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`

  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 7.0，则在Podfile文件的前面加上以下语句 platform :ios, '7.0'

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

  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 7.0

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，如下：

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

    // sdk调用
    self.manager = [NTESVerifyCodeManager sharedInstance];
    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"
    NSString *captchaId = @"a05f036b70ab447b87cc788af9a60974";
    [self.manager configureVerifyCode:self.captchaId timeout:5];
}
```

- 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 message 错误信息
*/
- (void)verifyCodeInitFailed:(NSString *)message{
    // 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、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     验证码语言选项
 *
 * @说明          验证码枚举类型NTESVerifyCodeLang，NTESVerifyCodeLangCN表示中文，NT
ESVerifyCodeLangEN表示英文
 *               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;
```

- 2、单例

```
/**
*  @abstract    单例
*
*  @return      返回NTESVerifyCodeManager对象
*/
+ (NTESVerifyCodeManager *)sharedInstance;
```

- 3、初始化

```
/**
 *  @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10s。这个时间尽量设置长一些,
比如5秒以上(5-10s)
 *
 */
- (void)configureVerifyCode:(NSString *)captcha_id
            timeout:(NSTimeInterval)timeoutInterval;
```

- 4、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，宽度至少为270;高度至少为:宽度/2.0 + 65.
*
*/
- (void)openVerifyCodeView:(UIView *)topView;
```

- 5、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
*/
```

四、效果演示

- 1、初始化

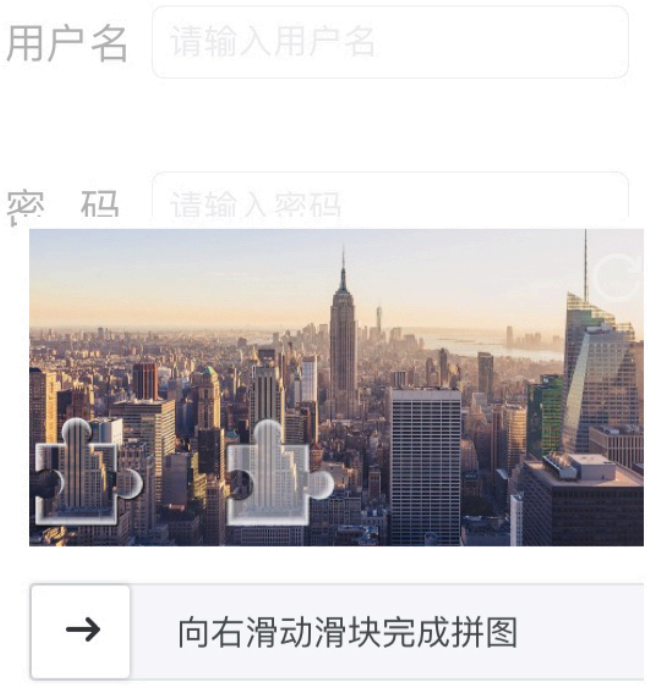
用户名 请输入用户名

密　码 请输入密码

验证码

登录

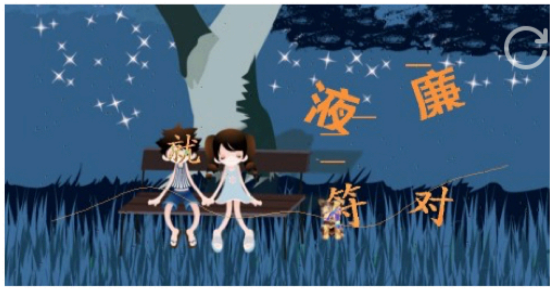
- 2、滑块验证



- 3、点选验证

用户名 请输入用户名

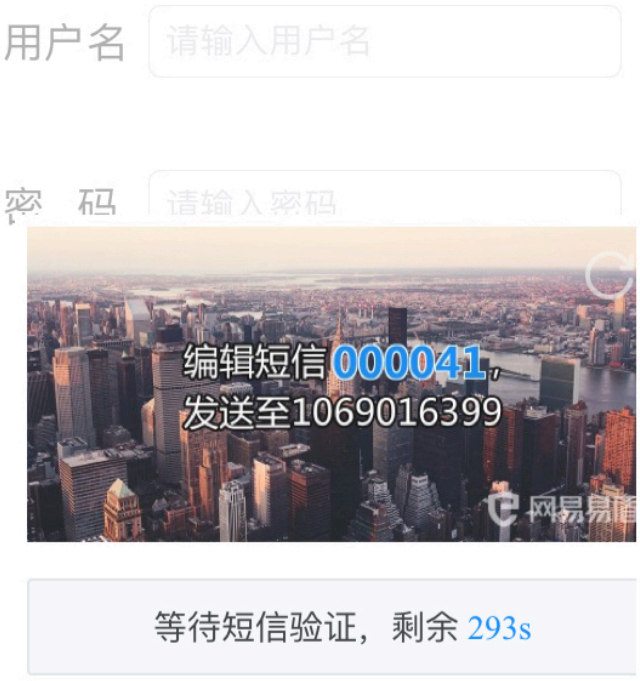
密　码 请输入密码



请依次点击 "符" "液" "对"

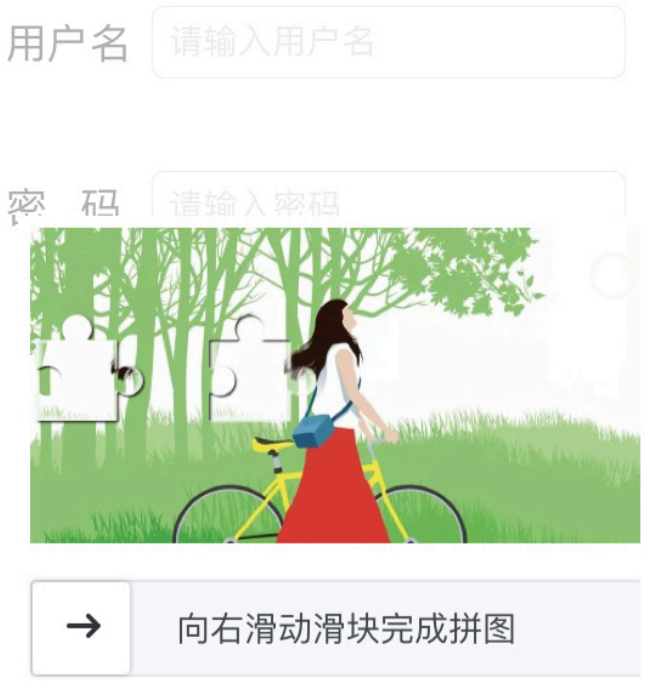
登录

- 4、短信验证



- 5、结果

  如果验证成功,则验证码图片消失;验证失败则弹出新的验证码



五、注意事项

- 1、如何自定义验证码视图?

  如果用户对验证码视图需要自定义，比如加一下UIButton控件，修改 `NTESVCController` 类的 `- (void)addVCViewOnBackgroundControl` 方法即可，比如:

```
- (void)addVCViewOnBackgroundControl{

    [self generateVerifyCodeView];
    [self.backGroundViewControl addSubview:self.verifyCodeView];

    // 自定义显示界面,验证码下加一个UIButton，UIButton的frame根据实际情况赋值
    UIButton *cancelButton = [[UIButton alloc] initWithFrame:CGRectMake(100, 1
00, 100, 20)];
    cancelButton.backgroundColor = [UIColor yellowColor];
    [cancelButton setTitleColor:[UIColor greenColor] forState:UIControlStateNo
rmal];
    [cancelButton setTitle:@"取消" forState:UIControlStateNormal];
    [self.backGroundViewControl addSubview:cancelButton];
}
```

- 2、验证码视图为什么没有显示?

  这种情况多见于APP不传递topView。没有显示的原因是topView获取的方式在不同的条件下，需要修改。 它的实现代码在 `NTESVCController.m` 文件里，对应的方法为:
  `- (UIView *)getTopView`

```
- (UIView *)getTopView{

   UIView *topView = [[[UIApplication sharedApplication] delegate] window];

   return topView;
}
```

这段代码的意思是，在多个UIWindow存在的情况下，获取正在使用的UIWindow作为topView。 不同的产品可以根据自己的需求修改这段代码，比如直接使用keyWindow或者任意自定义的UIWindow作为topView，或者以UIWindow的subviews的最前面的视图作为topView，例如:

```
- (UIView *)getTopView{

   UIView *topView = nil;

   UIApplication *app = [UIApplication sharedApplication];
   if (app) {
       UIWindow *topWindow = [[[UIApplication sharedApplication].windows  sort
edArrayUsingComparator:^NSComparisonResult(UIWindow *win1, UIWindow *win2) {
         return win1.windowLevel - win2.windowLevel;
       }] lastObject];
       topView = [[topWindow subviews] lastObject];
  }

  return topView;
}
```