阿里云
金融级实人认证

金融版开发指南

文档版本：20200413

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. 阿里云文档中所有内容，包括但不限于图片、架构设计、页面布局、文字描述，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	 **禁止：** 重置操作将丢失用户配置数据。
	该类警示信息可能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。
	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。
	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。	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 服务端接入指南

1.1 服务端接入

当您需要在 iOS 应用，Android 应用，支付宝小程序，或 H5 应用中集成金融级实人认证时，请参考本文完成服务端接入。

调用方式

金融级实人认证产品支持以下两种调用方式：

1. SDK 调用。关于如何通过 SDK 方式在服务端集成金融级实人认证，请参考：SDK 调用接入。

   说明：

   金融级实人认证提供了 Java、Python、PHP、C#、Golang、 Node.js 和 Ruby 语言的 SDK。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。

2. HTTPS 原生调用。关于如何通过 HTTPS 原生调用方式在服务端集成金融级实人认证，请参考：HTTPS 原生调用。

后续步骤

参看文档进行客户端接入开发：

- Android客户端接入；
- iOS 客户端接入；
- #unique_9；
- 支付宝 H5 页面接入；
- 端外唤起支付宝认证页面接入。

1.2 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。

发起认证请求字段说明

发起认证请求时，传入以下字段：

名称	类型	是否必需	描述	示例值
method	String	是	发起认证请求的操作。 取值必须是init。	init
sceneId	String	是	认证场景ID，在控制台创建认证场景后自动生成。	1000000006
outerOrderNo	String	是	商户请求的唯一标识，值为32位长度的字母数字组合前面几位字符是商户自定义的简称，中间可以使用一段时间，后段可以使用一个随机或递增序列。	e0c34a77f5ac40a5aa5e6ed20c353888
bizCode	String	是	认证场景码和用户发起认证的端有关： • 当用户在 iOS 或安卓平台发起认证时，认证场景码是FACE_SDK。 • 当用户在小程序中或 H5 页面中发起认证时，认证场景码是FACE。	FACE_SDK
identityType	String	是	身份信息参数类型，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。	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 端外唤起支付宝认证页面接入，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的应用的Scheme。详情请参看：回跳原应用。	https://www.aliyun.com

发起认证后，您会收到响应。相应参数列表如下：

名称	类型	是否必需	描述	示例值
certifyId	String	是	认证ID，刷脸认证唯一标识。	7eff3ad26a9c7b68c511b9f35eb1a354
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，传入以下字段：

名称	类型	是否必需	描述	示例值
method	String	是	查询人脸比对结果的操作。 取值必须为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。	7eff3ad26a9c7b68c511b9f35eb1a354
sceneId	String	是	认证场景ID，需与发起认证请求时的sceneId保持一致。	1000000006

发起查询后，您会收到响应。相应参数列表如下：

名称	类型	是否必需	描述	示例值
passed	String	是	是否通过，通过为T，不通过为F。	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":"330103xxxxxxxxxxx","cert_name":"张三"}"

名称	类型	是否必需	描述	示例值
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\"}}"

SDK 使用说明

1. JAVA SDK

下载 JAVA SDK 源码：aliyun-openapijava-sdk。

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 的环境：阿里云SDK开发指南;
- 您可以在此处获得对应的 Maven 依赖：站点1 和 站点2。

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 FACE_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", "330103xxxxxxxxxxx");
        serviceParams.put("certName", "张三");
        serviceParams.put("returnUrl", "https://www.aliyun.com");

        /*
        // 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();
        }
    }
}
```

2. Python SDK

下载 Python SDK 源码。

1. 安装 SDK 核心库。

   - 如果您使用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
     ```

2. 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、安装使用可参考：阿里云SDK开发指南。

3. PHP SDK

引用 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 access-key-id 和 access-key-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"=>      "330011200102257788",
   "certName"=>    "拉斯穆",
   "returnUrl"=>   "https://www.aliyun.com");
// --

try {
   // -- 下述参数无需调整
   $result = AlibabaCloud::rpcRequest()
                    ->product('saf')
                    ->scheme('https')
                    ->version('2017-03-31')
                    ->action('ExecuteRequest')
                    ->method('POST')
                    ->host('saf.cn-shanghai.aliyuncs.com')
                    ->options([
                            'query' => [
                               'Service' => 'fin_face_verify',
                               'ServiceParameters' => json_encode($serviceParams)
                            ],
                        ])
                    ->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、安装使用可参考：阿里云SDK开发指南。

4. C# SDK

下载 C# SDK 源码。

参考 C# SDK 的环境准备和安装使用：阿里云SDK开发指南。

5. Golang SDK

下载 Golang SDK。

Golang 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 access-key-id 和 access-key-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":      "330011202002257788",
        "certName":    "罗伯特·格瑞史莫",
        "returnUrl":   "https://www.aliyun.com",
    }
    // --
    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、安装使用可参考：阿里云SDK开发指南。

6. Node.js SDK

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 access-key-id 和 access-key-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":      "330011200102257788",
    "certName":    "瑞安·达尔",
    "returnUrl":   "https://www.aliyun.com"
}
// --

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、安装使用可参考：阿里云SDK开发指南。

7. Ruby SDK

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、安装使用可参考：阿里云SDK开发指南。

1.3 HTTPS 原生调用

调用方式

接口基本信息：

- 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	是否必需	描述	示例值
method	String	是	发起认证请求的操作。 取值必须是init。	init
sceneId	String	是	认证场景ID，在控制台创建认证场景后自动生成。	1000000006
outerOrderNo	String	是	商户请求的唯一标识，值为32位长度的字母数字组合前面几位字符是商户自定义的简称，中间可以使用一段时间，后段可以使用一个随机或递增序列。	e0c34a77f5ac40a5aa5e6ed20c353888
bizCode	String	是	认证场景码和用户发起认证的端有关： • 当用户在 iOS 或安卓平台发起认证时，认证场景码是FACE_SDK。 • 当用户在小程序中或 H5 页面中发起认证时，认证场景码是FACE。	FACE_SDK
identityType	String	是	身份信息参数类型，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。	330103xxxxxxxxxxxx
certName	String	是	用户姓名。	张三

名称	类型	是否必需	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 端外唤起支付宝认证页面接入，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的应用的Scheme。详情请参看：回跳原应用。	https://www.aliyun.com

发起认证后，您会收到响应。相应参数列表如下：

名称	类型	是否必需	描述	示例值
certifyId	String	是	认证ID，刷脸认证唯一标识。	7eff3ad26a9c7b68c511b9f35eb1a354
certifyUrl	String	是	认证流程入口 URL。	https://picker.antcloudauth.aliyuncs.com/gateway.do?...

响应示例如下：

```
{
  "code": 200,
  "requestId": "5CD69C04-8A6B-4CC3-A0C9-D18D5FEFA788",
  "data": {
    "certifyUrl": "https://picker.antcloudauth.aliyuncs.com/gateway.do?...",
    "certifyId": "7eff3ad26a9c7b68c511b9f35eb1a354"
  },
  "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	是否必需	描述	示例值
method	String	是	查询人脸比对结果的操作。 取值必须为query。	query

名称	类型	是否必需	描述	示例值
certifyId	String	是	认证ID，需与发起认证请求时返回的certifyId保持一致。	7eff3ad26a9c7b68c511b9f35eb1a354
sceneId	String	是	认证场景ID，需与发起认证请求时的sceneId保持一致。	1000000006

发起查询后，您会收到响应。相应参数列表如下：

名称	类型	是否必需	描述	示例值
passed	String	是	是否通过，通过为T，不通过为F。	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":"330103xxxxxxxxxxxx","cert_name":"张三"}"
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

在使用 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	是否必需	描述
Format	String	是	返回值的类型，支持JSON与XML，默认为XML。

名称	类型	是否必需	描述
Version	String	是	API版本号，为日期形式：YYYY-MM-DD，本版本对应为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#unique_12。
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，目前支持HMAC-SHA1。
Timestamp	String	是	请求的时间戳。日期格式按照ISO8601标准表示，并需要使用UTC时间。格式为：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。具体定义见下面各接口参数定义。

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**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	OK
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，或已到期、到量、未购买。
404	认证场景配置不存在

Code	Message
405	以下两种原因之一： • 身份认证记录不存在； • 您的年龄未满14周岁，不允许使用此产品。
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",  "1000000006");
        serviceParams.put("outerOrderNo",  "e0c34a77f5ac40a5aa5e6ed20c353888");
        serviceParams.put("bizCode", "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");

        /*
        // 查询认证结果
        serviceParams.put("method",  "query");
        serviceParams.put("certifyId", "7eff3ad26a9c7b68c511b9f35eb1a354");
        serviceParams.put("sceneId", "1000000006");
        */

        Map<String, String> params = new HashMap<>();
        // 接口业务参数-固定 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(servicePar
ameters));

        doPOPRequest(appKey, appSecret, safUrl, "ExecuteRequest", "2017-03-31", params);
    }

    public static String doPOPRequest(String appKey, String appSecret, String url, String
action, 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-DDThh: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
encoding) 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().getContent()));
        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,
String 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.
toString().getBytes(encoding));
    }

    private static String encode(String value, String encoding) throws Unsupporte
dEncodingException {
        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").
replace("%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 API签名机制

本文介绍了通过 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。

对于每一次HTTP或者HTTPS协议请求，我们会根据访问中的签名信息验证访问请求者身份。

具体由使用 AccessKeyID 和 AccessKeySecret 对称加密验证实现。AccessKey 的获取请参见#unique_12。

步骤一：构造规范化请求字符串

1. 参数排序。 将所有公共请求参数按照首字母顺序排序。

> **说明:**
> 当使用 GET 方法提交请求时，这些参数就是请求 URL 中的参数部分，即 URL 中? 之后由 & 连接的部分。

2. 参数编码。 使用 UTF-8 字符集按照RFC3986规则编码请求参数和参数取值，编码规则如下：
   - 不编码以下内容：字符 A~Z、a~z、0~9 以及字符 -、_、.、~。
   - 其它字符编码成 %XY 的格式，其中 XY 是字符对应 ASCII 码的16 进制。示例：半角双引号（"）对应 %22。
   - 扩展的 UTF-8 字符，编码成 %XY%ZA... 的格式。
   - 空格（ ）编码成 %20，而不是加号（+）。该编码方式与application/x-www-form-urlencodedMIME格式编码算法相似，但又有所不同。如果您使用的是 Java 标准库中的java.net.URLEncoder，可以先用标准库中 percentEncode 编码，然后将编码后的字符中加号（+）替换为%20、星号（*）替换为%2A、%7E替换为波浪号（~），即可得到上述规则描述的编码字符串。

```
private static final String ENCODING = "UTF-8";
private static String percentEncode(String value) throws UnsupportedEncodingE
xception {
  return value != null ? URLEncoder.encode(value, ENCODING).replace("+", "%20").
replace("*", "%2A").replace("%7E", "~") : null;
}
```

3. 使用等号（=）连接编码后的请求参数和参数取值。
4. 使用与号（&）连接编码后的请求参数，注意参数排序与第1步描述一致。

通过以上步骤得到了规范化请求字符串（CanonicalizedQueryString），其结构遵循请求结构的要求。

步骤二：构造签名字符串

1. 构造待签名字符串 StringToSign。您可以同样使用 percentEncode 处理上一步构造的规范化请求字符串，规则如下：

```
StringToSign=
  HTTPMethod + "&" + //HTTPMethod：发送请求的 HTTP 方法，例如 GET。
  percentEncode("/") + "&" + //percentEncode("/")：字符（/）UTF-8 编码得到的值，即 %
2F。
  percentEncode(CanonicalizedQueryString) //您的规范化请求字符串。
```

2. 按照RFC2104的定义，计算待签名字符串 StringToSign 的 HMAC-SHA1 值。示例中使用的是 Java Base64 编码方法。

```
Signature = Base64( HMAC-SHA1( AccessSecret, UTF-8-Encoding-Of(StringToSign) ) )
```

说明：

计算签名时，RFC2104 规定的 Key 值是您的 AccessKeySecret 并加上与号（&），其 ASCII 值为38。更多信息，请参见#unique_12。

3. 添加根据 RFC3986 规则编码后的参数Signature到规范化请求字符串URL中。

2 无线SDK接入

2.1 接入流程

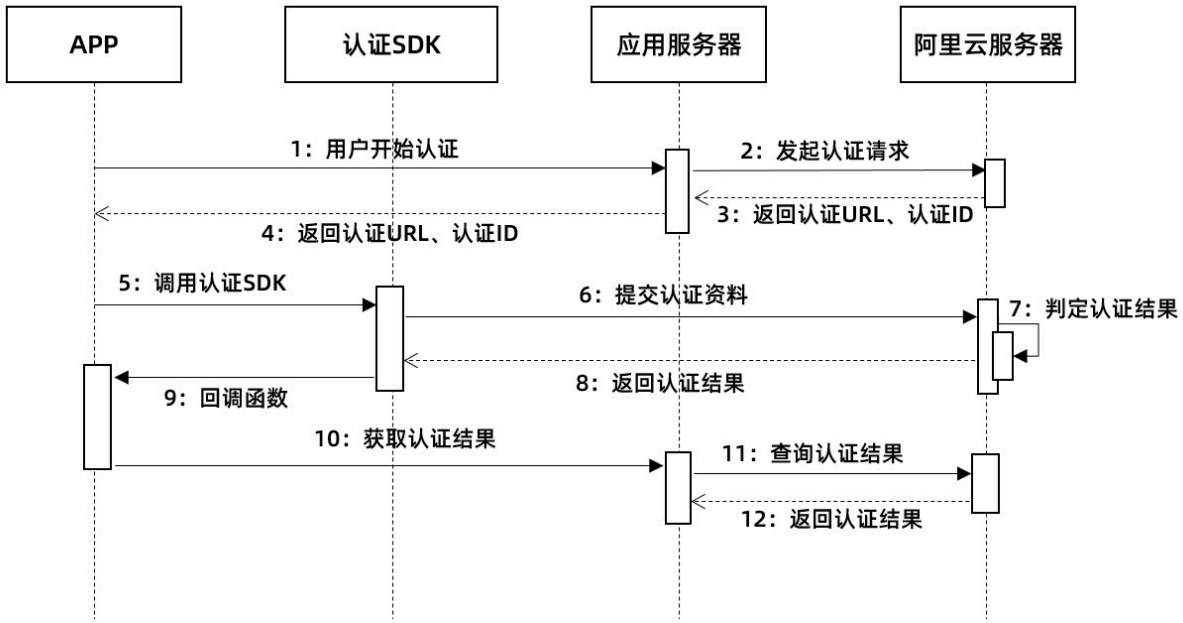
无线SDK+服务端接入适用于业务使用方自身已有手机APP应用，且希望通过该APP对用户进行线上认证的场景。

背景信息

如果您希望在 Android 或 iOS 应用中认证用户身份，您可以接入金融级实人认证服务。

调用时序

在接入前，请查看调用时序图：



系统调用时序说明：

序号	说明
1	用户开始认证。
2	应用服务端向金融级实人认证服务端发起认证请求。
3	金融级实人认证服务端返回认证 ID 和认证 URL 给应用服务端。
4	应用服务端将认证 ID 和认证 URL 传递给应用客户端。
5	应用客户端调用集成的金融级实人认证 SDK。
6	应用客户端将用户提交的认证资料传递给金融级实人认证服务端。

序号	说明
7	金融级实人认证服务端根据资料判定认证结果。
8	金融级实人认证服务端将认证结果返回给 SDK。
9	SDK 通过回调函数指引应用客户端获取认证状态。
10	应用客户端向服务端查询认证状态。
11	应用服务端向金融级实人认证服务端向发起查询。
12	金融级实人认证服务端将认证结果返回给应用服务端。

集成步骤

1. 在应用客户端集成相应 SDK。具体集成步骤，请根据平台系统，参见以下相应文档。
   - Android客户端接入
   - iOS 客户端接入
2. 在应用服务端集成相应 API。具体请参见服务端接入。

2.2 Android客户端接入

本文指导您在 Android 应用中集成金融级实人认证服务。

前提条件

应用必须在 Android 4.3+ 平台上运行。

获取 SDK 和 Demo 代码

- 下载 Android SDK 1.0.2 版本
- 下载 Android demo 1.0.2 版本

添加 SDK 依赖

如果您的应用客户端不依赖 mPaaS 基础库，则添加标准版 SDK；如果您的应用客户端依赖 mPaaS 基础库，则 添加 mPaaS 版 SDK。

当前 SDK 仅支持 `armeabi` 的 so 文件。如果您的应用中还有支持其他架构的 SDK，请添加以下代码以保留 `armeabi`：

```
ndk {
    abiFilters "armeabi"
}
```

添加标准版 SDK

1. 解压 AlipayVerifySDK-1.0.2.zip，将所有 .aar 包放入 libs 目录下：

2. 在工程的 build.gradle 文件中添加 libs 目录作为依赖仓库。

```
allprojects {
  repositories {
     // 添加以下内容
     flatDir {
        dirs 'libs'
     }
  }
}
```

3. 在应用的 build.gradle 文件中添加配置

```
android {
   // 添加以下内容
   useLibrary 'org.apache.http.legacy'
}
```

4. 在应用的 build.gradle 文件中添加依赖。

```
dependencies {
   // 添加如下内容
   compile 'com.android.support:appcompat-v7:26.1.0'
   compile (name:'verifysdk-1.0.0.190522154510', ext:'aar')
   compile (name:'releaseLoging-4.0.0.00000003', ext:'aar')
   compile (name:'bio-7.1.0.00000001', ext:'aar')
   compile (name:'hardware-7.1.0.00000001', ext:'aar')
   compile (name:'toyger-7.1.0.00000001', ext:'aar')
   compile (name:'zface-7.1.0.00000001', ext:'aar')
   compile (name:'zim-7.1.0.00000001', ext:'aar')
   compile (name:'rpc-2.1.0.180302121215-ZOLOZ', ext:'aar')
   compile (name:'zolozsdk-7.1.0.00000003', ext:'aar')
   compile (name:'securitysdk-open-6.0.5.20191014', ext:'aar')
   compile 'com.alibaba:fastjson:1.2.8@jar'
   compile 'com.squareup:otto:1.3.8'
}
```

添加 mPaaS 版 SDK

1. 解压 AlipayVerifySDK-1.0.2.zip，并将 verifysdk-1.0.0.190522154510.aar、securitysdk-open-6.0.5.20191014.aar 放入 libs 目录下。

2. 在工程的 build.gradle 文件中添加 libs 目录作为依赖仓库。

```
allprojects {
  repositories {
     // 添加以下内容
     flatDir {
        dirs 'libs'
     }
  }
}
```

3. 确认通过 mPaaS 插件构建的应用带有如下仓库依赖。

```
allprojects {
  repositories {
     maven {
```

```
        credentials {
            username "*******"
            password "*******"
        }
        url "http://mvn.cloud.alipay.com/nexus/content/repositories/releases/"
    }
    flatDir {
        dirs 'libs'
    }
    jcenter()
  }
}
```

4. 在 bundle 工程的 app 模块中，配置 build.gradle 文件，添加 SDK 的依赖。

```
dependencies {
    // 添加如下内容
    provided "com.alipay.android.phone.zoloz:zcloud-build:3.0.1.190704174711:api@
jar"
    compile (name:'verifysdk-1.0.0.190522154510', ext:'aar')
    compile (name:'securitysdk-open-6.0.5.20191014', ext:'aar')
    compile 'com.squareup:otto:1.3.8'
}
```

5. 在 portal 工程的主 module 模块中，配置 build.gradle 文件，添加 bundle 的依赖。

```
dependencies {
    // 添加如下内容
    bundle "com.alipay.android.phone.zoloz:zcloud-build:3.0.1.190704174711:nolog@
jar"
    manifest "com.alipay.android.phone.zoloz:zcloud-build:3.0.1.190704174711:
AndroidManifest@xml"
}
```

调用 API 发起认证

1. 封装发起认证的信息。

   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。

```
JSONObject requestInfo = new JSONObject();
requestInfo.put("url", url);
requestInfo.put("certifyId", certifyId);
```

2. 发起认证。

   调用 **startService** 接口请求认证，传入当前的 context、认证信息以及接收处理结果的回调。

```
ServiceFactory.build()
  .startService(context, requestInfo, new ICallback() {
      @Override
      public void onResponse(Map<String, String> response) {
      // TODO with sdk response
      }
```

```
  });
```

3. 获取认证结果。

   认证结束后，**ICallback.onResponse** 方法会被调用，参数 **response** 中包含本次调用的结果。

```
public interface ICallback {
  /**
   * 返回结果
   * resultStatus 结果状态码
   * result.certifyId 本次认证流水号
   *
   * @param response 结果对象
   */
  void onResponse(Map<String, String> response);
}
```

   其中，ResultStatus 枚举定义如下：

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 = 6001、6002 时，result 对象数据为空，接入者不需要获取 result 对象数据。
- resultStatus= 9000 时，业务方需要去查询认证结果接口查询最终状态（由于前端数据是可篡改的）。

   Status：4000 包含的部分 errorCode 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
...	其他

混淆配置参考

为避免接口混淆，您可以参考以下代码来保留类名：

```
-keepclassmembers class ** {
    @com.squareup.otto.Subscribe public *;
    @com.squareup.otto.Produce public *;
}
```

```
-keep public class com.alipay.mobile.security.zim.api.**{
    public <fields>;
    public <methods>;
}

-keep class com.alipay.mobile.security.zim.biz.ZIMFacadeBuilder {
  !private <fields>;
   !private <methods>;
}

-keep class com.alipay.android.phone.mobilecommon.logger.AlipayMonitorLogService {
    !private <fields>;
    !private <methods>;
}

-keep class com.alipay.android.phone.mobilecommon.rpc.AlipayRpcService {
    !private <fields>;
    !private <methods>;
}

-keep class com.alipay.android.phone.mobilecommon.apsecurity.AlipayApSecurityServ
ice {
    !private <fields>;
    !private <methods>;
}

-keep class com.alipay.zoloz.toyger.bean.ToygerMetaInfo {
    !private <fields>;
    !private <methods>;
}

-keep class com.alipay.zoloz.toyger.algorithm.** { *; }

-keep class com.alipay.zoloz.toyger.blob.** {
    !private <fields>;
    !private <methods>;
}

-keep class com.alipay.zoloz.toyger.face.** {
    !private <fields>;
    !private <methods>;
}

-keep class com.alipay.zoloz.hardware.camera.impl.** {
    !private <fields>;
    !private <methods>;
}


-keep public class com.alipay.mobile.security.zim.plugin.**{
    public <fields>;
    public <methods>;
}

-keep class * extends com.alipay.mobile.security.zim.gw.BaseGwService{
    !private <fields>;
    !private <methods>;
}

-keep class * extends com.alipay.mobile.security.bio.service.BioMetaInfo{
    !private <fields>;
    !private <methods>;
}
```

```
-keep class com.alipay.zoloz.toyger.workspace.FaceRemoteConfig{
   *;
}

-keep public class com.alipay.zoloz.toyger.**{
   *;
}

-keep public class com.alipay.mobile.security.zim.gw.**{
   *;
}

-keep class com.alipay.deviceid.module.senative.DeviceIdUtil { *;}

-repackageclass com.alipay.deviceid.module.x
-keep class com.alipay.deviceid.module.rpc.deviceFp.** { *; }
-keep class com.alipay.deviceid.module.rpc.report.open.** { *; }
-keep class com.alipay.deviceid.DeviceTokenClient { *; }
-keep class com.alipay.deviceid.DeviceTokenClient$InitResultListener { *; }
-keep class com.alipay.deviceid.DeviceTokenClient$TokenResult {*;}

-keep class com.alipay.rds.v2.face.RDSClient { *; }
-keep class com.alipay.rds.constant.* { *; }
```

2.3 iOS 客户端接入

本文指导您在 iOS 应用中集成金融级实人认证服务。

前提条件

- 应用必须在 iOS 9.0+ 平台上运行。
- 您必须采用 Objective C++ 集成金融级实人认证服务。
- 您的应用中未集成支付宝支付 SDK。如果您的应用中已经集成了支付宝 mPaaS 版支付SDK，请参见 iOS 客户端接入（mPaaS 版）。

获取 SDK 和 Demo 代码

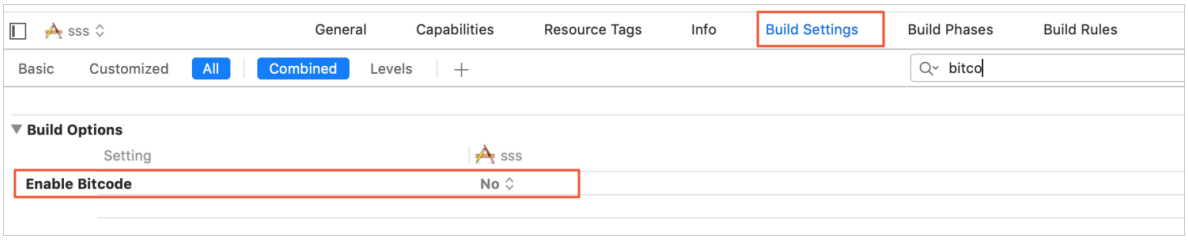
点击即可下载对应版本的 SDK 和 code demo。

您可以在这个文件的注释中确认您使用的 SDK 的版本号：AlipayVerifySDK.framework/Headers/APVerifyService.h 。

开发环境配置

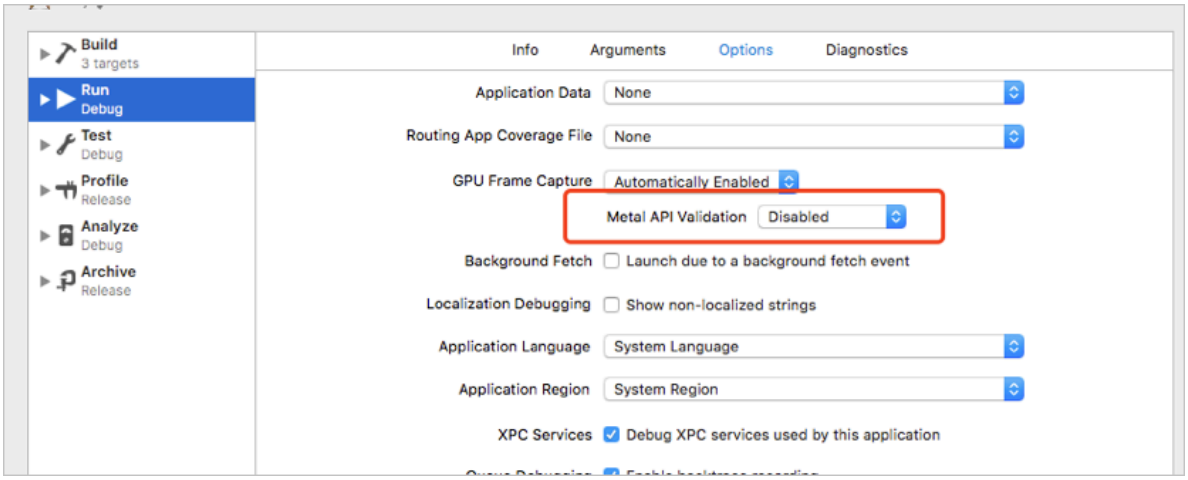
1. 在 info.plist 中配置摄像头权限请求。

2. 在 **Xcode** 的编译设置中关闭 **Bitcode** 选项。



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

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

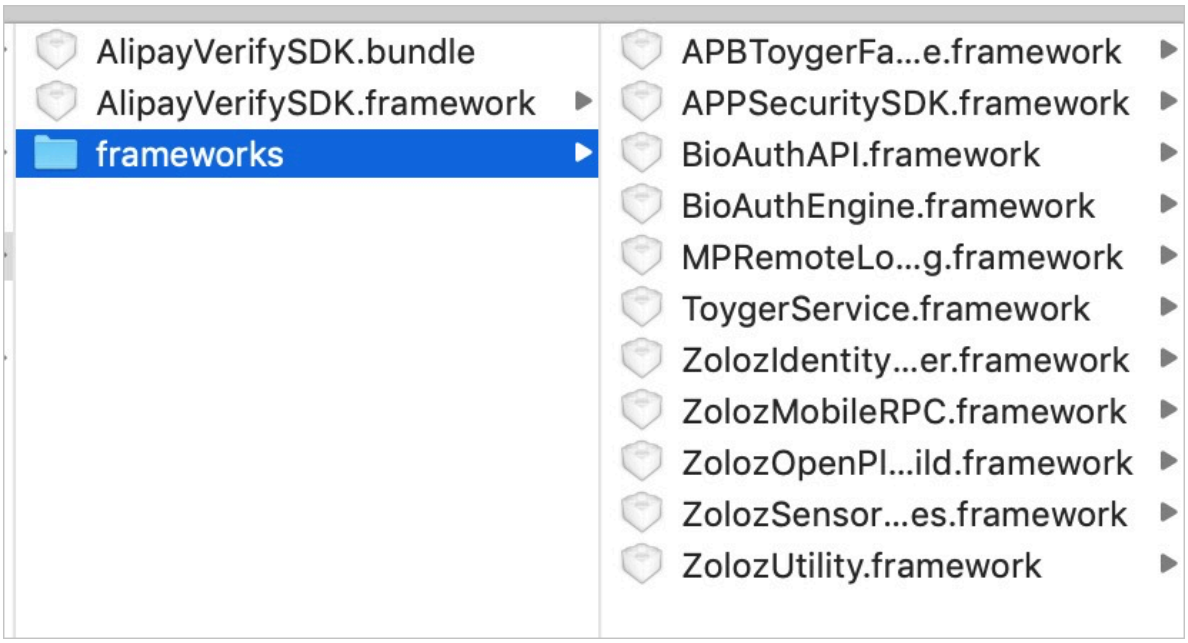


添加 framework 文件和 bundle 文件

注意：

步骤 1 和步骤 2 是开发过程中的必须步骤，请勿忽略。

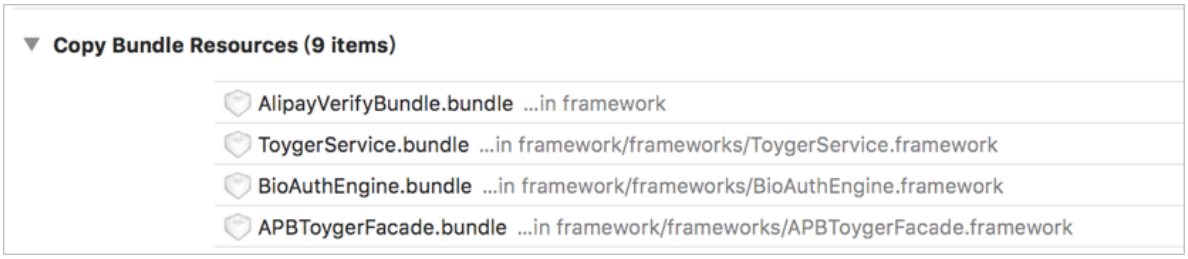
1. 引入 **SDK** 中 **frameworks** 文件夹下所有的 **framework** 到工程之中。



2. 找到并拷贝以下四个 bundle 文件，并将其手动链接到工程。bundle 文件在同名的 framework 文件中。

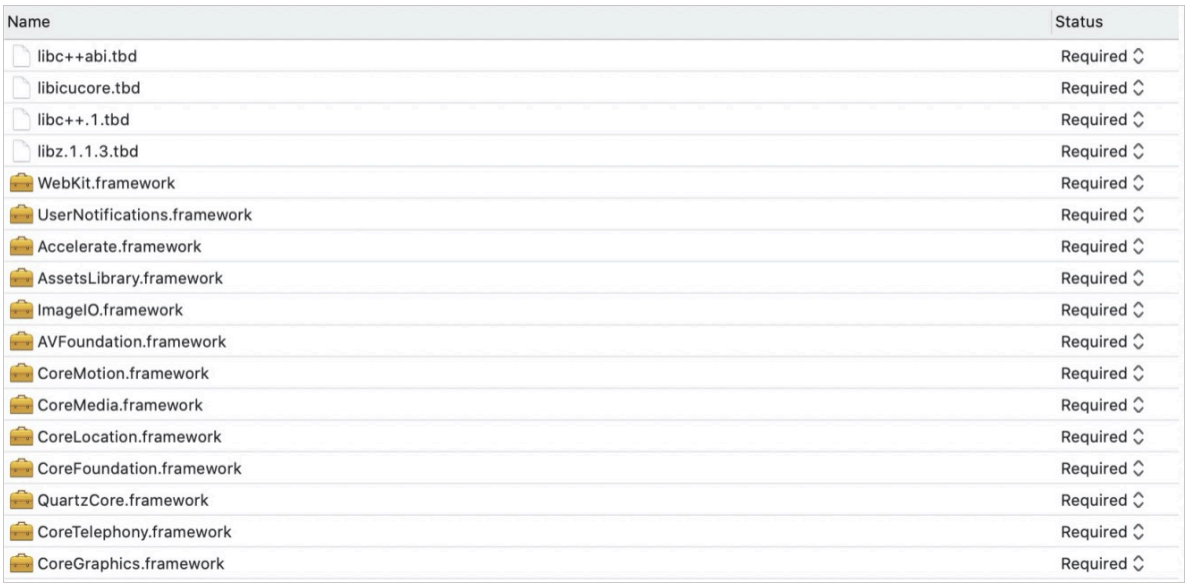
   选择 **TARGETS**，单击 **Build Phases** 标签页，在 **Copy Bundle Resources** 中添加如下四个 bundle：

   - APBToygerFacade.bundle：位于 APBToygerFacade.framework 中
   - ToygerService.bundle：位于 ToygerService.framework 中
   - BioAuthEngine.bundle：位于 BioAuthEngine.framework 中
   - AlipayVerifyBundle.bundle：位于framework中

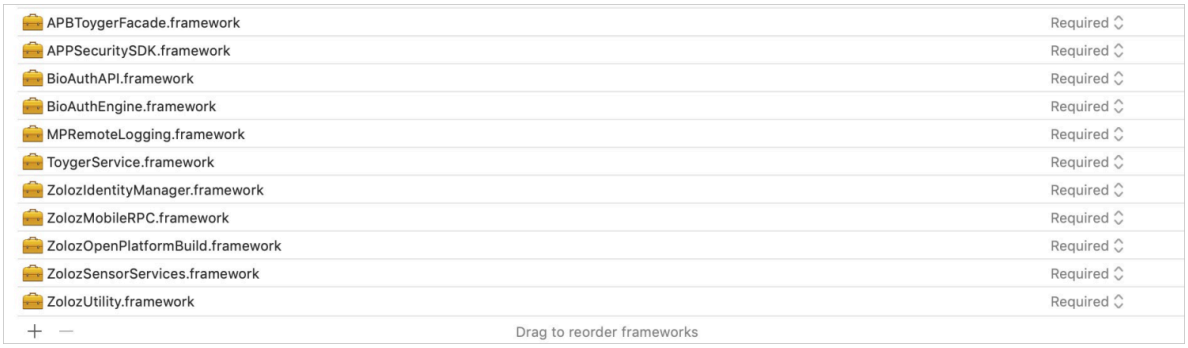


3. 添加系统库。

   在 **Linked Framweworks and Libraries** 中添加如下 iOS 系统库：



4. 添加支付宝认证库。



调用API

引用头文件，并尽早初始化。

```
#import<AlipayVerifySDK/APVerifyService.h>
[ZolozSdk init] // 尽早初始化
```

调用 startVerifyService

```
#import <AlipayVerifySDK/APVerifyService.h>
[[APVerifyService sharedService] startVerifyService:@{@"url": url?:@"",
                                        @"certifyId": @"test-certifyId",
                                            @"ext": @"test-extInfo"
                                               }
 target:self
 block:^(NSMutableDictionary * resultDic){
    NSLog(@"%@", resultDic);
}];
```

输入参数

参数名称	参数说明

options	包含两个参数：
	• 认证 url。该参数从应用服务端获得。
	• 本次认证的认证 ID。该参数从应用服务端获得。
	@"certifyId": @"test-certifyId",// 认证流水号，从商家服务端获得
	@{
	@"url":url?:@"",
	@"certifyId": @"test-certifyId",
	}

targetVC	当前 **controller**，必须有根 **navigation Controller**，且需要隐藏 **navigationBar** 再进入 SDK，回来后恢复，因为 SDK 内部已有自定义 **navigationBarView**。
block	返回的结果，在 **resultDic** 里面获取。

返回结果

回调函数带入的参数形式如下：

```
result: {resultStatus: 'xx',
        result: { }
       }
```

名称	类型	描述
resultStatus	string	认证流程结果状态码， 详见以下 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 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 枚举定义如下：

状态码	描述
9000	认证通过
6002	网络异常

状态码	描述
6001	用户取消了业务流程，主动退出
4000	业务异常

说明：

- resultStatus = 6001、6002 时，result 对象数据为空，接入者不需要获取 result 对象数据。
- resultStatus= 9000 时，业务方需要去查询认证结果接口查询最终状态（由于前端数据是可篡改的）。

Status：4000 包含的部分 errorCode 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
...	其他

2.4 iOS 客户端接入（mPaaS 版）

本文指导您在 iOS 应用中集成金融级实人认证服务。

前提条件

- 目前 SDK 只#持 iOS 9.0 或以上版本的#机系统。 SDK 不#持 swift #式调#。
- 您必须采用 Objective C++ 集成金融级实人认证服务。
- 您的应用中已集成支付宝支付 SDK。

背景信息

在 mPaaS 项目中集成金融级实人认证和在普通 iOS 原生项目中的步骤并不完全一致，需要处理好共享组件库之间的冲突，并在 mPaaS项目中注册好实人认证服务的拦截器。

说明：

mPaaS 拦截器概念请参见 拦截器。

整体流程

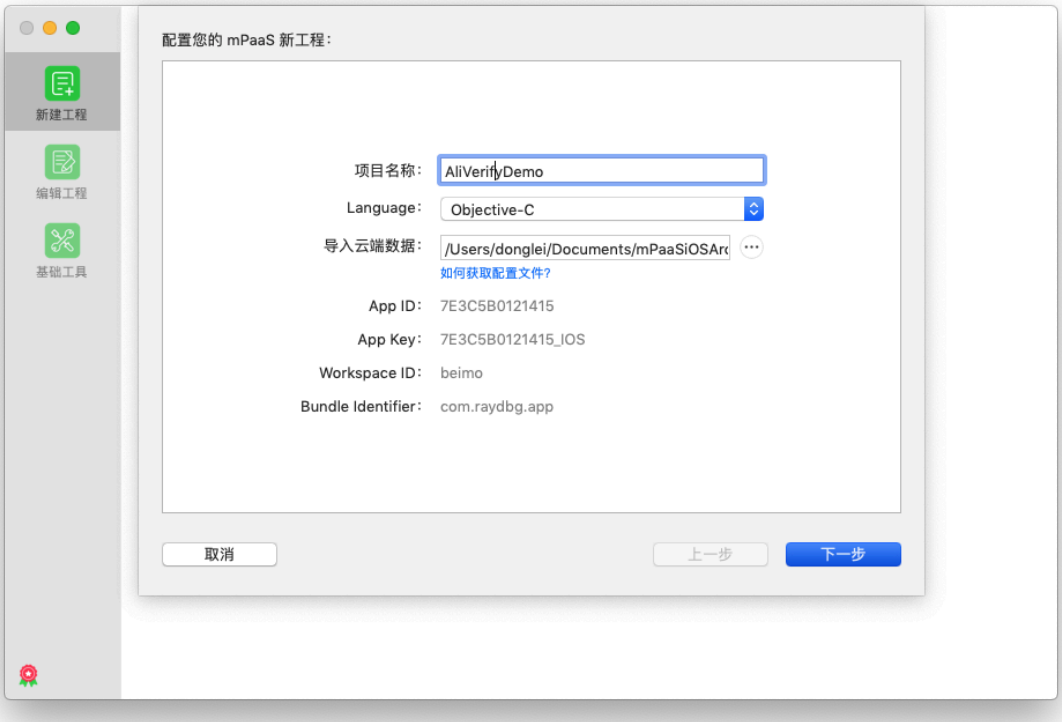
1. 服务端返回 url 给 iOS 客户端；
2. 启动客户端完成电#合约签订。

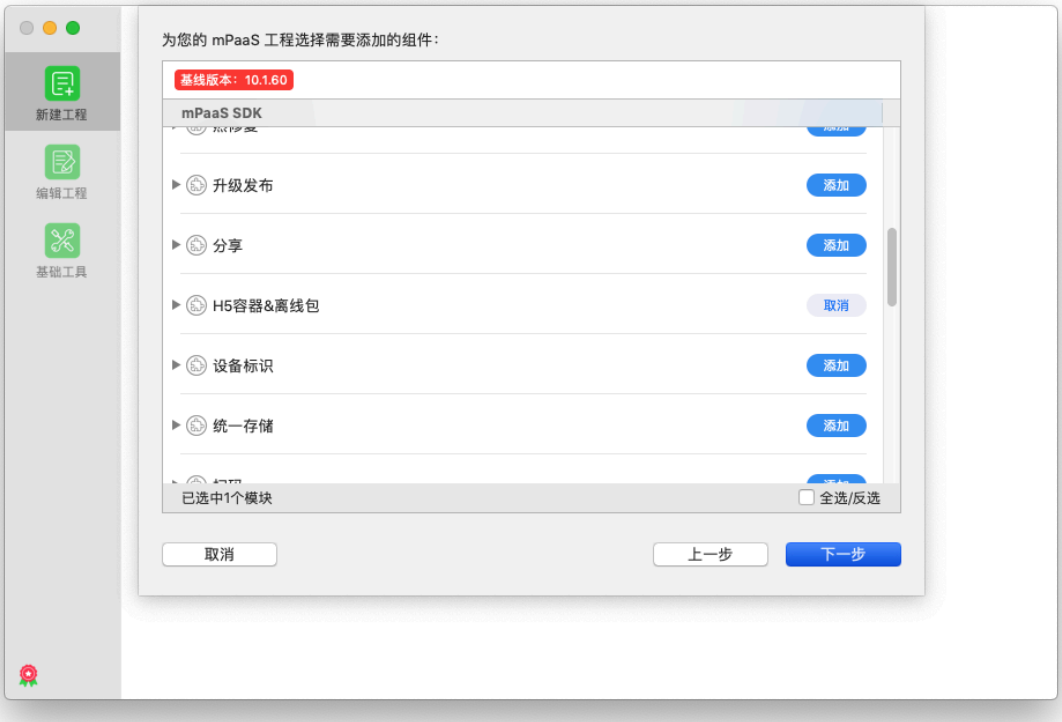
准备工作

1. 准备 mPaaS 工程

   本文以 mPaaS iOS 60 基线托管模式框架为准，介绍详细接入步骤。为了说明方便，我们从一个最简单的项目开始。如果 mPaaS 项目已经存在，可以跳过这一步。

   首先通过 mPaaS Extension 创建一个 iOS 项目，按需选择需要接入的组件。





创建完成后，务必确保 mPaaS 项目可以正常编译、调试和运行。

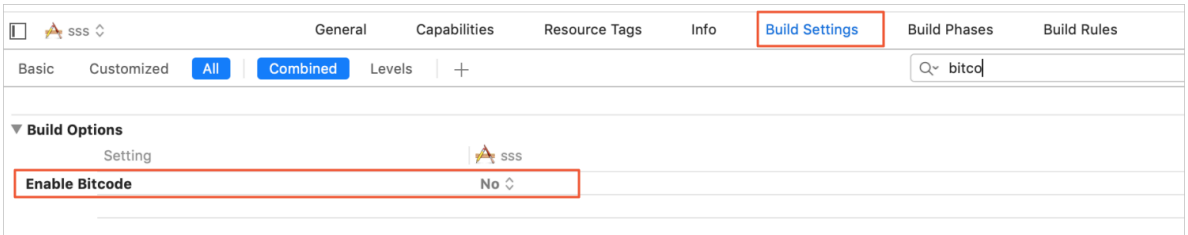
2. 准备实人认证服务

   请参考 #unique_17，开通实人认证服务并获取到正确的AccessKey。

系统配置

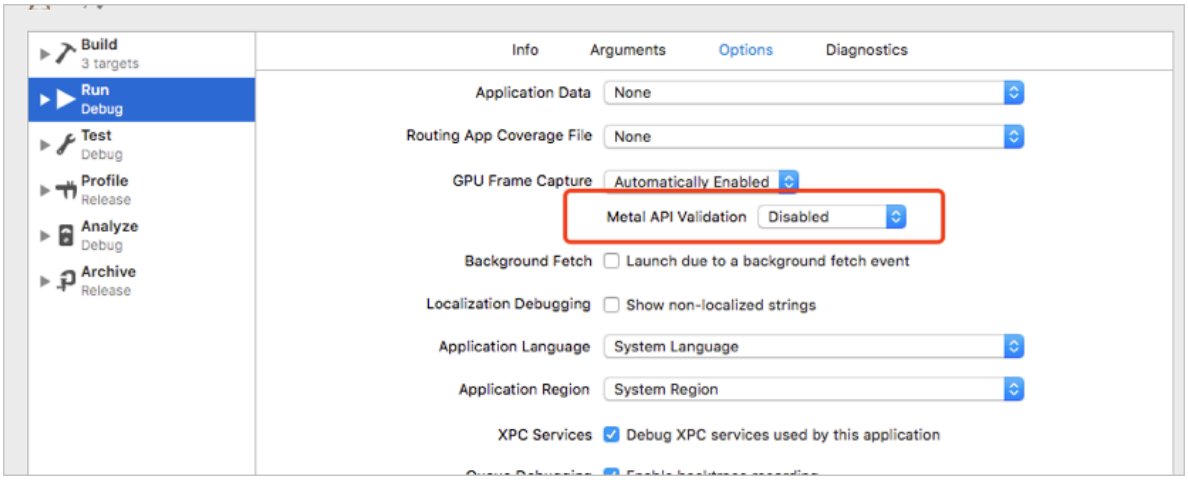
1. 在 info.plist 中配置摄像头权限请求。

2. 在 **Xcode** 的编译设置中关闭 **Bitcode** 选项。



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

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

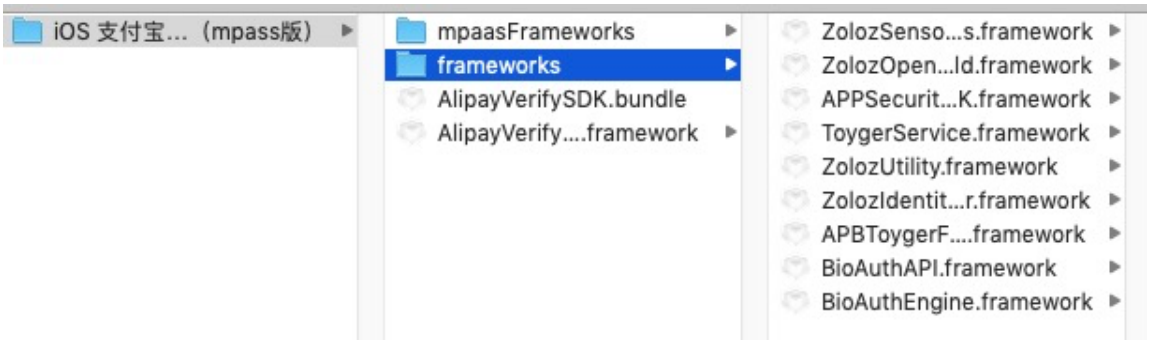


接入过程

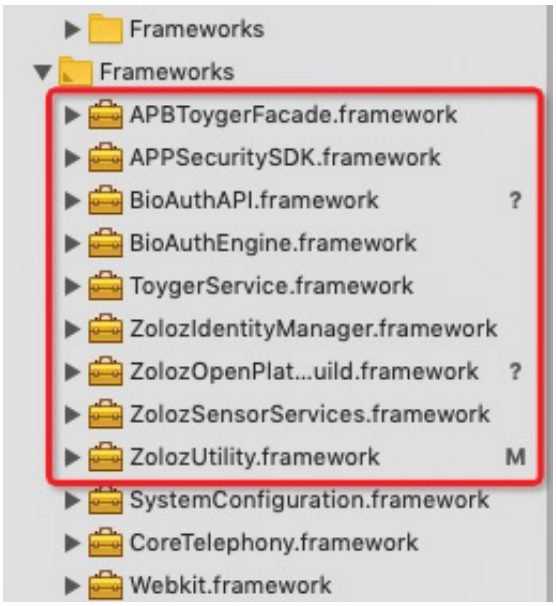
1. 实人认证 mPaaS 版 SDK 接入

a) 点击下载 实人认证mPaas版SDK 和 code demo 。

SDK 解压后得到如下文件：



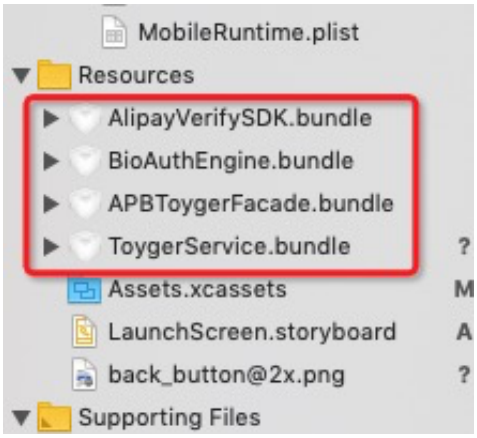
b) 将frameworks文件夹下所有的framework文件添加到工程中：



c) 找到并拷贝以下四个 bundle 文件，并将其手动链接到工程。bundle 文件在同名的 framework 文件中。

选择 **TARGETS**，单击 **Build Phases** 标签页，在 **Copy Bundle Resources** 中添加如下四个 bundle：

- APBToygerFacade.bundle：位于 APBToygerFacade.framework 中
- ToygerService.bundle：位于 ToygerService.framework 中
- BioAuthEngine.bundle：位于 BioAuthEngine.framework 中
- AlipayVerifyBundle.bundle：位于framework中



2. 实人认证拦截器配置

在 mPaaS 项目中，需要通过拦截器对认证业务的请求进行特殊处理。拦截器ZolozMobileInterceptor已经实现，并包含在实人认证 SDK 中，这里需要把该拦截器注册到 mPaaS 拦截器配置中。

另外，需要打开mPaaS容器shouldWKDispatchStartEvent开关。

参考代码：

```
// 位置：DTFrameworkInterface+DEMO.m
...
- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions
:(NSDictionary *)launchOptions{

    // Step 1. 初始化mPaaS H5容器
    [MPNebulaAdapterInterface initNebula];

    // Step 2. 初始化RPC拦截器
    [MPRpcInterface initRpc];

    // Step 3. 配置Zoloz必要的拦截器
    ZolozMobileInterceptor *zolozIcp = [[ZolozMobileInterceptor alloc] init];
    [MPRpcInterface addRpcInterceptor:zolozIcp];

    //Step 4.人脸模块初始化。
    [ZolozSdk init];
}

- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(
NSDictionary *)launchOptions{
    //Step 5. Nebula容器配置修改
    NBServiceConfigurationGet().shouldWKDispatchStartEvent = YES;
```

```
}
```

3. 实人认证业务调用

a) 引用头文件

```
#import<AlipayVerifySDK/APVerifyService.h>
```

b) 调#SDK

```
#import<AlipayVerifySDK/APVerifyService.h>
[ZolozSdk init] // 尽早初始化
```

说明:

请尽早调#初始化代码。

调用 startVerifyService， 让#户完成电#合约签订。

```
#import<AlipayVerifySDK/APVerifyService.h>
[[APVerifyService sharedService] startVerifyService:@{@"url": url?:@"",
                                    @"certifyId": @"testcertifyId",
                                    @"ext": @"testextInfo"
                                    }target:self block:^(NSMutableDictionary * resultDic){
    NSLog(@"%@", resultDic);
}];
```

c) 输入参数

参数名称	参数说明

options	包含两个参数：
	• 认证 url。该参数从应用服务端获得。
	• 本次认证的认证 ID。该参数从应用服务端获得。
	@"certifyId": @"test-certifyId",// 认证流水号，从商家服务端获得
	@{
	@"url":url?:@"",
	@"certifyId": @"test-certifyId",
	}

targetVC	当前 **controller**，必须有根 **navigationController**，且需要隐藏 **navigationBar** 再进入 SDK，回来后恢复，因为 SDK 内部已有自定义 **navigationBarView**。

参数名称	参数说明
block	返回的结果，在 **resultDic** 里面获取。

d) 返回结果

回调函数带入的参数形式如下：

```
result: {resultStatus: 'xx',
        result: { }
       }
```

名称	类型	描述
resultStatus	string	认证流程结果状态码， 详见以下 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 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。

e) ResultStatus 枚举定义

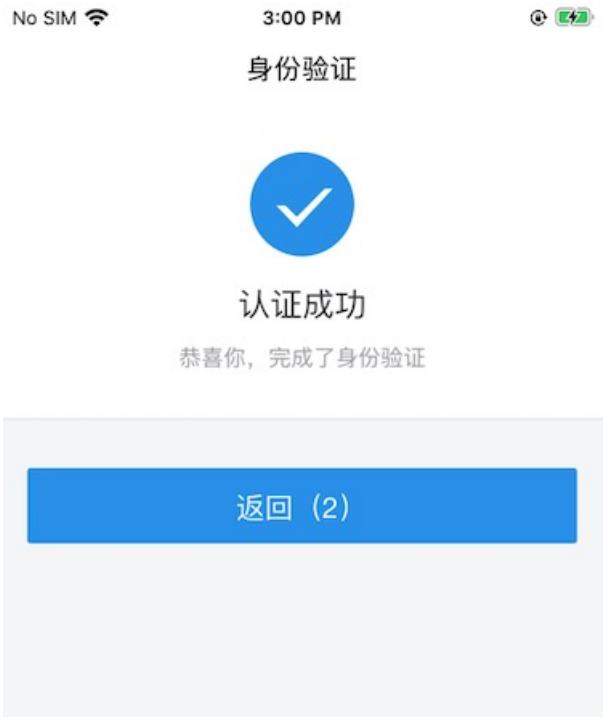
状态码	描述
9000	认证通过
6002	网络异常
6001	用户取消了业务流程，主动退出
4000	业务异常

说明：

- resultStatus = 6001、6002 时，result 对象数据为空，接入者不需要获取 result 对象数据。
- resultStatus= 9000 时，业务方需要去查询认证结果接口查询最终状态（由于前端数据是可篡改的）。

4. 验证

   正确拉起服务，完成验证后，应该见到如下界面：



说明

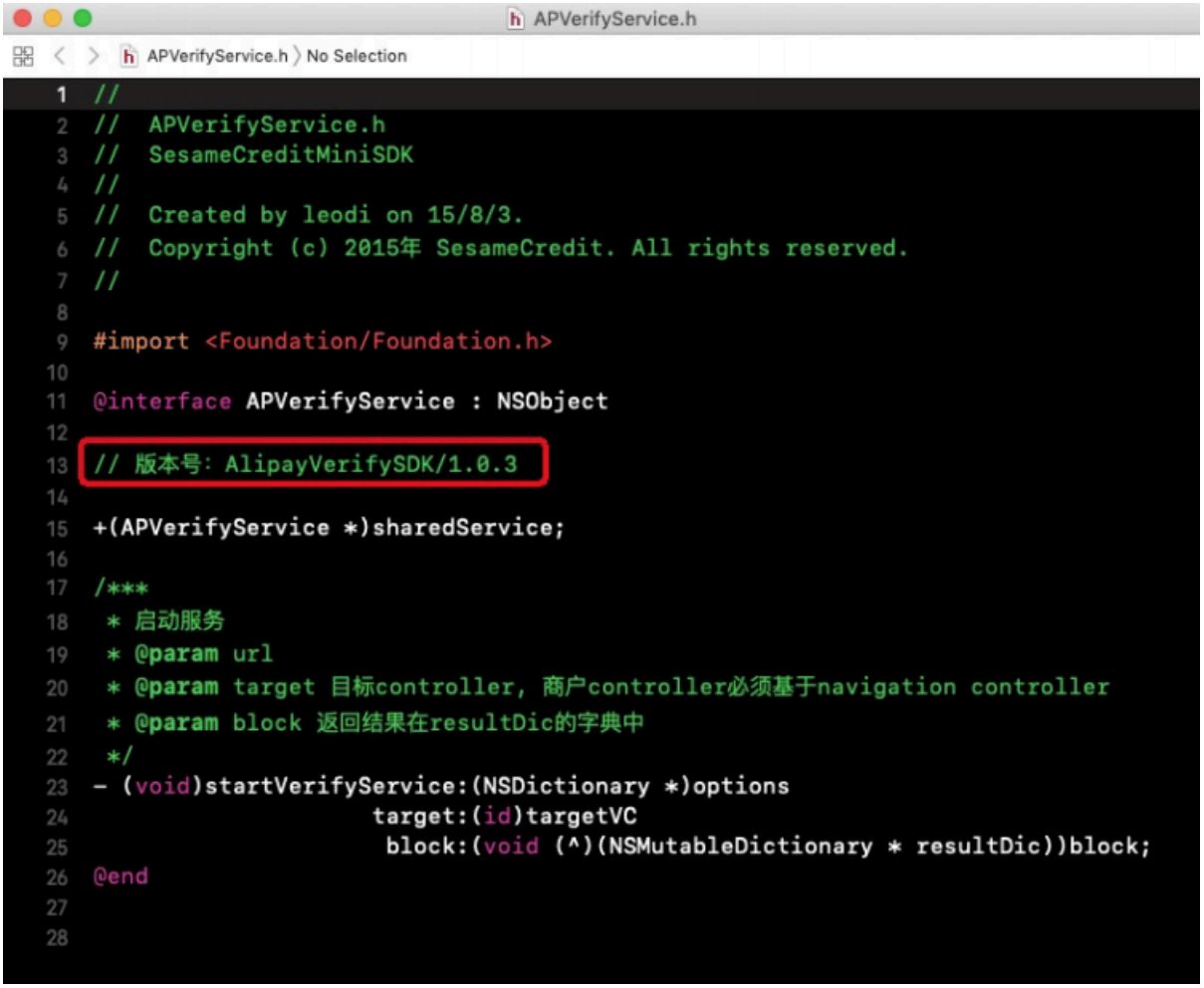
1. #定要替换所有.framework & bundle#件。

   AlipayVerifySDK的bundle在外部，其余bundle在.framework中。

2. 若 app 中有接入容器，需要防止容器拦截 scheme，否则点击唤不起扫脸：

```
NBServiceConfigurationGet().shouldWKDispatchStartEvent = YES;
```

3. 反馈问题时请加上版本号，版本号您可以在APVerifyService.h中找到。



3 支付宝小程序接入

3.1 接入流程

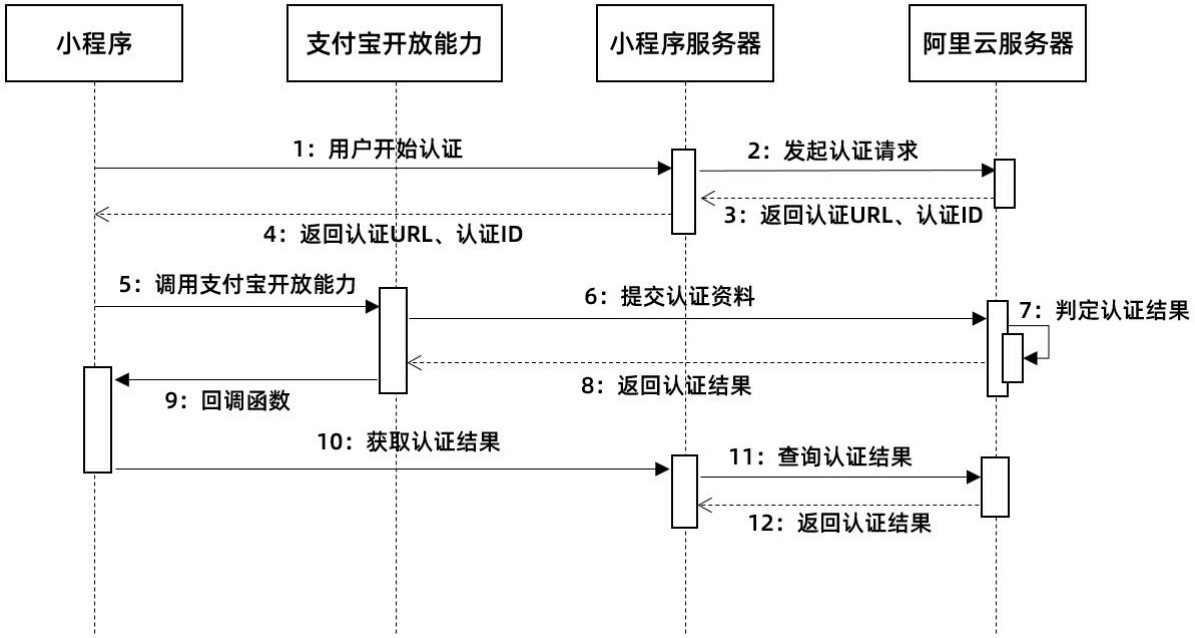
支付宝小程序接入适用于业务使用方希望在自己开发的支付宝小程序中对用户进行线上认证的场景。

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。

调用时序

在接入前，请查看调用时序图：



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。

序号	说明
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。

集成步骤

1. 在支付宝小程序中集成支付宝开放能力。具体集成步骤，请参见以下相应文档#unique_9。
2. 在小程序服务端集成相应 API。具体请参见服务端接入。

3.2 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

   a) 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);
}
```

   b) 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

a) 回调函数出参

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

名称	类型	描述
resultStatus	string	认证流程结果状态码， 详见以下 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 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。

b) ResultStatus

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 = 6001、6002 时，result 对象数据为空，接入者不需要获取 result 对象数据。
- resultStatus= 9000 时，业务方需要去查询认证结果接口查询最终状态（由于前端数据是可篡改的）。

Status：4000 包含的部分 errorCode 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
...	其他

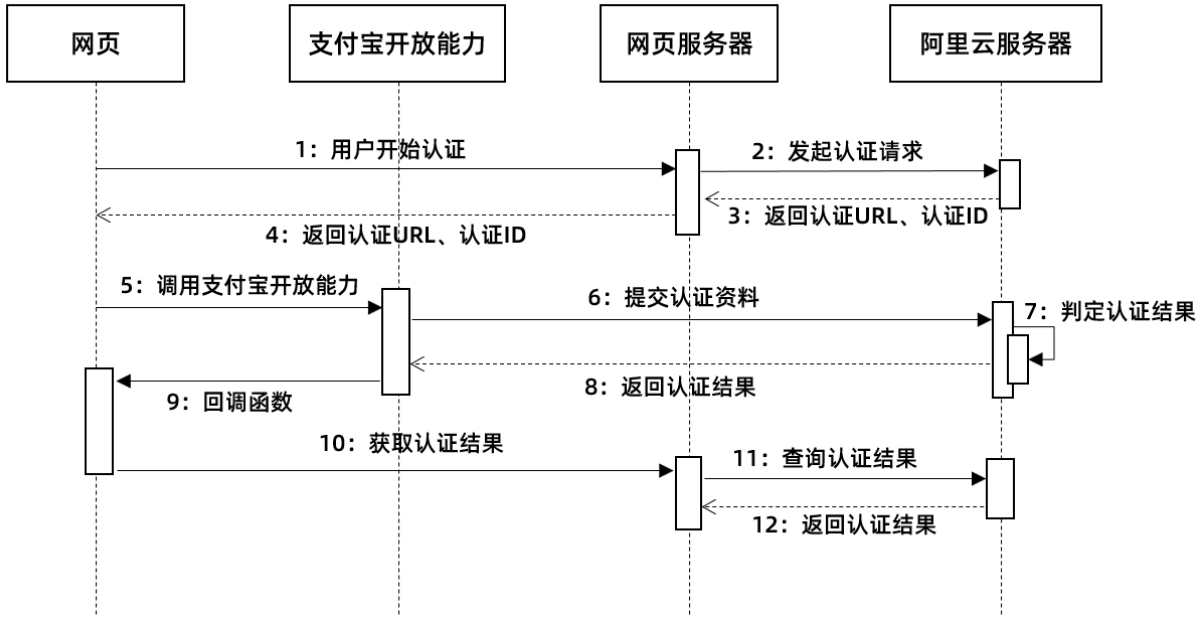
4 H5页面接入

4.1 接入流程

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

调用时序

在接入前，请查看调用时序图：



系统调用时序说明：

序号	说明
1	用户开始认证。
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。

序号	说明
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。

集成步骤

1. 在H5网页中集成支付宝开放能力。具体集成步骤，请参见相应文档：
   - 支付宝 H5 页面接入；
   - 端外唤起支付宝认证页面接入。
2. 在 H5 网页服务端集成相应 API。具体请参见服务端接入。

4.2 支付宝 H5 页面接入

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。如果您希望在支付宝端外 H5 页面中集成金融级实人认证服务，参看 端外唤起支付宝认证页面接入。

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

   a) 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);
  }
```

```
}
```

b) 发起认证服务。

调用 **startBizService** 接口请求认证。

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

c) 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。

```
ready(function() {
    // 需要确保在 AlipayJSBridge ready 之后才调用
    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

a) 回调函数出参

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

名称	类型	描述
resultStatus	string	认证流程结果状态码， 详见以下 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 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。

b) ResultStatus

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 = 6001、6002 时，result 对象数据为空，接入者不需要获取 result 对象数据。
- resultStatus= 9000 时，业务方需要去查询认证结果接口查询最终状态（由于前端数据是可篡改的）。

Status：4000 包含的部分 errorCode 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
...	其他

4.3 端外唤起支付宝认证页面接入

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。如果您希望在支付宝端内 H5 页面中进行集成，参看 支付宝 H5 页面接入。

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. 唤起支付宝客户端，在支付宝客户端中进行认证。

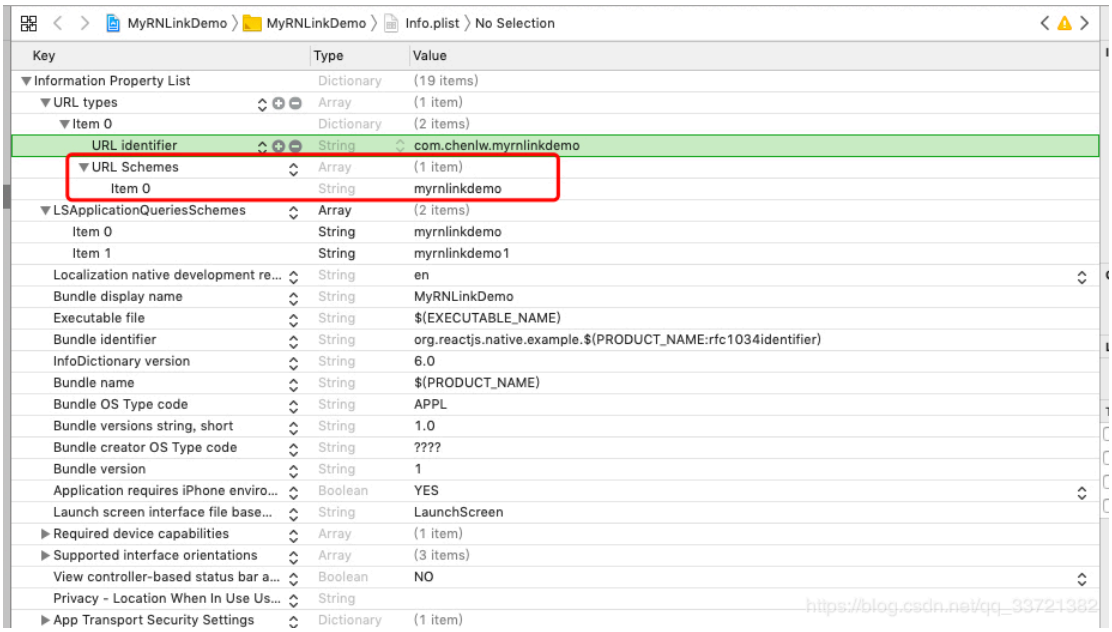
   a) 在服务端发起请求时获取的认证流程入口 url 的基础上拼接支付宝唤端逻辑：`const certifyUrl = alipays://platformapi/startapp?appId=20000067&url=encodeURIComponent(url);`

   b) 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。

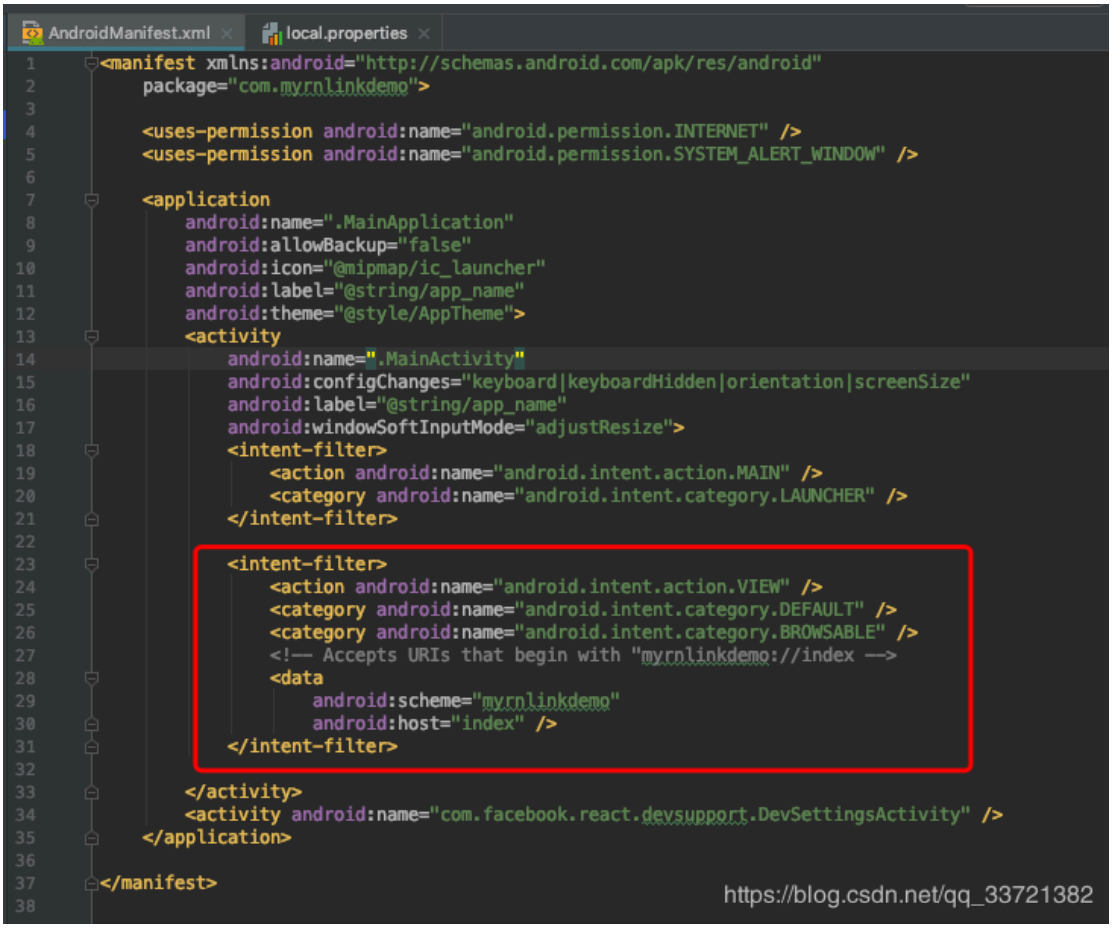
a) 准备您的应用的 Scheme。

- 对于 iOS 平台，在 info.plist 文件中可以设置 URL Schemes（URL types -> URL Schemes）。如图所示：



如果您设置正确，在 Safari 浏览器中输入 <您设置的 Scheme 名>://，就可以打开 APP了。

- 对于 Android 平台，您可以在配置清单文件 AndroidManifest.xml 中，设置 URLScheme。如图所示：



如果您设置正确，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，点击链接，即可打开 APP。

b) 在服务端发起请求时，在 **returnUrl** 中传入您的应用的 Scheme。详情参看 #unique_23/unique_23_Connect_42_section_gnq_5d4_1gb。