



# iOS活体检测集成文档1.2.0.4

## 一.准备工作

### 概述

本文结合示例代码指导您在 iOS应用中集成活体检测SDK，帮助您在 App 中实现刷脸认证功能。

### 前置条件

- 应用必须在 iOS 9.0+ 平台上运行。

### 创建应用

应用的创建流程及AppId的获取，请查看「账号创建」文档

**注意：AppId与bundle id一一对应的绑定关系，否则会校验不通过。**

### 认证流程

graph LR

A[SDK检测] --> B[SDK返回结果] --> C[云上校验] --> D[认证结果]

### 快速体验demo

- iOS压缩包中的demo文件夹中是示例工程源码，使用Xcode打开完成以下步骤配置，可直接运行测试：
  - 将项目中bundle id更改成您的创建应用时填写的bundle id
  - 将项目中的AppId换成您创建应用后生成的AppId

## 开发环境搭建

### 1. 添加隐私权限

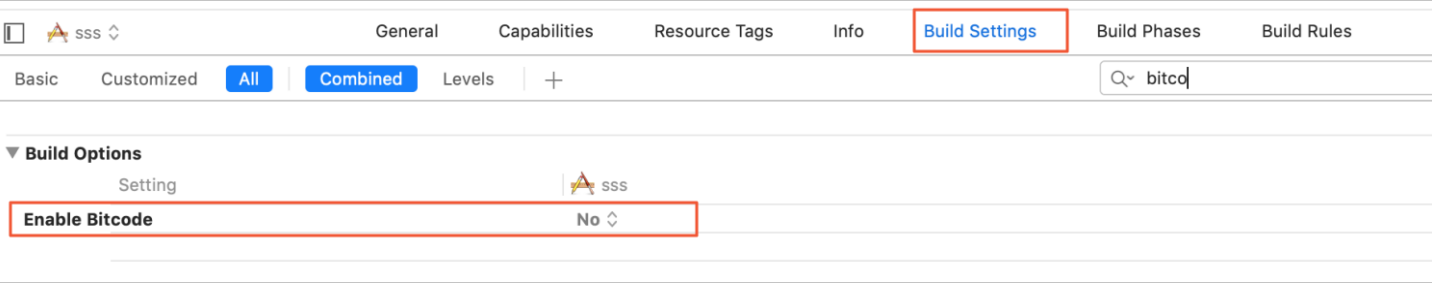
- Privacy - Camera Usage Description
- Privacy - NSUserTrackingUsageDescription

Bundle ID type code	String	\$(PRODUCT_BUNDLE_PACKAGE_TYPE)
Privacy - Camera Usage Description	String	活体 demo 想使用您的相机，允许吗？
Localization native development region	String	\$(DEVELOPMENT_LANGUAGE)
Privacy - Photo Library Additions Usage Description	String	活体 demo 想添加视频到相册，允许吗？

ps:参考文案“请放心，开启权限不会获取您在其他APP或网站的隐私信息，该权限仅用于标识设备并保障认证流程安全

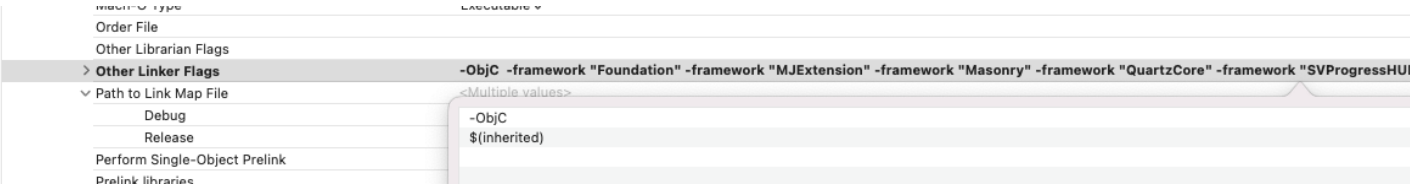
### 2. Xcode关闭BitCode选项

**Build Setting** -搜索Enable Bitcode -修改设置为**NO**



### 3. 编译配置

在Xcode变异设置的**Linking > Other Linker Flags**中，添加 **-ObjC**



### 4. 导入SDK包及系统依赖包

- SDK库

- 1 APBToygerFacade
- 2 APPSecuritySDK
- 3 BioAuthEngine
- 4 DTFIdentityManager

- 5 DTFUtility
- 6 ToygerNative
- 7 ToygerService
- 8 **CLLivingDetectSDK**

- 系统依赖库

- 1 CoreGraphics.framework
- 2 Accelerate.framework
- 3 SystemConfiguration.framework
- 4 AssetsLibrary.framework
- 5 CoreTelephony.framework
- 6 QuartzCore.framework
- 7 CoreFoundation.framework
- 8 CoreLocation.framework
- 9 ImageIO.framework
- 10 CoreMedia.framework
- 11 CoreMotion.framework
- 12 **AVFoundation**.framework
- 13 WebKit.framework
- 14 libresolv.tbd
- 15 libz.tbd
- 16 libc++.1.tbd
- 17 libc++abi.tbd
- 18 AudioToolbox.framework
- 19 **CFNetwork**.framework
- 20 MobileCoreServices.framework
- 21 libz.1.2.8.tbd
- 22 AdSupport.framework

- 拷贝资源文件

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

- 1 APBToygerFacade.bundle： 位于 APBToygerFacade.framework 中。
- 2 BioAuthEngine.bundle： 位于 BioAuthEngine.framework 中。
- 3 ToygerService.bundle： 所在位置为 ToygerService.framework 中。
- 4 **CLLivingDetect**.bundle： 所在位置为 **CLLivingDetectSDK**.framework中。

## 二.API说明

## 1. 初始化

导入头文件 `#import <CLLivingDetectSDK/CLLivingDetectSDK.h>`

- 为提高身份核验的用户体验，并为刷脸认证准备必要数据，iOS 客户端需要进行 SDK 初始化。

方法原型

```
1 /// 初始化
2 /// @param appId 控制台创建应用的appid
3 + (void)initWithAppId:(NSString *)appId;
```

示例代码

```
1 [CLLivingDetectManager initWithAppId:APPID];
```

## 2. 认证参数配置

- 【可选方法】每个配置都有默认值，可以不配置。如需修改默认配置，调用相应配置方法即可。

```
1 /// 扫脸圆圈颜色，默认为蓝色，颜色格式为 @"#FFFFFF"
2 @property(nonatomic, strong) NSString * faceCircleColor;
3
4 /// 验证请求超时时间设置
5 @property(nonatomic, strong) NSNumber * verifyOutTime;
6
7 /// 是否返回图片，默认为@NO
8 @property(nonatomic, strong)NSNumber * returnImage;
9
10 /// 是否返回录制视频地址，默认不返回，格式@(YES) or @(NO)
11 @property(nonatomic, assign) NSNumber * returnVideo;
12
13 /// 活体检测动作
14 /// （默认）眨眼动作活体检测；
15 /// CLLivingVerifyActionMulti: 多动作活体检测，眨眼+任意摇头检测
16 @property(nonatomic, assign) CLLivingVerifyAction verifyAction;
17
18 ///是否自定义协议，默认为@(YES),显示协议页，为@(NO)需要用户自定义协议页面
19 @property(nonatomic, assign)NSNumber * showProtocol;
20
21 + (CLLivingConfig *)defaultConfig;
```

## 示例代码

```
1 CLLivingConfig * config = [CLTestSettingModel defaultConfig];
2 [CLLivingDetectManager setLivingConfig:config];
```

## 3. 开始认证

- 调用此方法会启动SDK内部活体检测界面，并返回刷脸认证结果。

### 方法原型

```
1 /// 开启活体校验
2 /// @param viewController 当前显示viewController
3 /// @param completion 返回结果
4 + (void)startVerifyWithViewController:(UIViewController *)viewController
5                                completion:(void (^)(CLLivingResult * result))completion;
```

## 示例代码

```
1 [CLLivingDetectManager startVerifyWithViewController:self completion:^(CLLivingR
2     )];
```

## 刷脸认证结果

```
1 /**
2  SDK返回外层码
3  10000 : 刷脸结束
4  10001: 校验失败
5  10002: 验签失败
6  10003: 网络异常
7  10004: 本地异常
8  */
9 @property (nonatomic, assign, readonly) NSInteger code;
10 /// SDK返回响应描述
11 @property (nonatomic, strong, readonly) NSString * message;
12 /// SDK内层码
13 @property (nonatomic, assign) NSInteger innerCode;
```

```
14 /// SDK内层描述, 可查看具体原因
15 @property (nonatomic, strong) NSString * innerMessage;
16 /// SDK报错, 返回错误信息
17 @property (nonatomic, strong, readonly) NSError * error;
18 @property (nonatomic, strong) id ext;
19 /// SDK 活体检测后返回详细信息
20 @property (nonatomic, strong) CLLivingDetectResponse * response;
```

```
1 @interface CLLivingDetectResponse : NSObject
2 /// 如果采用视频返回, 这个字段返回视频的路径, 否则为空
3 @property(nonatomic, strong, nullable) NSString *videoFilePath;
4 /// 如果设置支持照片返回, 则返回照片, 否则为空
5 @property(nonatomic, strong, nullable) NSData *imageContent;
6 /// 验证id, 可用户服务器查询标识
7 @property(nonatomic, strong, nullable) NSString * certifyId;
```

## 4. 日志开关

- 放到初始化之前调用, 开启后可打印SDK内部调用日志

方法原型

```
1 /// 是否开启控制台日志打印
2 /// @param enable 默认为NO
3 + (void)setPrintConsoleEnable:(BOOL)enable;
```

示例代码

```
1 [CLLivingDetectManager setPrintConsoleEnable:YES];
```

## 三、返回码

外层码	外层描述	内层码	内层描述
10000	刷脸成功	1000	刷脸成功, 认证通过。可通过服务端查询接口获取认证详细资料信息 (Android端、iOS端)
10001	校验失败	1001	本地代码异常 (Android端)

			人脸识别算法初始化失败（Android端）
			不支持的CPU架构（Android端）
			Android系统版本过低（Android端）
			刷脸超时(单次)（Android端、iOS端）
			多次刷脸超时（Android端、iOS端）
			无前置摄像头（Android端）
			摄像头权限未赋予（Android端）
			打开摄像头失败（Android端）
			SDK认证流程正在进行中，请等待本地认证流程完成后再发起新调用（Android端）
			上传炫彩Meta信息失败（Android端）
			上传炫彩视频失败（Android端）
			用户点击Home键（Android端）
			抱歉，系统出错了，请您稍后再试（iOS端）
			拒绝开通相机权限（iOS端）
			无法启动相机（iOS端）
			本地活体检测出错（iOS端）
			验证中断（用户点击home键等导致验证停止）（iOS端）
			业务参数错误（iOS端）
			本地活体检测出错（iOS端）
1003			用户主动退出认证（Android端、iOS端）
			用户暂不认证（Android端）
2001			<del>用户OCR主动退出（iOS端）</del>
2002			客户端初始化网络错误（Android端）
			客户端初始化接口返回网络错误（Android端）
			信息上传网络错误（Android端）
			服务端认证接口网络错误（Android端）

			服务端接口并发请求超出限制（Android端）
			网络错误（iOS端）
		2003	客户端设备时间错误（iOS端）
		2006	刷脸失败，认证未通过。可通过服务端查询接口获取认证未通过具体原因（Android端、iOS端）
10002	验签失败	400001	参数校验异常（Android端、iOS端）
		600016	Android签名参数异常（Android端）
		600009	bundleId不能为空（iOS端）
		600017	平台类型非法（Android端、iOS端）
		600005	签名校验失败（Android端、iOS端）
		600018	签名失效（Android端、iOS端）
		600004	包名签名对应的appid不匹配 或 appid未匹配到应用（Android端、iOS端）
		500006	请求外部系统失败（Android端、iOS端）
		600019	包名签名校验失败（Android端、iOS端）
		500003	业务操作失败（Android端、iOS端）
		1010	服务端返回为空（Android端）
10003	网络异常	1023	网络原因导致的超时、域名解析异常等（Android端、iOS端）
10004	本地异常	1014	本地捕获异常（Android端、iOS端）