

# Secken iOS 平台 SDK 文档

编号：SECKEN\_iOS\_SDK

版本：SECKEN\_iOS\_SDK V1.0.8

<b>1. 概述及名词解释 .....</b>	<b>1</b>
1.1 概述 .....	1
1.2 名词解释 .....	1
<b>2. 功能简介 .....</b>	<b>1</b>
2.1 认证授权 .....	1
2.2 人脸功能 .....	2
2.3 声音功能 .....	2
2.4 二维码相关 .....	2
2.5 验证功能 .....	2
2.6 push 相关 .....	3
2.7 解除绑定 .....	3
<b>3. 申请应用 .....</b>	<b>3</b>
<b>4. 在 Xcode 中集成 SDK .....</b>	<b>6</b>
4.1 导入 SDK 开发包到你自己的工程项目 .....	6
5 在 Xcode 中集成 UI SDK .....	13
<b>6 错误码 .....</b>	<b>18</b>

## 1. 概述及名词解释

### 1.1 概述

洋葱 SDK 是帮助移动应用快速集成多维身份验证的开发工具集。

移动应用可通过集成洋葱 SDK，快速实现二维码登录、指纹识别、声纹识别或人脸识别功能，更加有效的提高识别的安全性和真实性，还能利用位置和网络等信息作为安全识别的重要依据。

### 1.2 名词解释

名词	注解
APP KEY	分配给每个第三方应用的 APP KEY，用于鉴权签名等功能。
APPID	分配给每个第三方应用的 APPID，用于识别应用的标示
USERNAME	第三方应用 APP 的对应的用户名
TOKEN	表示用户身份的 TOKEN，用于 SeckenSDK 其他接口的调用。
Auth_token	用于开发者服务端向洋葱服务端核对验证结果的凭据(验证成功之后返回的字段)

## 2. 功能简介

### 2.1 认证授权

在 Secken 开发者平台申请应用的客户端情况下，使用 SeckenSDK 授权即可授权客户端，之后才能使用 SeckenSDK 其他功能。

## 2.2 人脸功能

通过 Secken SDK，第三方应用能创建人脸（人脸训练），人脸验证以及删除人脸验证功能：

- 人脸训练

第三方应用上传带有人脸头像的图片，识别头像信息成功三次表示创建人脸验证成功。

- 人脸验证

第三方应用上传带有人脸头像的图片由服务器返回头像信息是否为创建的人脸信息作为人脸是否验证成功的标志。

- 人脸删除

删除人脸验证的一切相关信息。注：删除后用户将无法使用人脸验证

## 2.3 声音功能

通过 Secken SDK，第三方应用能创建声纹信息（声音训练），声音的验证以及删除声音验证功能：

- 声音训练

通过 SDK 声音训练接口采集用户声音信息，采集三次成功表示创建声音验证信息成功。

- 声音验证

通过 SDK 声音验证接口采集用户声音信息进行验证是否为创建声音的声音信息。

- 声音删除

删除声音验证的一切相关信息。注：删除后用户将无法使用声音验证

## 2.4 二维码相关

Secken SDK 以及 Server API，可实现客户端扫码，服务器端验证用户身份的功能。

## 2.5 验证功能

除了上面所述的人脸和声音验证外，SeckenSDK 还有一键验证功能，第三方收到验证推送信息或者扫描授权信息后，可以进行通过调用 SeckenSDK 验证接口进行一键验证。

## 2.6 push 相关

第三方应用可以通过 SeckenSDK 的 pull 接口主动拉取验证信息。也可以通过 updatePushId 更新推送的 id

## 2.7 解除绑定

即取消用户的授权，客户端将不能使用 SDK 功能，但是不删除授权的声音和人脸信息，下次授权用户仍然可以使用之前训练的人脸和声音进行验证。

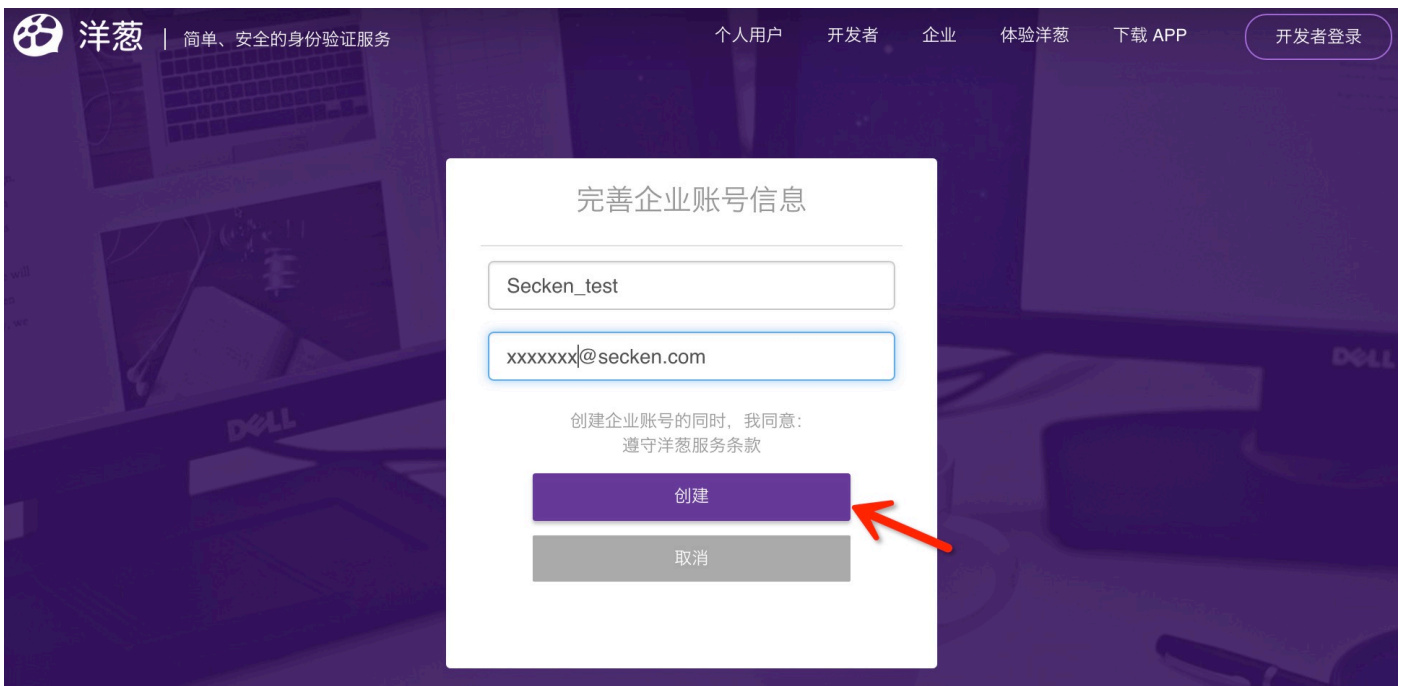
## 3. 申请应用

在使用 Secken SDK 为您提供的各种功能之前，您需要先到洋葱的开发者中心注册移动应用，并获取 APPKEY 和 APP\_ID。(注：洋葱官网的账号和开发者中心账号通用)

洋葱开发者中心地址：<https://www.yangcong.com/signin>

### (1) 登陆开发者中心





(2) .创建 SDK 应用，并获取 APPKEY 和 APPID。

洋葱开发者中心

+ 创建

主页

我的应用

账号管理

订阅

通知中心

创建

公有云应用


企业内网验证系统

SDK应用

填写基本信息

创建类型: SDK应用

上传 LOGO:



选择文件

 Maserati.jpg

应用名:

私有云SDKapp

应用描述:

私有云SDK 测试 app.

推送证书

选择文件

 未选择任何文件 (仅供 iOS 推送使用)

创建

洋葱开发者中心

+ 创建

主页

我的应用

账号管理

订阅

通知中心

我的应用

公有云应用

企业内网验证系统

SDK应用

时间	人脸验证	声音验证	指纹验证	一键验证	所有验证
今天	0	0	0	0	0
昨天	0	0	0	0	0
本周	0	0	0	0	0
本月	0	0	0	0	0

应用状态

正常

应用 ID

uGWzcj6JMySfmLd8rBbvFCxrAlnVmKTE

应用 KEY

5Hm6Jo020HALyDw5wt3c

授权 ID

vL841sSd2R3NrHCuh1ab

应用类型

SDK 应用

## 4. 在 Xcode 中集成 SDK

### 4.1 导入 SDK 开发包到你自己的工程项目

集成 SDK，以下分为三个步骤来进行，每个步骤都会详细的说明。

注意：不能从第一步直接跳到第三部，










开始第一步骤

欢迎使用 SeckenSDK.framework，在使用此库之前请先导入以下系统库

- 1, Security.framework
- 2, AudioToolBox.framework
- 3, AVFoundation.framework
- 4, Foundation.framework
- 5, UIKit.framework
- 6, MobileCoreServices.framework
- 7, SystemConfiguration.framework
- 8, Libz.tbd

效果如下图所示。

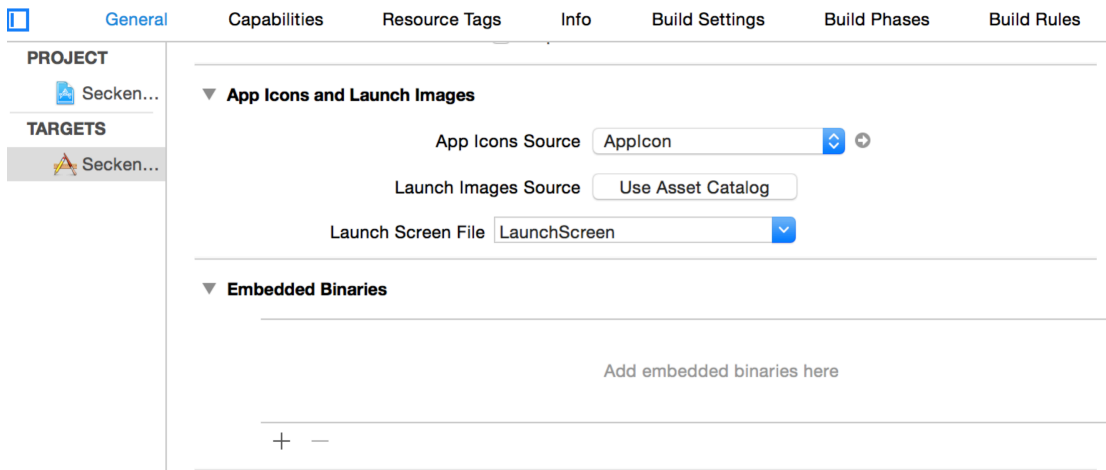
#### ▼ Linked Frameworks and Libraries

Name	Status
 Security.framework	Required ⚡
 SeckenSDK.framework	Required ⚡
 AudioToolbox.framework	Required ⚡
 AVFoundation.framework	Required ⚡
 Foundation.framework	Required ⚡
 UIKit.framework	Required ⚡
 MobileCoreServices.framework	Required ⚡
 SystemConfiguration.framework	Required ⚡
 libz.tbd	Required ⚡
+ -	

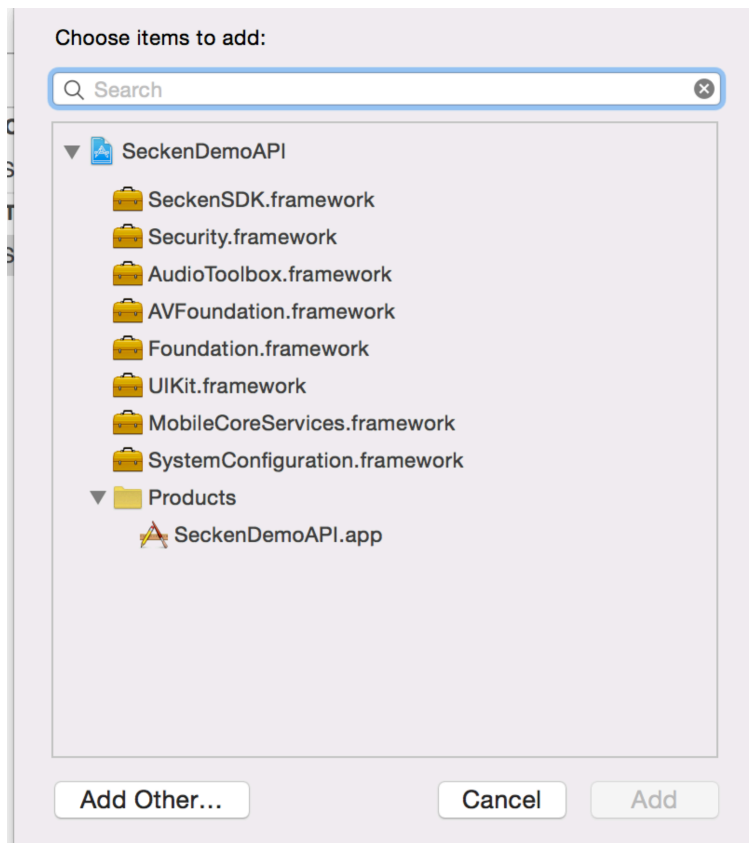
紧接着，接下来就开始导入 SeckenSDK.framework

首先到工程的 Targets -> General 界面

如下图所示。

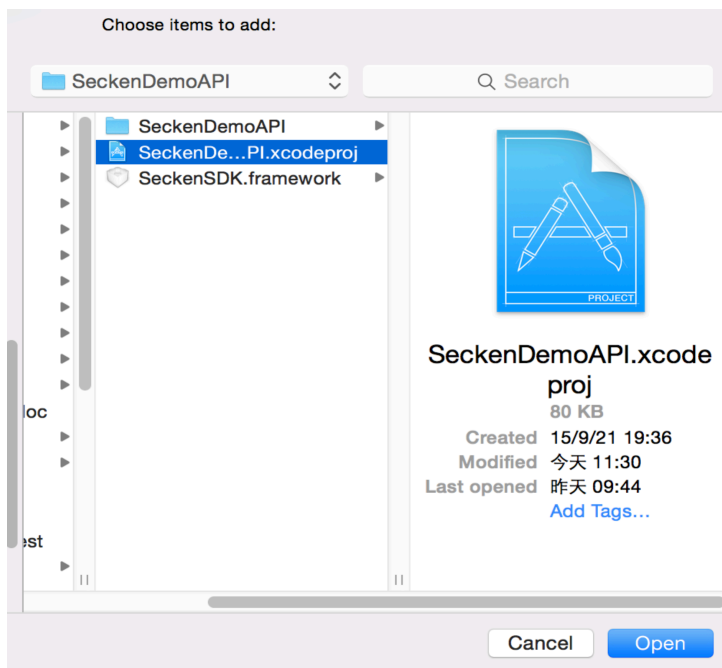


然后在“Embedded Binaries” 这栏点击 ‘+’ 加号  
这个时候弹出 下图 界面



如果 SeckenSDK.framework 存在这个列表当中，就选择，然后点击右下角的 Add 按钮。  
如果列表没有 SeckenSDK.framework 就点击 Add Other，就会弹出 下图 界面



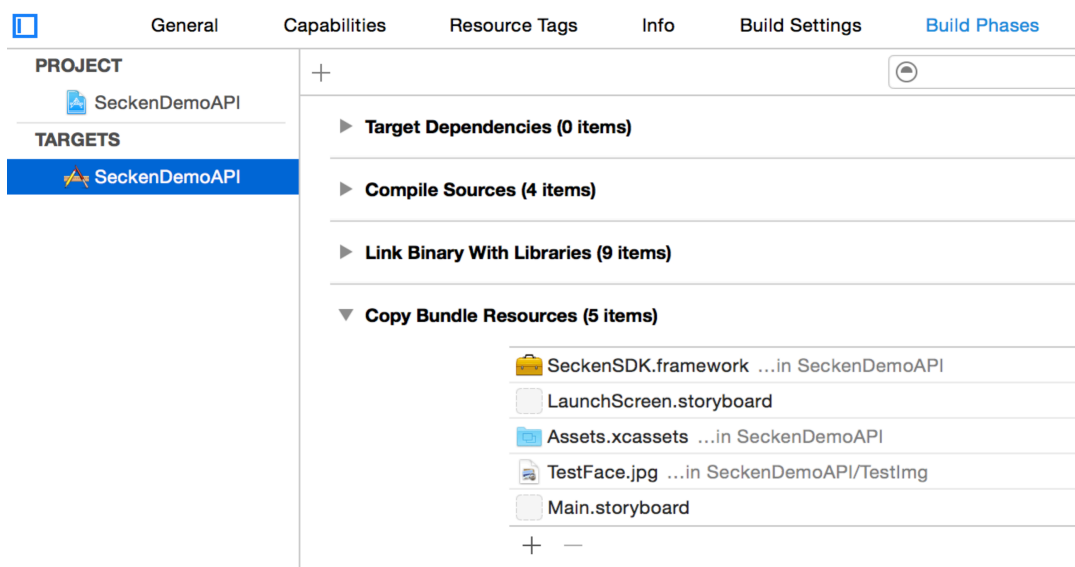


然后找到 SeckenSDK.framework 选择，点击 Open，弹出一个界面选择 Create Groups，点击右下角 Finish 按钮。

这个时候，第一步骤已经完成，紧接着，开始第二步骤。

点击 Targets -> Build Phases 选项。

如下图所示



这个时候，找到 Copy Bundle Resources，并且展开。

如果列表没有 SeckenSDK.framework，就点击 左下角的 ‘+’ 加号

然后添加 SeckenSDK.framework

添加完以后，或者展开之后列表存在的话，我们就开始进行第三步骤。

第三步骤。

在 AppDelegate.m 文件导入 #import "Secken/SeckenSDK.h"

然后通过以下代码授权，就可以使用了。

```
[SeckenSDK registerAppID:@" 您的 APPID" appKey:@" 您的 APPKEY" ];
```

调用人脸训练方法

```
[[SeckenSDK currSeckenSDK] faceTrainUserName:self.username  
    token:self.token  
    step:value  
    faceImage:image  
    faceTraninSuccess:^(id operation) {  
        NSLog(@" 成功" );  
    } faceTraninFail:^(id operation) {  
        NSLog(@" 失败" );  
    }  
];
```

调用人脸鉴定方法

```
[[SeckenSDK currSeckenSDK] faceCompareUserName:self.username  
    token:self.token  
    faceImage:[UIImage imageNamed:@"TestFace"]  
    faceCompareSuccess:^(id operation){  
        NSLog(@" 成功" );  
    } faceCompareFail:^(id operation) {  
        NSLog(@" 失败" );  
    }  
];
```

调用人脸删除方法

```
[[SeckenSDK currSeckenSDK] faceDeleteUserName:self.username
                        token:self.token
                        faceDeleteSuccess:^(id operation){
                            NSLog(@" 成功" );
                        } faceDeleteFail:^(id operation) {
                            NSLog(@" 失败" );
                        }
];
```

调用声音训练

/\*

首先要通过 SeckenVoiceTrain 训练，获取 voiceID 和 regionID

\*/

```
CGRect vTrainRect = CGRectMake( 0, 0,
                                [UIScreen mainScreen].bounds.size.width,
                                [UIScreen mainScreen].bounds.size.height);
```

```
SeckenVoiceTrain * voiceTrain = [[SeckenVoiceTrain alloc] initWithFrame:vTrainRect];
[self.view addSubview:voiceTrain];
```

```
voiceTrain.resultBlock = ^(id resultBlock){
    /*
    让后判断 hasVoice 和 hasRegID 参数的值是否为 1 或者为 0
    */
    if([[resultBlock objectForKey:@"hasVoice"] boolValue] == 1 ||
        [[resultBlock objectForKey:@"hasRegID"] boolValue] == 1)
    {
        [[SeckenSDK currSeckenSDK] voiceTrainUserName:userName
                                token:token
                                voiceTrainSuccess:^(id operation)
```

```

        {
            NSLog(@" 成功" );
        } voiceTrainFail:^(id operation) {
            NSLog(@" 失败" );
        };
    }
};

```

调用声音验证

/\*通过 SeckenVoiceConfirm 类来验证\*/

```

CGRect vConfirmRect = CGRectMake( 0, 0,
                                   [UIScreen mainScreen].bounds.size.width,
                                   [UIScreen mainScreen].bounds.size.height);
SeckenVoiceConfirm * voiceConfirm = [[SeckenVoiceConfirm alloc]
initWithFrame:vConfirmRect
userNmae:self.username
token:self.token
SuccessBlock:^(id operation) {
    NSLog(@" 成功" );
} FailBlock:^(id operation) {
    NSLog(@" 失败" );
}];
[self.view addSubview:voiceConfirm];

```

调用声音删除

/\*首先创建 SeckenVoiceDelete 类，然后在回调中删除声音\*/

```

SeckenVoiceDelete *voiceDelete = [[SeckenVoiceDelete alloc]
initWithVoiceDelSuccessBlock:^(id operation) {

    [[SeckenSDK currSeckenSDK] voiceDeleteUserName:self.username
    token:self.token
    voiceDelSuccess:^(id operation) {
        NSLog(@" 成功" );
    }
}];

```

```

        } voiceDelFail:^(id operation) {
            NSLog(@" 失败" );
        };

    } delFailBlock:^(id operation) {
        NSLog(@" 失败" );
    };
};

```

调用扫一扫

```

[[SeckenSDK currSeckenSDK] authQR:@"http://yc"
    token:self.token
    userName:self.username
    longitude:@"000.00"
    latitude:@"000.00"
    authSuccess:^(id operation) {
        NSLog(@" 成功" );
    } authFail:^(id operation) {
        NSLog(@" 失败" );
    };
};

```

调用确认

```

[[SeckenSDK currSeckenSDK] confirmToken:self.token
    userName:self.username
    agree:@"1"
    eventID:nil
    authSuccess:^(id operation)
    {
        NSLog(@"成功");
    }
    authFail:^(id operation) {
        NSLog(@"失败");
    }
};

```

```
});
```

调用推送方法，首先通过 SDK 的回调监听推送。

```
[[SeckenSDK currSeckenSDK] observeResultPullBlock:^(id operation) {

    /*然后通过 pull 拉取内容。比如人脸验证的推送或者声音验证的推送*/
    [[SeckenSDK currSeckenSDK] pullUserName:self.username
        token:self.token
        pullSuccess:^(id operation) {
            NSLog(@"成功");
        } pullFail:^(id operation) {
            NSLog(@"失败");
        }
    ];
}];
```

## 5 在 Xcode 中集成 UI SDK

注：如果没有使用 UI SDK 的话，可以忽略以下（第 5 项）内容。

由于 UI SDK 基于 API SDK，接下来开始进行集成 UI SDK 相关。

- 1，请按照标题 “集成 SDK” 进行 SeckenSDK 的集成。
- 2，然后 SeckenUI 集成方式与 SeckenSDK 一样。
- 3，开始使用 UI SDK

调用人脸训练类

首先通过 NSBundle 拿到 SeckenUI 的路径。

```
NSString * path = [[NSBundle mainBundle] pathForResource:@"SeckenUI"
ofType:@"framework"];
NSBundle * bundle = [NSBundle bundleWithPath:path];
```

//通过 bundle 加在 xib 文件。并传入 token 和 username 参数

```
SCFaceTrainController * face = [[SCFaceTrainController alloc]
```

```
initWithNibName:@"SCFaceTrainController" bundle:bundle];
    face.token = self.token;
    face.userName = self.username;

    //显示回调的结果。
    face.successBlock = ^(id operation){
        NSLog(@"operation = %@", operation);
    };
    face.failBlock = ^(id operation){
        NSLog(@"operation = %@", operation);
    };
    //push 到人脸训练的界面。
    [self.navigationController pushViewController:face animated:YES];
```

## 调用人脸验证类

```
//通过主 Bundle 拿到 SeckenUI.framework 的路径，并且加载 Bundle
NSString * path = [[NSBundle mainBundle] pathForResource:@"SeckenUI"
ofType:@"framework"];
NSBundle * bundle = [NSBundle bundleWithPath:path];

//通过 bundle 加在 xib 文件。并传入 token 和 username 参数
SCFaceConfirmController * faceConfirm = [[SCFaceConfirmController alloc]
initWithNibName:@"SCFaceConfirmController" bundle:bundle];
faceConfirm.token = self.token;
faceConfirm.userName = self.username;

faceConfirm.successBlock = ^(id operation){
    NSLog(@"operation = %@", operation);
};
faceConfirm.failBlock = ^(id operation){
    NSLog(@"operation = %@", operation);
```

```
};  
[self.navigationController pushViewController:faceConfirm animated:YES];
```

#### 调用人脸删除类

//通过调用 SeckenSDK , 然后删除人脸。

```
[[SeckenSDK currSeckenSDK] faceDeleteUserName:self.username token:self.token  
faceDeleteSuccess:^(id operation) {  
  
    NSLog(@"operation = %@", operation);  
  
} faceDeleteFail:^(id operation) {  
  
    NSLog(@"operation = %@", operation);  
}];
```

#### 调用声纹训练类

```
SCVoiceTrainController * scVoice = [[SCVoiceTrainController alloc] init];  
scVoice.token = self.token;  
scVoice.userName = self.username;  
  
scVoice.successBlock = ^(id operation){  
    NSLog(@"operation = %@", operation);  
};  
scVoice.failBlock = ^(id operation){  
    NSLog(@"operation = %@", operation);  
};  
  
[self.navigationController pushViewController:scVoice animated:YES];
```

#### 调用声纹验证类

```
SCVoiceConfirmController * scVoiceConfirm = [[SCVoiceConfirmController alloc] init];
```



```

scVoiceConfirm.strToken = self.token;
scVoiceConfirm.strUserName = self.username;
scVoiceConfirm.successBlock = ^(id operation){
    NSLog(@"operation = %@", operation);
};

scVoiceConfirm.failBlock = ^(id operation){
    [self.navigationController popViewControllerAnimated:YES];
};
[self.navigationController pushViewController:scVoiceConfirm animated:YES];

```

调用声纹删除

首先通过 SeckenVoiceDelete 类 标记删除，然后通过 SeckenSDK 进行删除。

如果不进行标记删除而直接调用 SeckenSDK 的删除，声纹并不会真正删除。

```

SeckenVoiceDelete * voiceDel = [[SeckenVoiceDelete alloc]
initWithVoiceDelSuccessBlock:^(id operation) {

    [[SeckenSDK currSeckenSDK] voiceDeleteUserName:self.username token:self.token
voiceDelSuccess:^(id operationDel){

    } voiceDelFail:^(id operationDel) {

    }

}];

} delFailBlock:^(id operation) {

}

}];

```

调用扫一扫类

//通过主 Bundle 拿到 SeckenUI.framework 的路径，并且加载 Bundle

```
NSString * path = [[NSBundle mainBundle] pathForResource:@"SeckenUI"
ofType:@"framework"];
```

```
NSBundle * bundle = [NSBundle bundleWithPath:path];
```

//通过 bundle 加载 xib 文件。并传入 token 和 username 参数

```
SCQRViewController * scanQR = [[SCQRViewController alloc]
initWithNibName:@"SCQRViewController" bundle:bundle];
```

```
scanQR.token = self.token;
```

```
scanQR.userName = self.username;
```

```
scanQR.longitude = @"123.00";
```

```
scanQR.latitude = @"123.00";
```

```
scanQR.successBlock = ^(id operation){
```

```
    NSLog(@"operation = %@", operation);
```

```
};
```

```
scanQR.failBlock = ^(id operation){
```

```
    NSLog(@"operation = %@", operation);
```

```
};
```

```
[self.navigationController pushViewController:scanQR animated:YES];
```

调用确认推送消息类型类 observeResultPullBlock 方法监听推送内容，  
然后通过 SCVerifyController 确认推送消息类型。

```
SCVerifyController * scVerify = [[SCVerifyController alloc] init];
```

```
scVerify.authType = authType;
```

```
scVerify.strUserName = self.username;
```

```
scVerify.strToken = self.token;
```

```
scVerify.strIPAddress = [operationSuccess objectForKey:@"ip"];
```

```
scVerify.strEventID = [operationSuccess objectForKey:@"event_id"];
```

```
scVerify.logoImage = [operationSuccess objectForKey:@"app_logo"];
```

```
scVerify.strDescContent = [operationSuccess objectForKey:@"action_details"];
```

```

scVerify.strTitle = [operationSuccess objectForKey:@"action_type"];
scVerify.strAppName = [operationSuccess objectForKey:@"app_name"];
scVerify.successBlock = ^(id operation){
    if ([[operation objectForKey:@"code"] intValue] == 200) {
        NSLog (@ "Success (成功)")
    };

    scVerify.failBlock = ^(id operation){
        NSLog (@ "Fail (失败)")
    };

[self.navigationController pushViewController:scVerify animated:YES];

```

## 6 错误码

客户端跟服务器交互错误：

错误码	错误消息
40001	上传参数格式错误
40002	解密错误
40003	无上传参数
40004	上传参数缺少 secret
40005	应用或者用户不存在
40006	token 过期
40007	二维码无效
40008	二维码信息不匹配
40009	人脸不匹配
40010	人脸信息不存在
40011	训练人脸中步数错误
40012	事件 id 不存在
40013	二维码未被扫描
40014	没有推送消息
40015	应用程序不存在
40016	人脸模糊，请重试

60001	网络异常
60002	数据格式错误
60003	解密失败
60004	连接服务器异常
60005	签名错误
60006	当前验证授权已停用或者失效

服务器错误：

50001	服务端异常
-------	-------

在开发或者使用过程中，遇到问题或者疑问，请及时联系洋葱团队，我们都将竭尽全力解决。

赛肯保留最终解释权

电话：010-64772882

QQ 群：154697540

邮箱：support@secken.com