(
            response.getResultObject() == null ? null : response.getResultObject().getSubCo
de());
        System.out.println(
            response.getResultObject() == null ? null
                : response.getResultObject().getIdentityInfo());
        System.out.println(
            response.getResultObject() == null ? null
                : response.getResultObject().getDeviceToken());
        System.out.println(
            response.getResultObject() == null ? null
                : response.getResultObject().getMaterialInfo());
    }
    private static DescribeFaceVerifyResponse describeFaceVerifyAutoRoute(DescribeFaceVerif
yRequest request) {
        // 第一个为主区域Endpoint，第二个为备区域Endpoint。
        List<String> endpoints = Arrays.asList("cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com");
        DescribeFaceVerifyResponse lastResponse = null;
        for (String endpoint : endpoints) {
            try {
                DescribeFaceVerifyResponse response = describeFaceVerify(endpoint, request)
;
                lastResponse = response;
                // 服务端错误，切换到下个区域调用。
                if (response != null && "500".equals(response.getCode())) {
                    continue;
                }
                return response;
            } catch (Exception e) {
                // 网络异常，切换到下个区域调用。
                if (e.getCause() instanceof TeaException) {
                    TeaException teaException = ((TeaException)e.getCause());
                    if (teaException.getData() != null && "ServiceUnavailable".equals(
                        teaException.getData().get("Code"))) {
                        continue;
                    }
                }
                if (e.getCause() instanceof TeaUnretryableException) {
                    continue;
                }
            }
        }
        return lastResponse;
    }
    private static DescribeFaceVerifyResponse describeFaceVerify(String endpoint, DescribeF
aceVerifyRequest request)
        throws Exception {
        Config config = new Config();
```

```
        Config config = new Config();
        config.setAccessKeyId("您的AccessKey ID");
        config.setAccessKeySecret("您的AccessKey Secret");
        config.setEndpoint(endpoint);
        // 设置http代理。
        //config.setHttpProxy("http://xx.xx.xx.xx:xxxx");
        // 设置https代理。
        //config.setHttpsProxy("http://xx.xx.xx.xx:xxxx");
        Client client = new Client(config);
        // 创建RuntimeObject实例并设置运行参数。
        RuntimeOptions runtime = new RuntimeOptions();
        runtime.readTimeout = 10000;
        runtime.connectTimeout = 10000;
        return client.describeFaceVerify(request, runtime);
    }
}
```

3.2.4. PHP SDK调用示例

本文介绍了金融级活体人脸验证方案的PHP SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的PHP版本在5.6版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于PHP SDK源代码，请参见PHP SDK包。

- 通过Composer来管理项目依赖（推荐）

```
composer require alibabacloud/cloudauth-20190307 1.0.7
```

- 使用composer.json文件

```
{
   "require": {
       "alibabacloud/cloudauth-20190307": "1.0.7"
    }
}
```

如果因网络问题导致无法按照上述操作安装SDK，您可以使用阿里云Composer全量镜像。

调用示例

InitFaceVerify接口调用示例

```
<?php
namespace AlibabaCloud\SDK\Sample {
use AlibabaCloud\SDK\Cloudauth\V20190307\Cloudauth;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\InitFaceVerifyRequest;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\InitFaceVerifyResponse;
use AlibabaCloud\Tea\Utils\Utils;
use \Exception;
use AlibabaCloud\Tea\Exception\TeaError;
use AlibabaCloud\Tea\Rpc\Rpc\Config;
use AlibabaCloud\Tea\Utils\Utils\RuntimeOptions;
class InitFaceVerify {
    /**
     * @param string $endpoint
     * @return Cloudauth
     */
    public static function createClient($endpoint){
        $config = new Config([
            // 您的AccessKey ID。
            "accessKeyId" => "<您的AccessKey ID>",
            // 您的AccessKey Secret。
            "accessKeySecret" => "<您的AccessKey Secret>",
            // 设置HTTP代理。
            // "httpProxy" => "<http://xx.xx.xx.xx:xxxx>",
            // 设置HTTPS代理。
            // "httpsProxy" => "<https://xx.xx.xx.xx:xxxx>",
            "endpoint" => $endpoint
        ]);
        return new Cloudauth($config);
    }
    /**
     * @return void
     */
    public static function main(){
        $request = new InitFaceVerifyRequest([
            // 请输入场景ID。
            "sceneId" => <xxx>,
            "outerOrderNo" => "<商户请求的唯一标识>",
            // 要接入的认证方案。
            "productCode" => "<PV_FV>",
            // 要进行活体检测的类型。
            "model" => "<LIVENESS>",
            "userId" => "<xxx>",
            "certType" => "<IDENTITY_CARD>",
            // "certName" => "<张三>",
            // "certNo" => "<xxx>",
            // MetaInfo环境参数，需要通过客户端SDK获取。
            "metaInfo" => '{"zimVer":"3.0.0","appVersion": "1","bioMetaInfo": "4.1.0:115015
68,0",
        "appName": "com.aliyun.antcloudauth","deviceType": "ios","osVersion": "iOS 10.3.2",
"apdidToken": "","deviceModel": "iPhone9,1"}',
            // 请在以下四种方式中选择其一。
            // 1.照片Base64编码。
            // 2.照片OSS Bucket名和文件名。
            // 3.照片OSS的URL地址。
```

```
            // 4.已经实人认证通过的CertifyId。
            // 方式一：照片Base64。
            // "faceContrastPicture" => "<人脸照片Base64编码>",
            // 方式二：照片OSS bucket名和文件名。
            // "ossBucketName" => "<cn-shanghai-aliyun-cloudauth-xxxxx>",
            // "ossObjectName" => "<verify/xxxxx/xxxxxx.jpeg>",
            // 方式三：照片OSS的URL地址，公网可访问。
            "faceContrastPictureUrl" => "<https://cn-shanghai-aliyun-cloudauth-****.oss-cn-
shanghai.aliyuncs.com/verify/****.jpeg>",
            // 方式四：已经实人认证通过的CertifyId。
            // "certifyId" => "<0bfa7c493f850e5178b9f8613634c9xx>",
            // "mobile" => "<xxx>",
            // "ip" => "<114.xxx.xxx.xxx>",
            // "callbackUrl" => "<https://www.aliyundoc.com>",
            // "callbackToken" => "<xxxxxxx>"
        ]);
        // 推荐，支持服务路由。
        $response = self::initFaceVerifyAutoRoute($request);
        // 不支持服务自动路由。
        // $response = self::initFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", $request)
;
        var_dump($response->requestId);
        var_dump($response->code);
        var_dump($response->message);
        var_dump($response->resultObject->certifyId);
    }
    /**
     * @param InitFaceVerifyRequest $request
     * @return InitFaceVerifyResponse
     */
    public static function initFaceVerifyAutoRoute(InitFaceVerifyRequest $request){
        $endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        foreach($endpoints as $endpoint){
            try {
                $response = self::initFaceVerify($endpoint, $request);
                if (Utils::equalString("500", $response->code)) {
                    continue;
                }
                return $response;
            }
            catch (Exception $err) {
                if (!($err instanceof TeaError)) {
                    $err = new TeaError([], $err->getMessage(), $err->getCode(), $err);
                }
                // 网络异常，切换到下个区域调用。
                if (Utils::equalString("ServiceUnavailable", $err->code)) {
                    continue;
                }
            }
        }
        return null;
```

```
    }
    /**
     * @param string $endpoint
     * @param InitFaceVerifyRequest $request
     * @return InitFaceVerifyResponse
     */
    public static function initFaceVerify($endpoint, $request){
        $client = self::createClient($endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        $runtime = new RuntimeOptions([]);
        $runtime->readTimeout = 10000;
        $runtime->connectTimeout = 10000;
        return $client->initFaceVerify($request, $runtime);
    }
}
$path = __DIR__ . \DIRECTORY_SEPARATOR . \DIRECTORY_SEPARATOR . 'vendor' . \DIRECTORY_SEPAR
ATOR . 'autoload.php';
if (file_exists($path)) {
    require_once $path;
}
InitFaceVerify::main();
}
```

DescribeFaceVerify接口调用示例

```
<?php
namespace AlibabaCloud\SDK\Sample {
use AlibabaCloud\SDK\Cloudauth\V20190307\Cloudauth;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\DescribeFaceVerifyRequest;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\DescribeFaceVerifyResponse;
use AlibabaCloud\Tea\Utils\Utils;
use \Exception;
use AlibabaCloud\Tea\Exception\TeaError;
use AlibabaCloud\Tea\Rpc\Rpc\Config;
use AlibabaCloud\Tea\Utils\Utils\RuntimeOptions;
class DescribeFaceVerify {
    /**
     * @param string $endpoint
     * @return Cloudauth
     */
    public static function createClient($endpoint){
        $config = new Config([
            // 您的AccessKey ID。
            "accessKeyId" => "<您的AccessKey ID>",
            // 您的AccessKey Secret。
            "accessKeySecret" => "<您的AccessKey Secret>",
            // 设置HTTP代理。
            // "httpProxy" => "<http://xx.xx.xx.xx:xxxx>",
            // 设置HTTPS代理。
            // "httpsProxy" => "<https://xx.xx.xx.xx:xxxx>",
            "endpoint" => $endpoint
        ]);
        return new Cloudauth($config);
    }
```

```
    /**
     * @return void
     */
    public static function main(){
        $request = new DescribeFaceVerifyRequest();
        // 场景ID。
        $request->sceneId=<xxx>;
        // 认证ID。
        $request->certifyId="<xxx>";
        // 推荐，支持服务路由。
        $response = self::describeFaceVerifyAutoRoute($request);
        // 不支持服务自动路由。
        // $response = self::describeFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", $requ
est);
        var_dump($response->requestId);
        var_dump($response->code);
        var_dump($response->message);
        var_dump($response->resultObject->passed);
        var_dump($response->resultObject->subCode);
        var_dump($response->resultObject->materialInfo);
        var_dump($response->resultObject->identityInfo);
        var_dump($response->resultObject->deviceToken);
    }
    /**
     * @param DescribeFaceVerifyRequest $request
     * @return DescribeFaceVerifyResponse
     */
    public static function describeFaceVerifyAutoRoute($request){
        $endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        foreach($endpoints as $endpoint){
            try {
                $response = self::describeFaceVerify($endpoint, $request);
                if (Utils::equalString("500", $response->code)) {
                    continue;
                }
                return $response;
            }
            catch (Exception $err) {
                if (!($err instanceof TeaError)) {
                    $err = new TeaError([], $err->getMessage(), $err->getCode(), $err);
                }
                // 网络异常，切换到下个区域调用。
                if (Utils::equalString("ServiceUnavailable", $err->code)) {
                    continue;
                }
            }
        }
        return null;
    }
    /**
     * @param string $endpoint
```

```
     * @param DescribeFaceVerifyRequest $request
     * @return DescribeFaceVerifyResponse
     */
    public static function describeFaceVerify($endpoint, $request){
        $client = self::createClient($endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        $runtime = new RuntimeOptions([]);
        $runtime->readTimeout = 10000;
        $runtime->connectTimeout = 10000;
        return $client->describeFaceVerify($request, $runtime);
    }
}
$path = __DIR__ . \DIRECTORY_SEPARATOR . \DIRECTORY_SEPARATOR . 'vendor' . \DIRECTORY_SEPAR
ATOR . 'autoload.php';
if (file_exists($path)) {
    require_once $path;
}
DescribeFaceVerify::main();
}
```

3.2.5. Python SDK调用示例

本文介绍了金融级活体人脸验证方案的Python SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的Python版本在3.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

引入依赖包

```
pip install alibabacloud_tea_openapi==0.2.5
```

安装SDK

关于Python SDK源代码，请参见Python SDK包。

```
pip install alibabacloud_cloudauth20190307==1.0.7
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
from alibabacloud_cloudauth20190307.client import Client as CloudauthClient
from alibabacloud_tea_openapi import models as open_api_models
from alibabacloud_cloudauth20190307 import models as cloudauth_models
from alibabacloud_tea_util.client import Client as UtilClient
from alibabacloud_tea_util import models as util_models
class InitFaceVerify:
    def __init__(self):
        pass
    @staticmethod
    def create_client(
            endpoint: str,
    ) -> CloudauthClient:
        config = open_api_models.Config(
            # 您的AccessKey ID。
            access_key_id='您的AccessKey ID',
            # 您的AccessKey Secret。
            access_key_secret='您的AccessKey Secret',
            # 设置HTTP代理。
            # http_proxy='http://xx.xx.xx.xx:xxxx',
            # 设置HTTPS代理。
            # https_proxy='https://username:password@xxx.xxx.xxx.xxx:9999',
            endpoint=endpoint
        )
        return CloudauthClient(config)
    @staticmethod
    def main() -> None:
        request = cloudauth_models.InitFaceVerifyRequest(
            # 请输入场景ID。
            scene_id=xxx,
            outer_order_no='商户请求的唯一标识',
            # 要接入的认证方案。
            product_code='PV_FV',
            model='LIVENESS',
            # 固定值IDENTITY_CARD。
            cert_type='IDENTITY_CARD',
            meta_info='{"zimVer":"3.0.0","appVersion": "1","bioMetaInfo": "4.1.0:11501568,0
","appName":"com.aliyun.antcloudauth","deviceType": "ios","osVersion": "iOS 10.3.2","apdidT
oken": "","deviceModel": "iPhone9,1"}',
            # cert_name='xxx',
            # cert_no='xxx',
            # mobile='130xxxxxxxx',
            # ip='114.xxx.xxx.xxx',
            user_id='12345xxxx',
            # callback_url='https://www.aliyundoc.com',
            # callback_token='xxxxxxx',
            # 请在以下四种方式中选择其一。
            # 1.照片Base64编码。
            # 2.照片OSS Bucket名和文件名。
            # 3.照片OSS的URL地址。
            # 4.已经实人认证通过的CertifyId。
            # 方式一：照片Base64。
            # face_contrast_picture='人脸照片Base64编码'
            # 方式二：照片OSS Bucket名和文件名。
            # oss_bucket_name='cn-shanghai-aliyun-cloudauth-xxxxx',
```

```
            # oss_object_name='verify/xxxxx/xxxxxx.jpeg',
            # 方式三：照片OSS的URL地址，公网可访问。
              face_contrast_picture_url='https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn
-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg',
            # 方式四：已经实人认证通过的CertifyId。
            # certify_id='0bfa7c493f850e5178b9f8613634c9xx'
        )
        response = InitFaceVerify.init_face_verify_auto_route(request)
        # 不支持服务自动路由。
        # response = InitFaceVerify.init_face_verify('cloudauth.cn-shanghai.aliyuncs.com',
request)
        print(response.request_id)
        print(response.code)
        print(response.message)
        print(response.result_object.certify_id)
    @staticmethod
    def init_face_verify_auto_route(
            request: cloudauth_models.InitFaceVerifyRequest,
    ) -> cloudauth_models.InitFaceVerifyResponse:
        endpoints = [
            'cloudauth.cn-shanghai.aliyuncs.com',
            'cloudauth.cn-beijing.aliyuncs.com'
        ]
        for endpoint in endpoints:
            try:
                response = InitFaceVerify.init_face_verify(endpoint, request)
                if UtilClient.equal_string('500', response.code):
                    continue
                return response
            except Exception as err:
                # 网络异常，切换到下个区域调用。
                continue
        return None
    @staticmethod
    def init_face_verify(
            endpoint: str,
            request: cloudauth_models.InitFaceVerifyRequest,
    ) -> cloudauth_models.InitFaceVerifyResponse:
        client = InitFaceVerify.create_client(endpoint)
        # 创建RuntimeObject实例并设置运行参数。
        runtime = util_models.RuntimeOptions()
        runtime.read_timeout = 10000
        runtime.connect_timeout = 10000
        return client.init_face_verify(request, runtime)
if __name__ == '__main__':
    InitFaceVerify.main()
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
from alibabacloud_cloudauth20190307.client import Client as CloudauthClient
from alibabacloud_tea_openapi import models as open_api_models
from alibabacloud_cloudauth20190307 import models as cloudauth_models
from alibabacloud_tea_util.client import Client as UtilClient
```

```
from alibabacloud_tea_util.client import client as utilclient
from alibabacloud_tea_util import models as util_models
class DescribeFaceVerify:
    def __init__(self):
        pass
    @staticmethod
    def create_client(
        endpoint: str,
    ) -> CloudauthClient:
        config = open_api_models.Config(
            # 您的AccessKey ID。
            access_key_id='您的AccessKey ID',
            # 您的AccessKey Secret。
            access_key_secret='您的AccessKey Secret',
            # 设置HTTP代理。
            # http_proxy='http://xx.xx.xx.xx:xxxx',
            # 设置HTTPS代理。
            # https_proxy='https://xx.xx.xx.xx:xxxx',
            endpoint=endpoint
        )
        return CloudauthClient(config)
    @staticmethod
    def main() -> None:
        request = cloudauth_models.DescribeFaceVerifyRequest(
            # 请输入场景ID。
            scene_id=xxx,
            certify_id='xxx'
        )
        response = DescribeFaceVerify.describe_face_verify_auto_route(request)
        # 不支持服务自动路由。
        # response = DescribeFaceVerify.describe_face_verify('cloudauth.cn-shanghai.aliyunc
s.com', request)
        print(response.request_id)
        print(response.code)
        print(response.message)
        print(response.result_object.passed)
        print(response.result_object.sub_code)
        print(response.result_object.material_info)
    @staticmethod
    def describe_face_verify_auto_route(
        request: cloudauth_models.DescribeFaceVerifyRequest,
    ) -> cloudauth_models.DescribeFaceVerifyResponse:
        endpoints = [
            'cloudauth.cn-shanghai.aliyuncs.com',
            'cloudauth.cn-beijing.aliyuncs.com'
        ]
        for endpoint in endpoints:
            try:
                response = DescribeFaceVerify.describe_face_verify(endpoint, request)
                if UtilClient.equal_string('500', response.code):
                    continue
                return response
            except Exception as err:
                    continue
        return None
```

```
    @staticmethod
    def describe_face_verify(
        endpoint: str,
        request: cloudauth_models.DescribeFaceVerifyRequest,
    ) -> cloudauth_models.DescribeFaceVerifyResponse:
        client = DescribeFaceVerify.create_client(endpoint)
        # 创建RuntimeObject实例并设置运行参数。
        runtime = util_models.RuntimeOptions()
        runtime.read_timeout = 10000
        runtime.connect_timeout = 10000
        return client.describe_face_verify(request, runtime)
if __name__ == '__main__':
    DescribeFaceVerify.main()
```

3.2.6. Go SDK调用示例

本文介绍了金融级活体人脸验证方案的Go SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 仅支持Go 1.12.0及以上版本。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

引入依赖包

```
go get github.com/alibabacloud-go/tea-rpc
go get github.com/alibabacloud-go/tea-utils
go get github.com/alibabacloud-go/tea
```

安装SDK

关于Go SDK源代码，请参见Go SDK包。

通过Go mod来管理项目依赖：

```
go get github.com/alibabacloud-go/cloudauth-20190307@vv1.0.7
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
package main
import (
```

```
  "fmt"
  cloudauth "github.com/alibabacloud-go/cloudauth-20190307/client"
  rpc "github.com/alibabacloud-go/tea-rpc/client"
  util "github.com/alibabacloud-go/tea-utils/service"
  "github.com/alibabacloud-go/tea/tea"
)
func CreateClient (endpoint *string) (_result *cloudauth.Client, _err error) {
  config := &rpc.Config{
    // 您的AccessKey ID。
    AccessKeyId: tea.String("您的AccessKey ID"),
    // 您的AccessKey Secret。
    AccessKeySecret: tea.String("您的AccessKey Secret"),
    // 设置HTTP代理。
    // HttpProxy: tea.String("http://xx.xx.xx.xx:xxxx"),
    // 设置HTTPS代理。
    // HttpsProxy: tea.String("https://username:password@xxx.xxx.xxx.xxx:9999"),
    Endpoint: endpoint,
  }
  _result = &cloudauth.Client{}
  _result, _err = cloudauth.NewClient(config)
  return _result, _err
}
func _main () (_err error) {
  request := &cloudauth.InitFaceVerifyRequest{
    // 固定值。
    ProductCode: tea.String("PV_FV"),
    // 请输入场景ID。
    SceneId: tea.Int64(xxxx),
    OuterOrderNo: tea.String("商户请求的唯一标识"),
    Model: tea.String("LIVENESS"),
    // 固定值IDENTITY_CARD。
    CertType: tea.String("IDENTITY_CARD"),
    // CertName: tea.String("xxx"),
    // CertNo: tea.String("xxx"),
    // Ip: tea.String("用户IP地址"),
    UserId: tea.String("用户ID"),
    // Mobile: tea.String("130xxxxxxxx"),
    // CallbackUrl: tea.String("https://www.aliyundoc.com"),
    // CallbackToken: tea.String("xxxxxxx"),
    MetaInfo: tea.String("{\"zimVer\":\"3.0.0\",\"appVersion\": \"1\",\"bioMetaInfo\": \"4.
1.0:11501568,0\",\"appName\": \"com.aliyun.antcloudauth\",\"deviceType\": \"ios\",\"osVersi
on\": \"iOS 10.3.2\",\"apdidToken\": \"\",\"deviceModel\": \"iPhone9,1\"}"),
    // 请在以下四种方式中选择其一。
    // 1.照片Base64编码。
    // 2.照片OSS Bucket名和文件名。
    // 3.照片OSS的URL地址。
    // 4.已经实人认证通过的CertifyId。
    // 方式一：照片Base64。
    // FaceContrastPicture: tea.String("人脸照片Base64编码"),
    // 方式二：照片OSS Bucket名和文件名。
    // OssBucketName: tea.String("cn-shanghai-aliyun-cloudauth-xxxxx"),
    // OssObjectName: tea.String("verify/xxxxx/xxxxxx.jpeg"),
    // 方式三：照片OSS的URL地址，公网可访问。
    FaceContrastPictureUrl: tea.String("https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-
shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg")
```

```
shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg"),
    // 方式四：已经实人认证通过的CertifyId。
    // CertifyId: tea.String("0bfa7c493f850e5178b9f8613634c9xx"),
  }
  response, _err := InitFaceVerifyAutoRoute(request)
  // 不支持服务自动路由。
  // response, _err := InitFaceVerify(tea.String("cloudauth.cn-shanghai.aliyuncs.com"), req
uest)
  if _err != nil {
    return _err
  }
  fmt.Println(*response.RequestId)
  fmt.Println(*response.Code)
  fmt.Println(*response.Message)
  fmt.Println(*response.RequestId)
  fmt.Println(*response.ResultObject.CertifyId)
  return _err
}
func InitFaceVerifyAutoRoute (request *cloudauth.InitFaceVerifyRequest) (_result *cloudauth
.InitFaceVerifyResponse, _err error) {
  endpoints := []*string{tea.String("cloudauth.cn-shanghai.aliyuncs.com"), tea.String("clou
dauth.cn-beijing.aliyuncs.com")}
  var lastResponse *cloudauth.InitFaceVerifyResponse
  for _, endpoint := range endpoints {
    response, _err := InitFaceVerify(endpoint, request)
    if _err != nil {
      var err = &tea.SDKError{}
      if _t, ok := _err.(*tea.SDKError); ok {
        err = _t
        // 网络异常，切换到下个区域调用。
        if tea.BoolValue(util.EqualString(tea.String("ServiceUnavailable"), err.Code)) {
          continue
        }
      }
      return _result, _err
    }
    if tea.BoolValue(util.EqualString(tea.String("500"), response.Code)) {
      continue
    }
    _result = response
    return _result, _err
  }
  _result = lastResponse
  return _result, _err
}
func InitFaceVerify (endpoint *string, request *cloudauth.InitFaceVerifyRequest) (_result *
cloudauth.InitFaceVerifyResponse, _err error) {
  client, _err := CreateClient(endpoint)
  if _err != nil {
    return _result, _err
  }
  // 创建RuntimeObject实例并设置运行参数。
  runtime := &util.RuntimeOptions{}
  runtime.ReadTimeout = tea.Int(10000)
  runtime.ConnectTimeout = tea.Int(10000)
```

```
  _result = &cloudauth.InitFaceVerifyResponse{}
  _body, _err := client.InitFaceVerify(request, runtime)
  if _err != nil {
    return _result, _err
  }
  _result = _body
  return _result, _err
}
func main() {
  err := _main()
  if err != nil {
    panic(err)
  }
}
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
package main
import (
  "fmt"
  cloudauth "github.com/alibabacloud-go/cloudauth-20190307/client"
  rpc "github.com/alibabacloud-go/tea-rpc/client"
  util "github.com/alibabacloud-go/tea-utils/service"
  "github.com/alibabacloud-go/tea/tea"
)
func CreateClient(endpoint *string) (_result *cloudauth.Client, _err error) {
  config := &rpc.Config{
    // 您的AccessKey ID。
    AccessKeyId: tea.String("您的AccessKey ID"),
    // 您的AccessKey Secret。
    AccessKeySecret: tea.String("您的AccessKey Secret"),
    Endpoint:        endpoint,
    // 设置HTTP代理。
    // HttpProxy: tea.String("http://xx.xx.xx.xx:xxxx"),
    // 设置HTTPS代理。
    // HttpsProxy: tea.String("https://xx.xx.xx.xx:xxxx"),
  }
  _result = &cloudauth.Client{}
  _result, _err = cloudauth.NewClient(config)
  return _result, _err
}
func _main() (_err error) {
  request := &cloudauth.DescribeFaceVerifyRequest{
    // 请输入场景ID。
    SceneId:   tea.Int64(xxx),
    CertifyId: tea.String("91707dc296d469ad38e4c5efa6a0xxxx"),
  }
  response, _err := DescribeFaceVerifyAutoRoute(request)
  // 不支持服务自动路由。
  // response, _err := DescribeFaceVerify(tea.String("cloudauth.cn-shanghai.aliyuncs.com"),
request)
  if _err != nil {
```

```
    return _err
  }
  if response != nil {
    fmt.Println(*response.RequestId)
    fmt.Println(*response.Code)
    fmt.Println(*response.Message)
    fmt.Println(*response.RequestId)
    fmt.Println(*response.ResultObject.Passed)
    fmt.Println(*response.ResultObject.SubCode)
    fmt.Println(*response.ResultObject.MaterialInfo)
    fmt.Println(*response.ResultObject.IdentityInfo)
    fmt.Println(*response.ResultObject.DeviceToken)
  }
  return _err
}
func DescribeFaceVerifyAutoRoute(request *cloudauth.DescribeFaceVerifyRequest) (_result *cl
oudauth.DescribeFaceVerifyResponse, _err error) {
  endpoints := []*string{tea.String("cloudauth.cn-shanghai.aliyuncs.com"), tea.String("clou
dauth.cn-beijing.aliyuncs.com")}
  var lastResponse *cloudauth.DescribeFaceVerifyResponse
  for _, endpoint := range endpoints {
    response, _err := DescribeFaceVerify(endpoint, request)
    if _err != nil {
      var err = &tea.SDKError{}
      if _t, ok := _err.(*tea.SDKError); ok {
        err = _t
        // 网络异常，切换到下个区域调用。
        if tea.BoolValue(util.EqualString(tea.String("ServiceUnavailable"), err.Code)) {
          continue
        }
      }
      return _result, _err
    }
    if tea.BoolValue(util.EqualString(tea.String("500"), response.Code)) {
      continue
    }
    _result = response
    return _result, _err
  }
  _result = lastResponse
  return _result, _err
}
func DescribeFaceVerify(endpoint *string, request *cloudauth.DescribeFaceVerifyRequest) (_r
esult *cloudauth.DescribeFaceVerifyResponse, _err error) {
  client, _err := CreateClient(endpoint)
  if _err != nil {
    return _result, _err
  }
  // 创建RuntimeObject实例并设置运行参数。
  runtime := &util.RuntimeOptions{}
  runtime.ReadTimeout = tea.Int(10000)
  runtime.ConnectTimeout = tea.Int(10000)
  _body, _err := client.DescribeFaceVerify(request, runtime)
  if _err != nil {
```

```
    return _result, _err
  }
  _result = _body
  return _result, _err
}
func main() {
  err := _main()
  if err != nil {
    panic(err)
  }
}
```

3.2.7. .NET SDK调用示例

本文介绍了金融级活体人脸验证方案的.NET SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的.NET Framework版本在4.5.0版本及以上或者.NET Standard版本在2.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于.NET SDK源代码，请参见.NET SDK包。

SDK有以下两种安装方式，请选择合适的方式安装：

- 通过.NET命令行安装。

```
dotnet add package AlibabaCloud.SDK.Cloudauth20190307 --version 1.0.5
```

- 使用包管理安装。

```
Install-Package AlibabaCloud.SDK.Cloudauth20190307 -Version 1.0.5
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
using System;
using System.Collections.Generic;
using Tea;
namespace InitFaceVerify
{
    public class Program
    {
```

```
        public static AlibabaCloud.SDK.Cloudauth20190307.Client CreateClient(string endpoin
t)
        {
            AlibabaCloud.RPCClient.Models.Config config = new AlibabaCloud.RPCClient.Models
.Config
            {
                // 您的AccessKey ID。
                AccessKeyId = "您的AccessKey ID",
                // 您的AccessKey Secret。
                AccessKeySecret = "您的AccessKey Secret",
                // 设置HTTP代理。
                // HttpProxy = "http://xx.xx.xx.xx:xxxx",
                // 设置HTTPS代理。
                // HttpsProxy = "https://username:password@xxx.xxx.xxx.xxx:9999",
                Endpoint = endpoint,
            };
            return new AlibabaCloud.SDK.Cloudauth20190307.Client(config);
        }
        public static void Main(string[] args)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest request = new A
libabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest
            {
                // 固定值。
                ProductCode = "PV_FV",
                // 请输入场景ID。
                SceneId = xxx,
                // 设置商户请求的唯一标识。
                OuterOrderNo = "xxxxx",
                // 模式。
                Model = "LIVENESS",
                // 固定值IDENTITY_CARD。
                CertType = "IDENTITY_CARD",
                //CertName = "xxx",
                //CertNo = "xxx",
                MetaInfo = "{\"zimVer\":\"3.0.0\",\"appVersion\": \"1\",\"bioMetaInfo\": \"
4.1.0:11501568,0\",\"appName\": \"com.aliyun.antcloudauth\",\"deviceType\": \"ios\",\"osVer
sion\": \"iOS 10.3.2\",\"apdidToken\": \"\",\"deviceModel\": \"iPhone9,1\"}",
                //Mobile = "130xxxxxxxx",
                //Ip = "114.xxx.xxx.xxx",
                UserId = "用户ID",
                //CallbackUrl = "https://www.aliyundoc.com",
                //CallbackToken = "xxxxxxx",
                // 请在以下四种方式中选择其一。
                // 1.照片Base64编码。
                // 2.照片OSS Bucket名和文件名。
                // 3.照片OSS的URL地址。
                // 4.已经实人认证通过的CertifyId。
                // 方式一：照片Base64。
                //FaceContrastPicture = "人脸照片Base64编码",
                // 方式二：照片OSS Bucket名和文件名。
                // OssBucketName = "cn-shanghai-aliyun-cloudauth-xxxxx",
                // OssObjectName = "verify/xxxxx/xxxxxx.jpeg",
                // 方式三：照片OSS的URL地址，公网可访问。
```

```
                FaceContrastPictureUrl = "https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-c
n-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg",
                // 方式四：已经实人认证通过的CertifyId。
                // CertifyId = "0bfa7c493f850e5178b9f8613634c9xx",
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse response = Ini
tFaceVerifyAutoRoute(request);
            // 不支持服务自动路由。
            // AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse response =
InitFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", request);
            Console.WriteLine(response.RequestId);
            Console.WriteLine(response.Code);
            Console.WriteLine(response.Message);
            Console.WriteLine(response.ResultObject.CertifyId);
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse Init
FaceVerifyAutoRoute(AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest request
)
        {
            List<string> endpoints = new List<string>
            {
                "cloudauth.cn-shanghai.aliyuncs.com",
                "cloudauth.cn-beijing.aliyuncs.com"
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse lastResponse =
null;
            foreach (var endpoint in endpoints)
            {
                try
                {
                    AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse respon
se = InitFaceVerify(endpoint, request);
                    if (AlibabaCloud.TeaUtil.Common.EqualString("500", response.Code))
                    {
                        continue;
                    }
                    return response;
                }
                catch (TeaException err)
                {
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
                catch (Exception _err)
                {
                    TeaException err = new TeaException(new Dictionary<string, object>
                    {
                        { "message", _err.Message }
                    });
                    // 网络异常，切换到下个区域调用。
```

```
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
            }
            return lastResponse;
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse Init
FaceVerify(string endpoint, AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest
request)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Client client = CreateClient(endpoint);
            // 创建RuntimeObject实例并设置运行参数。
            AlibabaCloud.TeaUtil.Models.RuntimeOptions runtime = new AlibabaCloud.TeaUtil.M
odels.RuntimeOptions();
            runtime.ReadTimeout = 10000;
            runtime.ConnectTimeout = 10000;
            return client.InitFaceVerify(request, runtime);
        }
    }
}
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
using System;
using System.Collections.Generic;
using Tea;
namespace DescribeFaceVerify
{
    public class Program
    {
        public static AlibabaCloud.SDK.Cloudauth20190307.Client CreateClient(string endpoin
t)
        {
            AlibabaCloud.RPCClient.Models.Config config = new AlibabaCloud.RPCClient.Models
.Config
            {
                // 您的AccessKey ID。
                AccessKeyId = "您的AccessKey ID",
                // 您的AccessKey Secret。
                AccessKeySecret = "您的AccessKey Secret",
                // 设置HTTP代理。
                // HttpProxy = "http://xx.xx.xx.xx:xxxx",
                // 设置HTTPS代理。
                // HttpsProxy = "https://xx.xx.xx.xx:xxxx",
                Endpoint = endpoint,
            };
            return new AlibabaCloud.SDK.Cloudauth20190307.Client(config);
        }
        public static void Main(string[] args)
        {
```

```
        {
            AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyRequest request = n
ew AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyRequest
            {
                // 请输入场景ID。
                SceneId = xxx,
                CertifyId = "91707dc296d469ad38e4c5efa6a0xxxx",
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse response =
DescribeFaceVerifyAutoRoute(request);
            // 不支持服务自动路由。
            // AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse respons
e = DescribeFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", request);
            Console.WriteLine(response.RequestId);
            Console.WriteLine(response.Code);
            Console.WriteLine(response.Message);
            Console.WriteLine(response.ResultObject.Passed);
            Console.WriteLine(response.ResultObject.SubCode);
            Console.WriteLine(response.ResultObject.MaterialInfo);
            Console.WriteLine(response.ResultObject.IdentityInfo);
            Console.WriteLine(response.ResultObject.DeviceToken);
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse
DescribeFaceVerifyAutoRoute(AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyReq
uest request)
        {
            List<string> endpoints = new List<string>
            {
                "cloudauth.cn-shanghai.aliyuncs.com",
                "cloudauth.cn-beijing.aliyuncs.com"
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse lastRespon
se = null;
            foreach (var endpoint in endpoints)
            {
                try
                {
                    AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse re
sponse = DescribeFaceVerify(endpoint, request);
                    if (AlibabaCloud.TeaUtil.Common.EqualString("500", response.Code))
                    {
                        continue;
                    }
                    return response;
                }
                catch (TeaException err)
                {
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
                catch (Exception  err)
```

```
                {
                    TeaException err = new TeaException(new Dictionary<string, object>
                    {
                        { "message", _err.Message }
                    });
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
            }
            return lastResponse;
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse
DescribeFaceVerify(string endpoint, AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceV
erifyRequest request)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Client client = CreateClient(endpoint);
            // 创建RuntimeObject实例并设置运行参数。
            AlibabaCloud.TeaUtil.Models.RuntimeOptions runtime = new AlibabaCloud.TeaUtil.M
odels.RuntimeOptions();
            runtime.ReadTimeout = 10000;
            runtime.ConnectTimeout = 10000;
            return client.DescribeFaceVerify(request, runtime);
        }
    }
}
```

3.2.8. Node.js SDK调用示例

本文介绍了金融级活体人脸验证方案的Node.js SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的Node.js版本在8.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于Node.js SDK源代码，请参见Node.js SDK。

建议您使用npm管理依赖包：

```
npm install @alicloud/tea-util
npm install @alicloud/rpc-client
npm install @alicloud/cloudauth20190307@1.0.7
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
const Config = require("@alicloud/rpc-client");
const CloudAuth = require("@alicloud/cloudauth20190307");
const CloudAuthClient = CloudAuth.default;
const {RuntimeOptions} = require("@alicloud/tea-util");
     function createClient(endpoint) {
        // 通过以下代码创建Config实例并初始化。
        const config = new Config.Config({
            // 您的AccessKey ID。
            accessKeyId: "您的AccessKey ID",
            // 您的AccessKey Secret。
            accessKeySecret: "您的AccessKey Secret",
            // 设置Endpoint。
            endpoint: endpoint,
            // 设置HTTP代理。
            // httpProxy: "http://xxx.xxx.xxx.xxx:9999",
            // 设置HTTPS代理。
            // httpsProxy: "https://username:password@xxx.xxx.xxx.xxx:9999",
        });
        return new CloudAuthClient(config)
    }
    async function initFaceVerify(endpoint, request) {
        const client = createClient(endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        const runtime = new RuntimeOptions([]);
        runtime.readTimeout=10000;
        runtime.connectTimeout=10000;
        return await client.initFaceVerify(request, runtime);
    }
    async function initFaceVerifyAutoRoute(request) {
        const endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        for (const endpoint of endpoints) {
            try {
                const response = await initFaceVerify(endpoint, request);
                if (response.code === '500') {
                    continue;
                }
                return response;
            } catch (err) {
                continue;
            }
        }
```

```
        return null;
    }
    async function main(){
        // 通过以下代码创建API请求并设置参数。
        const request = new CloudAuth.InitFaceVerifyRequest({
        // 请输入场景ID。
        sceneId: xxx,
        // 设置商户请求的唯一标识。
        outerOrderNo: "xxx",
        // 要接入的认证方案。
        productCode: "PV_FV",
        // 模式。
        model: "LIVENESS",
        // 固定值。
        certType: "IDENTITY_CARD",
        // certName: "张三",
        // certNo: "xxx",
        metaInfo: "{\"zimVer\":\"3.0.0\",\"appVersion\": \"1\",\"bioMetaInfo\": \"4.1.0:115
01568,0\",\n" +
            "        \"appName\": \"com.aliyun.antcloudauth\",\"deviceType\": \"ios\",\"osV
ersion\": \"iOS 10.3.2\",\"apdidToken\": \"\",\"deviceModel\": \"iPhone9,1\"}",
        // mobile: "130xxxxxxxx",
        // 请在以下四种方式中选择其一。
        // 1.人脸图片的Base64编码。
        // 2.已授权OSS空间的Bucket名和已授权OSS空间的文件名。
        // 3.人脸图片的HTTPS或HTTP地址。
        // 4.已通过智能核身得到的CertifyId，认证时的图片作为比对图片。
        // 方式一：人脸图片的Base64编码。
        // faceContrastPicture: "/9j/4AAQSkZJRgABAQAASxxxxxxx",
        // 方式二：已授权OSS空间的Bucket名和已授权OSS空间的文件名。
        // ossBucketName: "cn-shanghai-aliyun-cloudauth-xxxxx",
        // ossObjectName: "verify/xxxxx/xxxxxx.jpeg",
        //方式三：人脸图片的HTTPS或HTTP地址。
         faceContrastPictureUrl: "https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-shangh
ai.aliyuncs.com/smart/xxxxx/xxxxx.jpeg",
        // 方式四：已通过智能核身得到的CertifyId，认证时的图片作为比对图片。
        // certifyId: "0bfa7c493f850e5178b9f8613634c9xx",
        // ip: "114.xxx.xxx.xxx",
        userId: "12345xxxx",
        // callbackUrl: "https://www.aliyundoc.com",
        // callbackToken: "xxxxxxx"
        });
        // 推荐，支持服务路由。
        const response = await initFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        // const response = await initFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", requ
est);
        console.log(response.requestId);
        console.log(response.code);
        console.log(response.message);
        console.log(response.resultObject.certifyId);
        }
main().then(function (response) {});
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
const Config = require("@alicloud/rpc-client");
const CloudAuth = require("@alicloud/cloudauth20190307");
const CloudAuthClient = CloudAuth.default;
const {RuntimeOptions} = require("@alicloud/tea-util");
     function createClient(endpoint) {
        // 通过以下代码创建Config实例并初始化。
        const config = new Config.Config({
            // 您的AccessKey ID。
            accessKeyId: "您的AccessKey ID",
            // 您的AccessKey Secret。
            accessKeySecret: "您的AccessKey Secret",
            // 设置Endpoint。
            endpoint: endpoint,
            // 设置HTTP代理。
            // httpProxy: "http://xxx.xxx.xxx.xxx:9999",
            // 设置HTTPS代理。
            // httpsProxy: "https://username:password@xxx.xxx.xxx.xxx:9999",
        });
        return new CloudAuthClient(config)
    }
    async function describeFaceVerify(endpoint, request) {
        const client = createClient(endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        const runtime = new RuntimeOptions([]);
        runtime.readTimeout=10000;
        runtime.connectTimeout=10000;
        return await client.describeFaceVerify(request, runtime);
    }
    async function describeFaceVerifyAutoRoute(request) {
        const endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        for (const endpoint of endpoints) {
            try {
                const response = await describeFaceVerify(endpoint, request);
                if (response.code === '500') {
                    continue;
                }
                return response;
            } catch (err) {
                continue;
            }
        }
        return null;
    }
    async function main(){
        // 通过以下代码创建API请求并设置参数。
        const request = new CloudAuth.DescribeFaceVerifyRequest({
            sceneId: xxx,
            certifyId: "xxx"
```

```
        });
        // 推荐，支持服务路由。
        const response = await describeFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        //  const response = await describeFaceVerify("cloudauth.cn-shanghai.aliyuncs.com",
request);
        console.log(response.requestId);
        console.log(response.code);
        console.log(response.message);
        console.log(response.resultObject.passed);
        console.log(response.resultObject.identityInfo);
        console.log(response.resultObject.materialInfo);
        console.log(response.resultObject.deviceToken);
        console.log(response.resultObject.subCode);
        }
main().then(function (response) {});
```

3.3. App（SDK）接入

3.3.1. Android接入

金融级实人认证服务提供Android客户端SDK，帮助您在业务应用（App）中实现刷脸认证功能。您可以通过应用服务端认证初始化接口，向阿里云服务器调用发起认证请求接口并获取刷脸认证的唯一标识CertifyId，然后使用CertifyId唤起刷脸认证客户端SDK。本文结合示例代码介绍Android客户端的接入流程。

使用限制

金融级实人认证服务Android客户端SDK具有以下使用限制。

- 不支持模拟器模式调试。
- 仅支持Android 4.4及以上系统版本的移动智能设备（手机或Pad）接入。

依赖配置

下载金融级实人认证Android SDK，该SDK为Android标准AAR包，下载之后在*Android Studio*中进行AAR包引用。例如下载完AAR包之后，放入工程目录的*libs*目录下，并添加以下依赖：

```
# 添加阿里云实人认证SDK依赖包。
implementation files('libs/android-aliyunface-sdk-release-xxx.aar')
implementation files('libs/android-aliyunbasicstl-sdk-release-xxx.aar')
implementation files('libs/android-aliyuncomm-sdk-release-xxx.aar')
implementation files('libs/android-aliyunocr-sdk-release-xxx.aar')
implementation files('libs/photinus-1.0.1.211130153658.aar')
implementation files('libs/Android-AliyunDevice-xxx.aar')
implementation files('libs/deviceid-release-xxx.aar')
implementation files('libs/tygerservice-xxx.aar')
implementation 'com.squareup.okhttp3:okhttp:3.11.0'
implementation 'com.squareup.okio:okio:1.14.0'
implementation 'com.alibaba:fastjson:1.2.62'
implementation 'com.aliyun.dpa:oss-android-sdk:+'
```

说明 以上代码示例中xxx表示版本号，使用过程中，请替换为SDK包中文件的实际版本号。

接口定义

```
/**
 * 刷脸认证接口。
 */
public class ZIMFacade {
    /**
     * 初始化接口，建议接入方在调用其它接口前，先调用此接口。
     *
     * @context Context
     */
    public static void install(Context context);
    /**
     * 获取环境参数接口。
     *
     * @context Context
     */
    public static String getMetaInfos(Context context);
    /**
     * identify interface for android
     * 需在UI线程上调用，不会阻塞调用线程。
     *
     * @param certifyId：刷脸认证唯一标识，请从刷脸认证服务端认证初始化接口获取。
     * @param useMsgBox：当刷脸认证过程中出现异常情况，是否使用SDK内部的弹框提示。
     *                    如果取值为true，则SDK先弹框提示，确定之后，返回错误code。
     *                    如果取值为false，则不弹提示框，直接返回错误code，客户业务应用App自行决定
如何提示。
     * @param callback：认证结果的回调接口。
     */
    public void verify(String certifyId, boolean useMsgBox,, ZIMCallback callback);
    /**
     * 开始进行实名认证，带自定义参数的版本。
     * @param certifyId：刷脸认证唯一标识，请从刷脸认证服务端认证初始化接口获取。
     * @param useMsgBox：表示是否在SDK内部使用弹框提示错误信息。
     * @param extParams：扩展参数。
     * @param callback：identify result
     */
    public void verify(final String zimId, boolean useMsgBox, HashMap<String, String> extPa
rams, ZIMCallback callback);
    /**
     * 获取设备指纹DeviceToken。
     * @return
     */
    public ZIMSession getSession();
}
```

接口调用

1. 初始化SDK。

为了提高身份核验的用户体验，并为刷脸认证准备必要的数据，Android客户端需要进行SDK初始化。初始化的代码示例如下：

```
ZIMFacade.install(context);
```

2. 获取MetaInfo数据。

   您在调用刷脸认证服务端发起认证请求时，需要传入该MetaInfo值。代码示例如下：

```
String metaInfos = ZIMFacade.getMetaInfos(context);
```

3. 开始认证。

   Android客户端接入支持接入手机端和Pad端两种场景，请您根据自身业务需要选择接入的场景。两种接入场景的示例代码如下：

   - 手机端接入场景（自定义参数设置请参考Pad端设置）

```
ZIMFacade zimFacade = ZIMFacadeBuilder.create(MainActivity.this);
zimFacade.verify(certifyId, true, new ZIMCallback() {
 @Override
 public boolean response(ZIMResponse response) {
     // TODO: 根据实人认证回调结果处理自身的业务。
     if (null != response && 1000 == response.code) {
         // 认证成功。
     } else {
         // 认证失败。
     }
     return true;
 }
}
```

   - 手机端+Pad端接入场景（支持自定义参数）

> **说明** 金融级实人认证支持设置返回活体检测的视频，默认设置为不返回。如果设置为返回，在您使用金融级实人认证SDK完成认证（不区分认证结果为F或T），并在SDK回调时和查询接口返回视频路径，请您及时处理。

```
ZIMFacade zimFacade = ZIMFacadeBuilder.create(MainActivity.this);
HashMap<String, String> extParams = new HashMap<>();
// 如需指定活体检测UI界面方向（横屏+竖屏），请指定这一项。
extParams.put(ZIMFacade.ZIM_EXT_PARAMS_KEY_SCREEN_ORIENTATION, ZIMFacade.ZIM_EXT_PARA
MS_VAL_SCREEN_LAND);
// 如需支持活体视频返回，请指定这一项，并在response.videoFilePath或在服务端查询接口获取视频本
地路径。
extParams.put(ZIMFacade.ZIM_EXT_PARAMS_KEY_USE_VIDEO, ZIMFacade.ZIM_EXT_PARAMS_VAL_US
E_VIDEO_TRUE);
// 如需设置OCR的"下一步"按钮颜色（默认可不设置），请设置此项，如红色 #FF0000。
extParams.put(ZIMFacade.ZIM_EXT_PARAMS_KEY_OCR_BOTTOM_BUTTON_COLOR, "#FF0000");
// 如需自定义活体检测页面的进度条颜色（默认可不设置），请设置此项，如红色 #FF0000。
extParams.put(ZIMFacade.ZIM_EXT_PARAMS_KEY_FACE_PROGRESS_COLOR, "#FF0000");
zimFacade.verify(certifyId, true, extParams, new ZIMCallback() {
    @Override
    public boolean response(final ZIMResponse response) {
        if (1000 == response.code) {
            showMessageBox("认证通过");
        } else {
            showMessageBox("认证失败([" + response.code + "]" + response.reason + ")")
;
        }
        return true;
    }
});
```

说明

- certifyId：刷脸认证唯一标识，可从刷脸认证服务端发起认证请求获取。
- ZIMCallback的response回调需要有返回值，默认为true。

4. 获取设备指纹。

   获取设备指纹（deviceToken），然后通过后端接口查询设备风险。代码示例如下：

```
ZIMSession deviceSession = zimFacade.getSession();
 if(deviceSession.code == 0){
            // 获取session成功，可以通过后端接口查询风险标签。
 }else{
            // 获取session失败，不能通过后端接口查询到有效风险标签。
 }
```

5. 返回刷脸认证回调结果。

   刷脸认SDK回调结果中的ZIMResponse类，该类中定义了对应的结果编码和原因，代码示例如下：

```
/**
 * 刷脸认证回调结果。
 */
public class ZIMResponse {
    /**
     * 返回结果编码。
     */
    public int code;
    /**
     * 返回结果原因信息。
     */
    public String reason;
}
```

返回结果说明

错误码	错误码文案	错误码描述
1000	刷脸成功	代表刷脸成功，该结果仅供参考，可通过服务端查询认证结果获取最终认证结果。
1001	系统错误	表示系统错误。
1003	验证中断	表示验证中断。
2002	网络错误	表示网络错误。
2003	客户端设备时间错误	表示客户端设备时间错误。
2006	刷脸失败	表示刷脸失败，如需获取更详细的失败原因，需调用服务端查询认证结果。

说明 更多错误码信息，请参见Android客户端错误码详情。

接口混淆配置

为避免接口混淆，您可以参考以下代码来保留类名（一般是在封装release包的时候保留类名）：

```
-verbose
-keep class com.aliyun.aliyunface.network.model.** {*;}
-keep class com.aliyun.aliyunface.api.ZIMCallback {*;}
-keep class com.aliyun.aliyunface.api.ZIMFacade {*;}
-keep class com.aliyun.aliyunface.api.ZIMFacadeBuilder {*;}
-keep class com.aliyun.aliyunface.api.ZIMMetaInfo {*;}
-keep class com.aliyun.aliyunface.api.ZIMResponse {*;}
-keep class com.aliyun.aliyunface.api.ZIMSession {*;}
-keep class com.aliyun.aliyunface.config.**{*;}
-keep class com.aliyun.aliyunface.log.RecordBase {*;}
-keep class com.aliyun.aliyunface.ui.ToygerWebView {*;}
-keep class com.alipay.zoloz.toyger.**{*;}
-keep class com.alipay.zoloz.image.** {*;}
-keep class com.alipay.android.** {*;}
-keep class net.security.device.api.** {*;}
-keep class com.alipay.deviceid.** { *; }
-keep class com.alibaba.fastjson.** {*;}
-keep class com.alibaba.sdk.android.oss.** { *; }
-dontwarn okio.**
-dontwarn org.apache.commons.codec.binary.**
-keepclassmembers,allowobfuscation class * {
     @com.alibaba.fastjson.annotation.JSONField <fields>;
}
```

权限说明

金融级实人认证为您提供安全的全链路身份核验服务。为了达到更好的设备风险防范效果，建议您在使用业务应用App调用刷脸SDK之前，提前向您的用户获取如下权限。

```
// 授权获取手机状态的权限。
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
// 授权获取接入网络的权限。开发人员需注意，6.0以上系统属于动态权限。
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Demo代码包

您可以下载Android Demo代码包（aliyunfaceDemo.zip）进行体验。

SDK包大小裁剪说明

如果您不需要使用OCR功能，可以在SDK包中删除*android-aliyunocr-sdk-release-xxxx_version.aar*，并重新编译。

3.3.2. iOS接入

刷脸认证提供iOS客户端SDK，帮助您在您的业务应用（App）中实现刷脸认证功能。您可通过刷脸认证的服务端认证初始化接口，获取刷脸认证唯一标识CertifyId，并使用CertifyId唤起刷脸认证客户端SDK。本文将结合示例代码进行iOS客户端接入的详细说明。

前提条件

- 刷脸认证应用必须运行在iOS 9.0及以上版本的系统上。

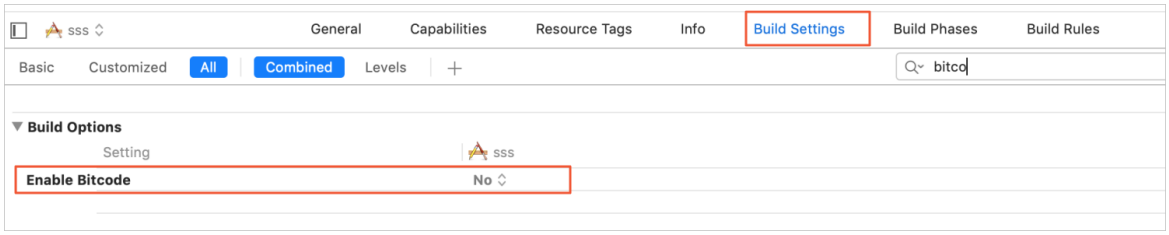
- 您的业务应用必须采用Objective C++或Swift集成金融级实人认证服务。
- 您的业务应用中未集成支付宝mPaaS SDK。

> **说明** iOS接入不支持模拟器模式调试。

配置开发环境

1. 在*info.plist*中配置摄像头权限请求。

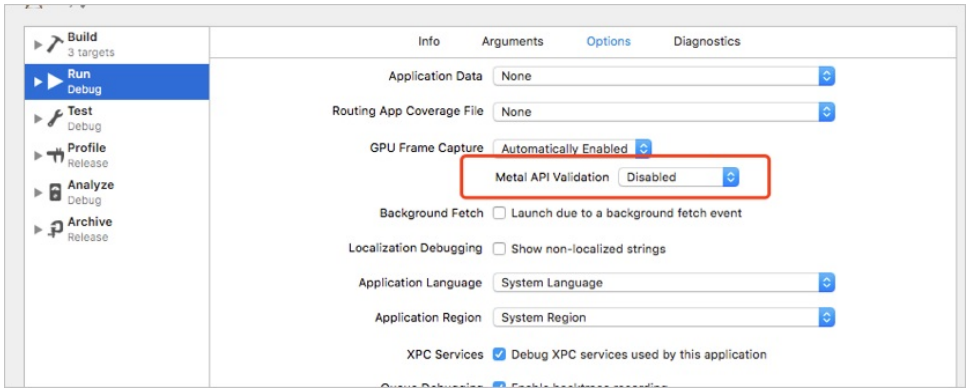
2. 在Xcode的编译设置中关闭Enable Bitcode选项。



3. 在Xcode编译设置的Linking > Other Linker Flags中，添加设置-ObjC -framework "BioAuthAPI" -lxml2。

   如果您的工程已设置了-force_load选项，则需要加入-force_load <framework path>/AliyunOSSiOS。

4. 在Xcode调试设置的Edit Scheme > Run > Options中禁用Metal API Validation。



配置依赖

1. 下载AliyunFace iOS SDK。

   该SDK为framework包。您需要在Xcode添加Link Binary With Libraries、SDK中的15个包和额外系统库依赖。

2. 解压下载的SDK包，并在解压的文件夹中执行以下命令提取所有的framework到Products目录。

```
for i in $(ls *.tgz);do tar xvf $i;done
```

- SDK中的包：

```
APPSecuritySDK
MPRemoteLogging
AliyunOSSiOS
BioAuthEngine
ZolozUtility
AliyunIdentityManager
APBToygerFacade
BioAuthAPI
ToygerService
ZolozSensorServices
ZolozOpenPlatformBuild
deviceiOS
ZolozMobileRPC
ZolozIdentityManager
OCRDetectSDKForTech
ToygerNative
```

- 系统库依赖：

```
CoreGraphics.framework
Accelerate.framework
SystemConfiguration.framework
AssetsLibrary.framework
CoreTelephony.framework
QuartzCore.framework
CoreFoundation.framework
CoreLocation.framework
ImageIO.framework
CoreMedia.framework
CoreMotion.framework
AVFoundation.framework
WebKit.framework
libresolv.tbd
libz.tbd
libc++.tbd
libc++.1.tbd
libc++abi.tbd
AudioToolbox.framework
CFNetwork.framework
MobileCoreServices.framework
libz.1.2.8.tbd
AdSupport.framework
```

拷贝资源文件

选择TARGETS，单击Build Phases页签，在Copy Bundle Resources中添加以下4个bundle：

- **APBToygerFacade.bundle**：位于*APBToygerFacade.framework*中。
- **ToygerNative.bundle**：位于*ToygerNative.framework*中。
- **OCRXMedia.bundle**：位于*OCRDetectSDKForTech.framework*中。
- **BioAuthEngine.bundle**：位于*BioAuthEngine.framework*中。

调用SDK

1. 引入头文件。

   命令如下：

```
#import <AliyunIdentityManager/AliyunIdentityPublicApi.h>
```

2. 初始化SDK。

   > **说明** 为了提升用户体验并为刷脸认证准备一些必备的数据，iOS客户端需要进行SDK初始化。您可以在调用 `appDelegate` 函数（如下示例）和步骤3获取metaInfo数据前，这两个位置二选一放置初始化SDK代码。

   `appDelegate` 函数调用，示例代码：

```
(BOOL)application:(UIApplication)application didFinishLaunchingWithOptions:(NSDictionary)launchOptions {
[AliyunSdk init];// 初始化SDK。
}
```

3. 获取metaInfo数据。

   命令如下：

```
[AliyunSdk init];// 初始化SDK，如果已经在appDelegate函数中调用过，此处无需再重复调用。
[AliyunIdentityManager getMetaInfo]
```

   > **说明** 返回值类型是NSDictionary，您的服务端需要将其转换为JSON字符串，调用服务端发起认证请求接口时，在入参metaInfo中传入该值。关于发起认证请求的具体操作，请参见InitFaceVerify。

4. （可选）获取设备指纹。

   命令如下：

```
/*获取设备指纹token。
1，SDK初始化后需要经过一段时间计算才能获取到deviceToken，建议间隔3秒以上。
2，deviceToken短期有效，不能长期缓存使用，不能作为设备唯一标识。
*/
NSString * astring = [AliyunSdk getMobileSession];
```

   > **说明** 调用 `[AliyunSdk init]` 后，才能获取到该结果。

5. 调用SDK开始认证。

   命令如下：

```
[[AliyunIdentityManager sharedInstance] verifyWith:certifyId extParams:extParams onComp
letion:^(ZIMResponse *response) {}];
```

参数说明如下表所示。

名称	类型	描述
certifyId	NSString	您可以通过服务端发起申请请求接口获取该参数。关于发起认证请求的具体操作，请参见InitFaceVerify。

extParams	NSDictionary	通过以下代码传入当前viewController，用于展现网络加载和presentViewController。
		`[extParams setValue:self forKey:@"currentCtr"];`
		○ 如需设置OCR的**下一步**按钮的颜色，在extParams中添加按钮的正常颜色和按钮按下时的颜色。代码示例如下：
		`[extParams setValue:@"00FF00" forKey:ZIM_EXT_PARAMS_KEY_OCR_BOTTOM_BUTTON_COLOR];`
		`[extParams setValue:@"0000FF" forKey:PARAMS_KEY_OCR_BOTTOM_BUTTON_CLICKED_COLZIM_EXT_OR];`
		○ 如需设置扫脸时的圆圈颜色，在extParams中添加圆圈颜色。代码示例如下：
		`[extParams setValue:@"FF0000" forKey:ZIM_EXT_PARAMS_KEY_OCR_FACE_CIRCLE_COLOR];`
		○ 如需返回本地视频，在extParams中添加 `returnVideo` ，并且设置值为 `true` 。代码示例如下：
		`[extParams setValue:@"true" forKey:@"returnVideo"];`
		○ 如需服务端查询接口返回视频，需要额外在extParams中添加 `ZIM_EXT_PARAMS_KEY_USE_VIDEO_UPLOAD` ，并且设置值为 `true` 。代码示例如下：
		`[extParams setValue:@"true" forKey:ZIM_EXT_PARAMS_KEY_USE_VIDEO_UPLOAD]`
		说明 金融级实人认证支持设置返回活体检测的视频，默认设置为不返回。如果设置为返回，在您使用金融级实人认证SDK完成认证（不区分认证结果为F或T），并在SDK回调和查询接口时返回视频路径，请您及时处理。

返回结果说明：

```
[[AliyunIdentityManager sharedInstance] verifyWith:certifyId extParams:extParams onComp
letion:^(ZIMResponse *response) {}];
```

`response.code` 包含以下返回参数，具体说明如下表所示。

错误码	错误码文案	错误码描述
1000（ZIMResponseSuccess）	刷脸成功	代表刷脸成功，该结果仅供参考，可通过服务端查询认证结果接口获取最终认证结果。关于发起认证请求的具体操作，请参见InitFaceVerify。
1001（ZIMInternalError）	系统错误	表示系统错误。
1003（ZIMInterrupt）	验证中断	表示验证中断。
2002（ZIMNetworkfail）	网络错误	表示网络错误。
2003（ZIMTIMEError）	客户端设备时间错误	表示客户端设备时间错误。
2006（ZIMResponseFail）	刷脸失败	表示刷脸失败，如需获取更详细的失败原因，需调用服务端查询认证结果接口。关于发起认证请求的具体操作，请参见InitFaceVerify。

说明 更多错误码信息，请参见iOS客户端错误码详情。

代码示例：

```
[[AliyunIdentityManager sharedInstance] verifyWith:certifyId extParams:extParams onCompletion:^(ZIMResponse *response) {
        dispatch_async(dispatch_get_main_queue(), ^{
            NSString *title = @"刷脸成功";
            switch (response.code) {
                case ZIMResponseSuccess: //1000。
                    break;
                case ZIMInterrupt:   //1003。
                    title = @"用户退出";
                    break;
                case ZIMNetworkfail: //2002。
                    title = @"网络错误";
                    break;
                case ZIMTIMEError: //2003。
                    title = @"设备时间设置不对";
                    break;
                case ZIMResponseFail: //2006。
                    title = @"刷脸失败";
                    break;
                case ZIMInternalError:   //1001。
                    title = @"初始化失败";
                    break;
                default:
                    break;
            }
```

Demo代码包

您可以下载以下iOS Demo代码包进行体验。在使用过程中，请替换iOS Demo代码包中的文件的版本为最新的版本：

- Objective-C Demo
- Swift Demo

SDK包大小裁剪说明

当您对SDK包大小有限制时，可以根据如下情况进行裁剪：

- 如果您之前没有使用过支付宝开放平台版，或开放平台版代码已经下线，可以在SDK包中删除ZolozIdentityManager、ZolozMobileRPC内容。
- 如果您不需要使用OCR功能，可以在SDK包中删除OCRDetectSDKForTech内容，并删除对应的bundle（OCRXMedia.bundle）。

3.3.3. UniApp接入

金融级实人认证服务提供UniApp插件，帮助您在您的业务应用（App）中实现刷脸认证功能。您可以通过您的应用服务端认证初始化接口，向阿里云服务器调用发起认证请求接口并获取刷脸认证的唯一标识CertifyId，然后使用CertifyId唤起刷脸认证客户端SDK。本文介绍了UniApp客户端接入的操作方法。

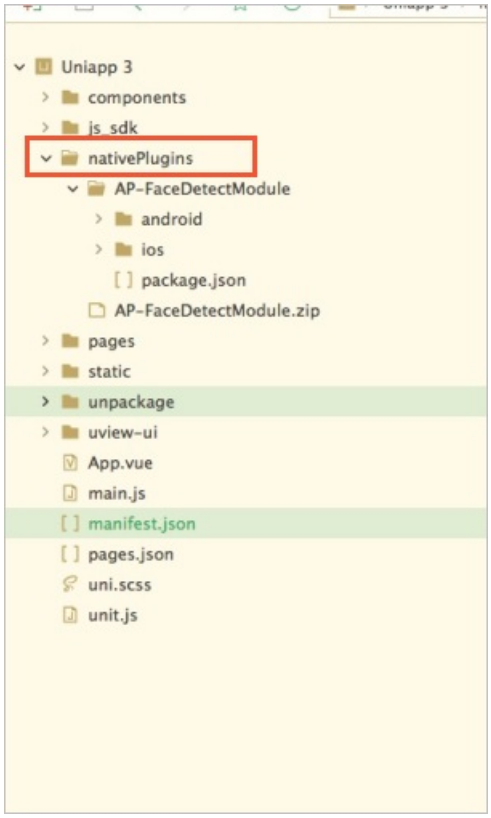
接入前必读

近期发现有个人开发者封装阿里云实人认证SDK后开发了第三方插件，并在某平台上提供付费下载服务。阿里云提醒您，如果您使用了此类非阿里云官方提供的插件或服务，阿里云概不对由此造成的一切后果负责，亦不承担任何法律责任。建议您使用阿里云官方SDK服务。

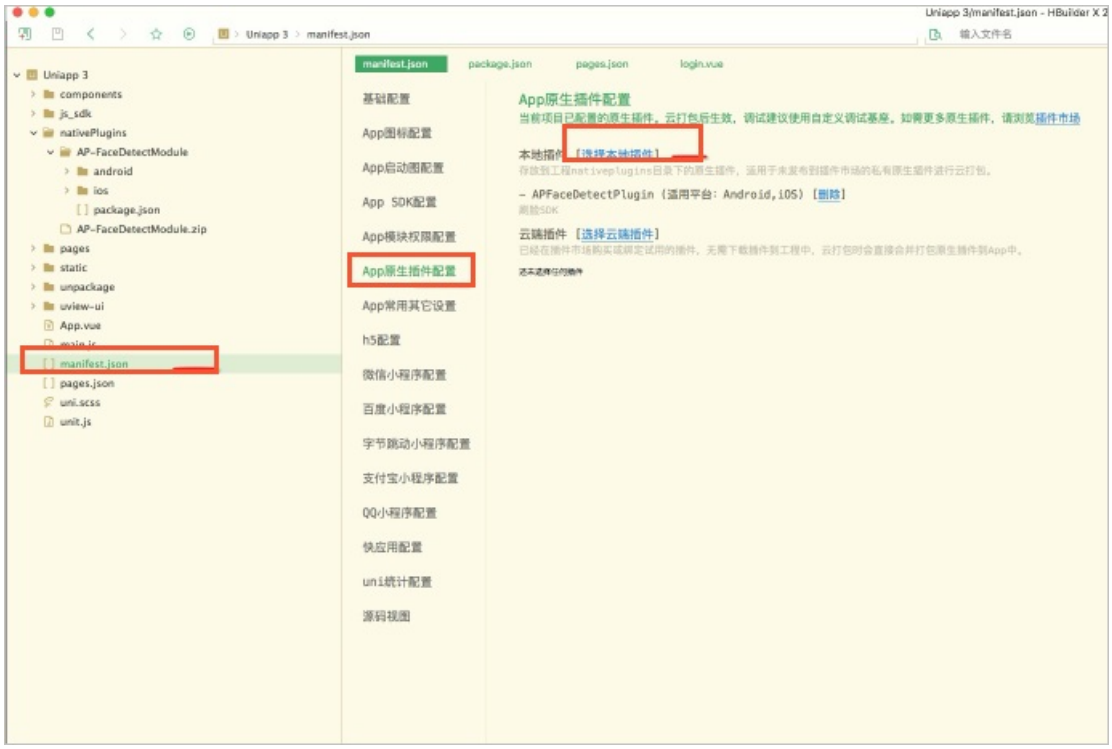
配置依赖

您可以参见DCloud官网文档接入UniApp插件，也可以参考以下步骤接入UniApp插件。

1. 下载金融级实人认证UniApp SDK，解压到项目中的*nativePlugins*目录下。

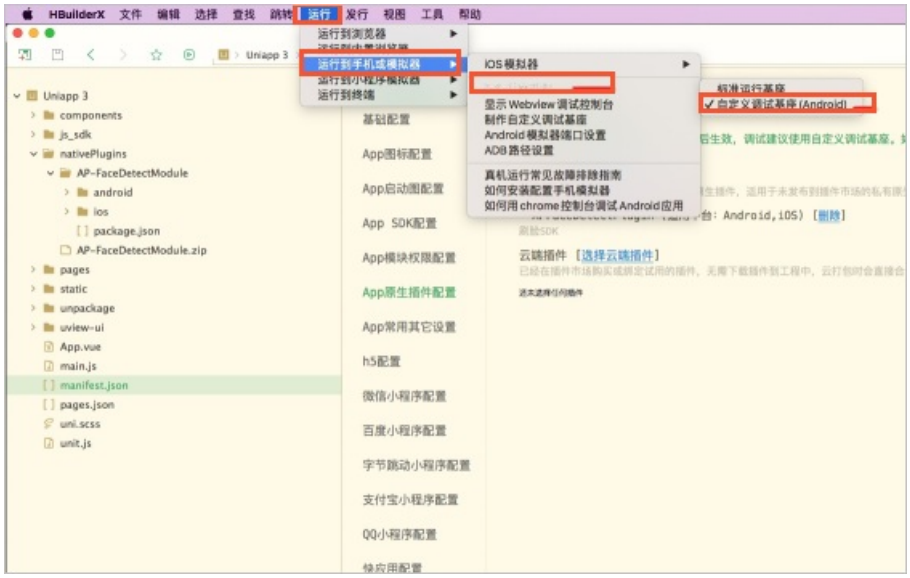


2. 导入本地插件。
   i. 在*manifest.json*文件下，单击**App原生插件配置**。
   ii. 在**App原生插件配置**区域，单击**选择本地插件**。

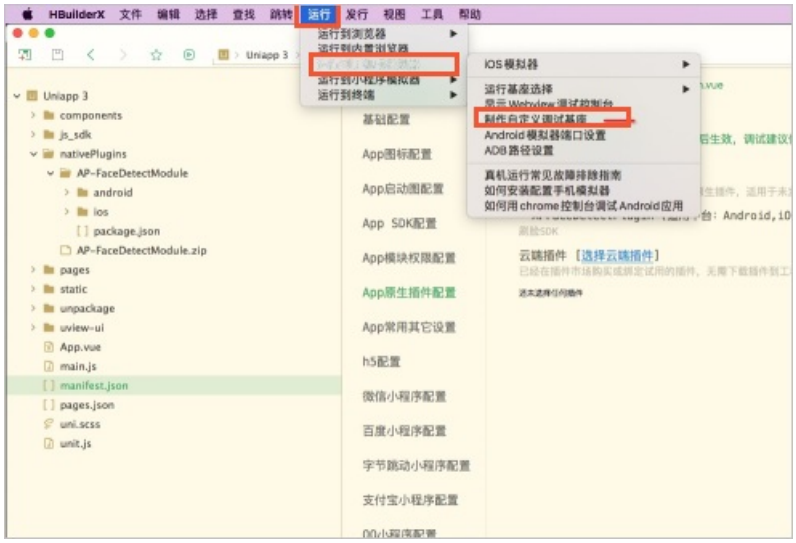


   iii. 在**本地插件选择**对话框，选择需要打包生效的插件。
3. 配置基座包。

i. 设置自定义调试基座。在*manifest.json*文件，选择**运行 > 运行到手机或模拟器 > 运行基座选择 > 自定义调试基座(Android)**。



ii. 制作自定义调试基座。在*manifest.json*文件，单击**运行 > 运行到手机或模拟器**，选择**制作自定义调试基座**。



4. 调用实人认证服务。

   i. 获取metaInfo数据。

   您在调用刷脸认证服务端发起认证请求时需要传入该值。具体操作，请参见发起认证请求。

   ii. 调用**verify**接口，传入certifyId进行认证。

   参数说明如下表所示。

名称	类型	描述
certifyId	String	认证的唯一标识。

名称	类型	描述
returnVideo	String	是否返回视频（iOS）。取值： ■ **true**：返回视频。 ■ **false**（默认）：不返回视频。
ext_params_key_use_video	String	是否返回视频（Android）。取值： ■ **true**：返回视频。 ■ **false**（默认）：不返回视频。
ext_params_key_ocr_bottom_button_color	String	OCR底部按钮的默认颜色。
ext_params_key_ocr_bottom_button_clicked_color	String	OCR底部按钮被操作时的颜色。
ext_params_key_ocr_face_circle_color	String	实人认证进度条颜色。

接入示例：

```
<template>
</template>
<script>
const n = uni.requireNativePlugin('AP-FaceDetectModule');
export default {
    methods: {
        verifyC: function() {
            n.verify(
                {
                    certifyId: this.certify
                },
                function(t) {
                    uni.showToast({
                        title: '返回的内容' + t,
                        icon: 'none'
                    });
                }
            );
        },
        //获取环境参数接口。
        getMetaInfo: function() {
            var t = n.getMetaInfo();
        },
    }
};
</script>
```

错误码说明

错误码	错误码文案	错误码描述
1000	刷脸成功	表示刷脸成功。 说明 该结果仅供参考。您可通过服务端查询认证结果接口获取最终认证结果。具体操作，请参见InitFaceVerify。
1001	系统错误	表示系统错误。
1003	验证中断	表示验证中断。
2002	网络错误	表示网络错误。
2003	客户端设备时间错误	表示客户端设备时间错误。
2006	刷脸失败	表示刷脸失败。 说明 如需获取失败原因的详细信息，需调用服务端查询认证结果接口获取。具体操作，请参见InitFaceVerify。

3.3.4. APICloud接入

金融级实人认证服务提供APICloud插件，帮助您在业务应用（App）中实现刷脸认证功能。您可以通过应用服务端认证初始化接口，向阿里云服务器调用发起认证请求接口并获取刷脸认证的唯一标识CertifyId，然后使用CertifyId唤起刷脸认证客户端SDK。本文介绍了APICloud客户端接入的操作方法。

背景信息

低代码平台APICloud的模块库提供多种模块。您可以使用已有模块，在应用中快速部署集成，缩短开发流程。当已有模块无法满足特殊需求时，您也可以自定义相关模块。

接入前必读

近期发现有个人开发者封装阿里云实人认证SDK后开发了第三方插件，并在某平台上提供付费下载服务。阿里云提醒您，如果您使用了此类非阿里云官方提供的插件或服务，阿里云概不对由此造成的一切后果负责，亦不承担任何法律责任。建议您使用阿里云官方SDK服务。

前提条件

已在APICloud官网中创建应用。

配置依赖

1. 下载金融级实人认证APICloud SDK。
2. 在APICloud控制台上传刷脸认证APICloud模块。
   i. 登录APICloud控制台。

ii. 在左侧导航栏，选择**端开发 > 模块**。

iii. 在**模块**页签，选择**自定义模块**。

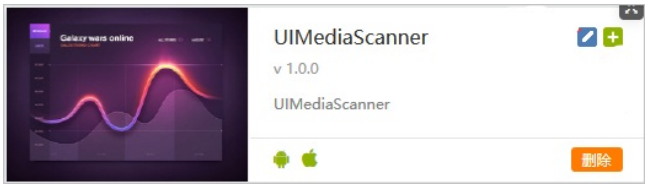
iv. 在**自定义模块**页签，输入有效的**模块名称**、**模块概要**及版本号，选择并上传已下载的APICloud模块。

v. 单击**保存**。

3. 将APICloud模块添加到应用中。

i. 在**模块**页签，选择**模块库**。

ii. 在**模块库**页签，单击已上传APICloud模块右上角的图标，将APICloud模块添加到应用中。



> **说明** 自定义模块仅对自己的账号可见。如果需要对外开放，您可以在**模块管理**页面提交该模块，官方审核通过后将会发布到模块库。

当您需要更新已上传的自定义模块时，可以通过已上传自定义模块右上角的图标，重新上传自定义模块包，然后单击**保存**。

函数说明

APICloud插件的接入流程包含获取MetaInfo和开始认证函数。具体说明如下。

- **获取MetaInfo数据**

  调用getMetaInfo函数获取当前移动设备端的环境信息，并将这些信息发送到业务服务器端。业务服务器端会将这些信息作为参数之一（MetaInfo）调用服务端初始化认证接口（InitSmartVerify），从而获取CertifyId，用于后续验证环节。

  示例代码：

```
var aliyunFace = api.require('aliyunFaceVerifyModule');
aliyunFace.getMetaInfo(function(ret, err) {
    alert(ret);
});
```

- **开始认证**

  调用Verify函数发起金融级实人认证。

  参数说明：

名称	类型	说明

名称	类型	说明
certifyId	String	从服务端初始化认证接口（InitSmartVerify）获取的CertifyId。 **说明** 每个CertifyId只能调用一次**Verify**函数，每次调用**Verify**函数之前都必须重新获取CertifyId。
extParams	Map<String, String>	用户自定义参数，可以为空。有关目前支持自定义的字段，请参见用户自定义参数extParams支持的字段说明。

callback	IdentityCallback	认证结果回调。回调格式如下：
		`public class IdentityResponse {`
		`// 结果Code，具体含义请参见本文档“返回值说明”章节。`
		`public int code;`
		`// 结果Code描述。`
		`public String message;`
		`}`
		`public interface IdentityCallback {`
		`boolean response(IdentityResponse response);`
		`}`

用户自定义参数extParams支持的字段说明

名称	描述	示例值
ext_params_key_use_video	是否需要返回视频留证。取值： ○ **false（默认）**：不返回视频留证。 ○ **true**：返回视频留证。	false
kIdentityParamKeyNextButtonColor	OCR识别结果确认页底部按钮的底色。	#FF0000
ext_params_key_face_progress_color	定制实人认证进度条的颜色。	#FF0000

示例代码：

```
var aliyunFace = api.require('aliyunFaceVerifyModule');
aliyunFace.verify({
    certifyId: '1adffa'
}, function(ret, err) {
    if (ret.code == 1000) {
        alert('认证成功');
    } else {
        alert(err.code);
    }
});
```

返回值说明

服务器端的验证结果通过 `IdentityResponse.code` 返回，取值范围如下。

错误码	说明
1000	刷脸成功。
1001	系统错误。
1003	验证中断。
2002	网络错误。
2003	客户端设备时间错误。
2006	刷脸失败。

4.金融级活体检测方案

4.1. 服务端接入

4.1.1. InitFaceVerify

本文介绍了如何调用InitFaceVerify接口在服务端接入流程中发起认证请求。

接入方式说明

如果采用服务端接入方式，您可以配合SDK调用InitFaceVerify接口发起认证请求，获取CertifyId，唤起客户端上传人脸信息。用户上传人脸信息后，您可以调用DescribeFaceVerify接口查询认证结果。

发起认证请求

接口名：InitFaceVerify。

服务地址：`cloudauth.aliyuncs.com`。

请求方法：HTTPS POST和GET。

接口说明：每次开始认证前通过InitFaceVerify接口获取CertifyId，用来串联认证请求中的各个接口。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	认证场景ID。该ID在控制台创建认证场景后自动生成。	100000****
OuterOrderNo	String	是	客户服务端自定义的业务唯一标识，用于后续定位排查问题使用。值最长为32位长度的字母数字组合，请确保唯一。	e0c34a77f5ac40a5aa5e6ed20c353888
ProductCode	String	是	认证方案。 唯一取值：LR_FR。	LR_FR

名称	类型	是否必选	描述	示例值
Model	String	否	活体检测类型。取值： • LIVENESS（默认）：眨眼动作活体检测。 • PHOTINUS_LIVENESS：眨眼动作活体+炫彩活体双重检测。 • MULTI_ACTION：多动作活体检测。当前为眨眼+任意摇头检测。 说明 活体检测模式仅支持运行在Android SDK 1.2.6及以上版本和iOS SDK 1.2.4及以上版本；多动作活体检测模式仅支持运行在Android SDK 1.6.4及以上版本和iOS SDK 1.5.1及以上版本。	LIVENESS
UserId	String	是	客户业务自定义的用户ID，请保持唯一。	123456789
CertType	String	否	证件类型。 当前仅支持身份证，必须传入IDENTITY_CARD。 说明 该方案中，不验证姓名和证件号码的真实性、一致性。如需验证，请选择实人认证方案。	IDENTITY_CARD
CertName	String	否	真实姓名。	张三
CertNo	String	否	证件号码。	330103xxxxxxxxxxxx
MetaInfo	String	是	MetaInfo环境参数，需要通过客户端SDK获取。	{"zimVer":"3.0.0","appVersion":"1","bioMetaInfo":"4.1.0:11501568,0","appName":"com.aliyun.antcloudauth","deviceType":"ios","osVersion":"iOS 10.3.2","apdidToken":"","deviceModel":"iPhone9,1"}
Mobile	String	否	用户手机号码。	130xxxxxxxx
Ip	String	否	用户网络IP。	114.xxx.xxx.xxx

名称	类型	是否必选	描述	示例值
CallbackUrl	String	否	认证结果的回调通知地址，必须以 `https` 开头。 平台在完成认证后会回调该地址，并自动添加 `certifyId` 、 `passed` 和 `subcode` 字段，示例： `https://www.aliyun.com?certifyId=xxxx&passed=T&subcode=200` 。	https://www.aliyun.com
CallbackToken	String	否	安全Token，由您自行生成，用于防重复、防篡改校验。 如果设置了该值会在回调地址显示**CallbackToken**字段。	NMjvQanQgplBSaEI0sL86WnQplB
AuthId	String	否	用户授权ID，最大长度为64位字符。	92d46b9e9e2d703f2897f350d5bd4149

返回数据

名称	类型	描述	示例值
RequestId	String	请求ID。	130A2C10-B9EE-4D84-88E3-5384FF039795
Message	String	返回信息。	success
Code	String	返回码。200表示成功，其他表示失败。关于返回码的具体信息，请参见返回Code和Message说明。	200
ResultObject.CertifyId	String	实人认证唯一标识。	91707dc296d469ad38e4c5efa6a0****

返回Code和Message说明

Code	Message	描述
200	success	成功。
400	参数不能为空	参数不能为空。
401	参数非法	非法参数。传入的姓名、身份证号码长度必须符合国家标准且不得包含英文字母等特殊字符。
402	应用配置不存在	应用配置不存在。

Code	Message	描述
404	认证场景配置不存在	认证场景配置不存在，请先在控制台上创建认证场景。
410	未开通服务	未开通OSS产品或未完成OSS读写授权，请登录控制台完成授权。具体操作，请参见授权金融级实人认证访问OSS存储空间。
411	RAM无权限	需要为RAM用户授予AliyunAntCloudAuthFullAccess的操作权限。
412	欠费中	金融级实人认证或OSS存在欠费，请充值后操作。
414	设备类型不支持	当前移动设备不支持刷脸认证，请更换设备后操作。
415	SDK版本不支持	当前认证SDK版本不支持刷脸认证，请升级SDK后操作。
416	系统版本不支持	当前操作系统版本不支持刷脸认证，请升级系统或更换设备操作。
418	刷脸失败次数过多	当天刷脸认证次数过多，请明天再试。
500	系统错误	系统内部错误，请反馈工程师排查。

4.1.2. DescribeFaceVerify

本文介绍了如何通过DescribeFaceVerify接口在服务端接入流程中查询活体检测的结果。

接入方式说明

如果采用服务端接入方式，您可以配合SDK调用InitFaceVerify接口发起认证请求，获取CertifyId，唤起客户端上传人脸信息。用户上传人脸信息后，您可以调用DescribeFaceVerify接口查询认证结果。

查询认证结果

接口名：DescribeFaceVerify。

服务地址：`cloudauth.aliyuncs.com`。

请求方法：HTTPS POST和GET。

接口说明：当接入方移动端收到回调之后，其服务端可以调用此接口，来获取相应的认证状态和认证资料。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	认证场景ID。	100000****
CertifyId	String	是	实人认证唯一标识。该值需要调用InitFaceVerify获取。	91707dc296d469ad38e4c5efa6a0****

返回数据

名称	类型	是否必选	描述	示例值
RequestId	String	是	请求ID。	130A2C10-B9EE-4D84-88E3-5384FF039795
Message	String	否	错误信息。	success
Code	String	是	返回码。200表示接口响应成功。 详细认证结果判断请参见ResultObject.SubCode错误码说明。	200
ResultObject.Passed	String	是	认证结果。取值： • T：通过。 • F：不通过。 说明 判断认证结果请以ResultObject.Passed字段为准。	T
ResultObject.SubCode	String	否	认证结果描述。 具体请参见ResultObject.SubCode错误码说明。 说明 判断认证结果请以ResultObject.Passed字段为准。	200
ResultObject.IdentityInfo	String	否	认证的主体信息，一般的认证场景返回为空。	null
ResultObject.DeviceToken	String	否	设备Token。	McozS1ZWRcRZStlERcZZo_QOytx5jcgZoZJEoRLOxxxxxxx

名称	类型	是否必选	描述	示例值
ResultObject.MaterialInfo	String	否	认证主体附件信息，主要为图片类材料。JSON格式，请参见ResultObject.MaterialInfo的JSON格式示例。	{"faceAttack": "F","faceOcclusion": "F","facialPictureFront": {"faceAttackScore": 0.00008597839769208804,"qualityScore": 88.3615493774414,"pictureUrl": "https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg","ossBucketName": "cn-shanghai-aliyun-cloudauth-1260051251634779","ossObjectName": "verify/1260051251634779/6ba7bcfccf33f56cdb44ed086f36ce3e0.jpeg"}}

返回示例

ResultObject.MaterialInfo的JSON格式示例：

```
{
     // 是否为攻击：攻击为T，非攻击为F。
    "faceAttack": "F",
     // 是否有脸部遮挡：有脸部遮挡为T，否则为F。
    "faceOcclusion": "F",
    // 认证的照片信息。
    "facialPictureFront": {
         // 人脸攻击分。
        "faceAttackScore": 0.00008597839769208804,
        // 活体人脸质量分数。
        "qualityScore": 88.3615493774414,
        // 照片https地址，有效期15分钟，从查询时开始计时。
        "pictureUrl": "https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-shanghai.aliyuncs
.com/verify/xxxxx/xxxxx.jpeg",
        // 照片存放客户上海区域的OSS里，此为照片的bucket名。
        "ossBucketName": "cn-shanghai-aliyun-cloudauth-1260051251634779",
        // 照片存放客户上海区域的OSS里，此为照片的文件名。
        "ossObjectName": "verify/1260051251634779/6ba7bcfccf33f56cdb44ed086f36ce3e0.jpeg",
        // 视频存放客户上海区域的OSS里，此为视频的文件名（仅当客户端设置视频参数时返回）。
        "ossVerifyVideoObjectName": "verify/1260051251634779/03a081bd96328aedf83f635****_ve
rifyvideo_dcb7.mov",
        // 视频HTTPS地址，有效期15分钟，从查询时开始计时（仅当客户端设置视频参数时返回）。
        "verifyVideoUrl": "http://cn-shanghai-aliyun-cloudauth-1260051251634779.oss-cn-shan
ghai.aliyuncs.com/verify/1260051251634779/03a081bd96328aedf83f635f39a50c57__verifyvideo_dcb
7.mov?Expires=1625371140&OSSAccessKeyId=STS.NTX1ngfr6Acg2Pmnn2RYM****&Signature=Hw5BF9WxJs6
wI68IxKs41cxCU8****&security-token=CAISjgJ1q6Ft5B2yfSjIr5ftetTTi60X9qGMMHbcim5nXtZhu7GT1Dz2
IH1PdXFgA%2Bgds%2Fswmm5U7vgalrkqEcEdHRGdN5YpsM8LrlzwO1h2TGRsq%2B5qsoasPETOITyZtZagToeUZdfZf
ejXGDKgvyRvwLz8WCy%2FVli%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGfLC1dysQcO4gEW
q4bHm5fCskKD1Qenk7FO%2B9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWwIMvErYbbaMrIV
1WwZ9Z7knHaVAq%2BXwnOBkuuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAF%2FHZbelNLYWtipAu5X5oG1t22SqI
r1p3TMK5FrjDIVeWOppcxeRXtU%2BjR7hGcwd25scGhiraoxTXV3sxw%2F6dhFSswQ37O4j%2B%2FrCPtiGauqn2ZMd
Mj%2FvYwKb6KmpQqa%2BtR%2F9cxhS6hoIQRq0SUIfyXl5ZUiZbTjW22iwuX%2BPwzVCw%3D%3D"
    }
}
```


Result Object.SubCode错误码说明

错误码	认证描述文案	是否计费	可能原因和运营建议
200	认证通过	是	成功。
205	活体检测存在风险	是	可能存在攻击风险，建议人工审核分层处理，若为真人可重新操作。
206	业务策略限制	是	个性化业务安全策略限制，如有疑问可咨询客服。

返回Code和Message说明

Code	Message	描述
200	success	成功。

Code	Message	描述
400	参数不能为空	参数不能为空。
401	参数非法	非法参数。传入的姓名、身份证号码长度必须符合国家标准且不得包含英文字母等特殊字符。
403	认证未完成	认证未完成。
404	认证场景配置不存在	认证场景配置不存在，请先在控制台上创建认证场景。
406	无效的CertifyId	无效的CertifyId。
410	未开通服务	未开通OSS产品或未完成OSS读写授权，请登录控制台完成授权。具体操作，请参见授权金融级实人认证访问OSS存储空间。
411	RAM无权限	需要给子账号授予AliyunAntCloudAuthFullAccess的操作权限。
424	身份认证记录不存在	当前CertifyId还未完成认证，建议您客户端收到认证回调为1000或2006时，再请求查询接口获取详情。 说明 如果出现CertifyId未认证、认证过程中放弃、认证处理中的情况，均会导致查询接口返回此错误码。
500	系统错误	系统内部错误。

4.1.3. Java SDK调用示例

本文介绍了金融级活体检测方案Java SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

本章节提供的是最新版本的Java SDK调用示例，如果您需要查看之前版本，请参见Java SDK调用示例（V1版本）。

前提条件

需要使用的Java版本在1.8版本及以上。

使用Maven引入SDK

```
<dependency>
        <groupId>com.aliyun</groupId>
        <artifactId>cloudauth20190307</artifactId>
         <version>1.0.7</version>
</dependency>
```

调用示例

InitFaceVerify接口调用示例

```
import java.util.Arrays;
import java.util.List;
import com.aliyun.cloudauth20190307.Client;
import com.aliyun.cloudauth20190307.models.*;
import com.aliyun.tea.TeaException;
import com.aliyun.tea.TeaUnretryableException;
import com.aliyun.tearpc.models.Config;
import com.aliyun.teautil.models.RuntimeOptions;
public class InitFaceVerify {
    public static void main(String[] args) throws Exception {
        InitFaceVerifyRequest request = new InitFaceVerifyRequest();
        // 请输入场景ID+L。
        request.setSceneId(xxxxL);
        // 设置商户请求的唯一标识。
        request.setOuterOrderNo("xxxx");
        // 认证方案。
        request.setProductCode("LR_FR");
        request.setUserId("12345xxxx");
        // 模式。
        request.setModel("LIVENESS");
        //request.setCertType("IDENTITY_CARD");
        //request.setCertName("张三");
        //request.setCertNo("xxxx");
        // MetaInfo环境参数。
        request.setMetaInfo("xxxxxx");
        //request.setMobile("130xxxxxxxx");
        //request.setIp("114.xxx.xxx.xxx");
        // 推荐，支持服务路由。
        InitFaceVerifyResponse response = initFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        //InitFaceVerifyResponse response = initFaceVerify("cloudauth.cn-shanghai.aliyuncs.
com", request);
        System.out.println(response.getRequestId());
        System.out.println(response.getCode());
        System.out.println(response.getMessage());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getCertifyId());
    }
    private static InitFaceVerifyResponse initFaceVerifyAutoRoute(InitFaceVerifyRequest req
uest) {
        // 第一个为主区域Endpoint，第二个为备区域Endpoint。
        List<String> endpoints = Arrays.asList("cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com");
        InitFaceVerifyResponse lastResponse = null;
        for (String endpoint : endpoints) {
            try {
                InitFaceVerifyResponse response = initFaceVerify(endpoint, request);
                lastResponse = response;
                // 服务端错误，切换到下个区域调用。
                if (response != null && "500".equals(response.getCode())) {
                    continue;
```

```
                    continue;
                }
                return response;
            } catch (Exception e) {
                // 网络异常，切换到下个区域调用。
                if (e.getCause() instanceof TeaException) {
                    TeaException teaException = ((TeaException)e.getCause());
                    if (teaException.getData() != null && "ServiceUnavailable".equals(
                        teaException.getData().get("Code"))) {
                        continue;
                    }
                }
                if (e.getCause() instanceof TeaUnretryableException) {
                    continue;
                }
            }
        }
        return lastResponse;
    }
    private static InitFaceVerifyResponse initFaceVerify(String endpoint, InitFaceVerifyReq
uest request)
        throws Exception {
        Config config = new Config();
        config.setAccessKeyId("您的AccessKey ID");
        config.setAccessKeySecret("您的AccessKey Secret");
        config.setEndpoint(endpoint);
        // 设置http代理。
        //config.setHttpProxy("http://xx.xx.xx.xx:xxxx");
        // 设置https代理。
        //config.setHttpsProxy("https://xx.xx.xx.xx:xxxx");
        Client client = new Client(config);
        // 创建RuntimeObject实例并设置运行参数。
        RuntimeOptions runtime = new RuntimeOptions();
        runtime.readTimeout = 10000;
        runtime.connectTimeout = 10000;
        return client.initFaceVerify(request, runtime);
    }
}
```

DescribeFaceVerify接口调用示例

```
import java.util.Arrays;
import java.util.List;
import com.aliyun.cloudauth20190307.Client;
import com.aliyun.cloudauth20190307.models.DescribeFaceVerifyRequest;
import com.aliyun.cloudauth20190307.models.DescribeFaceVerifyResponse;
import com.aliyun.tea.TeaException;
import com.aliyun.tea.TeaUnretryableException;
import com.aliyun.tearpc.models.Config;
import com.aliyun.teautil.models.RuntimeOptions;
public class DescribeFaceVerify {
    public static void main(String[] args) throws Exception {
        // 通过以下代码创建API请求并设置参数。
        DescribeFaceVerifyRequest request = new DescribeFaceVerifyRequest();
```

```
        // 请输入场景ID+L。
        request.setSceneId(xxxxL);
        request.setCertifyId("xxxx");
        // 推荐，支持服务路由。
        DescribeFaceVerifyResponse response = describeFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        //DescribeFaceVerifyResponse response = describeFaceVerify("cloudauth.cn-shanghai.a
liyuncs.com", request);
        System.out.println(response.getRequestId());
        System.out.println(response.getCode());
        System.out.println(response.getMessage());
        System.out.println(
            response.getResultObject() == null ? null : response.getResultObject().getPasse
d());
        System.out.println(
            response.getResultObject() == null ? null : response.getResultObject().getSubCo
de());
        System.out.println(
            response.getResultObject() == null ? null
                : response.getResultObject().getIdentityInfo());
        System.out.println(
            response.getResultObject() == null ? null
                : response.getResultObject().getDeviceToken());
        System.out.println(
            response.getResultObject() == null ? null
                : response.getResultObject().getMaterialInfo());
    }
    private static DescribeFaceVerifyResponse describeFaceVerifyAutoRoute(DescribeFaceVerif
yRequest request) {
        // 第一个为主区域Endpoint，第二个为备区域Endpoint。
        List<String> endpoints = Arrays.asList("cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com");
        DescribeFaceVerifyResponse lastResponse = null;
        for (String endpoint : endpoints) {
            try {
                DescribeFaceVerifyResponse response = describeFaceVerify(endpoint, request)
;
                lastResponse = response;
                // 服务端错误，切换到下个区域调用。
                if (response != null && "500".equals(response.getCode())) {
                    continue;
                }
                return response;
            } catch (Exception e) {
                // 网络异常，切换到下个区域调用。
                if (e.getCause() instanceof TeaException) {
                    TeaException teaException = ((TeaException)e.getCause());
                    if (teaException.getData() != null && "ServiceUnavailable".equals(
                        teaException.getData().get("Code"))) {
                        continue;
                    }
                }
                if (e.getCause() instanceof TeaUnretryableException) {
                    continue;
```

```
                }
            }
        }
        return lastResponse;
    }
    private static DescribeFaceVerifyResponse describeFaceVerify(String endpoint, DescribeF
aceVerifyRequest request)
        throws Exception {
        Config config = new Config();
        config.setAccessKeyId("您的AccessKey ID");
        config.setAccessKeySecret("您的AccessKey Secret");
        config.setEndpoint(endpoint);
        // 设置http代理。
        //config.setHttpProxy("http://xx.xx.xx.xx:xxxx");
        // 设置https代理。
        //config.setHttpsProxy("http://xx.xx.xx.xx:xxxx");
        Client client = new Client(config);
        // 创建RuntimeObject实例并设置运行参数。
        RuntimeOptions runtime = new RuntimeOptions();
        runtime.readTimeout = 10000;
        runtime.connectTimeout = 10000;
        return client.describeFaceVerify(request, runtime);
    }
}
```

4.1.4. PHP SDK调用示例

本文介绍了金融级活体检测方案的PHP SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的PHP版本在5.6版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于PHP SDK源代码，请参见PHP SDK包。

- 通过Composer来管理项目依赖（推荐）

```
composer require alibabacloud/cloudauth-20190307 1.0.7
```

- 使用composer.json文件

```
{
   "require": {
       "alibabacloud/cloudauth-20190307": "1.0.7"
    }
}
```

如果因网络问题导致无法按照上述操作安装SDK，您可以使用阿里云Composer全量镜像。

调用示例

InitFaceVerify接口调用示例

```
<?php
namespace AlibabaCloud\SDK\Sample {
use AlibabaCloud\SDK\Cloudauth\V20190307\Cloudauth;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\InitFaceVerifyRequest;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\InitFaceVerifyResponse;
use AlibabaCloud\Tea\Utils\Utils;
use \Exception;
use AlibabaCloud\Tea\Exception\TeaError;
use AlibabaCloud\Tea\Rpc\Rpc\Config;
use AlibabaCloud\Tea\Utils\Utils\RuntimeOptions;
class InitFaceVerify {
    /**
     * @param string $endpoint
     * @return Cloudauth
     */
    public static function createClient($endpoint){
        $config = new Config([
            // 您的AccessKey ID。
            "accessKeyId" => "<您的AccessKey ID>",
            // 您的AccessKey Secret。
            "accessKeySecret" => "<您的AccessKey Secret>",
            // 设置HTTP代理。
            // "httpProxy" => "<http://xx.xx.xx.xx:xxxx>",
            // 设置HTTPS代理。
            // "httpsProxy" => "<https://xx.xx.xx.xx:xxxx>",
            "endpoint" => $endpoint
        ]);
        return new Cloudauth($config);
    }
    /**
     * @return void
     */
    public static function main(){
        $request = new InitFaceVerifyRequest([
            // 请输入场景ID。
            "sceneId" => <xxx>,
            "outerOrderNo" => "<商户请求的唯一标识>",
            // 要接入的认证方案。
            "productCode" => "<LR_FR>",
            // 要进行活体检测的类型。
            "model" => "<LIVENESS>",
            "userId" => "<xxx>",
            "certType" => "<IDENTITY_CARD>"
```

```
            "certType" => "<IDENTITY_CARD>",
            // "certName" => "<张三>",
            // "certNo" => "<xxx>",
            // MetaInfo环境参数，需要通过客户端SDK获取。
            "metaInfo" => '{"zimVer":"3.0.0","appVersion": "1","bioMetaInfo": "4.1.0:115015
68,0",
        "appName": "com.aliyun.antcloudauth","deviceType": "ios","osVersion": "iOS 10.3.2",
"apdidToken": "","deviceModel": "iPhone9,1"}',
            // "mobile" => "<xxx>",
            // "ip" => "<114.xxx.xxx.xxx>",
            // "callbackUrl" => "<https://www.aliyundoc.com>",
            // "callbackToken" => "<xxxxxxx>"
        ]);
        // 推荐，支持服务路由。
        $response = self::initFaceVerifyAutoRoute($request);
        // 不支持服务自动路由。
        // $response = self::initFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", $request)
;
        var_dump($response->requestId);
        var_dump($response->code);
        var_dump($response->message);
        var_dump($response->resultObject->certifyId);
    }
    /**
     * @param InitFaceVerifyRequest $request
     * @return InitFaceVerifyResponse
     */
    public static function initFaceVerifyAutoRoute(InitFaceVerifyRequest $request){
        $endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        foreach($endpoints as $endpoint){
            try {
                $response = self::initFaceVerify($endpoint, $request);
                if (Utils::equalString("500", $response->code)) {
                    continue;
                }
                return $response;
            }
            catch (Exception $err) {
                if (!($err instanceof TeaError)) {
                    $err = new TeaError([], $err->getMessage(), $err->getCode(), $err);
                }
                // 网络异常，切换到下个区域调用。
                if (Utils::equalString("ServiceUnavailable", $err->code)) {
                    continue;
                }
            }
        }
        return null;
    }
    /**
     * @param string $endpoint
     * @param InitFaceVerifyRequest $request
```

```
     * @return InitFaceVerifyResponse
     */
    public static function initFaceVerify($endpoint, $request){
        $client = self::createClient($endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        $runtime = new RuntimeOptions([]);
        $runtime->readTimeout = 10000;
        $runtime->connectTimeout = 10000;
        return $client->initFaceVerify($request, $runtime);
    }
}
$path = __DIR__ . \DIRECTORY_SEPARATOR . \DIRECTORY_SEPARATOR . 'vendor' . \DIRECTORY_SEPAR
ATOR . 'autoload.php';
if (file_exists($path)) {
    require_once $path;
}
InitFaceVerify::main();
}
```

DescribeFaceVerify接口调用示例

```
<?php
namespace AlibabaCloud\SDK\Sample {
use AlibabaCloud\SDK\Cloudauth\V20190307\Cloudauth;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\DescribeFaceVerifyRequest;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\DescribeFaceVerifyResponse;
use AlibabaCloud\Tea\Utils\Utils;
use \Exception;
use AlibabaCloud\Tea\Exception\TeaError;
use AlibabaCloud\Tea\Rpc\Rpc\Config;
use AlibabaCloud\Tea\Utils\Utils\RuntimeOptions;
class DescribeFaceVerify {
    /**
     * @param string $endpoint
     * @return Cloudauth
     */
    public static function createClient($endpoint){
        $config = new Config([
            // 您的AccessKey ID。
            "accessKeyId" => "<您的AccessKey ID>",
            // 您的AccessKey Secret。
            "accessKeySecret" => "<您的AccessKey Secret>",
            // 设置HTTP代理。
            // "httpProxy" => "<http://xx.xx.xx.xx:xxxx>",
            // 设置HTTPS代理。
            // "httpsProxy" => "<https://xx.xx.xx.xx:xxxx>",
            "endpoint" => $endpoint
        ]);
        return new Cloudauth($config);
    }
    /**
     * @return void
     */
```

```
    public static function main(){
        $request = new DescribeFaceVerifyRequest();
        // 场景ID。
        $request->sceneId=<xxx>;
        // 认证ID。
        $request->certifyId="<xxx>";
        // 推荐，支持服务路由。
        $response = self::describeFaceVerifyAutoRoute($request);
        // 不支持服务自动路由。
        // $response = self::describeFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", $requ
est);
        var_dump($response->requestId);
        var_dump($response->code);
        var_dump($response->message);
        var_dump($response->resultObject->passed);
        var_dump($response->resultObject->subCode);
        var_dump($response->resultObject->materialInfo);
        var_dump($response->resultObject->identityInfo);
        var_dump($response->resultObject->deviceToken);
    }
    /**
     * @param DescribeFaceVerifyRequest $request
     * @return DescribeFaceVerifyResponse
     */
    public static function describeFaceVerifyAutoRoute($request){
        $endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        foreach($endpoints as $endpoint){
            try {
                $response = self::describeFaceVerify($endpoint, $request);
                if (Utils::equalString("500", $response->code)) {
                    continue;
                }
                return $response;
            }
            catch (Exception $err) {
                if (!($err instanceof TeaError)) {
                    $err = new TeaError([], $err->getMessage(), $err->getCode(), $err);
                }
                // 网络异常，切换到下个区域调用。
                if (Utils::equalString("ServiceUnavailable", $err->code)) {
                    continue;
                }
            }
        }
        return null;
    }
    /**
     * @param string $endpoint
     * @param DescribeFaceVerifyRequest $request
     * @return DescribeFaceVerifyResponse
     */
    public static function describeFaceVerify($endpoint, $request){
```

```
    public static function describeFaceVerify($endpoint, $request){
        $client = self::createClient($endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        $runtime = new RuntimeOptions([]);
        $runtime->readTimeout = 10000;
        $runtime->connectTimeout = 10000;
        return $client->describeFaceVerify($request, $runtime);
    }
}
$path = __DIR__ . \DIRECTORY_SEPARATOR . \DIRECTORY_SEPARATOR . 'vendor' . \DIRECTORY_SEPAR
ATOR . 'autoload.php';
if (file_exists($path)) {
    require_once $path;
}
DescribeFaceVerify::main();
}
```

4.1.5. Python SDK调用示例

本文介绍了金融级活体检测方案的Python SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的Python版本在3.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

引入依赖包

```
pip install alibabacloud_tea_openapi==0.2.5
```

安装SDK

关于Python SDK源代码，请参见Python SDK包。

```
pip install alibabacloud_cloudauth20190307==1.0.7
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
from alibabacloud_cloudauth20190307.client import Client as CloudauthClient
from alibabacloud_tea_openapi import models as open_api_models
from alibabacloud_cloudauth20190307 import models as cloudauth_models
from alibabacloud_tea_util.client import Client as UtilClient
```

```
from alibabacloud_tea_util import models as util_models
class InitFaceVerify:
    def __init__(self):
        pass
    @staticmethod
    def create_client(
            endpoint: str,
    ) -> CloudauthClient:
        config = open_api_models.Config(
            # 您的AccessKey ID。
            access_key_id='您的AccessKey ID',
            # 您的AccessKey Secret。
            access_key_secret='您的AccessKey Secret',
            # 设置HTTP代理。
            # http_proxy='http://xx.xx.xx.xx:xxxx',
            # 设置HTTPS代理。
            # https_proxy='https://username:password@xxx.xxx.xxx.xxx:9999',
            endpoint=endpoint
        )
        return CloudauthClient(config)
    @staticmethod
    def main() -> None:
        request = cloudauth_models.InitFaceVerifyRequest(
            # 请输入场景ID。
            scene_id=xxx,
            outer_order_no='商户请求的唯一标识',
            # 要接入的认证方案。
            product_code='LR_FR',
            model='LIVENESS',
            # 固定值IDENTITY_CARD。
            cert_type='IDENTITY_CARD',
            meta_info='{"zimVer":"3.0.0","appVersion": "1","bioMetaInfo": "4.1.0:11501568,0
","appName":"com.aliyun.antcloudauth","deviceType": "ios","osVersion": "iOS 10.3.2","apdidT
oken": "","deviceModel": "iPhone9,1"}',
            # cert_name='xxx',
            # cert_no='xxx',
            # mobile='130xxxxxxxx',
            # ip='114.xxx.xxx.xxx',
            user_id='12345xxxx',
            # callback_url='https://www.aliyundoc.com',
            # callback_token='xxxxxxx'
        )
        response = InitFaceVerify.init_face_verify_auto_route(request)
        # 不支持服务自动路由。
        # response = InitFaceVerify.init_face_verify('cloudauth.cn-shanghai.aliyuncs.com',
request)
        print(response.request_id)
        print(response.code)
        print(response.message)
        print(response.result_object.certify_id)
    @staticmethod
    def init_face_verify_auto_route(
            request: cloudauth_models.InitFaceVerifyRequest,
    ) -> cloudauth_models.InitFaceVerifyResponse:
```

```
        endpoints = [
            'cloudauth.cn-shanghai.aliyuncs.com',
            'cloudauth.cn-beijing.aliyuncs.com'
        ]
        for endpoint in endpoints:
            try:
                response = InitFaceVerify.init_face_verify(endpoint, request)
                if UtilClient.equal_string('500', response.code):
                    continue
                return response
            except Exception as err:
                # 网络异常，切换到下个区域调用。
                continue
        return None
    @staticmethod
    def init_face_verify(
            endpoint: str,
            request: cloudauth_models.InitFaceVerifyRequest,
    ) -> cloudauth_models.InitFaceVerifyResponse:
        client = InitFaceVerify.create_client(endpoint)
        # 创建RuntimeObject实例并设置运行参数。
        runtime = util_models.RuntimeOptions()
        runtime.read_timeout = 10000
        runtime.connect_timeout = 10000
        return client.init_face_verify(request, runtime)
if __name__ == '__main__':
    InitFaceVerify.main()
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
from alibabacloud_cloudauth20190307.client import Client as CloudauthClient
from alibabacloud_tea_openapi import models as open_api_models
from alibabacloud_cloudauth20190307 import models as cloudauth_models
from alibabacloud_tea_util.client import Client as UtilClient
from alibabacloud_tea_util import models as util_models
class DescribeFaceVerify:
    def __init__(self):
        pass
    @staticmethod
    def create_client(
        endpoint: str,
    ) -> CloudauthClient:
        config = open_api_models.Config(
            # 您的AccessKey ID。
            access_key_id='您的AccessKey ID',
            # 您的AccessKey Secret。
            access_key_secret='您的AccessKey Secret',
            # 设置HTTP代理。
            # http_proxy='http://xx.xx.xx.xx:xxxx',
            # 设置HTTPS代理。
            # https_proxy='https://xx.xx.xx.xx:xxxx',
            endpoint=endpoint
```

```
        )
        return CloudauthClient(config)
    @staticmethod
    def main() -> None:
        request = cloudauth_models.DescribeFaceVerifyRequest(
            # 请输入场景ID。
            scene_id=xxx,
            certify_id='xxx'
        )
        response = DescribeFaceVerify.describe_face_verify_auto_route(request)
        # 不支持服务自动路由。
        # response = DescribeFaceVerify.describe_face_verify('cloudauth.cn-shanghai.aliyunc
s.com', request)
        print(response.request_id)
        print(response.code)
        print(response.message)
        print(response.result_object.passed)
        print(response.result_object.sub_code)
        print(response.result_object.material_info)
    @staticmethod
    def describe_face_verify_auto_route(
        request: cloudauth_models.DescribeFaceVerifyRequest,
    ) -> cloudauth_models.DescribeFaceVerifyResponse:
        endpoints = [
            'cloudauth.cn-shanghai.aliyuncs.com',
            'cloudauth.cn-beijing.aliyuncs.com'
        ]
        for endpoint in endpoints:
            try:
                response = DescribeFaceVerify.describe_face_verify(endpoint, request)
                if UtilClient.equal_string('500', response.code):
                    continue
                return response
            except Exception as err:
                    continue
        return None
    @staticmethod
    def describe_face_verify(
        endpoint: str,
        request: cloudauth_models.DescribeFaceVerifyRequest,
    ) -> cloudauth_models.DescribeFaceVerifyResponse:
        client = DescribeFaceVerify.create_client(endpoint)
        # 创建RuntimeObject实例并设置运行参数。
        runtime = util_models.RuntimeOptions()
        runtime.read_timeout = 10000
        runtime.connect_timeout = 10000
        return client.describe_face_verify(request, runtime)
if __name__ == '__main__':
    DescribeFaceVerify.main()
```

4.1.6. Go SDK调用示例

本文介绍了金融级活体检测方案的Go SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 仅支持Go 1.12.0及以上版本。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

引入依赖包

```
go get github.com/alibabacloud-go/tea-rpc
go get github.com/alibabacloud-go/tea-utils
go get github.com/alibabacloud-go/tea
```

安装SDK

关于Go SDK源代码，请参见Go SDK包。

通过Go mod来管理项目依赖：

```
go get github.com/alibabacloud-go/cloudauth-20190307@vv1.0.7
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
package main
import (
  "fmt"
  cloudauth "github.com/alibabacloud-go/cloudauth-20190307/client"
  rpc "github.com/alibabacloud-go/tea-rpc/client"
  util "github.com/alibabacloud-go/tea-utils/service"
  "github.com/alibabacloud-go/tea/tea"
)
func CreateClient (endpoint *string) (_result *cloudauth.Client, _err error) {
  config := &rpc.Config{
    // 您的AccessKey ID。
    AccessKeyId: tea.String("您的AccessKey ID"),
    // 您的AccessKey Secret。
    AccessKeySecret: tea.String("您的AccessKey Secret"),
    // 设置HTTP代理。
    // HttpProxy: tea.String("http://xx.xx.xx.xx:xxxx"),
    // 设置HTTPS代理。
    // HttpsProxy: tea.String("https://username:password@xxx.xxx.xxx.xxx:9999"),
    Endpoint: endpoint,
  }
```

```
  _result = &cloudauth.Client{}
  _result, _err = cloudauth.NewClient(config)
  return _result, _err
}
func _main () (_err error) {
  request := &cloudauth.InitFaceVerifyRequest{
    // 固定值。
    ProductCode: tea.String("LR_FR"),
    // 请输入场景ID。
    SceneId: tea.Int64(xxx),
    OuterOrderNo: tea.String("商户请求的唯一标识"),
    Model: tea.String("LIVENESS"),
    // 固定值IDENTITY_CARD。
    CertType: tea.String("IDENTITY_CARD"),
    // CertName: tea.String("xxx"),
    // CertNo: tea.String("xxx"),
    // Ip: tea.String("用户IP地址"),
    UserId: tea.String("用户ID"),
    // Mobile: tea.String("130xxxxxxxx"),
    // CallbackUrl: tea.String("https://www.aliyundoc.com"),
    // CallbackToken: tea.String("xxxxxxx"),
    MetaInfo: tea.String("{\"zimVer\":\"3.0.0\",\"appVersion\": \"1\",\"bioMetaInfo\": \"4.
1.0:11501568,0\",\"appName\": \"com.aliyun.antcloudauth\",\"deviceType\": \"ios\",\"osVersi
on\": \"iOS 10.3.2\",\"apdidToken\": \"\",\"deviceModel\": \"iPhone9,1\"}")
  }
  response, _err := InitFaceVerifyAutoRoute(request)
  // 不支持服务自动路由。
  // response, _err := InitFaceVerify(tea.String("cloudauth.cn-shanghai.aliyuncs.com"), req
uest)
  if _err != nil {
    return _err
  }
  fmt.Println(*response.RequestId)
  fmt.Println(*response.Code)
  fmt.Println(*response.Message)
  fmt.Println(*response.RequestId)
  fmt.Println(*response.ResultObject.CertifyId)
  return _err
}
func InitFaceVerifyAutoRoute (request *cloudauth.InitFaceVerifyRequest) (_result *cloudauth
.InitFaceVerifyResponse, _err error) {
  endpoints := []*string{tea.String("cloudauth.cn-shanghai.aliyuncs.com"), tea.String("clou
dauth.cn-beijing.aliyuncs.com")}
  var lastResponse *cloudauth.InitFaceVerifyResponse
  for _, endpoint := range endpoints {
    response, _err := InitFaceVerify(endpoint, request)
    if _err != nil {
      var err = &tea.SDKError{}
      if _t, ok := _err.(*tea.SDKError); ok {
        err = _t
        // 网络异常，切换到下个区域调用。
        if tea.BoolValue(util.EqualString(tea.String("ServiceUnavailable"), err.Code)) {
          continue
        }
```

```
      }
      return _result, _err
    }
    if tea.BoolValue(util.EqualString(tea.String("500"), response.Code)) {
      continue
    }
    _result = response
    return _result, _err
  }
  _result = lastResponse
  return _result, _err
}
func InitFaceVerify (endpoint *string, request *cloudauth.InitFaceVerifyRequest) (_result *
cloudauth.InitFaceVerifyResponse, _err error) {
  client, _err := CreateClient(endpoint)
  if _err != nil {
    return _result, _err
  }
  // 创建RuntimeObject实例并设置运行参数。
  runtime := &util.RuntimeOptions{}
  runtime.ReadTimeout = tea.Int(10000)
  runtime.ConnectTimeout = tea.Int(10000)
  _result = &cloudauth.InitFaceVerifyResponse{}
  _body, _err := client.InitFaceVerify(request, runtime)
  if _err != nil {
    return _result, _err
  }
  _result = _body
  return _result, _err
}
func main() {
  err := _main()
  if err != nil {
    panic(err)
  }
}
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
package main
import (
  "fmt"
  cloudauth "github.com/alibabacloud-go/cloudauth-20190307/client"
  rpc "github.com/alibabacloud-go/tea-rpc/client"
  util "github.com/alibabacloud-go/tea-utils/service"
  "github.com/alibabacloud-go/tea/tea"
)
func CreateClient(endpoint *string) (_result *cloudauth.Client, _err error) {
  config := &rpc.Config{
    // 您的AccessKey ID。
    AccessKeyId: tea.String("您的AccessKey ID"),
    // 您的AccessKey Secret。
    AccessKeySecret: tea.String("您的AccessKey Secret"),
```

```
    AccessKeySecret: tea.String("您的AccessKey Secret"),
    Endpoint:        endpoint,
    // 设置HTTP代理。
    // HttpProxy: tea.String("http://xx.xx.xx.xx:xxxx"),
    // 设置HTTPS代理。
    // HttpsProxy: tea.String("https://xx.xx.xx.xx:xxxx"),
  }
  _result = &cloudauth.Client{}
  _result, _err = cloudauth.NewClient(config)
  return _result, _err
}
func _main() (_err error) {
  request := &cloudauth.DescribeFaceVerifyRequest{
    // 请输入场景ID。
    SceneId:   tea.Int64(xxx),
    CertifyId: tea.String("91707dc296d469ad38e4c5efa6a0xxxx"),
  }
  response, _err := DescribeFaceVerifyAutoRoute(request)
  // 不支持服务自动路由。
  // response, _err := DescribeFaceVerify(tea.String("cloudauth.cn-shanghai.aliyuncs.com"),
request)
  if _err != nil {
    return _err
  }
  if response != nil {
    fmt.Println(*response.RequestId)
    fmt.Println(*response.Code)
    fmt.Println(*response.Message)
    fmt.Println(*response.RequestId)
    fmt.Println(*response.ResultObject.Passed)
    fmt.Println(*response.ResultObject.SubCode)
    fmt.Println(*response.ResultObject.MaterialInfo)
    fmt.Println(*response.ResultObject.IdentityInfo)
    fmt.Println(*response.ResultObject.DeviceToken)
  }
  return _err
}
func DescribeFaceVerifyAutoRoute(request *cloudauth.DescribeFaceVerifyRequest) (_result *cl
oudauth.DescribeFaceVerifyResponse, _err error) {
  endpoints := []*string{tea.String("cloudauth.cn-shanghai.aliyuncs.com"), tea.String("clou
dauth.cn-beijing.aliyuncs.com")}
  var lastResponse *cloudauth.DescribeFaceVerifyResponse
  for _, endpoint := range endpoints {
    response, _err := DescribeFaceVerify(endpoint, request)
    if _err != nil {
      var err = &tea.SDKError{}
      if _t, ok := _err.(*tea.SDKError); ok {
        err = _t
        // 网络异常，切换到下个区域调用。
        if tea.BoolValue(util.EqualString(tea.String("ServiceUnavailable"), err.Code)) {
          continue
        }
      }
      return _result, _err
    }
```

```
    }
    if tea.BoolValue(util.EqualString(tea.String("500"), response.Code)) {
      continue
    }
    _result = response
    return _result, _err
  }
  _result = lastResponse
  return _result, _err
}
func DescribeFaceVerify(endpoint *string, request *cloudauth.DescribeFaceVerifyRequest) (_r
esult *cloudauth.DescribeFaceVerifyResponse, _err error) {
  client, _err := CreateClient(endpoint)
  if _err != nil {
    return _result, _err
  }
  // 创建RuntimeObject实例并设置运行参数。
  runtime := &util.RuntimeOptions{}
  runtime.ReadTimeout = tea.Int(10000)
  runtime.ConnectTimeout = tea.Int(10000)
  _body, _err := client.DescribeFaceVerify(request, runtime)
  if _err != nil {
    return _result, _err
  }
  _result = _body
  return _result, _err
}
func main() {
  err := _main()
  if err != nil {
    panic(err)
  }
}
```

4.1.7. .NET SDK调用示例

本文介绍了金融级活体检测方案的.NET SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的.NET Framework版本在4.5.0版本及以上或者.NET Standard版本在2.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于.NET SDK源代码，请参见.NET SDK包。

SDK有以下两种安装方式，请选择合适的方式安装：

- 通过.NET命令行安装。

```
dotnet add package AlibabaCloud.SDK.Cloudauth20190307 --version 1.0.7
```

- 使用包管理安装。

```
Install-Package AlibabaCloud.SDK.Cloudauth20190307 -Version 1.0.7
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
using System;
using System.Collections.Generic;
using Tea;
namespace InitFaceVerify
{
    public class Program
    {
        public static AlibabaCloud.SDK.Cloudauth20190307.Client CreateClient(string endpoin
t)
        {
            AlibabaCloud.RPCClient.Models.Config config = new AlibabaCloud.RPCClient.Models
.Config
            {
                // 您的AccessKey ID。
                AccessKeyId = "您的AccessKey ID",
                // 您的AccessKey Secret。
                AccessKeySecret = "您的AccessKey Secret",
                // 设置HTTP代理。
                // HttpProxy = "http://xx.xx.xx.xx:xxxx",
                // 设置HTTPS代理。
                // HttpsProxy = "https://username:password@xxx.xxx.xxx.xxx:9999",
                Endpoint = endpoint,
            };
            return new AlibabaCloud.SDK.Cloudauth20190307.Client(config);
        }
        public static void Main(string[] args)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest request = new A
libabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest
            {
                // 固定值。
                ProductCode = "LR_FR",
                // 请输入场景ID。
                SceneId = xxx,
                // 设置商户请求的唯一标识。
                OuterOrderNo = "xxxxx",
                // 模式。
                Model = "LIVENESS",
                // 固定值IDENTITY_CARD。
                CertType = "IDENTITY_CARD",
```

```
                //CertName = "xxx",
                //CertNo = "xxx",
                MetaInfo = "{\"zimVer\":\"3.0.0\",\"appVersion\": \"1\",\"bioMetaInfo\": \"
4.1.0:11501568,0\",\"appName\": \"com.aliyun.antcloudauth\",\"deviceType\": \"ios\",\"osVer
sion\": \"iOS 10.3.2\",\"apdidToken\": \"\",\"deviceModel\": \"iPhone9,1\"}",
                //Mobile = "130xxxxxxxx",
                //Ip = "",
                UserId = "用户ID",
                //CallbackUrl = "https://www.aliyundoc.com",
                //CallbackToken = "xxxxxxx"
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse response = Ini
tFaceVerifyAutoRoute(request);
            // 不支持服务自动路由。
            // AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse response =
InitFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", request);
            Console.WriteLine(response.RequestId);
            Console.WriteLine(response.Code);
            Console.WriteLine(response.Message);
            Console.WriteLine(response.ResultObject.CertifyId);
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse Init
FaceVerifyAutoRoute(AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest request
)
        {
            List<string> endpoints = new List<string>
            {
                "cloudauth.cn-shanghai.aliyuncs.com",
                "cloudauth.cn-beijing.aliyuncs.com"
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse lastResponse =
null;
            foreach (var endpoint in endpoints)
            {
                try
                {
                    AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse respon
se = InitFaceVerify(endpoint, request);
                    if (AlibabaCloud.TeaUtil.Common.EqualString("500", response.Code))
                    {
                        continue;
                    }
                    return response;
                }
                catch (TeaException err)
                {
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
                catch (Exception _err)
```

```
                {
                    TeaException err = new TeaException(new Dictionary<string, object>
                    {
                        { "message", _err.Message }
                    });
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
            }
            return lastResponse;
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyResponse Init
FaceVerify(string endpoint, AlibabaCloud.SDK.Cloudauth20190307.Models.InitFaceVerifyRequest
request)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Client client = CreateClient(endpoint);
            // 创建RuntimeObject实例并设置运行参数。
            AlibabaCloud.TeaUtil.Models.RuntimeOptions runtime = new AlibabaCloud.TeaUtil.M
odels.RuntimeOptions();
            runtime.ReadTimeout = 10000;
            runtime.ConnectTimeout = 10000;
            return client.InitFaceVerify(request, runtime);
        }
    }
}
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
using System;
using System.Collections.Generic;
using Tea;
namespace DescribeFaceVerify
{
    public class Program
    {
        public static AlibabaCloud.SDK.Cloudauth20190307.Client CreateClient(string endpoin
t)
        {
            AlibabaCloud.RPCClient.Models.Config config = new AlibabaCloud.RPCClient.Models
.Config
            {
                // 您的AccessKey ID。
                AccessKeyId = "您的AccessKey ID",
                // 您的AccessKey Secret。
                AccessKeySecret = "您的AccessKey Secret",
                // 设置HTTP代理。
                // HttpProxy = "http://xx.xx.xx.xx:xxxx",
                // 设置HTTPS代理。
```

```
                // HttpsProxy = "https://xx.xx.xx.xx:xxxx",
                Endpoint = endpoint,
            };
            return new AlibabaCloud.SDK.Cloudauth20190307.Client(config);
        }
        public static void Main(string[] args)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyRequest request = n
ew AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyRequest
            {
                // 请输入场景ID。
                SceneId = xxx,
                CertifyId = "91707dc296d469ad38e4c5efa6a0xxxx",
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse response =
DescribeFaceVerifyAutoRoute(request);
            // 不支持服务自动路由。
            // AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse respons
e = DescribeFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", request);
            Console.WriteLine(response.RequestId);
            Console.WriteLine(response.Code);
            Console.WriteLine(response.Message);
            Console.WriteLine(response.ResultObject.Passed);
            Console.WriteLine(response.ResultObject.SubCode);
            Console.WriteLine(response.ResultObject.MaterialInfo);
            Console.WriteLine(response.ResultObject.IdentityInfo);
            Console.WriteLine(response.ResultObject.DeviceToken);
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse
DescribeFaceVerifyAutoRoute(AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyReq
uest request)
        {
            List<string> endpoints = new List<string>
            {
                "cloudauth.cn-shanghai.aliyuncs.com",
                "cloudauth.cn-beijing.aliyuncs.com"
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse lastRespon
se = null;
            foreach (var endpoint in endpoints)
            {
                try
                {
                    AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse re
sponse = DescribeFaceVerify(endpoint, request);
                    if (AlibabaCloud.TeaUtil.Common.EqualString("500", response.Code))
                    {
                        continue;
                    }
                    return response;
                }
                catch (TeaException err)
                {
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
```

```
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
                catch (Exception _err)
                {
                    TeaException err = new TeaException(new Dictionary<string, object>
                    {
                        { "message", _err.Message }
                    });
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
            }
            return lastResponse;
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceVerifyResponse
DescribeFaceVerify(string endpoint, AlibabaCloud.SDK.Cloudauth20190307.Models.DescribeFaceV
erifyRequest request)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Client client = CreateClient(endpoint);
            // 创建RuntimeObject实例并设置运行参数。
            AlibabaCloud.TeaUtil.Models.RuntimeOptions runtime = new AlibabaCloud.TeaUtil.M
odels.RuntimeOptions();
            runtime.ReadTimeout = 10000;
            runtime.ConnectTimeout = 10000;
            return client.DescribeFaceVerify(request, runtime);
        }
    }
}
```

4.1.8. Node.js SDK调用示例

本文介绍了金融级活体检测方案的Node.js SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码，详细信息，请参见：

- 调试InitFaceVerify接口
- 调试DescribeFaceVerify接口

前提条件

- 需要使用的Node.js版本在8.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于Node.js SDK源代码，请参见Node.js SDK。

建议您使用npm管理依赖包：

```
npm install @alicloud/tea-util
npm install @alicloud/rpc-client
npm install @alicloud/cloudauth20190307@1.0.7
```

调用示例

InitFaceVerify接口调用示例

关于参数的具体描述，请参见InitFaceVerify。

```
const Config = require("@alicloud/rpc-client");
const CloudAuth = require("@alicloud/cloudauth20190307");
const CloudAuthClient = CloudAuth.default;
const {RuntimeOptions} = require("@alicloud/tea-util");
     function createClient(endpoint) {
        // 通过以下代码创建Config实例并初始化。
        const config = new Config.Config({
            // 您的AccessKey ID。
            accessKeyId: "您的AccessKey ID",
            // 您的AccessKey Secret。
            accessKeySecret: "您的AccessKey Secret",
            // 设置Endpoint。
            endpoint: endpoint,
            // 设置HTTP代理。
            // httpProxy: "http://xxx.xxx.xxx.xxx:9999",
            // 设置HTTPS代理。
            // httpsProxy: "https://username:password@xxx.xxx.xxx.xxx:9999",
        });
        return new CloudAuthClient(config)
    }
    async function initFaceVerify(endpoint, request) {
        const client = createClient(endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        const runtime = new RuntimeOptions([]);
        runtime.readTimeout=10000;
        runtime.connectTimeout=10000;
        return await client.initFaceVerify(request, runtime);
    }
    async function initFaceVerifyAutoRoute(request) {
        const endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        for (const endpoint of endpoints) {
            try {
                const response = await initFaceVerify(endpoint, request);
                if (response.code === '500') {
                    continue;
                }
```

```
                return response;
            } catch (err) {
                continue;
            }
        }
        return null;
    }
    async function main(){
        // 通过以下代码创建API请求并设置参数。
        const request = new CloudAuth.InitFaceVerifyRequest({
        // 请输入场景ID。
        sceneId: xxx,
        // 设置商户请求的唯一标识。
        outerOrderNo: "xxx",
        // 要接入的认证方案。
        productCode: "LR_FR",
        // 模式。
        model: "LIVENESS",
        // 固定值。
        certType: "IDENTITY_CARD",
        // certName: "张三",
        // certNo: "xxx",
        metaInfo: "{\"zimVer\":\"3.0.0\",\"appVersion\": \"1\",\"bioMetaInfo\": \"4.1.0:115
01568,0\",\n" +
            "        \"appName\": \"com.aliyun.antcloudauth\",\"deviceType\": \"ios\",\"osV
ersion\": \"iOS 10.3.2\",\"apdidToken\": \"\",\"deviceModel\": \"iPhone9,1\"}",
        // mobile: "130xxxxxxxx",
        // ip: "114.xxx.xxx.xxx",
        userId: "12345xxxx",
        // callbackUrl: "https://www.aliyundoc.com",
        // callbackToken: "xxxxxxx"
        });
        // 推荐，支持服务路由。
        const response = await initFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        // const response = await initFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", requ
est);
        console.log(response.requestId);
        console.log(response.code);
        console.log(response.message);
        console.log(response.resultObject.certifyId);
        }
main().then(function (response) {});
```

DescribeFaceVerify接口调用示例

关于参数的具体描述，请参见DescribeFaceVerify。

```
const Config = require("@alicloud/rpc-client");
const CloudAuth = require("@alicloud/cloudauth20190307");
const CloudAuthClient = CloudAuth.default;
const {RuntimeOptions} = require("@alicloud/tea-util");
     function createClient(endpoint) {
        // 通过以下代码创建Config实例并初始化。
```

```
        const config = new Config.Config({
            // 您的AccessKey ID。
            accessKeyId: "您的AccessKey ID",
            // 您的AccessKey Secret。
            accessKeySecret: "您的AccessKey Secret",
            // 设置Endpoint。
            endpoint: endpoint,
            // 设置HTTP代理。
            // httpProxy: "http://xxx.xxx.xxx.xxx:9999",
            // 设置HTTPS代理。
            // httpsProxy: "https://username:password@xxx.xxx.xxx.xxx:9999",
        });
        return new CloudAuthClient(config)
    }
    async function describeFaceVerify(endpoint, request) {
        const client = createClient(endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        const runtime = new RuntimeOptions([]);
        runtime.readTimeout=10000;
        runtime.connectTimeout=10000;
        return await client.describeFaceVerify(request, runtime);
    }
    async function describeFaceVerifyAutoRoute(request) {
        const endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        for (const endpoint of endpoints) {
            try {
                const response = await describeFaceVerify(endpoint, request);
                if (response.code === '500') {
                    continue;
                }
                return response;
            } catch (err) {
                continue;
            }
        }
        return null;
    }
    async function main(){
        // 通过以下代码创建API请求并设置参数。
        const request = new CloudAuth.DescribeFaceVerifyRequest({
            sceneId: xxx,
            certifyId: "xxx"
        });
        // 推荐，支持服务路由。
        const response = await describeFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        //  const response = await describeFaceVerify("cloudauth.cn-shanghai.aliyuncs.com",
request);
        console.log(response.requestId);
        console.log(response.code);
        console.log(response.message);
        console.log(response.resultObject.passed);
```

```
        console.log(response.resultObject.passed);
        console.log(response.resultObject.identityInfo);
        console.log(response.resultObject.materialInfo);
        console.log(response.resultObject.deviceToken);
        console.log(response.resultObject.subCode);
        }
main().then(function (response) {});
```

4.2. App（SDK）接入

4.2.1. 接入流程

无线SDK和服务端接入适用于您自身已有手机App应用，且希望通过该App对用户进行线上认证的场景。

前置条件

您必须了解通过HTTP、HTTPS原生调用金融级实人认证服务接口时，构造请求签名的方法。更多签名机制的信息，请参见API签名机制。

操作步骤

1. 开通金融级实人认证服务。
2. 添加认证场景。

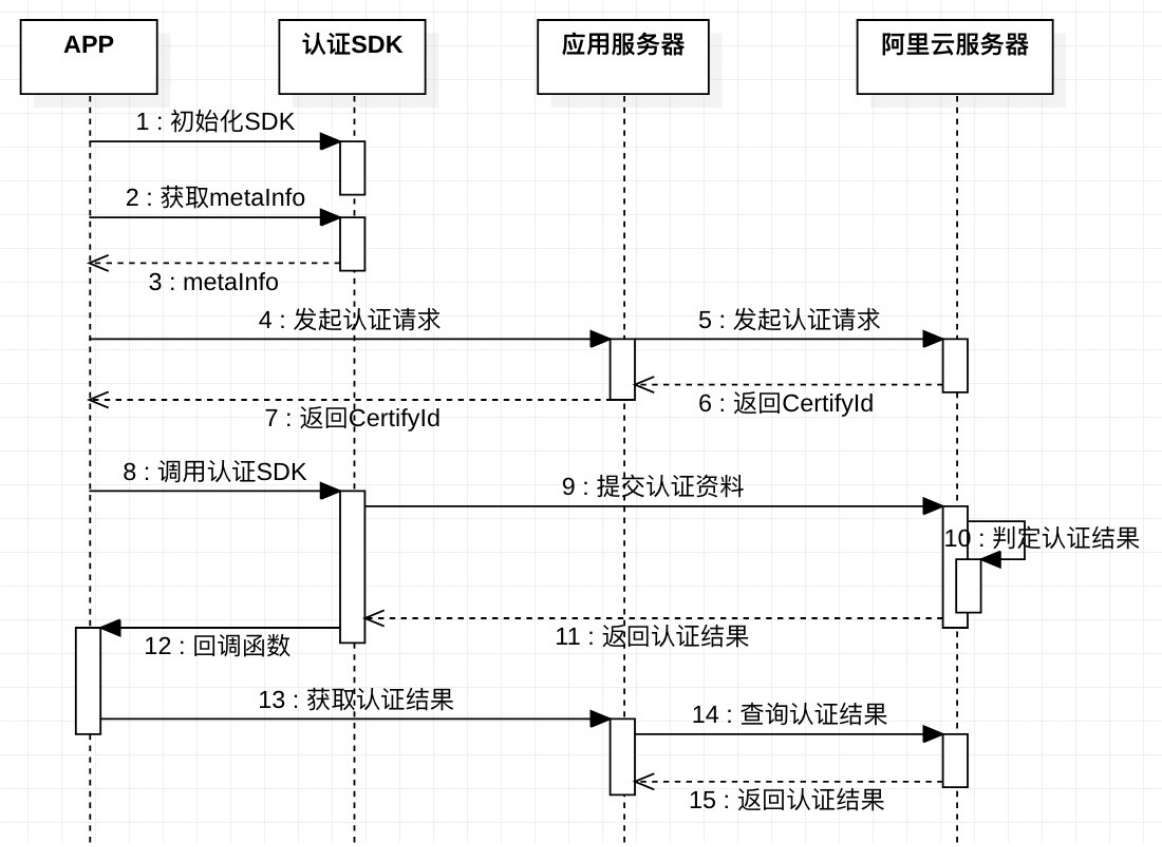
   > **说明** 基于不同业务场景，您可以在添加认证场景时，手动配置是否授权阿里云访问指定OSS存储空间用于存储认证的资料图片。

3. 如果需要将认证过程中的人脸采集照片存储到OSS空间，您需要授权金融级实人认证访问OSS存储空间。
4. 接入活体检测方案。

   在您完成上述步骤后，您可以进行无线SDK+服务端接入，接入步骤如下：

   i. 调用服务端初始化接口。
   ii. 调用接入客户端SDK。
   iii. 调用服务端查询接口。

   您可以查看下方调用时序图了解详细接入步骤。



系统调用时序说明表

序号	说明
1	初始化认证SDK。
2	App调用认证SDK获取MetaInfo。
3	认证SDK返回MetaInfo。
4	App向应用服务端发起认证请求并传递MetaInfo。
5	应用服务端向阿里云服务端调用发起认证请求接口。
6	阿里云服务端返回CertifyId给应用服务端。
7	应用服务端将CertifyId传递给App。
8	App调用认证SDK。
9	认证SDK将用户提交的认证资料传递给阿里云服务端。
10	阿里云服务端根据资料判定认证结果。
11	阿里云服务端将认证结果返回给SDK。
12	SDK通过回调函数指引App获取认证状态。

序号	说明
13	App向应用服务端查询认证状态。
14	应用服务端向阿里云服务端调用查询认证结果接口。
15	阿里云服务端将认证结果返回给应用服务端。

在您了解接入步骤后，您可以根据以下方式接入活体检测方案。

- 进行服务端接入，请参见InitFaceVerify。
- 在应用客户端集成SDK，请参见以下文档：
  - Android接入
  - iOS接入
  - UniApp接入

4.2.2. Android接入

金融级实人认证服务提供Android客户端SDK，帮助您在业务应用（App）中实现刷脸认证功能。您可以通过应用服务端认证初始化接口，向阿里云服务器调用发起认证请求接口并获取刷脸认证的唯一标识CertifyId，然后使用CertifyId唤起刷脸认证客户端SDK。本文结合示例代码介绍Android客户端的接入流程。

使用限制

金融级实人认证服务Android客户端SDK具有以下使用限制。

- 不支持模拟器模式调试。
- 仅支持Android 4.4及以上系统版本的移动智能设备（手机或Pad）接入。

依赖配置

下载金融级实人认证Android SDK，该SDK为Android标准AAR包，下载之后在*Android Studio*中进行AAR包引用。例如下载完AAR包之后，放入工程目录的*libs*目录下，并添加以下依赖：

```
# 添加阿里云实人认证SDK依赖包。
implementation files('libs/android-aliyunface-sdk-release-xxx.aar')
implementation files('libs/android-aliyunbasicstl-sdk-release-xxx.aar')
implementation files('libs/android-aliyuncomm-sdk-release-xxx.aar')
implementation files('libs/android-aliyunocr-sdk-release-xxx.aar')
implementation files('libs/photinus-1.0.1.211130153658.aar')
implementation files('libs/Android-AliyunDevice-xxx.aar')
implementation files('libs/deviceid-release-xxx.aar')
implementation files('libs/tygerservice-xxx.aar')
implementation 'com.squareup.okhttp3:okhttp:3.11.0'
implementation 'com.squareup.okio:okio:1.14.0'
implementation 'com.alibaba:fastjson:1.2.62'
implementation 'com.aliyun.dpa:oss-android-sdk:+'
```

说明 以上代码示例中xxx表示版本号，使用过程中，请替换为SDK包中文件的实际版本号。

接口定义

```
/**
 * 刷脸认证接口。
 */
public class ZIMFacade {
    /**
     * 初始化接口，建议接入方在调用其它接口前，先调用此接口。
     *
     * @context Context
     */
    public static void install(Context context);
    /**
     * 获取环境参数接口。
     *
     * @context Context
     */
    public static String getMetaInfos(Context context);
    /**
     * identify interface for android
     * 需在UI线程上调用，不会阻塞调用线程。
     *
     * @param certifyId：刷脸认证唯一标识，请从刷脸认证服务端认证初始化接口获取。
     * @param useMsgBox：当刷脸认证过程中出现异常情况，是否使用SDK内部的弹框提示。
     *                    如果取值为true，则SDK先弹框提示，确定之后，返回错误code。
     *                    如果取值为false，则不弹提示框，直接返回错误code，客户业务应用App自行决定
如何提示。
     * @param callback：认证结果的回调接口。
     */
    public void verify(String certifyId, boolean useMsgBox,, ZIMCallback callback);
    /**
     * 开始进行实名认证，带自定义参数的版本。
     * @param certifyId：刷脸认证唯一标识，请从刷脸认证服务端认证初始化接口获取。
     * @param useMsgBox：表示是否在SDK内部使用弹框提示错误信息。
     * @param extParams：扩展参数。
     * @param callback：identify result
     */
    public void verify(final String zimId, boolean useMsgBox, HashMap<String, String> extPa
rams, ZIMCallback callback);
    /**
     * 获取设备指纹DeviceToken。
     * @return
     */
    public ZIMSession getSession();
}
```

接口调用

1. 初始化SDK。

   为了提高身份核验的用户体验，并为刷脸认证准备必要的数据，Android客户端需要进行SDK初始化。初始化的代码示例如下：

```
ZIMFacade.install(context);
```

2. 获取MetaInfo数据。

   您在调用刷脸认证服务端发起认证请求时，需要传入该MetaInfo值。代码示例如下：

```
String metaInfos = ZIMFacade.getMetaInfos(context);
```

3. 开始认证。

   Android客户端接入支持接入手机端和Pad端两种场景，请您根据自身业务需要选择接入的场景。两种接入场景的示例代码如下：

   - 手机端接入场景（自定义参数设置请参考Pad端设置）

```
ZIMFacade zimFacade = ZIMFacadeBuilder.create(MainActivity.this);
zimFacade.verify(certifyId, true, new ZIMCallback() {
 @Override
 public boolean response(ZIMResponse response) {
     // TODO: 根据实人认证回调结果处理自身的业务。
     if (null != response && 1000 == response.code) {
         // 认证成功。
     } else {
         // 认证失败。
     }
     return true;
 }
}
```

   - 手机端+Pad端接入场景（支持自定义参数）

> **说明** 金融级实人认证支持设置返回活体检测的视频，默认设置为不返回。如果设置为返回，在您使用金融级实人认证SDK完成认证（不区分认证结果为F或T），并在SDK回调时和查询接口返回视频路径，请您及时处理。

```
ZIMFacade zimFacade = ZIMFacadeBuilder.create(MainActivity.this);
HashMap<String, String> extParams = new HashMap<>();
// 如需指定活体检测UI界面方向（横屏+竖屏），请指定这一项。
extParams.put(ZIMFacade.ZIM_EXT_PARAMS_KEY_SCREEN_ORIENTATION, ZIMFacade.ZIM_EXT_PARA
MS_VAL_SCREEN_LAND);
// 如需支持活体视频返回，请指定这一项，并在response.videoFilePath或在服务端查询接口获取视频本
地路径。
extParams.put(ZIMFacade.ZIM_EXT_PARAMS_KEY_USE_VIDEO, ZIMFacade.ZIM_EXT_PARAMS_VAL_US
E_VIDEO_TRUE);
// 如需设置OCR的"下一步"按钮颜色（默认可不设置），请设置此项，如红色 #FF0000。
extParams.put(ZIMFacade.ZIM_EXT_PARAMS_KEY_OCR_BOTTOM_BUTTON_COLOR, "#FF0000");
// 如需自定义活体检测页面的进度条颜色（默认可不设置），请设置此项，如红色 #FF0000。
extParams.put(ZIMFacade.ZIM_EXT_PARAMS_KEY_FACE_PROGRESS_COLOR, "#FF0000");
zimFacade.verify(certifyId, true, extParams, new ZIMCallback() {
    @Override
    public boolean response(final ZIMResponse response) {
        if (1000 == response.code) {
            showMessageBox("认证通过");
        } else {
            showMessageBox("认证失败([" + response.code + "]" + response.reason + ")")
;
        }
        return true;
    }
});
```

说明

- certifyId：刷脸认证唯一标识，可从刷脸认证服务端发起认证请求获取。
- ZIMCallback的response回调需要有返回值，默认为true。

4. 获取设备指纹。

   获取设备指纹（deviceToken），然后通过后端接口查询设备风险。代码示例如下：

```
ZIMSession deviceSession = zimFacade.getSession();
 if(deviceSession.code == 0){
            // 获取session成功，可以通过后端接口查询风险标签。
 }else{
            // 获取session失败，不能通过后端接口查询到有效风险标签。
 }
```

5. 返回刷脸认证回调结果。

   刷脸认SDK回调结果中的ZIMResponse类，该类中定义了对应的结果编码和原因，代码示例如下：

```
/**
 * 刷脸认证回调结果。
 */
public class ZIMResponse {
    /**
     * 返回结果编码。
     */
    public int code;
    /**
     * 返回结果原因信息。
     */
    public String reason;
}
```

返回结果说明

错误码	错误码文案	错误码描述
1000	刷脸成功	代表刷脸成功，该结果仅供参考，可通过服务端查询认证结果获取最终认证结果。
1001	系统错误	表示系统错误。
1003	验证中断	表示验证中断。
2002	网络错误	表示网络错误。
2003	客户端设备时间错误	表示客户端设备时间错误。
2006	刷脸失败	表示刷脸失败，如需获取更详细的失败原因，需调用服务端查询认证结果。

说明 更多错误码信息，请参见Android客户端错误码详情。

接口混淆配置

为避免接口混淆，您可以参考以下代码来保留类名（一般是在封装release包的时候保留类名）：

```
-verbose
-keep class com.aliyun.aliyunface.network.model.** {*;}
-keep class com.aliyun.aliyunface.api.ZIMCallback {*;}
-keep class com.aliyun.aliyunface.api.ZIMFacade {*;}
-keep class com.aliyun.aliyunface.api.ZIMFacadeBuilder {*;}
-keep class com.aliyun.aliyunface.api.ZIMMetaInfo {*;}
-keep class com.aliyun.aliyunface.api.ZIMResponse {*;}
-keep class com.aliyun.aliyunface.api.ZIMSession {*;}
-keep class com.aliyun.aliyunface.config.**{*;}
-keep class com.aliyun.aliyunface.log.RecordBase {*;}
-keep class com.aliyun.aliyunface.ui.ToygerWebView {*;}
-keep class com.alipay.zoloz.toyger.**{*;}
-keep class com.alipay.zoloz.image.** {*;}
-keep class com.alipay.android.** {*;}
-keep class net.security.device.api.** {*;}
-keep class com.alipay.deviceid.** { *; }
-keep class com.alibaba.fastjson.** {*;}
-keep class com.alibaba.sdk.android.oss.** { *; }
-dontwarn okio.**
-dontwarn org.apache.commons.codec.binary.**
-keepclassmembers,allowobfuscation class * {
     @com.alibaba.fastjson.annotation.JSONField <fields>;
}
```

权限说明

金融级实人认证为您提供安全的全链路身份核验服务。为了达到更好的设备风险防范效果，建议您在使用业务应用App调用刷脸SDK之前，提前向您的用户获取如下权限。

```
// 授权获取手机状态的权限。
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
// 授权获取接入网络的权限。开发人员需注意，6.0以上系统属于动态权限。
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Demo代码包

您可以下载Android Demo代码包（aliyunfaceDemo.zip）进行体验。

SDK包大小裁剪说明

如果您不需要使用OCR功能，可以在SDK包中删除*android-aliyunocr-sdk-release-xxxx_version.aar*，并重新编译。

4.2.3. iOS接入

刷脸认证提供iOS客户端SDK，帮助您在您的业务应用（App）中实现刷脸认证功能。您可通过刷脸认证的服务端认证初始化接口，获取刷脸认证唯一标识CertifyId，并使用CertifyId唤起刷脸认证客户端SDK。本文将结合示例代码进行iOS客户端接入的详细说明。

前提条件

- 刷脸认证应用必须运行在iOS 9.0及以上版本的系统上。

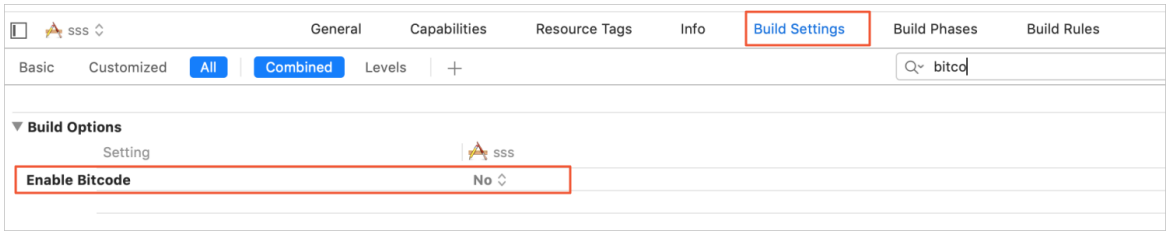
- 您的业务应用必须采用Objective C++或Swift集成金融级实人认证服务。
- 您的业务应用中未集成支付宝mPaaS SDK。

> **说明** iOS接入不支持模拟器模式调试。

配置开发环境

1. 在*info.plist*中配置摄像头权限请求。

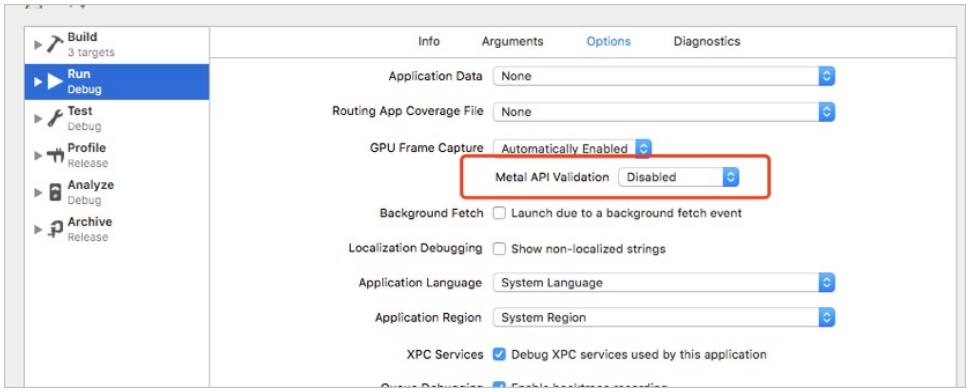
2. 在Xcode的编译设置中关闭Enable Bitcode选项。



3. 在Xcode编译设置的Linking > Other Linker Flags中，添加设置-ObjC -framework "BioAuthAPI" -lxml2。

   如果您的工程已设置了-force_load选项，则需要加入-force_load <framework path>/AliyunOSSiOS。

4. 在Xcode调试设置的Edit Scheme > Run > Options中禁用Metal API Validation。



配置依赖

1. 下载AliyunFace iOS SDK。

   该SDK为framework包。您需要在Xcode添加Link Binary With Libraries、SDK中的15个包和额外系统库依赖。

2. 解压下载的SDK包，并在解压的文件夹中执行以下命令提取所有的framework到Products目录。

```
for i in $(ls *.tgz);do tar xvf $i;done
```

- SDK中的包：

```
APPSecuritySDK
MPRemoteLogging
AliyunOSSiOS
BioAuthEngine
ZolozUtility
AliyunIdentityManager
APBToygerFacade
BioAuthAPI
ToygerService
ZolozSensorServices
ZolozOpenPlatformBuild
deviceiOS
ZolozMobileRPC
ZolozIdentityManager
OCRDetectSDKForTech
ToygerNative
```

- 系统库依赖：

```
CoreGraphics.framework
Accelerate.framework
SystemConfiguration.framework
AssetsLibrary.framework
CoreTelephony.framework
QuartzCore.framework
CoreFoundation.framework
CoreLocation.framework
ImageIO.framework
CoreMedia.framework
CoreMotion.framework
AVFoundation.framework
WebKit.framework
libresolv.tbd
libz.tbd
libc++.tbd
libc++.1.tbd
libc++abi.tbd
AudioToolbox.framework
CFNetwork.framework
MobileCoreServices.framework
libz.1.2.8.tbd
AdSupport.framework
```

拷贝资源文件

选择TARGETS，单击Build Phases页签，在Copy Bundle Resources中添加以下4个bundle：

- **APBToygerFacade.bundle**：位于*APBToygerFacade.framework*中。
- **ToygerNative.bundle**：位于*ToygerNative.framework*中。
- **OCRXMedia.bundle**：位于*OCRDetectSDKForTech.framework*中。
- **BioAuthEngine.bundle**：位于*BioAuthEngine.framework*中。

调用SDK

1. 引入头文件。

   命令如下：

```
#import <AliyunIdentityManager/AliyunIdentityPublicApi.h>
```

2. 初始化SDK。

   > **说明** 为了提升用户体验并为刷脸认证准备一些必备的数据，iOS客户端需要初始化SDK。您可以在调用 `appDelegate` 函数时（如下例代码所示）放置初始化SDK代码。您也可以在获取metaInfo数据前（如步骤3所示）放置初始化SDK代码。

   `appDelegate` 函数调用，示例代码：

```
(BOOL)application:(UIApplication)application didFinishLaunchingWithOptions:(NSDictionary)launchOptions {
[AliyunSdk init];// 初始化SDK。
}
```

3. 获取metaInfo数据。

   命令如下：

```
[AliyunSdk init];// 初始化SDK，如果已经在appDelegate函数中调用过，此处无需再重复调用。
[AliyunIdentityManager getMetaInfo];
```

   > **说明** 返回值类型是NSDictionary，您的服务端需要将其转换为JSON字符串。调用服务端发起认证请求接口时，在入参metaInfo中传入该值。关于发起认证请求的具体操作，请参见发起认证请求。

4. （可选）获取设备指纹。

   命令如下：

```
/*获取设备指纹token。
1，SDK初始化后需要经过一段时间计算才能获取到deviceToken，建议间隔3秒以上。
2，deviceToken短期有效，不能长期缓存使用，不能作为设备唯一标识。
*/
NSString * astring = [AliyunSdk getMobileSession];
```

   > **说明** 调用 `[AliyunSdk init]` 后，才能获取到该结果。

5. 调用SDK开始认证。

   命令如下：

```
[[AliyunIdentityManager sharedInstance] verifyWith:certifyId extParams:extParams onCompletion:^(ZIMResponse *response) {}];
```

参数说明如下表所示。

名称	类型	描述
certifyId	NSString	您可以通过服务端发起申请请求接口获取该参数。关于发起认证请求的具体操作，请参见发起认证请求。

名称	类型	描述

extParams	NSDictionary	通过以下代码传入当前viewController，用于展现网络加载和presentViewController。
		`[extParams setValue:self forKey:@"currentCtr"];`
		○ 如需设置OCR的**下一步**按钮的颜色，在extParams中添加按钮的正常颜色和按钮按下时的颜色。代码示例如下：
		`[extParams setValue:@"00FF00" forKey:ZIM_EXT_PARAMS_KEY_OCR_BOTTOM_BUTTON_COLOR];`
		`[extParams setValue:@"0000FF" forKey:PARAMS_KEY_OCR_BOTTOM_BUTTON_CLICKED_COLZIM_EXT_OR];`
		○ 如需设置扫脸时的圆圈颜色，在extParams中添加圆圈颜色。代码示例如下：
		`[extParams setValue:@"FF0000" forKey:ZIM_EXT_PARAMS_KEY_OCR_FACE_CIRCLE_COLOR];`
		○ 如需返回本地视频，在extParams中添加 `returnVideo` ，并且设置值为 `true` 。代码示例如下：
		`[extParams setValue:@"true" forKey:@"returnVideo"];`
		○ 如需服务端查询接口返回视频，需要额外在extParams中添加 `ZIM_EXT_PARAMS_KEY_USE_VIDEO_UPLOAD` ，并且设置值为 `true` 。代码示例如下：
		`[extParams setValue:@"true" forKey:ZIM_EXT_PARAMS_KEY_USE_VIDEO_UPLOAD]`
		说明 金融级实人认证支持设置返回活体检测的视频，默认设置为不返回。如果设置为返回，在您使用金融级实人认证SDK完成认证（不区分认证结果为F或T），并在SDK回调和查询接口时返回视频路径，请您及时处理。
		○ 如需自定义X号（页面关闭按钮），您可以参照以下步骤：
		■ 如需自定义X号的图片，需要额外在extParams中添加cancelIImage，并设置值为UIImage。代码示例如下：
		`[extParams setValue:UIImage forKey:@"cancelImage"];`
		■ 如需自定义X号的图片位置，需要额外在extParams中添加imageOrientation，并设置值为right或left 。默认为left。代码示例如下：
		`[extParams setValue:@"right" forKey:@"imageOrientation"];`

返回结果说明：

```
[[AliyunIdentityManager sharedInstance] verifyWith:certifyId extParams:extParams onCompletion:^(ZIMResponse *response) {}];
```

`response.code` 包含以下返回参数，具体说明如下表所示。

错误码	错误码文案	错误码描述
1000（ZIMResponseSuccess）	刷脸成功	表示刷脸成功。 说明 该结果仅供参考，可通过服务端查询认证结果接口获取最终认证结果。关于发起认证请求的具体操作，请参见发起认证请求。
1001（ZIMInternalError）	系统错误	表示系统错误。
1003（ZIMInterrupt）	验证中断	表示验证中断。
2002（ZIMNetworkfail）	网络错误	表示网络错误。
2003（ZIMTIMEError）	客户端设备时间错误	表示客户端设备时间错误。
2006（ZIMResponseFail）	刷脸失败	表示刷脸失败。 说明 如需获取失败原因的详细信息，需调用服务端查询认证结果接口。关于发起认证请求的具体操作，请参见发起认证请求。

说明 关于错误码的更多信息，请参见iOS客户端错误码详情。

代码示例：

```
[[AliyunIdentityManager sharedInstance] verifyWith:certifyId extParams:extParams onComp
letion:^(ZIMResponse *response) {
        dispatch_async(dispatch_get_main_queue(), ^{
            NSString *title = @"刷脸成功";
            switch (response.code) {
                case ZIMResponseSuccess: //1000。
                    break;
                case ZIMInterrupt:   //1003。
                    title = @"用户退出";
                    break;
                case ZIMNetworkfail: //2002。
                    title = @"网络错误";
                    break;
                case ZIMTIMEError: //2003。
                    title = @"设备时间设置不对";
                    break;
                case ZIMResponseFail: //2006。
                    title = @"刷脸失败";
                    break;
                case ZIMInternalError:   //1001。
                    title = @"初始化失败";
                    break;
                default:
                    break;
            }
```

Demo代码包

您可以下载以下iOS Demo代码包进行体验。在使用过程中，请替换iOS Demo代码包中的文件的版本为最新的版本：

- Objective-C Demo
- Swift Demo

SDK包大小裁剪说明

当您对SDK包大小有限制时，可以根据如下情况进行裁剪：

- 如果您之前没有使用过支付宝开放平台版，或开放平台版代码已经下线，可以在SDK包中删除ZolozIdentityManager、ZolozMobileRPC内容。
- 如果您不需要使用OCR功能，可以在SDK包中删除OCRDetectSDKForTech内容，并删除对应的bundle（OCRXMedia.bundle）。

4.2.4. UniApp接入

金融级实人认证服务提供UniApp插件，帮助您在您的业务应用（App）中实现刷脸认证功能。您可以通过您的应用服务端认证初始化接口，向阿里云服务器调用发起认证请求接口并获取刷脸认证的唯一标识CertifyId，然后使用CertifyId唤起刷脸认证客户端SDK。本文介绍了UniApp客户端接入的操作方法。

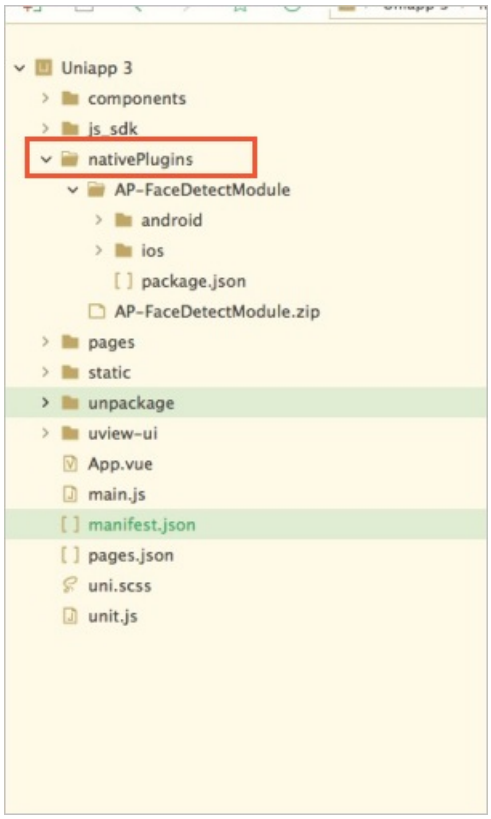
接入前必读

近期发现有个人开发者封装阿里云实人认证SDK后开发了第三方插件，并在某平台上提供付费下载服务。阿里云提醒您，如果您使用了此类非阿里云官方提供的插件或服务，阿里云概不对由此造成的一切后果负责，亦不承担任何法律责任。建议您使用阿里云官方SDK服务。

配置依赖

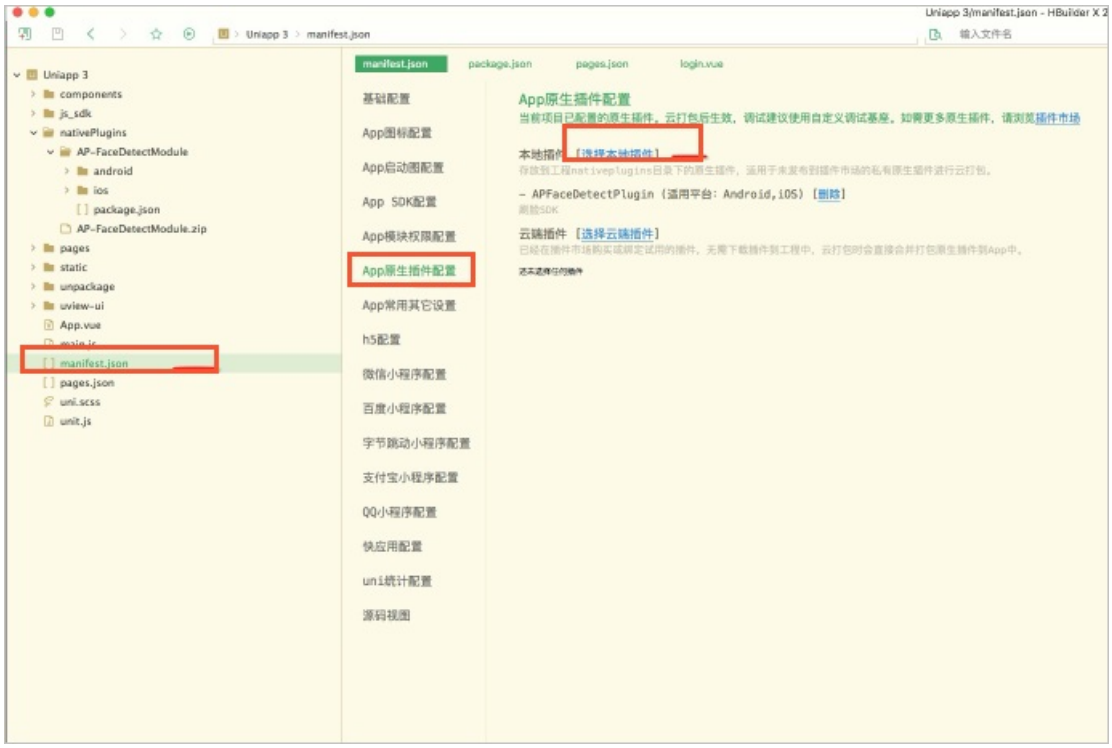
您可以参见DCloud官网文档接入UniApp插件，也可以参考以下步骤接入UniApp插件。

1. 下载金融级实人认证UniApp SDK，解压到项目中的*nativePlugins*目录下。



2. 导入本地插件。
   i. 在*manifest.json*文件下，单击**App原生插件配置**。

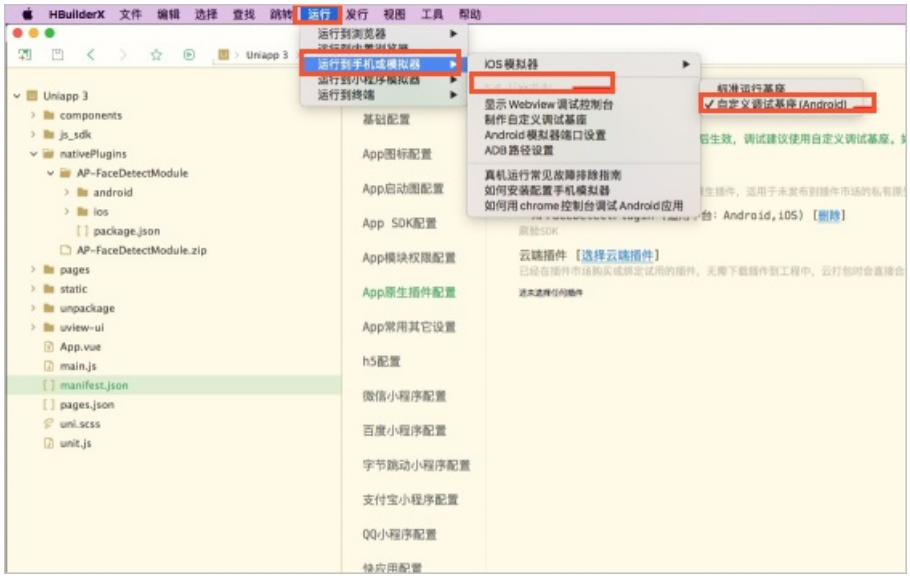
ii. 在**App原生插件配置**区域，单击**选择本地插件**。



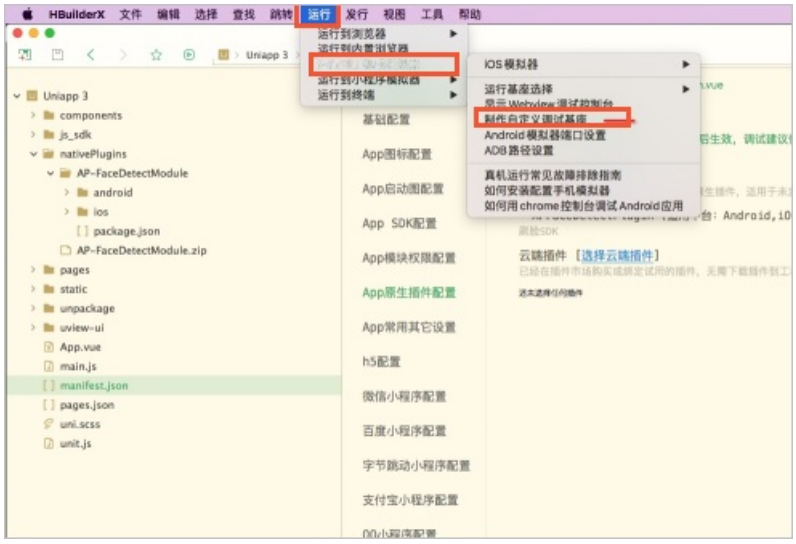
iii. 在**本地插件选择**对话框，选择需要打包生效的插件。

3. 配置基座包。

i. 设置自定义调试基座。在*manifest.json*文件，选择**运行 > 运行到手机或模拟器 > 运行基座选择 > 自定义调试基座(Android)**。



ii. 制作自定义调试基座。在*manifest.json*文件，单击**运行 > 运行到手机或模拟器**，选择**制作自定义调试基座**。



4. 调用实人认证服务。

   i. 获取metaInfo数据。

   您在调用刷脸认证服务端发起认证请求时需要传入该值。具体操作，请参见发起认证请求。

   ii. 调用**verify**接口，传入certifyId进行认证。

   参数说明如下表所示。

名称	类型	描述
certifyId	String	认证的唯一标识。
returnVideo	String	是否返回视频（iOS）。取值： ■ **true**：返回视频。 ■ **false**（默认）：不返回视频。
ext_params_key_use_video	String	是否返回视频（Android）。取值： ■ **true**：返回视频。 ■ **false**（默认）：不返回视频。
ext_params_key_ocr_bottom_button_color	String	OCR底部按钮的默认颜色。
ext_params_key_ocr_bottom_button_clicked_color	String	OCR底部按钮被操作时的颜色。
ext_params_key_ocr_face_circle_color	String	实人认证进度条颜色。

接入示例：

```
<template>
</template>
<script>
const n = uni.requireNativePlugin('AP-FaceDetectModule');
export default {
    methods: {
        verifyC: function() {
            n.verify(
                {
                    certifyId: this.certify
                },
                function(t) {
                    uni.showToast({
                        title: '返回的内容' + t,
                        icon: 'none'
                    });
                }
            );
        },
        //获取环境参数接口。
        getMetaInfo: function() {
            var t = n.getMetaInfo();
        },
    }
};
</script>
```

错误码说明

错误码	错误码文案	错误码描述
1000	刷脸成功	表示刷脸成功。 **说明** 该结果仅供参考。您可通过服务端查询认证结果接口获取最终认证结果。具体操作，请参见InitFaceVerify。
1001	系统错误	表示系统错误。
1003	验证中断	表示验证中断。
2002	网络错误	表示网络错误。
2003	客户端设备时间错误	表示客户端设备时间错误。

错误码	错误码文案	错误码描述
2006	刷脸失败	表示刷脸失败。 **说明** 如需获取失败原因的详细信息，需调用服务端查询认证结果接口获取。具体操作，请参见InitFaceVerify。

4.2.5. APICloud接入

金融级实人认证服务提供APICloud插件，帮助您在业务应用（App）中实现刷脸认证功能。您可以通过应用服务端认证初始化接口，向阿里云服务器调用发起认证请求接口并获取刷脸认证的唯一标识CertifyId，然后使用CertifyId唤起刷脸认证客户端SDK。本文介绍了APICloud客户端接入的操作方法。

背景信息

低代码平台APICloud的模块库提供多种模块。您可以使用已有模块，在应用中快速部署集成，缩短开发流程。当已有模块无法满足特殊需求时，您也可以自定义相关模块。

接入前必读

近期发现有个人开发者封装阿里云实人认证SDK后开发了第三方插件，并在某平台上提供付费下载服务。阿里云提醒您，如果您使用了此类非阿里云官方提供的插件或服务，阿里云概不对由此造成的一切后果负责，亦不承担任何法律责任。建议您使用阿里云官方SDK服务。

前提条件

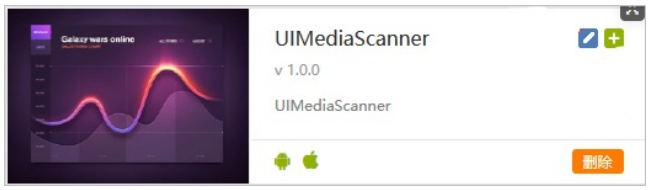
已在APICloud官网中创建应用。

配置依赖

1. 下载金融级实人认证APICloud SDK。
2. 在APICloud控制台上传刷脸认证APICloud模块。
   i. 登录APICloud控制台。
   ii. 在左侧导航栏，选择**端开发 > 模块**。
   iii. 在**模块**页签，选择**自定义模块**。
   iv. 在**自定义模块**页签，输入有效的**模块名称**、**模块概要**及版本号，选择并上传已下载的APICloud模块。

   v. 单击**保存**。
3. 将APICloud模块添加到应用中。
   i. 在**模块**页签，选择**模块库**。

ii. 在**模块库**页签，单击已上传APICloud模块右上角的图标，将APICloud模块添加到应用中。



> **说明** 自定义模块仅对自己的账号可见。如果需要对外开放，您可以在**模块管理**页面提交该模块，官方审核通过后将会发布到模块库。

当您需要更新已上传的自定义模块时，可以通过已上传自定义模块右上角的图标，重新上传自定义模块包，然后单击**保存**。

函数说明

APICloud插件的接入流程包含获取MetaInfo和开始认证函数。具体说明如下。

- **获取MetaInfo数据**

  调用getMetaInfo函数获取当前移动设备端的环境信息，并将这些信息发送到业务服务器端。业务服务器端会将这些信息作为参数之一（MetaInfo）调用服务端初始化认证接口（InitSmartVerify），从而获取CertifyId，用于后续验证环节。

  示例代码：

```
var aliyunFace = api.require('aliyunFaceVerifyModule');
aliyunFace.getMetaInfo(function(ret, err) {
    alert(ret);
});
```

- **开始认证**

  调用Verify函数发起金融级实人认证。

  参数说明：

名称	类型	说明
certifyId	String	从服务端初始化认证接口（InitSmartVerify）获取的CertifyId。 **说明** 每个CertifyId只能调用一次**Verify**函数，每次调用**Verify**函数之前都必须重新获取CertifyId。
extParams	Map<String, String>	用户自定义参数，可以为空。有关目前支持自定义的字段，请参见用户自定义参数extParams支持的字段说明。

名称	类型	说明

callback	IdentityCallback	认证结果回调。回调格式如下：
		`public class IdentityResponse {`
		`// 结果Code，具体含义请参见本文档“返回值说明”章节。`
		`public int code;`
		`// 结果Code描述。`
		`public String message;`
		`}`
		`public interface IdentityCallback {`
		`boolean response(IdentityResponse response);`
		`}`

用户自定义参数extParams支持的字段说明

名称	描述	示例值
ext_params_key_use_video	是否需要返回视频留证。取值： ○ **false（默认）**：不返回视频留证。 ○ **true**：返回视频留证。	false
kIdentityParamKeyNextButtonColor	OCR识别结果确认页底部按钮的底色。	#FF0000
ext_params_key_face_progress_color	定制实人认证进度条的颜色。	#FF0000

示例代码：

```
var aliyunFace = api.require('aliyunFaceVerifyModule');
aliyunFace.verify({
    certifyId: '1adffa'
}, function(ret, err) {
    if (ret.code == 1000) {
        alert('认证成功');
    } else {
        alert(err.code);
    }
});
```

返回值说明

服务器端的验证结果通过 `IdentityResponse.code` 返回，取值范围如下。

错误码	说明
1000	刷脸成功。
1001	系统错误。

错误码	说明
1003	验证中断。
2002	网络错误。
2003	客户端设备时间错误。
2006	刷脸失败。

4.3. PC或移动端H5网页接入

4.3.1. 接入流程

实人认证提供Web SDK，帮助您在浏览器或内嵌webview中实现实人认证功能。

前提条件

您必须了解通过HTTP或HTTPS原生调用金融级实人认证服务接口时，构造请求签名的方法。更多签名机制的信息，请参见API签名机制。

操作步骤

1. 开通金融级实人认证服务。
2. 添加认证场景。

   > **说明** 基于不同业务场景，您可以在添加认证场景时，手动配置是否授权阿里云访问指定OSS存储空间用于存储认证的资料图片。

3. 如果需要将认证过程中的人脸采集照片存储到OSS空间，您需要授权金融级实人认证访问OSS存储空间。
4. 接入PC或移动端H5网页认证方案。
   i. 进行服务端接入，获取认证URL。

      具体步骤，请参见服务端接入。

   ii. 在应用客户端接入认证URL。

      具体步骤，请参见PC或移动端H5网页接入。

4.3.2. 服务端接入

本文介绍了活体检测方案PC或移动端H5网页认证方案的接入流程。

发起认证请求

接口名：InitFaceVerify。

服务地址：cloudauth.aliyuncs.com。

请求方法：HTTPS POST和GET。

接口说明：每次开始认证前通过本接口获取CertifyId，用来串联认证请求中的各个接口。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	认证场景ID。您可以在控制台创建认证场景后获得的场景ID，具体操作，请参见添加认证场景。	1000000006
OuterOrderNo	String	是	客户服务端自定义的业务唯一标识，用于后续定位排查问题使用。值最长为32位长度的字母数字组合，请确保唯一。	e0c34a77f5ac40a5aa5e6ed20c353888
ProductCode	String	是	认证方案。 唯一取值：**LR_FR**。	LR_FR
ReturnUrl	String	是	客户业务页面回跳的目标地址。	http://www.aliyun.com
MetaInfo	String	是	MetaInfo环境参数，需要通过JS获取。	{"zimVer":"3.0.0","appVersion": "1","bioMetaInfo": "4.1.0:11501568,0","appName": "com.aliyun.antcloudauth","deviceType": "ios","osVersion": "iOS 10.3.2","apdidToken": "","deviceModel": "iPhone9,1"}
Mobile	String	否	用户手机号码。	130xxxxxxxx
Ip	String	否	用户IP。	114.xxx.xxx.xxx
UserId	String	是	客户业务自定义的用户ID，请保持唯一。	123456789
CallbackUrl	String	否	认证结果的回调通知地址，必须以 `https` 开头。 平台在完成认证后会回调该地址，并自动添加 `certifyId` 、 `passed` 和 `subcode` 字段，示例： `https://www.aliyun.com?certifyId=xxxx&passed=T&subcode=200` 。	https://www.aliyun.com
CallbackToken	Callback Token	否	安全Token，由您自行生成，用于防重复、防篡改校验。 如果设置了该值会在回调地址显示**CallbackToken**字段。	NMjvQanQgplBSaEI0sL86WnQplB

名称	类型	是否必选	描述	示例值
CertifyUrlType	String	否	Web SDK设备类型。取值**WEB**或者**H5**。 说明 只支持Web SDK设备类型。	WEB

返回数据

名称	类型	描述	示例值
RequestId	String	请求ID。	130A2C10-B9EE-4D84-88E3-5384FF039795
Message	String	请求参数的响应信息。	success
Code	String	返回码：200为成功，其他为失败。	200
ResultObject.CertifyId	String	实人认证唯一标识。	91707dc296d469ad38e4c5efa6a0f24b
ResultObject.CertifyUrl	String	Web认证URL，当您完成服务端初始化，阿里云会将Web认证URL返回给您。认证结束后根据请求参数中设置的ReturnUrl进行跳转。	http://www.test.com

返回Code和Message说明

Code	Message	描述
200	success	成功。
400	参数不能为空	参数不能为空。
401	参数非法	非法参数。传入的姓名、身份证号码长度必须符合国家标准且不得包含英文字母等特殊字符。
402	应用配置不存在	应用配置不存在。
404	认证场景配置不存在	认证场景配置不存在，请先在控制台上创建认证场景。
410	未开通服务	未开通OSS产品或未完成OSS读写授权，请登录控制台完成授权。关于开通OSS产品和完成OSS读写授权，请参见授权金融级实人认证访问OSS存储空间。
411	RAM无权限	需要给RAM用户授予**AliyunAntCloudAuthFullAccess**的操作权限。
412	欠费中	金融级实人认证或OSS存在欠费，请充值后操作。

Code	Message	描述
414	设备类型不支持	当前移动设备不支持刷脸认证，请更换设备后操作。
415	SDK版本不支持	当前认证SDK版本不支持刷脸认证，请升级SDK后操作。
416	系统版本不支持	当前操作系统版本不支持刷脸认证，请升级系统或更换设备操作。
417	无法使用刷脸服务	当前身份信息比对源不可用。若信息正确，建议人工审核。
418	刷脸失败次数过多	当天刷脸认证次数过多，请明天再试。
500	系统错误	系统内部错误，请反馈工程师排查。

查询认证结果

接口名：**DescribeFaceVerify**。

服务地址：cloudauth.aliyuncs.com。

请求方法：HTTPS POST和GET。

接口说明：当接入方移动端收到回调之后，其服务端可以调用此接口，来获取相应的认证状态和认证资料。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	认证场景ID。	1000000006
CertifyId	String	是	实人认证唯一标识。	91707dc296d469ad38e4c5efa6a0f24b

返回数据

名称	类型	描述	示例值
RequestId	String	请求ID。	130A2C10-B9EE-4D84-88E3-5384FF039795
Message	String	请求消息的响应信息。	success
Code	String	返回码。关于认证结果的详细说明，请参见下文**ResultObject.SubCode错误码说明**。	200

名称	类型	描述	示例值
ResultObject.Passed	String	认证结果。取值： • **T**：通过。 • **F**：未通过。 **说明** 判断认证结果请以ResultObject.Passed字段为准。	T
ResultObject.SubCode	String	认证结果描述。具体请参见下文**ResultObject.SubCode错误码说明**。 **说明** 判断认证结果请以ResultObject.Passed字段为准。	200
ResultObject.MaterialInfo	String	认证主体附件信息，主要为图片类材料。 本参数取值中涉及的字段解释，请参见下文的返回示例。	{"faceAttack": "F","faceOcclusion":"F","facialPictureFront": {"faceAttackScore": 0.00008597839769208804,"pictureUrl": "https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg","ossBucketName": "cn-shanghai-aliyun-cloudauth-1260051251634779","ossObjectName": "verify/1260051251634779/6ba7bcfccf33f56cdb44ed086f36ce3e0.jpeg","qualityScore": 99.93476867675781}}

返回示例

ResultObject.MaterialInfo的JSON格式示例：

```
{
    // 是否为攻击：攻击为T，非攻击为F。
    "faceAttack": "T",
    // 是否有脸部遮挡：有脸部遮挡为T，否则为F。
    "faceOcclusion": "F",
    // 认证的照片信息。
    "facialPictureFront": {
         // 人脸攻击分。
         "faceAttackScore": 0.00008597839769208804,
        // 照片存放客户上海区域的OSS里，此为照片的bucket名。
        "ossBucketName": "cn-shanghai-aliyun-cloudauth-1260051251634779",
        // 照片存放客户上海区域的OSS里，此为照片的文件名。
        "ossObjectName": "verify/1260051251634779/03a081bd96328aedf83f635f39a50c57_0.jpeg",
        // 照片https地址，有效期15分钟，从查询时开始计时。
        "pictureUrl": "http://cn-shanghai-aliyun-cloudauth-1260051251634779.oss-cn-shanghai
.aliyuncs.com/verify/1260051251634779/03a081bd96328aedf83f635f39a50c57_0.jpeg?Expires=16253
71140&OSSAccessKeyId=STS.NTX1ngfr6Acg2Pmnn2RYM****&Signature=Hw5BF9WxJs6wI68IxKs41cxCU8****
&security-token=CAISjgJ1q6Ft5B2yfSjIr5ftetTTi60X9qGMMHbcim5nXtZhu7GT1Dz2IH1PdXFgA%2Bgds%2Fs
wmm5U7vgalrkqEcEdHRGdN5YpsM8LrlzwO1h2TGRsq%2B5qsoasPETOITyZtZagToeUZdfZfejXGDKgvyRvwLz8WCy%
2FVli%2BS%2FOggoJmadJlNWvRL0AxZrFsKxBltdUROFbIKP%2BpKWSKuGfLC1dysQcO4gEWq4bHm5fCskKD1Qenk7F
O%2B9uuc6LJNZc8YM1NNP6ux%2FFze6b71ypd1gNH7q8ejtYfpGyW5oHHWwIMvErYbbaMrIV1WwZ9Z7knHaVAq%2BXw
nOBkuuXYnIns0BdLMuZOSD7YQI2wYWEwgBcxt78agAF%2FHZbelNLYWtipAu5X5oG1t22SqIr1p3TMK5FrjDIVeWOpp
cxeRXtU%2BjR7hGcwd25scGhiraoxTXV3sxw%2F6dhFSswQ37O4j%2B%2FrCPtiGauqn2ZMdMj%2FvYwKb6KmpQqa%2
BtR%2F9cxhS6hoIQRq0SUIfyXl5ZUiZbTjW22iwuX%2BPwzVCw%3D%3D",
        // 活体人脸质量分数。
        "qualityScore": 99.93476867675781
    },
}
```

ResultObject.SubCode错误码说明

错误码	认证描述文案	是否计费	可能原因和处理建议
200	认证通过	是	成功。
205	活体检测存在风险	是	可能存在攻击风险，建议人工审核分层处理，若为真人可重新操作。
206	业务策略限制	是	个性化业务安全策略限制，如有疑问可咨询客服。

说明 如果您有个性化需求，您可以根据业务情况，参考返回的比对分和阈值，自定义认证结果。

返回Code和Message说明

Code	Message	描述
200	success	成功。
400	参数不能为空	参数不能为空。

Code	Message	描述
401	参数非法	非法参数。传入的姓名、身份证号码长度必须符合国家标准且不得包含英文字母等特殊字符。
403	认证未完成	认证未完成。
404	认证场景配置不存在	认证场景配置不存在，请先在控制台上创建认证场景。
406	无效的CertifyId	无效的CertifyId。
410	未开通服务	未开通OSS产品或未完成OSS读写授权。
411	RAM无权限	需要给RAM用户授予AliyunAntCloudAuthFullAccess的操作权限。
424	身份认证记录不存在	当前CertifyId还未完成认证，建议您客户端收到认证回调为1000或2006时，再请求查询接口获取详情。 **说明** 如果出现CertifyId未认证、认证过程中放弃、认证处理中的情况，均会导致查询接口返回此错误码。
500	系统错误	系统内部错误。

4.3.3. PC或移动端H5网页接入

本文介绍PC或移动端H5网页的接入流程。

手机浏览器支持列表

浏览器名称	支持情况
Edge	支持
FireFox	支持
Chrome	支持
Safari	支持（iOS 11及以上）
Opera	支持
Android Browser 5.0+	支持
UC	支持
QQ	支持
厂商内置浏览器	小米、三星等部分机型支持

> **说明** 受制于浏览器兼容性碎片化问题，建议您在流程设计上引导用户使用推荐的浏览器完成认证。若您在手机应用App内集成该方案，可能会因为内嵌浏览器原因无法兼容，推荐您使用SDK模式接入。若您是通过微信公众号或者小程序集成，由于微信运营审核规则的限制，可能出现无法避免的兼容性问题，建议您使用纯服务端（API）接入方式。

扫脸认证

为了落实用户个人信息保护相关法律法规要求，提升用户刷脸认证体验，URL默认自带扫脸认证引导页，用户勾选同意认证服务协议，同意授权后，可以开始使用扫脸认证服务。微信公众号场景下，扫脸认证引导页是必须存在的，否则会导致摄像头无法唤起，引发黑屏现象。

> **说明** 目前，新增的用户授权页面处于灰度中。如果需要在您的应用客户端增加该页面，您可以提交工单，获取技术支持。

启动刷脸

```
<script type="text/javascript"src="https://cn-shanghai-aliyun-cloudauth.oss-cn-shanghai.aliyuncs.com/web_sdk_js/jsvm_all.js" ></script>
```

1. 引入上述JS文件，调用函数getMetaInfo() 获取MetaInfo。
2. 您在调用实人认证服务端发起请求时需要传入该MetaInfo值。
3. 获取服务端初始化返回的CertifyUrl，在浏览器中加载该链接进行实人认证。

结果解析

CertifyUrl中ReturnUrl参数为商户自定义，查询结果会在扫脸认证完成后拼接在ReturnUrl后面作为参数返回给商户。格式如下：

```
response=%7B"code"%3A1001%2C"subCode"%3A"Z5051"%2C"reason"%3A"success"%2C"extInfo"%3A%7B%7D%7D
```

从URL中获取Response参数，URL解码后格式如下：

```
{
  "code":"",
  "subCode":"",
  "reason":"",
  "extInfo" : {}
}
```

说明 前端路由模式为history，只需对response解码一次即可。若您使用的是hash路由模式，则需要对response进行二次解码（例如 `decodeURIComponent(decodeURIComponent('responseValue'))` ）。

错误码

错误码	错误码文案	错误码描述
1000	刷脸成功	代表刷脸成功，该结果仅供参考，可通过服务端获取最终认证结果。具体内容，请参见查询认证结果。
1001	系统错误	表示系统错误。
1003	验证中断	表示验证中断。
2002	网络错误	表示网络错误。
2003	客户端设备时间错误	表示客户端设备时间错误。
2006	刷脸失败	表示刷脸失败，如需获取更详细的失败原因，需调用服务端查询认证结果。具体内容，请参见查询认证结果。

subCode说明

错误码	错误码文案	错误码描述
Z5052	数据错误或程序异常	表示数据错误或程序异常。
Z5056	重试次数过多	表示重试次数过多。
Z5055	用户退出	表示用户退出。
Z5054	摄像头无权限或者无法获取摄像头数据出错	表示摄像头无权限或者无法获取摄像头数据出错。

4.4. 纯服务端（API）接入

4.4.1. LivenessFaceVerify

如果您的业务场景无法通过SDK方式集成活体检测，推荐您选择人脸静默活体（LivenessFaceVerify）API。通过调用LivenessFaceVerify接口，您可以对用户自拍图片或线下设备采集的用户人脸图片进行活体检测，适用于SDK无法覆盖的场景。本文介绍了调用LivenessFaceVerify接口对人脸图片进行活体检测的纯服务端接入流程。

人脸静默活体检测API

接口名：LivenessFaceVerify。

服务地址：`cloudauth.aliyuncs.com`。

请求方法：HTTPS POST和GET。

接口说明：调用LivenessFaceVerify接口对人脸图片进行活体检测。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	认证场景ID。	100000****
OuterOrderNo	String	是	客户服务端自定义的业务唯一标识，用于后续定位排查问题。取值最长为32位长度的字母数字组合，请确保唯一。	e0c34a77f5ac40a5aa5e6ed20c353888
ProductCode	String	是	固定值：**LR_FR_MIN**。	LR_FR_MIN
Model	String	否	活体检测类型。取值： • **FRONT_CAMERA_LIVENESS**：移动设备前置摄像头采集的人脸图片活体检测。 • **REAR_CAMERA_LIVENESS**：其他场景采集的人脸图片活体检测。	FRONT_CAMERA_LIVENESS
DeviceToken	String	否	设备Token，用于风险识别。	McozS1ZWRcRZStlERcZZo_QOytx5jcgZoZJEoRLOxxxxxxx
Mobile	String	否	用户手机号码。	130xxxxxxxx
Ip	String	否	用户网络IP地址。	114.xxx.xxx.xxx
UserId	String	否	客户业务自定义的用户ID，请保持唯一。	123456789

名称	类型	是否必选	描述	示例值
FaceContrastPicture	String	否	照片Base64编码。如果您选择FaceContrastPicture（照片Base64编码）方式传入人脸照片，请注意检查照片大小，不要传入过大的照片；您可以通过OSS方式上传较大的人脸照片。 说明 您需要在下述三种传入人脸照片方式中，选择其中一种： • FaceContrastPicture • FaceContrastPictureUrl • OSS	/9j/4AAQSkZJRgABAQAASxxxxxxx
FaceContrastPictureUrl	String	否	人像地址，公网可访问的HTTP、HTTPS链接。 说明 您需要在下述三种传入人脸照片方式中，选择其中一种： • FaceContrastPicture • FaceContrastPictureUrl • OSS	https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg
OssBucketName	String	否	已授权OSS空间的文件名。如果您选择OSS方式传入人脸照片，您必须传入OssBucketName和OssObjectName参数。OSS请求参数格式仅支持开通实人认证服务时授权的OSS生成的URL、Bucket和文件名。 说明 您需要在下述三种传入人脸照片方式中，选择其中一种： • FaceContrastPicture • FaceContrastPictureUrl • OSS	cn-shanghai-aliyun-cloudauth-xxxxx

名称	类型	是否必选	描述	示例值
OssObjectName	String	否	已授权OSS空间的文件名。如果您选择OSS方式传入人脸照片，您必须传入OssBucketName和OssObjectName参数。OSS请求参数格式仅支持开通实人认证服务时授权的OSS生成的URL、Bucket和文件名。 **说明** 您需要在下述三种传入人脸照片方式中，选择其中一种： • FaceContrastPicture • FaceContrastPictureUrl • OSS	verify/xxxxx/xxxxx.jpeg
Crop	String	否	是否允许裁剪人脸图片，默认不允许。 • T：表示允许剪裁。 • F：表示不允许剪裁。 **说明** 若您请求的图片并非来自标准活体检测SDK采集，建议您允许裁剪人脸图片。 开启该功能后，将会对请求的图片先进行人脸裁剪与矫正，完成后会发起请求服务。	T

返回数据

名称	类型	描述	示例值
RequestId	String	请求ID。	130A2C10-B9EE-4D84-88E3-5384FF039795
Message	String	返回信息的响应消息。	success
Code	String	返回码。取值： • **200**：表示响应成功。 • 其他：表示响应失败。具体请参见返回Code和Message说明。 详细认证结果判断请参考ResultObject.SubCode的结果。	200

名称	类型	描述	示例值
ResultObject.Passed	String	认证结果。取值： • T：认证通过。 • F：认证未通过。	T
ResultObject.CertifyId	String	08573be80f944d95ac812e019e3655a	实人认证请求的唯一标识。
ResultObject.SubCode	String	认证结果描述。具体请参见ResultObject.SubCode错误码说明。	200
ResultObject.MaterialInfo	String	认证主体附件信息，主要为图片类材料，JSON格式，具体请参见ResultObject.MaterialInfo的JSON格式示例。	{"faceAttack":"F","faceOcclusion":"F","facialPictureFront":{"faceAttackScore":0.00008597839769208804,"qualityScore":88.3615493774414}}

ResultObject.MaterialInfo的JSON格式示例

```
{
    // 是否为攻击。T表示是，F表示否。
    "faceAttack": "F",
    // 是否有脸部遮挡。T表示是，F表示否。
    "faceOcclusion": "F",
    "facialPictureFront": {
        // 人脸攻击分。
        "faceAttackScore": 0.00008597839769208804,
        // 活体人脸质量分数。
       "qualityScore": 88.3615493774414
    }
}
```

返回Code和Message说明

Code	Message	描述
200	success	成功。
400	参数不能为空	参数不能为空。
401	参数非法	非法参数。传入的姓名、身份证号码长度必须符合国家标准且不得包含英文字母等特殊字符。
404	认证场景配置不存在	认证场景配置不存在，请先在控制台上创建认证场景。
410	未开通服务	未开通OSS产品或未完成OSS读写授权，请登录控制台完成授权。具体操作，请参见授权金融级实人认证访问OSS存储空间。
411	RAM无权限	需要为子账号授予AliyunAntCloudAuthFullAccess的操作权限。

Code	Message	描述
412	欠费中	金融级实人认证或OSS存在欠费，请充值后操作。
419	传入图片不可用	图片无法下载、图片内容为空、图片分辨率不符合要求或提取不到人脸特征，建议更换图片。
420	数据重复	图片传入方式超过了一种。支持以下三种传入人脸图片的方式，必须且只能使用其中一种： • FaceContrastPicture • FaceContrastPictureUrl • OSS
421	传入图片过大	图片超过了1 MB，建议压缩图片或更换图片上传方式。
422	下载图片超时	图片下载超过了3秒，请排查网络后重新操作。
500	系统错误	系统内部错误，请反馈工程师排查。

ResultObject.SubCode错误码说明

错误码	认证描述文案	是否计费	可能原因和运营建议
200	认证通过	是	无。
205	活体检测存在风险	是	可能存在攻击风险，建议人工审核分层处理。如果是真人，可重新操作。

4.4.2. Java SDK调用示例

本文介绍了金融级活体检测方案人脸静默活体检测API的Java SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

调试LivenessFaceVerify接口

本章节提供的是最新版本的Java SDK调用示例，如果您需要查看之前版本，请参见Java SDK调用示例（V1版本）。

前提条件

需要使用的Java版本在1.8版本及以上。

使用Maven引入SDK

```
<dependency>
        <groupId>com.aliyun</groupId>
        <artifactId>cloudauth20190307</artifactId>
         <version>1.0.5</version>
</dependency>
```

调用示例

```
import java.util.Arrays;
import java.util.List;
import com.aliyun.cloudauth20190307.Client;
import com.aliyun.cloudauth20190307.models.LivenessFaceVerifyRequest;
import com.aliyun.cloudauth20190307.models.LivenessFaceVerifyResponse;
import com.aliyun.tea.TeaException;
import com.aliyun.tea.TeaUnretryableException;
import com.aliyun.tearpc.models.Config;
import com.aliyun.teautil.models.RuntimeOptions;
public class LivenessFaceVerify {
    public static void main(String[] args) throws Exception {
        // 通过以下代码创建API请求并设置参数。
        LivenessFaceVerifyRequest request = new LivenessFaceVerifyRequest();
        // 请输入场景ID+L。
        request.setSceneId(xxxxL);
        request.setOuterOrderNo("商户请求的唯一标识");
        request.setProductCode("LR_FR_MIN");
        // 卡证核身类型，固定值。
        request.setModel("FRONT_CAMERA_LIVENESS");
        //request.setDeviceToken("xxxx");
        //request.setMobile("130xxxxxxxx");
        //request.setIp("114.xxx.xxx.xxx");
        //request.setUserId("12345xxxx");
        // 请在以下方式中选择一种，传入人脸图片。
        // 方式一：人脸图片Base64。
        request.setFaceContrastPicture("人脸图片Base64编码");
        // 方式二：人脸图片OSS bucket名和文件名。
        //request.setOssBucketName("cn-shanghai-aliyun-cloudauth-xxxxx");
        //request.setOssObjectName("verify/xxxxx/xxxxxx.jpeg");
        // 方式三：人脸图片的OSS URL地址。该地址必须公网可访问。
        //request.setFaceContrastPictureUrl("https://cn-shanghai-aliyun-cloudauth-xxxxxx.os
s-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg");
        // 推荐，支持服务路由。
        LivenessFaceVerifyResponse response = livenessFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        //LivenessFaceVerifyResponse response = livenessFaceVerify("cloudauth.cn-shanghai.a
liyuncs.com", request);
        System.out.println(response.getRequestId());
        System.out.println(response.getCode());
        System.out.println(response.getMessage());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getPassed());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getSubCode());
        System.out.println(response.getResultObject() == null ? null
```

```
            : response.getResultObject().getMaterialInfo());
    }
    private static LivenessFaceVerifyResponse livenessFaceVerifyAutoRoute(LivenessFaceVerif
yRequest request) {
        // 第一个为主区域Endpoint，第二个为备区域Endpoint。
        List<String> endpoints = Arrays.asList("cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com");
        LivenessFaceVerifyResponse lastResponse = null;
        for (String endpoint : endpoints) {
            try {
                LivenessFaceVerifyResponse response = livenessFaceVerify(endpoint, request)
;
                lastResponse = response;
                // 服务端错误，切换到下个区域调用。
                if (response != null && "500".equals(response.getCode())) {
                    continue;
                }
                return response;
            } catch (Exception e) {
                // 网络异常，切换到下个区域调用。
                if (e.getCause() instanceof TeaException) {
                    TeaException teaException = ((TeaException)e.getCause());
                    if (teaException.getData() != null && "ServiceUnavailable".equals(
                        teaException.getData().get("Code"))) {
                        continue;
                    }
                }
                if (e.getCause() instanceof TeaUnretryableException) {
                    continue;
                }
            }
        }
        return lastResponse;
    }
    private static LivenessFaceVerifyResponse livenessFaceVerify(String endpoint, LivenessF
aceVerifyRequest request)
        throws Exception {
        Config config = new Config();
        config.setAccessKeyId("您的AccessKey ID");
        config.setAccessKeySecret("您的AccessKey Secret");
        config.setEndpoint(endpoint);
        // 设置https代理。
        //config.setHttpsProxy("https://xx.xx.xx.xx:xxxx");
        // 设置http代理。
        //config.setHttpProxy("http://xx.xx.xx.xx:xxxx");
        Client client = new Client(config);
        // 创建RuntimeObject实例并设置运行参数。
        RuntimeOptions runtime = new RuntimeOptions();
        runtime.readTimeout = 10000;
        runtime.connectTimeout = 10000;
        return client.livenessFaceVerify(request, runtime);
    }
}
```

4.4.3. PHP SDK调用示例

本文介绍了人脸静默活体检测API的PHP SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

调试LivenessFaceVerify接口

前提条件

- 需要使用的PHP版本在5.6版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于PHP SDK源代码，请参见PHP SDK包。

- 通过Composer来管理项目依赖（推荐）

```
composer require alibabacloud/cloudauth-20190307 1.0.5
```

- 使用composer.json文件

```
{
   "require": {
       "alibabacloud/cloudauth-20190307": "1.0.5"
    }
}
```

如果因网络问题导致无法按照上述操作安装SDK，您可以使用阿里云Composer全量镜像。

调用示例

LivenessFaceVerify接口调用示例

```
<?php
namespace AlibabaCloud\SDK\Sample {
use AlibabaCloud\SDK\Cloudauth\V20190307\Cloudauth;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\LivenessFaceVerifyRequest;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\LivenessFaceVerifyResponse;
use AlibabaCloud\Tea\Utils\Utils;
use \Exception;
use AlibabaCloud\Tea\Exception\TeaError;
use AlibabaCloud\Tea\Rpc\Rpc\Config;
use AlibabaCloud\Tea\Utils\Utils\RuntimeOptions;
class LivenessFaceVerify {
    /**
     * @param string $endpoint
     * @return Cloudauth
     */
    public static function createClient($endpoint){
        $config = new Config([
            // 您的AccessKey ID。
```

```
            "accessKeyId" => "<您的AccessKey ID>",
            // 您的AccessKey Secret。
            "accessKeySecret" => "<您的AccessKey Secret>",
            // 设置HTTP代理。
            // "httpProxy" => "<http://xx.xx.xx.xx:xxxx>",
            // 设置HTTPS代理。
            // "httpsProxy" => "<https://xx.xx.xx.xx:xxxx>",
            "endpoint" => $endpoint
        ]);
        return new Cloudauth($config);
    }
    /**
     * @return void
     */
    public static function main(){
        $request = new LivenessFaceVerifyRequest([
            // 请输入场景ID。
            "sceneId" => <xxx>,
            "outerOrderNo" => "<商户请求的唯一标识>",
            // 要接入的认证方案，固定值：LR_FR_MIN。
            "productCode" => "<LR_FR_MIN>",
            // 活体检测类型。
            "model" => "<FRONT_CAMERA_LIVENESS>",
            // 设备Token，用于风险识别。
            // "deviceToken" => "<McozS1ZWRcRZStlERcZZo_QOytx5jcgZoZJEoRLOxxxxxxx>",
            // "mobile" => "<LR_FR_MIN>",
            // "ip" => "<114.xxx.xxx.xxx>",
            // "userId" => "<123456789>",
            // 请在以下方式中选择一种，传入人脸图片。
            // 方式一：人脸图片Base64。
            // "faceContrastPicture" => "<人脸图片Base64编码>",
            // 方式二：人脸图片OSS Bucket名和文件名。
            // "ossBucketName" => "<cn-shanghai-aliyun-cloudauth-xxxxx>",
            // "ossObjectName" => "<verify/xxxxx/xxxxxx.jpeg>",
            // 方式三：人脸图片的OSS URL地址。该地址必须公网可访问。
            "faceContrastPictureUrl" => "<https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-c
n-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg>"
        ]);
        // 推荐，支持服务路由。
        $response = self::livenessFaceVerifyAutoRoute($request);
        // 不支持服务自动路由。
        // $response = self::livenessFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", $requ
est);
        var_dump($response->requestId);
        var_dump($response->code);
        var_dump($response->message);
        var_dump($response->resultObject->materialInfo);
        var_dump($response->resultObject->subCode);
        var_dump($response->resultObject->passed);
    }
    /**
     * @param LivenessFaceVerifyRequest $request
     * @return LivenessFaceVerifyResponse
     */
    public static function livenessFaceVerifyAutoRoute(LivenessFaceVerifyRequest $request){
```

```
    public static function livenessFaceVerifyAutoRoute(LivenessFaceVerifyRequest $request){
        $endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        foreach($endpoints as $endpoint){
            try {
                $response = self::livenessFaceVerify($endpoint, $request);
                if (Utils::equalString("500", $response->code)) {
                    continue;
                }
                return $response;
            }
            catch (Exception $err) {
                if (!($err instanceof TeaError)) {
                    $err = new TeaError([], $err->getMessage(), $err->getCode(), $err);
                }
                // 网络异常，切换到下个区域调用。
                if (Utils::equalString("ServiceUnavailable", $err->code)) {
                    continue;
                }
            }
        }
        return null;
    }
    /**
     * @param string $endpoint
     * @param LivenessFaceVerifyRequest $request
     * @return LivenessFaceVerifyResponse
     */
    public static function livenessFaceVerify($endpoint, $request){
        $client = self::createClient($endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        $runtime = new RuntimeOptions([]);
        $runtime->readTimeout = 10000;
        $runtime->connectTimeout = 10000;
        return $client->livenessFaceVerify($request, $runtime);
    }
}
$path = __DIR__ . \DIRECTORY_SEPARATOR . \DIRECTORY_SEPARATOR . 'vendor' . \DIRECTORY_SEPAR
ATOR . 'autoload.php';
if (file_exists($path)) {
    require_once $path;
}
LivenessFaceVerify::main();
}
```

4.4.4. Python SDK调用示例

本文介绍了人脸静默活体检测API的Python SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

详细信息，请参见调试LivenessFaceVerify接口。

前提条件

- 需要使用的Python版本在3.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

引入依赖包

```
pip install alibabacloud_tea_openapi==0.2.5
```

安装SDK

关于Python SDK源代码，请参见Python SDK包。

```
pip install alibabacloud_cloudauth20190307==1.0.5
```

调用示例

LivenessFaceVerify接口调用示例

关于参数的具体描述，请参见LivenessFaceVerify。

```
from alibabacloud_cloudauth20190307.client import Client as CloudauthClient
from alibabacloud_tea_openapi import models as open_api_models
from alibabacloud_cloudauth20190307 import models as cloudauth_models
from alibabacloud_tea_util.client import Client as UtilClient
from alibabacloud_tea_util import models as util_models
class LivenessFaceVerify:
    def __init__(self):
        pass
    @staticmethod
    def create_client(
        endpoint: str,
    ) -> CloudauthClient:
        config = open_api_models.Config(
            # 您的AccessKey ID。
            access_key_id='您的AccessKey ID',
            # 您的AccessKey Secret。
            access_key_secret='您的AccessKey Secret',
            # 设置HTTP代理。
            # http_proxy='http://xx.xx.xx.xx:xxxx',
            # 设置HTTPS代理。
            # https_proxy='https://username:password@xxx.xxx.xxx.xxx:9999',
            endpoint=endpoint
        )
        return CloudauthClient(config)
    @staticmethod
    def main() -> None:
        request = cloudauth_models.LivenessFaceVerifyRequest(
            # 请输入场景ID。
            scene_id=xxx
```

```
            scene_id=xxx,
            outer_order_no='商户请求的唯一标识',
            # 固定值。
            product_code='LR_FR_MIN',
            model='FRONT_CAMERA_LIVENESS',
            # ip='用户IP地址',
            # user_id='用户ID',
            # mobile='130xxxxxxxx',
            # device_token='通过设备指纹获取的deviceToken',
            # 请在以下四种方式中选择其一。
            # 1.照片Base64编码。
            # 2.照片OSS Bucket名和文件名。
            # 3.照片OSS的URL地址。
            # 方式一：照片Base64。
            # face_contrast_picture='人脸照片Base64编码'
            # 方式二：照片OSS Bucket名和文件名。
            # oss_bucket_name='cn-shanghai-aliyun-cloudauth-xxxxx',
            # oss_object_name='verify/xxxxx/xxxxxx.jpeg',
            # 方式三：照片OSS的URL地址，公网可访问。
            face_contrast_picture_url='https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-s
hanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg'
        )
        response = LivenessFaceVerify.liveness_face_verify_auto_route(request)
        # 不支持服务自动路由。
        # response = LivenessFaceVerify.liveness_face_verify('cloudauth.cn-shanghai.aliyunc
s.com', request)
        print(response.request_id)
        print(response.code)
        print(response.message)
        print(response.result_object.passed)
        print(response.result_object.sub_code)
        print(response.result_object.material_info)
        print(response.result_object.certify_id)
    @staticmethod
    def liveness_face_verify_auto_route(
        request: cloudauth_models.LivenessFaceVerifyRequest,
    ) -> cloudauth_models.LivenessFaceVerifyResponse:
        endpoints = [
            'cloudauth.cn-shanghai.aliyuncs.com',
            'cloudauth.cn-beijing.aliyuncs.com'
        ]
        for endpoint in endpoints:
            try:
                response = LivenessFaceVerify.liveness_face_verify(endpoint, request)
                if UtilClient.equal_string('500', response.code):
                    continue
                return response
            except Exception as err:
                    continue
        return None
    @staticmethod
    def liveness_face_verify(
        endpoint: str,
        request: cloudauth_models.LivenessFaceVerifyRequest,
    ) -> cloudauth_models.LivenessFaceVerifyResponse:
```

```
        client = LivenessFaceVerify.create_client(endpoint)
        # 创建RuntimeObject实例并设置运行参数。
        runtime = util_models.RuntimeOptions()
        runtime.read_timeout = 10000
        runtime.connect_timeout = 10000
        return client.liveness_face_verify(request, runtime)
if __name__ == '__main__':
    LivenessFaceVerify.main()
```

4.4.5. Go SDK调用示例

本文介绍了人脸静默活体检测API的Go SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

详细信息，请参见调试LivenessFaceVerify接口。

前提条件

- 仅支持Go 1.12.0及以上版本。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

引入依赖包

```
go get github.com/alibabacloud-go/tea-rpc
go get github.com/alibabacloud-go/tea-utils
go get github.com/alibabacloud-go/tea
```

安装SDK

关于Go SDK源代码，请参见Go SDK包。

通过Go mod来管理项目依赖：

```
go get github.com/alibabacloud-go/cloudauth-20190307@vv1.0.5
```

调用示例

LivenessFaceVerify接口调用示例

关于参数的具体描述，请参见LivenessFaceVerify。

```
package main
import (
  "fmt"
  cloudauth "github.com/alibabacloud-go/cloudauth-20190307/client"
  rpc "github.com/alibabacloud-go/tea-rpc/client"
  util "github.com/alibabacloud-go/tea-utils/service"
  "github.com/alibabacloud-go/tea/tea"
)
func CreateClient(endpoint *string) (_result *cloudauth.Client, _err error) {
```

```
func CreateClient(endpoint *String) (_result *cloudauth.Client, _err error) {
  config := &rpc.Config{
    // 您的AccessKey ID。
    AccessKeyId: tea.String("您的AccessKey ID"),
    // 您的AccessKey Secret。
    AccessKeySecret: tea.String("您的AccessKey Secret"),
    Endpoint:        endpoint,
    // 设置HTTP代理。
    // HttpProxy: tea.String("http://xx.xx.xx.xx:xxxx"),
    // 设置HTTPS代理。
    // HttpsProxy: tea.String("https://xx.xx.xx.xx:xxxx"),
  }
  _result = &cloudauth.Client{}
  _result, _err = cloudauth.NewClient(config)
  return _result, _err
}
func _main() (_err error) {
  request := &cloudauth.LivenessFaceVerifyRequest{
    // 固定值。
    ProductCode: tea.String("LR_FR_MIN"),
    // 请输入场景ID。
    SceneId: tea.Int64(xxx),
    OuterOrderNo: tea.String("商户请求的唯一标识"),
    Model: tea.String("FRONT_CAMERA_LIVENESS"),
    // Ip: tea.String("用户IP地址"),
    // UserId: tea.String("用户ID"),
    // Mobile: tea.String("130xxxxxxxx"),
    // DeviceToken: tea.String("通过设备指纹获取的deviceToken"),
    // 请在以下四种方式中选择其一。
    // 1.照片Base64编码。
    // 2.照片OSS Bucket名和文件名。
    // 3.照片OSS的URL地址。
    // 方式一：照片Base64。
    // FaceContrastPicture: tea.String("人脸照片Base64编码"),
    // 方式二：照片OSS Bucket名和文件名。
    // OssBucketName: tea.String("cn-shanghai-aliyun-cloudauth-xxxxx"),
    // OssObjectName: tea.String("verify/xxxxx/xxxxxx.jpeg"),
    // 方式三：照片OSS的URL地址，公网可访问。
    FaceContrastPictureUrl: tea.String("https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-
shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg"),
  }
    response, _err := LivenessFaceVerifyAutoRoute(request)
  // 不支持服务自动路由。
  // response, _err := LivenessFaceVerify(tea.String("cloudauth.cn-shanghai.aliyuncs.com"),
request)
  if _err != nil {
    return _err
  }
  if response != nil {
    fmt.Println(*response.RequestId)
    fmt.Println(*response.Code)
    fmt.Println(*response.Message)
    fmt.Println(*response.RequestId)
    fmt.Println(*response.ResultObject.Passed)
    fmt.Println(*response.ResultObject.SubCode)
```

```
    fmt.Println(*response.ResultObject.MaterialInfo)
    fmt.Println(*response.ResultObject.CertifyId)
  }
  return _err
}
func LivenessFaceVerifyAutoRoute(request *cloudauth.LivenessFaceVerifyRequest) (_result *cl
oudauth.LivenessFaceVerifyResponse, _err error) {
  endpoints := []*string{tea.String("cloudauth.cn-shanghai.aliyuncs.com"), tea.String("clou
dauth.cn-beijing.aliyuncs.com")}
  var lastResponse *cloudauth.LivenessFaceVerifyResponse
  for _, endpoint := range endpoints {
    response, _err := LivenessFaceVerify(endpoint, request)
    if _err != nil {
      var err = &tea.SDKError{}
      if _t, ok := _err.(*tea.SDKError); ok {
        err = _t
        // 网络异常，切换到下个区域调用。
        if tea.BoolValue(util.EqualString(tea.String("ServiceUnavailable"), err.Code)) {
          continue
        }
      }
      return _result, _err
    }
    if tea.BoolValue(util.EqualString(tea.String("500"), response.Code)) {
      continue
    }
    _result = response
    return _result, _err
  }
  _result = lastResponse
  return _result, _err
}
func LivenessFaceVerify(endpoint *string, request *cloudauth.LivenessFaceVerifyRequest) (_r
esult *cloudauth.LivenessFaceVerifyResponse, _err error) {
  client, _err := CreateClient(endpoint)
  if _err != nil {
    return _result, _err
  }
  // 创建RuntimeObject实例并设置运行参数。
  runtime := &util.RuntimeOptions{}
  runtime.ReadTimeout = tea.Int(10000)
  runtime.ConnectTimeout = tea.Int(10000)
  _body, _err := client.LivenessFaceVerify(request, runtime)
  if _err != nil {
    return _result, _err
  }
  _result = _body
  return _result, _err
}
func main() {
  err := _main()
  if err != nil {
    panic(err)
  }
```

```
}
```

4.4.6. .NET SDK调用示例

本文介绍了人脸静默活体检测API的.NET SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

详细信息，请参见调试LivenessFaceVerify接口。

前提条件

- 需要使用的.NET Framework版本在4.5.0版本及以上或者.NET Standard版本在2.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于.NET SDK源代码，请参见.NET SDK包。

SDK有以下两种安装方式，请选择合适的方式安装：

- 通过.NET命令行安装。

```
dotnet add package AlibabaCloud.SDK.Cloudauth20190307 --version 1.0.5
```

- 使用包管理安装。

```
Install-Package AlibabaCloud.SDK.Cloudauth20190307 -Version 1.0.5
```

调用示例

LivenessFaceVerify接口调用示例

关于参数的具体描述，请参见LivenessFaceVerify。

```
using System;
using System.Collections.Generic;
using Tea;
namespace LivenessFaceVerify
{
    public class Program
    {
        public static AlibabaCloud.SDK.Cloudauth20190307.Client CreateClient(string endpoin
t)
        {
            AlibabaCloud.RPCClient.Models.Config config = new AlibabaCloud.RPCClient.Models
.Config
            {
                // 您的AccessKey ID。
                AccessKeyId = "您的AccessKey ID",
                // 您的AccessKey Secret。
                AccessKeySecret = "您的AccessKey Secret",
                // 设置HTTP代理。
```

```
                // HttpProxy = "http://xx.xx.xx.xx:xxxx",
                // 设置HTTPS代理。
                // HttpsProxy = "https://username:password@xxx.xxx.xxx.xxx:9999",
                Endpoint = endpoint,
            };
            return new AlibabaCloud.SDK.Cloudauth20190307.Client(config);
        }
        public static void Main(string[] args)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Models.LivenessFaceVerifyRequest request = n
ew AlibabaCloud.SDK.Cloudauth20190307.Models.LivenessFaceVerifyRequest
            {
                // 固定值。
                ProductCode = "LR_FR_MIN",
                // 请输入场景ID。
                SceneId = xxx,
                OuterOrderNo = "商户请求的唯一标识",
                Model = "FRONT_CAMERA_LIVENESS",
                // Ip = "用户IP地址",
                // UserId = "用户ID",
                // Mobile = "130xxxxxxxx",
                // DeviceToken = "通过设备指纹获取的deviceToken",
                // 请在以下四种方式中选择其一。
                // 1.照片Base64编码。
                // 2.照片OSS Bucket名和文件名。
                // 3.照片OSS的URL地址。
                // 方式一：照片Base64。
                //FaceContrastPicture = "人脸照片Base64编码",
                // 方式二：照片OSS Bucket名和文件名。
                // OssBucketName = "cn-shanghai-aliyun-cloudauth-xxxxx",
                // OssObjectName = "verify/xxxxx/xxxxxx.jpeg",
                // 方式三：照片OSS的URL地址，公网可访问。
                FaceContrastPictureUrl = "https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-c
n-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg",
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.LivenessFaceVerifyResponse response =
LivenessFaceVerifyAutoRoute(request);
            // 不支持服务自动路由。
            // AlibabaCloud.SDK.Cloudauth20190307.Models.LivenessFaceVerifyResponse respons
e = LivenessFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", request);
            Console.WriteLine(response.RequestId);
            Console.WriteLine(response.Code);
            Console.WriteLine(response.Message);
            Console.WriteLine(response.ResultObject.Passed);
            Console.WriteLine(response.ResultObject.SubCode);
            Console.WriteLine(response.ResultObject.MaterialInfo);
            Console.WriteLine(response.ResultObject.CertifyId);
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.LivenessFaceVerifyResponse
LivenessFaceVerifyAutoRoute(AlibabaCloud.SDK.Cloudauth20190307.Models.LivenessFaceVerifyReq
uest request)
        {
            List<string> endpoints = new List<string>
            {
```

```
                "cloudauth.cn-shanghai.aliyuncs.com",
                "cloudauth.cn-beijing.aliyuncs.com"
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.LivenessFaceVerifyResponse lastRespon
se = null;
            foreach (var endpoint in endpoints)
            {
                try
                {
                    AlibabaCloud.SDK.Cloudauth20190307.Models.LivenessFaceVerifyResponse re
sponse = LivenessFaceVerify(endpoint, request);
                    if (AlibabaCloud.TeaUtil.Common.EqualString("500", response.Code))
                    {
                        // continue;
                    }
                    return response;
                }
                catch (TeaException err)
                {
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        // continue;
                    }
                }
                catch (Exception _err)
                {
                    TeaException err = new TeaException(new Dictionary<string, object>
                    {
                        { "message", _err.Message }
                    });
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        // continue;
                    }
                }
            }
            return lastResponse;
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.LivenessFaceVerifyResponse
LivenessFaceVerify(string endpoint, AlibabaCloud.SDK.Cloudauth20190307.Models.LivenessFaceV
erifyRequest request)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Client client = CreateClient(endpoint);
            // 创建RuntimeObject实例并设置运行参数。
            AlibabaCloud.TeaUtil.Models.RuntimeOptions runtime = new AlibabaCloud.TeaUtil.M
odels.RuntimeOptions();
            runtime.ReadTimeout = 10000;
            runtime.ConnectTimeout = 10000;
            return client.LivenessFaceVerify(request, runtime);
        }
    }
```

```
}
```

4.4.7. Node.js SDK调用示例

本文介绍了人脸静默活体检测API的Node.js SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

详细信息，请参见调试LivenessFaceVerify接口。

前提条件

- 需要使用的Node.js版本在8.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于Node.js SDK源代码，请参见Node.js SDK。

建议您使用npm管理依赖包：

```
npm install @alicloud/tea-util
npm install @alicloud/rpc-client
npm install @alicloud/cloudauth20190307@1.0.5
```

调用示例

LivenessFaceVerify接口调用示例

关于参数的具体描述，请参见LivenessFaceVerify。

```
const Config = require("@alicloud/rpc-client");
const CloudAuth = require("@alicloud/cloudauth20190307");
const CloudAuthClient = CloudAuth.default;
const {RuntimeOptions} = require("@alicloud/tea-util");
const fs = require("fs");
     function createClient(endpoint) {
        // 通过以下代码创建Config实例并初始化。
        const config = new Config.Config({
            // 您的AccessKey ID。
            accessKeyId: "您的AccessKey ID",
            // 您的AccessKey Secret。
            accessKeySecret: "您的AccessKey Secret",
            // 设置Endpoint。
            endpoint: endpoint,
            // 设置HTTP代理。
            // httpProxy: "http://xxx.xxx.xxx.xxx:9999",
            // 设置HTTPS代理。
            // httpsProxy: "https://username:password@xxx.xxx.xxx.xxx:9999",
        });
        return new CloudAuthClient(config)
```

```
    }
    async function livenessFaceVerify(endpoint, request) {
        const client = createClient(endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        const runtime = new RuntimeOptions([]);
        runtime.readTimeout=10000;
        runtime.connectTimeout=10000;
        return await client.livenessFaceVerify(request, runtime);
    }
    async function livenessFaceVerifyAutoRoute(request) {
        const endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        for (const endpoint of endpoints) {
            try {
                const response = await livenessFaceVerify(endpoint, request);
                if (response.code === '500') {
                    continue;
                }
                return response;
            } catch (err) {
                continue;
            }
        }
        return null;
    }
    async function main(){
        // 通过以下代码创建API请求并设置参数。
        const request = new CloudAuth.LivenessFaceVerifyRequest({
        // 请输入场景ID。
        sceneId: xxx,
        // 设置商户请求的唯一标识。
        outerOrderNo: "xxx",
        // 要接入的认证方案。
        productCode: "LR_FR_MIN",
        // 模式。
        model: "FRONT_CAMERA_LIVENESS",
        // 请在以下方式中选择一种，传入人脸图片。
        // 方式一：人脸图片Base64。
        // faceContrastPicture: "人脸图片Base64编码",
        // 方式二：人脸图片OSS bucket名和文件名。
        // ossBucketName: "cn-shanghai-aliyun-cloudauth-xxxxx",
        // ossObjectName: "verify/xxxxx/xxxxxx.jpeg",
        // 方式三：人脸图片的OSS URL地址。该地址必须公网可访问。
        faceContrastPictureUrl: "https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-shangha
i.aliyuncs.com/verify/xxxxx/xxxxx.jpeg"
        // mobile: "130xxxxxxxx",
        // ip: "114.xxx.xxx.xxx",
        // userId: "12345xxxx",
        // deviceToken: "xxxxxxx"
        });
        // 推荐，支持服务路由。
        const response = await livenessFaceVerifyAutoRoute(request);
        // 不支持服务自动路由
```

```
        // 不支持服务自动路由。
        // const response = await livenessFaceVerify("cloudauth.cn-shanghai.aliyuncs.com",
request);
        console.log(response.requestId);
        console.log(response.code);
        console.log(response.message);
        console.log(response.resultObject.certifyId);
        console.log(response.resultObject.materialInfo);
        console.log(response.resultObject.subCode);
        console.log(response.resultObject.passed);
    }
main().then(function (response) {});
```

5.金融级人脸比对方案

5.1. 接入流程

本文介绍了人脸比对服务端接入的详细操作。

前置条件

您必须了解通过HTTP、HTTPS原生调用金融级实人认证服务接口时，构造请求签名的方法。更多签名机制的信息，请参见API签名机制。

操作步骤

1. 开通金融级实人认证服务。
2. 添加认证场景。
3. 授权金融级实人认证访问OSS存储空间。
4. 进行服务端接入。

5.2. 纯服务端（API）接入

传入照片格式要求

当您在进行人脸比对的时候，请传入满足以下所有条件的2张人脸照片。

- 近期照片或底库近期照片，人脸完整清晰无遮挡、表情自然、正对摄像头。
- 照片清晰、曝光正常，脸部无过黑、过亮、光晕等情况、角度无太大偏差。
- 分辨率不超过1920*1080（高*宽），至少640*480，推荐短边缩放到720像素，压缩率大于0.9。照片高大于宽，如果传入的照片宽大于高，可能会影响检测效果。
- 照片大小：小于1 MB。
- 支持90、180和270度照片，多人脸的情况下会选取最大脸。

示例如下：

发起认证请求

接口名：**CompareFaceVerify**。

服务地址：cloudauth.aliyuncs.com。

请求方法：HTTPS POST和GET。

接口说明：通过服务端接入实现实人认证的接口。

请求参数

名称	类型	是否必选	描述	示例值
SceneId	Long	是	认证场景ID。	1000000006
OuterOrderNo	String	是	客户服务端自定义的业务唯一标识，用于后续定位排查问题使用。值最长为32位长度的字母数字组合，请确保唯一。	e0c34a77f5ac40a5aa5e6ed20c353888
ProductCode	String	是	固定值：*PV_FC*。	PV_FC

名称	类型	是否必选	描述	示例值
SourceFaceContrastPicture	String	必填，在以下四种传入人脸照片方式中，选择其中一种： • *FaceContrastPicture* • *FaceContrastPictureUrl* • *CertifyId* • *OSS*	照片Base64编码。	/9j/4AAQSkZJRgABAQAASxxxxxx
SourceFaceContrastPictureUrl	String		OSS照片地址，目前只支持已授权OSS照片地址。	https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg
SourceCertifyId	String		之前实人认证通过的CertifyId，认证时的照片作为人脸比对照片。	0bfa7c493f850e5178b9f8613634c9xx
SourceOssBucketName	String		已授权OSS空间的Bucket名。	cn-shanghai-aliyun-cloudauth-xxxxx
SourceOssObjectName	String		已授权OSS空间的文件名。	verify/xxxxx/xxxxxx.jpeg
TargetFaceContrastPicture	String	必填，在以下四种传入底库人脸照片方式中，选择其中一种： • *FaceContrastPicture* • *FaceContrastPictureUrl* • *CertifyId* • *OSS*	底库照片Base64编码。	/9j/4AAQSkZJRgABAQAASxxxxxx
TargetFaceContrastPictureUrl	String		底库OSS照片地址，目前只支持已授权OSS照片地址。	https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg
TargetCertifyId	String		之前实人认证通过的CertifyId，认证时的照片作为人脸比对照片。	0bfa7c493f850e5178b9f8613634c9xx
TargetOssBucketName	String		已授权OSS空间的Bucket名 。	cn-shanghai-aliyun-cloudauth-xxxxx
TargetOssObjectName	String		已授权OSS空间的文件名。	verify/xxxxx/xxxxxx.jpeg

名称	类型	是否必选	描述	示例值
Crop	String	否	是否允许裁剪人脸图片，默认不允许。 ● T：表示允许剪裁。 ● F：表示不允许剪裁。 说明 若您请求的图片并非来自标准活体检测SDK采集，建议您允许裁剪人脸图片。 开启该功能后，将会对请求的图片先进行人脸裁剪与矫正，完成后会发起请求服务。	T

说明

以Source开头的参数代表您要上传的近期人脸照片参数，以Target开头的参数代表您要上传的底库近期人脸照片参数。传入照片的格式要求请参见传入照片格式要求。

如果您选择OSS方式传入人脸照片，您必须同时传入OssBucketName和OssObjectName字段。

OSS入参格式仅支持开通实人认证时授权的OSS生成的URL、Bucket和文件名 。关于如何查询授权的OSS，请参见授权金融级实人认证访问OSS存储空间。

如果您选择*FaceContrastPicture*（照片Base64编码）方式传入人脸照片，请注意检查照片大小，不要传入过大的照片；您可以通过OSS方式上传较大的人脸照片。

返回数据

名称	类型	描述	示例值
RequestId	String	请求ID。	130A2C10-B9EE-4D84-88E3-5384FF039795
Message	String	错误信息。	success
Code	String	返回码：200表示成功，其他返回码表示失败。	200
ResultObject.Passed	String	认证结果。取值： ● *T*：认证通过。 ● *F*：认证失败。	T

名称	类型	描述	示例值
ResultObject.CertifyId	String	08573be80f944d95ac812e019e3655a8	实人认证请求的唯一标识。
ResultObject.VerifyScore	Float	人脸比对分。	99.60875

返回示例

```
{
    "Code": "200",
    "Message": "success",
    "RequestId": "23699272-B263-4B6C-BC04-0EBA1AB6F6D5",
    "ResultObject": {
        "Passed": "T",
        "VerifyScore": 99.60875
    }
}
```

返回Code和Message

Code	Message	描述
200	success	成功。
400	参数不能为空	参数不能为空。
401	参数非法	非法参数。传入的姓名、身份证号码长度必须符合国家标准且不得包含英文字母等特殊字符。
402	应用配置不存在	应用配置不存在。
404	认证场景配置不存在	认证场景配置不存在，请先在控制台上创建认证场景。
410	未开通服务	未开通OSS产品或未完成OSS读写授权，请登录控制台完成授权。
411	RAM 无权限	需要给子账号授予AliyunAntCloudAuthFullAccess的操作权限。
412	欠费中	金融级实人认证或OSS存在欠费，请充值后操作。
419	传入图片不可用	图片无法下载、图片内容为空、图片分辨率不符合要求或提取不到人脸特征，建议更换图片。
420	数据重复	图片传入方式超过2种。
421	传入图片过大	图片超过了1 MB，建议压缩图片或更换图片上传方式。
422	下载图片超时	图片下载超过了3秒，请排查网络后重新操作。

Code	Message	描述
423	状态错误	传入的CertifyId认证状态需要为T，您也可以更换其他方式传入图片。
500	系统错误	系统内部错误，请联系工程师排查。

5.3. Java SDK调用示例

本文介绍了金融级人脸对比方案的Java SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

调试CompareFaceVerify接口

本章节提供的是最新版本的Java SDK调用示例，如果您需要查看之前版本，请参见Java SDK调用示例（V1版本）。

前提条件

需要使用的Java版本在1.8版本及以上。

使用Maven引入SDK

```
<dependency>
       <groupId>com.aliyun</groupId>
       <artifactId>cloudauth20190307</artifactId>
        <version>1.0.5</version>
</dependency>
```

调用示例

```
import java.util.Arrays;
import java.util.List;
import com.aliyun.cloudauth20190307.Client;
import com.aliyun.cloudauth20190307.models.CompareFaceVerifyRequest;
import com.aliyun.cloudauth20190307.models.CompareFaceVerifyResponse;
import com.aliyun.tea.TeaException;
import com.aliyun.tea.TeaUnretryableException;
import com.aliyun.tearpc.models.Config;
import com.aliyun.teautil.models.RuntimeOptions;
public class CompareFaceVerify {
    public static void main(String[] args) {
        // 通过以下代码创建API请求并设置参数。
        CompareFaceVerifyRequest request = new CompareFaceVerifyRequest();
        // 请输入场景ID+L。
        request.setSceneId(xxxxL);
        request.setOuterOrderNo("商户请求的唯一标识");
        request.setProductCode("PV_FC");
        // 请在以下四种传入人脸照片方式中，选择一种。
        // 1. 照片Base64编码。
```

```
        // 1.照片Base64编码。
        // 2.照片OSS bucket名和文件名。
        // 3.照片OSS的URL地址。
        // 4.已经实人认证通过的CertifyId。
        // 方式一：照片Base64。
        request.setSourceFaceContrastPicture("人脸照片Base64编码");
        // 方式二：照片OSS bucket名和文件名。
        //request.setSourceOssBucketName("cn-shanghai-aliyun-cloudauth-xxxxx");
        //request.setSourceOssObjectName("verify/xxxxx/xxxxxx.jpeg");
        // 方式三：照片OSS的URL地址，公网可访问。
        //request.setSourceFaceContrastPictureUrl("https://cn-shanghai-aliyun-cloudauth-xxx
xxx.oss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg")
        // 方式四：已经实人认证通过的CertifyId。
        //request.setSourceCertifyId("0bfa7c493f850e5178b9f8613634c9xx");
        // 请在以下四种传入底库人脸照片方式中，选择一种
        // 1.照片Base64编码。
        // 2.照片OSS bucket名和文件名。
        // 3.照片OSS的URL地址。
        // 4.已经实人认证通过的CertifyId。
        // 方式一：照片Base64。
        request.setTargetFaceContrastPicture("底库人脸照片Base64编码");
        // 方式二：照片OSS bucket名和文件名。
        //request.setTargetOssBucketName("cn-shanghai-aliyun-cloudauth-xxxxx");
        //request.setTargetOssObjectName("verify/xxxxx/xxxxxx.jpeg");
        // 方式三：照片OSS的URL地址，公网可访问。
        //request.setTargetFaceContrastPictureUrl("https://cn-shanghai-aliyun-cloudauth-xxx
xxx.oss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg")
        // 方式四：已经实人认证通过的CertifyId。
        //request.setTargetCertifyId("0bfa7c493f850e5178b9f8613634c9xx");
        // 推荐，支持服务路由。
        CompareFaceVerifyResponse response = compareFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        //CompareFaceVerifyResponse response = compareFaceVerify("cloudauth.cn-shanghai.ali
yuncs.com", request);
        System.out.println(response.getRequestId());
        System.out.println(response.getCode());
        System.out.println(response.getMessage());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getPassed());
        System.out.println(response.getResultObject() == null ? null
            : response.getResultObject().getVerifyScore());
    }
    private static CompareFaceVerifyResponse compareFaceVerifyAutoRoute(CompareFaceVerifyRe
quest request) {
        // 第一个为主区域Endpoint，第二个为备区域Endpoint。
        List<String> endpoints = Arrays.asList("cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com");
        CompareFaceVerifyResponse lastResponse = null;
        for (String endpoint : endpoints) {
            try {
                CompareFaceVerifyResponse response = compareFaceVerify(endpoint, request);
                lastResponse = response;
                // 服务端错误，切换到下个区域调用。
                if (response != null && "500".equals(response.getCode())) {
                    continue;
```

```
                }
                return response;
            } catch (Exception e) {
                // 网络异常，切换到下个区域调用。
                if (e.getCause() instanceof TeaException) {
                    TeaException teaException = ((TeaException)e.getCause());
                    if (teaException.getData() != null && "ServiceUnavailable".equals(
                        teaException.getData().get("Code"))) {
                        continue;
                    }
                }
                if (e.getCause() instanceof TeaUnretryableException) {
                    continue;
                }
            }
        }
        return lastResponse;
    }
    private static CompareFaceVerifyResponse compareFaceVerify(String endpoint, CompareFace
VerifyRequest request)
        throws Exception {
        Config config = new Config();
        config.setAccessKeyId("您的AccessKey ID");
        config.setAccessKeySecret("您的AccessKey Secret");
        config.setEndpoint(endpoint);
        // 设置https代理。
        //config.setHttpsProxy("https://xx.xx.xx.xx:xxxx");
        // 设置http代理。
        //config.setHttpProxy("http://xx.xx.xx.xx:xxxx");
        Client client = new Client(config);
        // 创建RuntimeObject实例并设置运行参数。
        RuntimeOptions runtime = new RuntimeOptions();
        runtime.readTimeout = 10000;
        runtime.connectTimeout = 10000;
        return client.compareFaceVerify(request, runtime);
    }
}
```

5.4. PHP SDK调用示例

本文介绍了金融级人脸比对方案的PHP SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

调试CompareFaceVerify接口

前提条件

- 需要使用的PHP版本在5.6版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于PHP SDK源代码，请参见PHP SDK包。

- 通过Composer来管理项目依赖（推荐）

```
composer require alibabacloud/cloudauth-20190307 1.0.5
```

- 使用composer.json文件

```
{
   "require": {
       "alibabacloud/cloudauth-20190307": "1.0.5"
    }
}
```

如果因网络问题导致无法按照上述操作安装SDK，您可以使用阿里云Composer全量镜像。

调用示例

CompareFaceVerify接口调用示例

```
<?php
namespace AlibabaCloud\SDK\Sample {
use AlibabaCloud\SDK\Cloudauth\V20190307\Cloudauth;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\CompareFaceVerifyRequest;
use AlibabaCloud\SDK\Cloudauth\V20190307\Models\CompareFaceVerifyResponse;
use AlibabaCloud\Tea\Utils\Utils;
use \Exception;
use AlibabaCloud\Tea\Exception\TeaError;
use AlibabaCloud\Tea\Rpc\Rpc\Config;
use AlibabaCloud\Tea\Utils\Utils\RuntimeOptions;
class CompareFaceVerify {
    /**
     * @param string $endpoint
     * @return Cloudauth
     */
    public static function createClient($endpoint){
        $config = new Config([
            // 您的AccessKey ID。
            "accessKeyId" => "<您的AccessKey ID>",
            // 您的AccessKey Secret。
            "accessKeySecret" => "<您的AccessKey Secret>",
            // 设置HTTP代理。
            // "httpProxy" => "<http://xx.xx.xx.xx:xxxx>",
            // 设置HTTPS代理。
            // "httpsProxy" => "<https://xx.xx.xx.xx:xxxx>",
            "endpoint" => $endpoint
        ]);
        return new Cloudauth($config);
    }
    /**
     * @return void
     */
    public static function main(){
        $request = new CompareFaceVerifyRequest([
```

```
            // 请输入场景ID。
            "sceneId" => <xxx>,
            "outerOrderNo" => "<商户请求的唯一标识>",
            // 要接入的认证方案。
            "productCode" => "<PV_FC>",
            // 请在以下四种传入人脸照片方式中，选择一种。
            // 1.照片Base64编码。
            // 2.照片OSS Bucket名和文件名。
            // 3.照片OSS的URL地址。
            // 4.已经实人认证通过的CertifyId。
            // 方式一：照片Base64。
            // "sourceFaceContrastPicture" => "<人脸照片Base64编码>",
            // 方式二：照片OSS Bucket名和文件名。
            // "sourceOssBucketName" => "<cn-shanghai-aliyun-cloudauth-xxxxx>",
            // "sourceOssObjectName" => "<verify/xxxxx/xxxxxx.jpeg>",
            // 方式三：照片OSS的URL地址，公网可访问。
            "sourceFaceContrastPictureUrl" => "<https://cn-shanghai-aliyun-cloudauth-xxxxxx
.oss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg>",
            // 方式四：已经实人认证通过的CertifyId。
            // "sourceCertifyId" => "<0bfa7c493f850e5178b9f8613634c9xx>",
            // 请在以下四种传入底库人脸照片方式中，选择一种。
            // 1.照片Base64编码。
            // 2.照片OSS Bucket名和文件名。
            // 3.照片OSS的URL地址。
            // 4.已经实人认证通过的CertifyId。
            // 方式一：照片Base64。
            // "targetFaceContrastPicture" => "<人脸照片Base64编码>",
            // 方式二：照片OSS Bucket名和文件名。
            // "targetOssBucketName" => "<cn-shanghai-aliyun-cloudauth-xxxxx>",
            // "targetOssObjectName" => "<verify/xxxxx/xxxxxx.jpeg>",
            // 方式三：照片OSS的URL地址，公网可访问。
            "targetFaceContrastPictureUrl" => "<https://cn-shanghai-aliyun-cloudauth-xxxxxx
.oss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg>",
            // 方式四：已经实人认证通过的CertifyId。
            // "targetCertifyId" => "<0bfa7c493f850e5178b9f8613634c9xx>",
        ]);
        // 推荐，支持服务路由。
        $response = self::compareFaceVerifyAutoRoute($request);
        // 不支持服务自动路由。
        // $response = self::compareFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", $reque
st);
        var_dump($response->requestId);
        var_dump($response->code);
        var_dump($response->message);
        var_dump($response->resultObject->verifyScore);
        var_dump($response->resultObject->passed);
    }
    /**
     * @param CompareFaceVerifyRequest $request
     * @return CompareFaceVerifyResponse
     */
    public static function compareFaceVerifyAutoRoute(CompareFaceVerifyRequest $request){
        $endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
```

```
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        foreach($endpoints as $endpoint){
            try {
                $response = self::compareFaceVerify($endpoint, $request);
                if (Utils::equalString("500", $response->code)) {
                    continue;
                }
                return $response;
            }
            catch (Exception $err) {
                if (!($err instanceof TeaError)) {
                    $err = new TeaError([], $err->getMessage(), $err->getCode(), $err);
                }
                // 网络异常，切换到下个区域调用。
                if (Utils::equalString("ServiceUnavailable", $err->code)) {
                    continue;
                }
            }
        }
        return null;
    }
    /**
     * @param string $endpoint
     * @param CompareFaceVerifyRequest $request
     * @return CompareFaceVerifyResponse
     */
    public static function compareFaceVerify($endpoint, $request){
        $client = self::createClient($endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        $runtime = new RuntimeOptions([]);
        $runtime->readTimeout = 10000;
        $runtime->connectTimeout = 10000;
        return $client->compareFaceVerify($request, $runtime);
    }
}
$path = __DIR__ . \DIRECTORY_SEPARATOR . \DIRECTORY_SEPARATOR . 'vendor' . \DIRECTORY_SEPAR
ATOR . 'autoload.php';
if (file_exists($path)) {
    require_once $path;
}
CompareFaceVerify::main();
}
```

5.5. Python SDK调用示例

本文介绍了金融级人脸比对方案的Python SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

详细信息，请参见调试CompareFaceVerify接口。

前提条件

- 需要使用的Python版本在3.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

引入依赖包

```
pip install alibabacloud_tea_openapi==0.2.5
```

安装SDK

关于Python SDK源代码，请参见Python SDK包。

```
pip install alibabacloud_cloudauth20190307==1.0.5
```

调用示例

CompareFaceVerify接口调用示例

关于参数的具体描述，请参见纯服务端（API）接入。

```
from alibabacloud_cloudauth20190307.client import Client as CloudauthClient
from alibabacloud_tea_openapi import models as open_api_models
from alibabacloud_cloudauth20190307 import models as cloudauth_models
from alibabacloud_tea_util.client import Client as UtilClient
from alibabacloud_tea_util import models as util_models
class CompareFaceVerify:
    def __init__(self):
        pass
    @staticmethod
    def create_client(
        endpoint: str,
    ) -> CloudauthClient:
        config = open_api_models.Config(
            # 您的AccessKey ID。
            access_key_id='您的AccessKey ID',
            # 您的AccessKey Secret。
            access_key_secret='您的AccessKey Secret',
            # 设置HTTP代理。
            # http_proxy='http://xx.xx.xx.xx:xxxx',
            # 设置HTTPS代理。
            # https_proxy='https://xx.xx.xx.xx:xxxx',
            endpoint=endpoint
        )
        return CloudauthClient(config)
    @staticmethod
    def main() -> None:
        request = cloudauth_models.CompareFaceVerifyRequest(
            # 请输入场景ID。
            scene_id=xxx,
            outer_order_no='商户请求的唯一标识',
            product_code='PV_FC',
```

```
            # 请在以下四种传入人脸照片方式中，选择一种。
            # 1.照片Base64编码。
            # 2.照片OSS Bucket名和文件名。
            # 3.照片OSS的URL地址。
            # 4.已经实人认证通过的CertifyId。
            # 方式一：照片Base64。
            # source_face_contrast_picture='人脸照片Base64编码',
            # 方式二：照片OSS Bucket名和文件名。
            # source_oss_bucket_name='cn-shanghai-aliyun-cloudauth-xxxxx',
            # source_oss_object_name='verify/xxxxx/xxxxxx.jpeg',
            # 方式三：照片OSS的URL地址，公网可访问。
            source_face_contrast_picture_url='https=//cn-shanghai-aliyun-cloudauth-xxxxxx.o
ss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg',
            # 方式四：已经实人认证通过的CertifyId。
            # source_certify_id='0bfa7c493f850e5178b9f8613634c9xx',
            # 请在以下四种传入底库人脸照片方式中，选择一种：
            # 1.照片Base64编码。
            # 2.照片OSS Bucket名和文件名。
            # 3.照片OSS的URL地址。
            # 4.已经实人认证通过的CertifyId。
            # 方式一：照片Base64。
            # target_face_contrast_picture='底库人脸照片Base64编码'
            # 方式二：照片OSS Bucket名和文件名。
            # target_oss_bucket_name='cn-shanghai-aliyun-cloudauth-xxxxx',
            # target_oss_object_name='verify/xxxxx/xxxxxx.jpeg',
            # 方式三：照片OSS的URL地址，公网可访问。
            target_face_contrast_picture_url='https://cn-shanghai-aliyun-cloudauth-xxxxxx.o
ss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg',
            # 方式四：已经实人认证通过的CertifyId。
            # target_certify_id='0bfa7c493f850e5178b9f8613634c9xx'
        )
        response = CompareFaceVerify.compare_face_verify_auto_route(request)
        # 不支持服务自动路由。
        # response = CompareFaceVerify.compare_face_verify('cloudauth.cn-shanghai.aliyuncs.
com', request)
        print(response.request_id)
        print(response.code)
        print(response.message)
        print(response.result_object.passed)
        print(response.result_object.verify_score)
        print(response.result_object.certify_id)
    @staticmethod
    def compare_face_verify_auto_route(
        request: cloudauth_models.CompareFaceVerifyRequest,
    ) -> cloudauth_models.CompareFaceVerifyResponse:
        endpoints = [
            'cloudauth.cn-shanghai.aliyuncs.com',
            'cloudauth.cn-beijing.aliyuncs.com'
        ]
        for endpoint in endpoints:
            try:
                response = CompareFaceVerify.compare_face_verify(endpoint, request)
                if UtilClient.equal_string('500', response.code):
                    continue
```

```
                return response
            except Exception as err:
                    continue
        return None
    @staticmethod
    def compare_face_verify(
        endpoint: str,
        request: cloudauth_models.CompareFaceVerifyRequest,
    ) -> cloudauth_models.CompareFaceVerifyResponse:
        client = CompareFaceVerify.create_client(endpoint)
        # 创建RuntimeObject实例并设置运行参数。
        runtime = util_models.RuntimeOptions()
        runtime.read_timeout = 10000
        runtime.connect_timeout = 10000
        return client.compare_face_verify(request, runtime)
if __name__ == '__main__':
    CompareFaceVerify.main()
```

5.6. Go SDK调用示例

本文介绍了金融级人脸比对方案的Go SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

详细信息，请参见调试CompareFaceVerify接口。

前提条件

- 仅支持Go 1.12.0及以上版本。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

引入依赖包

```
go get github.com/alibabacloud-go/tea-rpc
go get github.com/alibabacloud-go/tea-utils
go get github.com/alibabacloud-go/tea
```

安装SDK

关于Go SDK源代码，请参见Go SDK包。

通过Go mod来管理项目依赖：

```
go get github.com/alibabacloud-go/cloudauth-20190307@vv1.0.5
```

调用示例

CompareFaceVerify接口调用示例

关于参数的具体描述，请参见纯服务端（API）接入。

```
package main
import (
  "fmt"
  cloudauth "github.com/alibabacloud-go/cloudauth-20190307/client"
  rpc "github.com/alibabacloud-go/tea-rpc/client"
  util "github.com/alibabacloud-go/tea-utils/service"
  "github.com/alibabacloud-go/tea/tea"
)
func CreateClient(endpoint *string) (_result *cloudauth.Client, _err error) {
  config := &rpc.Config{
    // 您的AccessKey ID。
    AccessKeyId: tea.String("您的AccessKey ID"),
    // 您的AccessKey Secret。
    AccessKeySecret: tea.String("您的AccessKey Secret"),
    Endpoint:        endpoint,
    // 设置HTTP代理。
    // HttpProxy: tea.String("http://xx.xx.xx.xx:xxxx"),
    // 设置HTTPS代理。
    // HttpsProxy: tea.String("https://xx.xx.xx.xx:xxxx"),
  }
  _result = &cloudauth.Client{}
  _result, _err = cloudauth.NewClient(config)
  return _result, _err
}
func _main() (_err error) {
  request := &cloudauth.CompareFaceVerifyRequest{
    // 请输入场景ID。
    SceneId:      tea.Int64(xxx),
    OuterOrderNo: tea.String("商户请求的唯一标识"),
    ProductCode:  tea.String("PV_FC"),
    // 请在以下四种传入人脸照片方式中，选择一种。
    // 1.照片Base64编码。
    // 2.照片OSS Bucket名和文件名。
    // 3.照片OSS的URL地址。
    // 4.已经实人认证通过的CertifyId。
    // 方式一：照片Base64。
    // SourceFaceContrastPicture: tea.String("人脸照片Base64编码"),
    // 方式二：照片OSS Bucket名和文件名。
    // SourceOssBucketName: tea.String("cn-shanghai-aliyun-cloudauth-xxxxx"),
    // SourceOssBucketName: tea.String("verify/xxxxx/xxxxxx.jpeg"),
    // 方式三：照片OSS的URL地址，公网可访问。
    SourceFaceContrastPictureUrl: tea.String("https://cn-shanghai-aliyun-cloudauth-xxxxxx.o
ss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg"),
    // 方式四：已经实人认证通过的CertifyId。
    // SourceCertifyId: tea.String("0bfa7c493f850e5178b9f8613634c9xx"),
    // 请在以下四种传入底库人脸照片方式中，选择一种
    // 1.照片Base64编码。
    // 2.照片OSS Bucket名和文件名。
    // 3.照片OSS的URL地址。
    // 4.已经实人认证通过的CertifyId。
    // 方式一：照片Base64。
    // TargetFaceContrastPicture: tea.String("人脸照片Base64编码"),
    // 方式二：照片OSS Bucket名和文件名。
    // TargetOssBucketName: tea.String("cn-shanghai-aliyun-cloudauth-xxxxx"),
```

```
    // TargetOssObjectName: tea.String("verify/xxxxx/xxxxxx.jpeg"),
    // 方式三：照片OSS的URL地址，公网可访问。
    TargetFaceContrastPictureUrl: tea.String("https://cn-shanghai-aliyun-cloudauth-xxxxxx.o
ss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg"),
    // 方式四：已经实人认证通过的CertifyId。
    // TargetCertifyId: tea.String("0bfa7c493f850e5178b9f8613634c9xx")
  }
  // 推荐，支持服务路由。
  response, _err := CompareFaceVerifyAutoRoute(request)
  // 不支持服务自动路由。
  // response, _err := CompareFaceVerify(tea.String("cloudauth.cn-shanghai.aliyuncs.com"),
request)
  if _err != nil {
    return _err
  }
  if response != nil {
    fmt.Println(*response.RequestId)
    fmt.Println(*response.Code)
    fmt.Println(*response.Message)
    fmt.Println(*response.ResultObject.CertifyId)
    fmt.Println(*response.ResultObject.Passed)
    fmt.Println(*response.ResultObject.VerifyScore)
  }
  return _err
}
func CompareFaceVerifyAutoRoute(request *cloudauth.CompareFaceVerifyRequest) (_result *clou
dauth.CompareFaceVerifyResponse, _err error) {
  // 第一个为主区域Endpoint，第二个为备区域Endpoint。
  endpoints := []*string{tea.String("cloudauth.cn-shanghai.aliyuncs.com"), tea.String("clou
dauth.cn-beijing.aliyuncs.com")}
  var lastResponse *cloudauth.CompareFaceVerifyResponse
  for _, endpoint := range endpoints {
    response, _err := CompareFaceVerify(endpoint, request)
    if _err != nil {
      var err = &tea.SDKError{}
      if _t, ok := _err.(*tea.SDKError); ok {
        err = _t
        // 网络异常，切换到下个区域调用。
        if tea.BoolValue(util.EqualString(tea.String("ServiceUnavailable"), err.Code)) {
          continue
        }
      }
      return _result, _err
    }
    // 服务端错误，切换到下个区域调用。
    if tea.BoolValue(util.EqualString(tea.String("500"), response.Code)) {
      continue
    }
    _result = response
    return _result, _err
  }
  _result = lastResponse
  return _result, _err
}
func CompareFaceVerify(endpoint *string, request *cloudauth.CompareFaceVerifyRequest) (_res
```

```
func CompareFaceVerify(endpoint *string, request *cloudauth.CompareFaceVerifyRequest) (_res
ult *cloudauth.CompareFaceVerifyResponse, _err error) {
  client, _err := CreateClient(endpoint)
  if _err != nil {
    return _result, _err
  }
  // 创建RuntimeObject实例并设置运行参数。
  runtime := &util.RuntimeOptions{}
  runtime.ReadTimeout = tea.Int(10000)
  runtime.ConnectTimeout = tea.Int(10000)
  _body, _err := client.CompareFaceVerify(request, runtime)
  if _err != nil {
    return _result, _err
  }
  _result = _body
  return _result, _err
}
func main() {
  err := _main()
  if err != nil {
    panic(err)
  }
}
```

5.7. .NET SDK调用示例

本文介绍了金融级人脸比对方案的.NET SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

详细信息，请参见调试CompareFaceVerify接口。

前提条件

- 需要使用的.NET Framework版本在4.5.0版本及以上或者.NET Standard版本在2.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于.NET SDK源代码，请参见.NET SDK包。

SDK有以下两种安装方式，请选择合适的方式安装：

- 通过.NET命令行安装。

```
dotnet add package AlibabaCloud.SDK.Cloudauth20190307 --version 1.0.5
```

- 使用包管理安装。

```
Install-Package AlibabaCloud.SDK.Cloudauth20190307 -Version 1.0.5
```

调用示例

CompareFaceVerify接口调用示例

关于参数的具体描述，请参见纯服务端（API）接入。

```
using System;
using System.Collections.Generic;
using Tea;
namespace CompareFaceVerify
{
    public class Program
    {
        public static AlibabaCloud.SDK.Cloudauth20190307.Client CreateClient(string endpoin
t)
        {
            AlibabaCloud.RPCClient.Models.Config config = new AlibabaCloud.RPCClient.Models
.Config
            {
                // 您的AccessKey ID。
                AccessKeyId = "您的AccessKey ID",
                // 您的AccessKey Secret。
                AccessKeySecret = "您的AccessKey Secret",
                // 设置HTTP代理。
                // HttpProxy = "http://xx.xx.xx.xx:xxxx",
                // 设置HTTPS代理。
                // HttpsProxy = "https://xx.xx.xx.xx:xxxx",
                Endpoint = endpoint,
            };
            return new AlibabaCloud.SDK.Cloudauth20190307.Client(config);
        }
        public static void Main(string[] args)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Models.CompareFaceVerifyRequest request = ne
w AlibabaCloud.SDK.Cloudauth20190307.Models.CompareFaceVerifyRequest
            {
                // 请输入场景ID。
                SceneId = xxx,
                OuterOrderNo = "商户请求的唯一标识",
                ProductCode = "PV_FC",
                // 请在以下四种传入人脸照片方式中，选择一种。
                // 1.照片Base64编码。
                // 2.照片OSS Bucket名和文件名。
                // 3.照片OSS的URL地址。
                // 4.已经实人认证通过的CertifyId。
                // 方式一：照片Base64。
                //SourceFaceContrastPicture = "人脸照片Base64编码",
                // 方式二：照片OSS Bucket名和文件名。
                // SourceOssBucketName = "cn-shanghai-aliyun-cloudauth-xxxxx",
                // SourceOssBucketName = "verify/xxxxx/xxxxxx.jpeg",
                // 方式三：照片OSS的URL地址，公网可访问。
                SourceFaceContrastPictureUrl = "https=//cn-shanghai-aliyun-cloudauth-xxxxxx
.oss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg",
                // 方式四：已经实人认证通过的CertifyId。
                // SourceCertifyId = "0bfa7c493f850e5178b9f8613634c9xx",
                // 请在以下四种传入底库人脸照片方式中，选择一种
                // 1.照片Base64编码。
```

```
                // 2.照片OSS Bucket名和文件名。
                // 3.照片OSS的URL地址。
                // 4.已经实人认证通过的CertifyId。
                // 方式一：照片Base64。
                //TargetFaceContrastPicture = "底库人脸照片Base64编码",
                // 方式二：照片OSS Bucket名和文件名。
                // TargetOssBucketName = "cn-shanghai-aliyun-cloudauth-xxxxx",
                // TargetOssObjectName = "verify/xxxxx/xxxxxx.jpeg",
                // 方式三：照片OSS的URL地址，公网可访问。
                TargetFaceContrastPictureUrl = "https://cn-shanghai-aliyun-cloudauth-xxxxxx
.oss-cn-shanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg",
                // 方式四：已经实人认证通过的CertifyId。
                // TargetCertifyId = "0bfa7c493f850e5178b9f8613634c9xx",
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.CompareFaceVerifyResponse response =
CompareFaceVerifyAutoRoute(request);
            // 不支持服务自动路由。
            // AlibabaCloud.SDK.Cloudauth20190307.Models.CompareFaceVerifyResponse response
= CompareFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", request);
            Console.WriteLine(response.RequestId);
            Console.WriteLine(response.Code);
            Console.WriteLine(response.Message);
            Console.WriteLine(response.ResultObject.Passed);
            Console.WriteLine(response.ResultObject.VerifyScore);
            Console.WriteLine(response.ResultObject.CertifyId);
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.CompareFaceVerifyResponse C
ompareFaceVerifyAutoRoute(AlibabaCloud.SDK.Cloudauth20190307.Models.CompareFaceVerifyReques
t request)
        {
            List<string> endpoints = new List<string>
            {
                "cloudauth.cn-shanghai.aliyuncs.com",
                "cloudauth.cn-beijing.aliyuncs.com"
            };
            AlibabaCloud.SDK.Cloudauth20190307.Models.CompareFaceVerifyResponse lastRespons
e = null;
            foreach (var endpoint in endpoints)
            {
                try
                {
                    AlibabaCloud.SDK.Cloudauth20190307.Models.CompareFaceVerifyResponse res
ponse = CompareFaceVerify(endpoint, request);
                    if (AlibabaCloud.TeaUtil.Common.EqualString("500", response.Code))
                    {
                        continue;
                    }
                    return response;
                }
                catch (TeaException err)
                {
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
```

```
                    {
                        continue;
                    }
                }
                catch (Exception _err)
                {
                    TeaException err = new TeaException(new Dictionary<string, object>
                    {
                        { "message", _err.Message }
                    });
                    // 网络异常，切换到下个区域调用。
                    if (AlibabaCloud.TeaUtil.Common.EqualString("ServiceUnavailable", err.C
ode))
                    {
                        continue;
                    }
                }
            }
            return lastResponse;
        }
        public static AlibabaCloud.SDK.Cloudauth20190307.Models.CompareFaceVerifyResponse C
ompareFaceVerify(string endpoint, AlibabaCloud.SDK.Cloudauth20190307.Models.CompareFaceVeri
fyRequest request)
        {
            AlibabaCloud.SDK.Cloudauth20190307.Client client = CreateClient(endpoint);
            // 创建RuntimeObject实例并设置运行参数。
            AlibabaCloud.TeaUtil.Models.RuntimeOptions runtime = new AlibabaCloud.TeaUtil.M
odels.RuntimeOptions();
            runtime.ReadTimeout = 10000;
            runtime.ConnectTimeout = 10000;
            return client.CompareFaceVerify(request, runtime);
        }
    }
}
```

5.8. Node.js SDK调用示例

本文介绍了金融级人脸比对方案的Node.js SDK调用示例。

SDK使用说明

如果需要其他语言的SDK示例代码，您可以通过OpenAPI开发者门户在线调试工具调试API接口。该工具会自动生成相应API的SDK调用示例代码。

详细信息，请参见调试CompareFaceVerify接口。

前提条件

- 需要使用的Node.js版本在8.0版本及以上。
- 已添加认证场景，并获取对应的场景ID。具体操作，请参见添加认证场景。

安装SDK

关于Node.js SDK源代码，请参见Node.js SDK。

建议您使用npm管理依赖包：

```
npm install @alicloud/tea-util
npm install @alicloud/rpc-client
npm install @alicloud/cloudauth20190307@1.0.5
```

调用示例

CompareFaceVerify接口调用示例

关于参数的具体描述，请参见纯服务端（API）接入。

```
const Config = require("@alicloud/rpc-client");
const CloudAuth = require("@alicloud/cloudauth20190307");
const CloudAuthClient = CloudAuth.default;
const {RuntimeOptions} = require("@alicloud/tea-util");
const fs = require("fs");
     function createClient(endpoint) {
        // 通过以下代码创建Config实例并初始化。
        const config = new Config.Config({
            // 您的AccessKey ID。
            accessKeyId: "您的AccessKey ID",
            // 您的AccessKey Secret。
            accessKeySecret: "您的AccessKey Secret",
            // 设置Endpoint。
            endpoint: endpoint,
            // 设置HTTP代理。
            // httpProxy: "http://xxx.xxx.xxx.xxx:9999",
            // 设置HTTPS代理。
            // httpsProxy: "https://username:password@xxx.xxx.xxx.xxx:9999",
        });
        return new CloudAuthClient(config)
    }
    async function compareFaceVerify(endpoint, request) {
        const client = createClient(endpoint);
        // 创建RuntimeObject实例并设置运行参数。
        const runtime = new RuntimeOptions([]);
        runtime.readTimeout=10000;
        runtime.connectTimeout=10000;
        return await client.compareFaceVerify(request, runtime);
    }
    async function compareFaceVerifyAutoRoute(request) {
        const endpoints = [
            "cloudauth.cn-shanghai.aliyuncs.com",
            "cloudauth.cn-beijing.aliyuncs.com"
        ];
        for (const endpoint of endpoints) {
            try {
                const response = await compareFaceVerify(endpoint, request);
                if (response.code === '500') {
                    continue;
                }
                return response;
```

```
            } catch (err) {
                continue;
            }
        }
        return null;
    }
    async function main(){
        // 通过以下代码创建API请求并设置参数。
        const request = new CloudAuth.CompareFaceVerifyRequest({
        // 请输入场景ID。
        sceneId: xxx,
        // 设置商户请求的唯一标识。
        outerOrderNo: "xxx",
        // 要接入的认证方案。
        productCode: "PV_FC",
        // 方式一：照片Base64。
        // sourceFaceContrastPicture: "人脸照片Base64编码",
        // 方式二：照片OSS bucket名和文件名。
        // sourceOssBucketName: "cn-shanghai-aliyun-cloudauth-xxxxx",
        // sourceOssObjectName: "verify/xxxxx/xxxxxx.jpeg",
        // 方式三：照片OSS的URL地址，公网可访问。
        sourceFaceContrastPictureUrl: "https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-s
hanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg",
        // 方式四：已经实人认证通过的CertifyId。
        // sourceCertifyId: "0bfa7c493f850e5178b9f8613634c9xx",
        // 请在以下四种传入底库人脸照片方式中，选择一种
        // 1.照片Base64编码。
        // 2.照片OSS bucket名和文件名。
        // 3.照片OSS的URL地址。
        // 4.已经实人认证通过的CertifyId。
        // 方式一：照片Base64。
        // targetFaceContrastPicture: "人脸照片Base64编码",
        // 方式二：照片OSS bucket名和文件名。
        // targetOssBucketName: "cn-shanghai-aliyun-cloudauth-xxxxx",
        // targetOssObjectName: "verify/xxxxx/xxxxxx.jpeg",
        // 方式三：照片OSS的URL地址，公网可访问。
        targetFaceContrastPictureUrl: "https://cn-shanghai-aliyun-cloudauth-xxxxxx.oss-cn-s
hanghai.aliyuncs.com/verify/xxxxx/xxxxx.jpeg",
        // 方式四：已经实人认证通过的CertifyId。
        // targetCertifyId: "0bfa7c493f850e5178b9f8613634c9xx"
        });
        // 推荐，支持服务路由。
        const response = await compareFaceVerifyAutoRoute(request);
        // 不支持服务自动路由。
        // const response = await compareFaceVerify("cloudauth.cn-shanghai.aliyuncs.com", r
equest);
        console.log(response.requestId);
        console.log(response.code);
        console.log(response.message);
        console.log(response.resultObject.verifyScore);
        console.log(response.resultObject.passed);
    }
main().then(function (response) {});
```