VerifyCode iOS SDK 接入指南

一、SDK集成

CocoaPods集成方式

- 1、更新Podfile文件

  在工程的 Podfile 里对应的 Target 中添加以下代码

  ```
  pod 'VerifyCode'
  ```

- 2、集成SDK

  在工程的当前目录下, 运行 `pod install` 或者 `pod update`

- 3、工程设置

  在工程target目录内，需将Build Settings —> other link flags设置为-ObjC。

  备注:

  (1). 命令行下执行 `pod search VerifyCode` ,如显示的 `VerifyCode` 版本不是最新的，则先执行 `pod update` 操作更新本地repo的内容

  (2). 如果想使用最新版本的SDK，则执行 `pod update`

  (3). 如果你的工程设置的"Deplyment Target"低于 9.0，则在Podfile文件的前面加上以下语句 platform :ios, '9.0'

手动集成方式

- 1、下载VerifyCode SDK包
  - 地址: `https://github.com/yidun/captcha-ios-demo`
- 2、导入 `VerifyCode.framework` 到XCode工程:
  - 拖拽 `VerifyCode.framework` 文件到Xcode工程内(请勾选Copy items if needed选项)
- 3、导入 `NTESVerifyCodeResources.bundle` 到工程中:
  - 进入 `Build Phase` ，在 `Copy Bundle Resources` 选项中，添加 `NTESVerifyCodeResources.bundle` 文件(请勾选Copy items if needed选项) 。

- 4、添加依赖库 `SystemConfiguration.framework` `JavaScriptCore.framework` 、 `WebKit.framework`

- 5、工程设置

  在工程target目录内，需将Build Settings —> other link flags设置为-ObjC。

  备注:
  (1)如果已存在上述的系统framework，则忽略

  (2)SDK 最低兼容系统版本 iOS 9.0

二、SDK 使用

2.1 Object-C 工程

- 1、在项目需要使用SDK的文件中引入VerifyCode SDK头文件，如下：

```
#import <VerifyCode/NTESVerifyCodeManager.h>
```

- 2、在页面初始化的地方初始化 SDK，SDK同时支持无感知验证码和普通验证码，需在官网申请不同的captchaID，如下：

```
- (void)viewDidLoad {
    [super viewDidLoad];

    // sdk调用
    self.manager = [NTESVerifyCodeManager getInstance];
    self.manager.delegate = self;

    // 设置透明度
    self.manager.alpha = 0.7;

    // 设置frame
    self.manager.frame = CGRectNull;

    // captchaId从网易申请，比如@"a05f036b70ab447b87cc788af9a60974"

    // 普通验证码
    // NSString *captchaid = @"deecf3951a614b71b4b1502c072be1c1";
    // self.manager.mode = NTESVerifyCodeNormal;

    // 无感知验证码
    NSString *captchaid = @"6a5cab86b0eb4c309ccb61073c4ab672";
    self.manager.mode = NTESVerifyCodeBind;
}
```

- 3、在需要验证码验证的地方，调用SDK的openVerifyCodeView接口，如下:

```
[self.manager openVerifyCodeView:nil];
```

- 4、如果需要处理VerifyCode SDK的回调信息，则实现NTESVerifyCodeManagerDelegate即可

(1) 初始化完成

```
/**
* 验证码组件初始化完成
*/
- (void)verifyCodeInitFinish{
    // App添加自己的处理逻辑
}
```

(2) 初始化出错

```
/**
* 验证码组件初始化出错
*
* @param error 错误信息
*/
- (void)verifyCodeInitFailed:(NSArray *)error{
    // App添加自己的处理逻辑
}
```

(3) 验证结果回调

```
/**
* 完成验证之后的回调
*
* @param result 验证结果 BOOL:YES/NO
* @param validate 二次校验数据，如果验证结果为false，validate返回空
* @param message 结果描述信息
*
*/
- (void)verifyCodeValidateFinish:(BOOL)result
                            validate:(NSString *)validate
                            message:(NSString *)message{
    // App添加自己的处理逻辑
}
```

(4) 关闭验证码窗口的回调

```
/**
* 关闭验证码窗口后的回调
*/
- (void)verifyCodeCloseWindow{
    //App添加自己的处理逻辑
}
```

(5) 网络错误

```
/**
* 网络错误
*
* @param error 网络错误信息
*/
- (void)verifyCodeNetError:(NSError *)error{
    //App添加自己的处理逻辑
}
```

备注: 如果不需要处理VerifyCode SDK的回调信息，也可不实现

2.2 Swift 工程

- 1、在项目对应的 bridging-header.h 中引入头文件，如下：

```
#import <VerifyCode/NTESVerifyCodeManager.h>
```

备注: Swift 调用 Objective-C 需要一个名为 `<工程名>-Bridging-Header.h` 的桥接头文件。文件的作用为 Swift 调用 Objective-C 对象提供桥接。

- 2、其他调用同上

三、SDK 接口

- 1、枚举

```
/**
* @abstract    设置验证码语言类型
*/
typedef NS_ENUM(NSInteger, NTESVerifyCodeLang) {
    // 中文
    NTESVerifyCodeLangCN = 1,
    // 英文
    NTESVerifyCodeLangEN,
    // 繁体
    NTESVerifyCodeLangTW,
    // 日文
    NTESVerifyCodeLangJP,
    // 韩文
    NTESVerifyCodeLangKR,
    // 泰文
    NTESVerifyCodeLangTL,
    // 越南语
    NTESVerifyCodeLangVT,
    // 法语
    NTESVerifyCodeLangFRA,
    // 俄语
    NTESVerifyCodeLangRUS,
    // 阿拉伯语
    NTESVerifyCodeLangKSA,
    // 德语
    NTESVerifyCodeLangDE,
    // 意大利语
    NTESVerifyCodeLangIT,
    // 希伯来语
    NTESVerifyCodeLangHE,
    // 印地语
    NTESVerifyCodeLangHI,
```

```
    // 印尼语
    NTESVerifyCodeLangID,
    // 缅甸语
    NTESVerifyCodeLangMY,
    // 老挝语
    NTESVerifyCodeLangLO,
    // 马来语
    NTESVerifyCodeLangMS,
    // 波兰语
    NTESVerifyCodeLangPL,
    // 葡萄牙语
    NTESVerifyCodeLangPT,
    // 西班牙语
    NTESVerifyCodeLangES,
    // 土耳其语
    NTESVerifyCodeLangTR,
};

/**
* @abstract    设置验证码类型
*/
typedef NS_ENUM(NSInteger, NTESVerifyCodeMode) {
    // 普通验证码
    NTESVerifyCodeNormal = 1,
    // 无感知验证码
    NTESVerifyCodeBind,
};
```

- 2、属性

```
/**
* @abstract    验证码图片显示的frame
*
* @说明          验证码控件显示的位置,可以不传递。
*               (1)如果不传递或者传递为CGRectNull(CGRectZero),则使用默认值:topView的居中显示,宽度为屏幕宽度的4/5,高度:view宽度/2.0 + 65
*               (2)如果传递,则frame的宽度至少为270;高度至少为:宽度/2.0 + 65.
*/
@property(nonatomic) CGRect            frame;
```

```
/**
* @abstract    验证码图片背景的透明度
*
* @说明          范围:0~1，0表示全透明，1表示不透明。默认值:0.8
*/
@property(nonatomic) CGFloat           alpha;
```

```
/**
 * @abstract    验证码图片背景的颜色
 *
 * @说明          默认值:黑色
 */
@property(nonatomic) UIColor           *color;
```

```
/**
 * @abstract    验证码语言选项
 *
 * @说明          验证码枚举类型NTESVerifyCodeLang，NTESVerifyCodeLangCN表示中文，NTES
 *               不传默认中文。
 */
 @property(nonatomic) NTESVerifyCodeLang    lang;
```

```
/**
* @abstract    验证码滑块icon url，不传则使用易盾默认滑块显示。
*/
@property(nonatomic) NSString *slideIconURL;
```

```
/**
 * @abstract    验证码验证成功的滑块icon url，不传则使用易盾默认滑块显示。
 */
@property(nonatomic) NSString *slideIconSuccessURL;
```

```
/**
 * @abstract    验证码滑块滑动过程中的icon url，不传则使用易盾默认滑块显示。
 */
@property(nonatomic) NSString *slideIconMovingURL;
```

```
/**
 * @abstract    验证码验证失败的滑块icon url，不传则使用易盾默认滑块显示。
 */
@property(nonatomic) NSString *slideIconErrorURL;
```

```
/**
 * @abstract     设置验证码类型
 *
 * @说明          验证码枚举类型NTESVerifyCodeMode，可选类型见枚举定义
 *               不传默认普通验证码。
 *
 */
@property(nonatomic) NTESVerifyCodeMode mode;
```

```
/**
 * @abstract     设置极端情况下，当验证码服务不可用时，是否开启降级方案。
 *               默认开启，当触发降级开关时，将直接通过验证，进入下一步。
 */
@property(nonatomic) BOOL openFallBack;
```

```
/**
 * @abstract     设置发生第fallBackCount次错误时，将触发降级。取值范围 >=1
 *               默认设置为3次，第三次服务器发生错误时，触发降级，直接通过验证。
 */
@property(nonatomic) NSUInteger fallBackCount;
```

```
/**
* @abstract     是否隐藏关闭按钮
*               默认不隐藏，设置为YES隐藏，NO不隐藏
*/
@property(nonatomic) BOOL closeButtonHidden;
```

- 3、初始化

```
/**
 *  @abstract   初始化方法
 *
 *  @return     返回NTESVerifyCodeManager实例对象
 */
+ (NTESVerifyCodeManager *)getInstance;
```

- 4、配置参数

```
/**
 *  @abstract   配置参数
 *
 *  @param      captcha_id      验证码id
 *  @param      timeoutInterval 加载验证码的超时时间,最长12s。这个时间尽量设置长一些，比如
 *
 */
- (void)configureVerifyCode:(NSString *)captcha_id
            timeout:(NSTimeInterval)timeoutInterval;
```

- 5、弹出验证码

```
/**
*  @abstract 展示验证码视图
*
*  @说明      展示位置:[[[UIApplication sharedApplication] delegate] window];全屏居中显示,宽度为屏幕宽度的4/5,高度:view宽度/2.0 + 65.
*/
- (void)openVerifyCodeView;
```

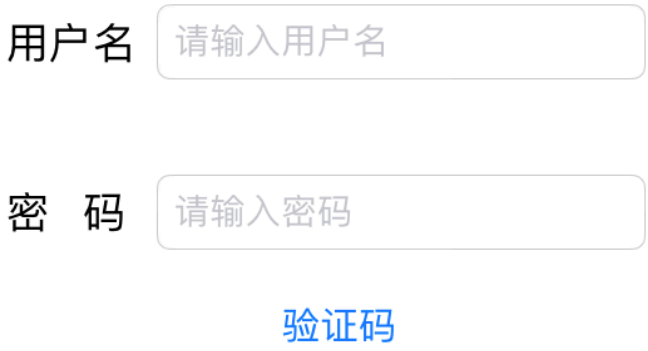
```
/**
*  @abstract   在指定的视图上展示验证码视图
*
*  @param      topView         加载验证码控件的父视图,可以为nil。
*                              (1)如果传递值为nil,则使用默认值:[[[UIApplication sharedApplication] delegate] window]
*                              (2)如果传递值不为nil，则注意topView的宽高值，宽度需为验证码frame宽度加上左右padding:frame.size.width + 16*2;高度至少为验证码framek高度加上上下padding:frame.size.height + 48 + 25
*
*/
- (void)openVerifyCodeView:(UIView *)topView;
```

- 6、log打印

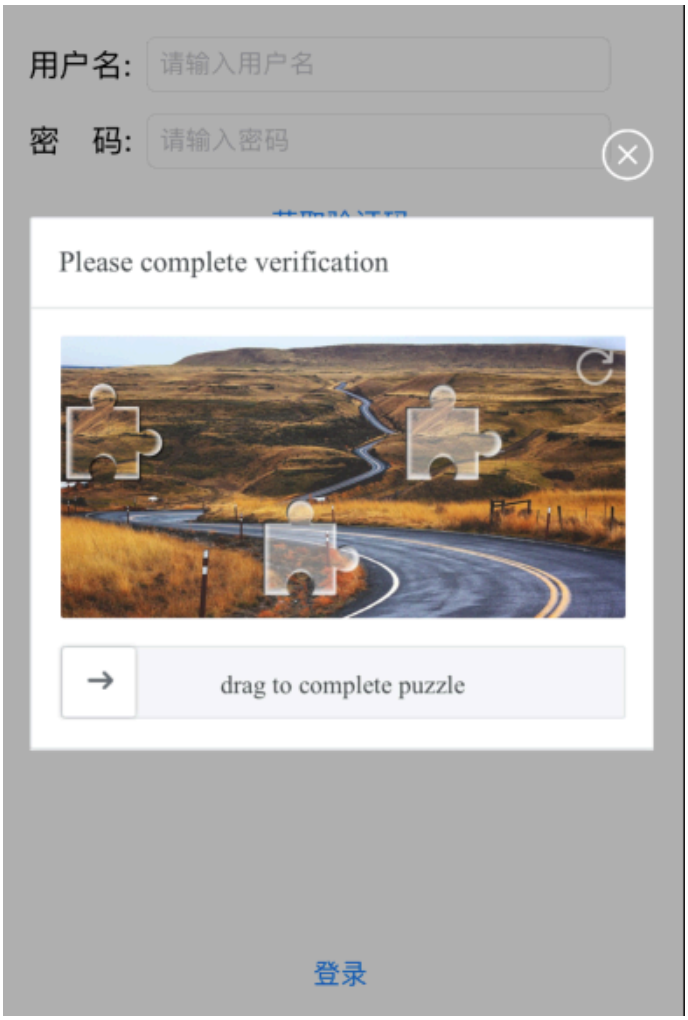
```
/**
*  @abstract    是否开启sdk日志打印
*
*  @param       enabled         YES:开启;NO:不开启
*
*  @说明         默认为NO,只打印workflow;设为YES后，Release下只会打印workflow和BGRLogLevelError
*/
- (void)enableLog:(BOOL)enabled;
```

四、效果演示

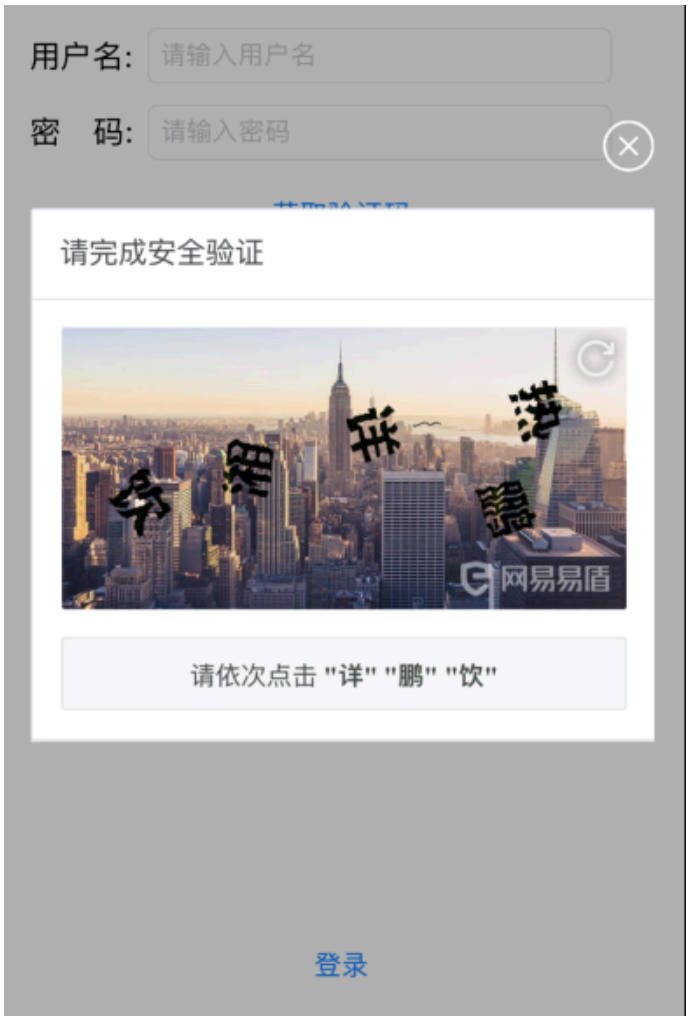
- 1、初始化



- 2、滑块验证


用户名:
请输入用户名
密 码:
请输入密码
Please complete verification
drag to complete puzzle
登录

- 3、点选验证

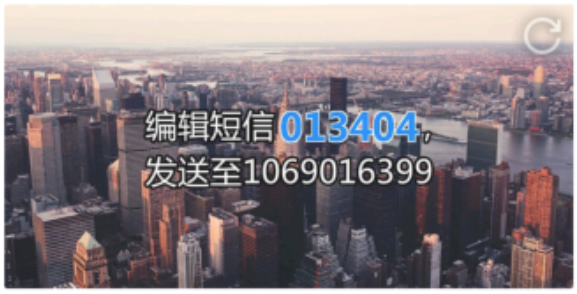


- 4、短信验证

用户名: 请输入用户名

密　码: 请输入密码

请完成安全验证



等待短信验证，剩余 83s

登录