title: 验证-ios-部署文档 type: 验证-ios-部署文档

order: 0

概述及资源

行为验证 iOS SDK 提供给集成 iOS 原生客户端开发的开发者使用, SDK 不依赖任何第三方库。

环境需求

条目	资源
开发目标	兼容iOS8+
开发环境	Xcode 11+
系统依赖	`Webkit.framework`, `JavascriptCore.framework`
sdk三方依赖	无

相关开发资源

条目	资源
产品结构流程	通讯流程, 交互流程
SDK接口文档	gt3-ios-docs 或查看头文件注释
错误码列表	Error Code 列表

安装

获取SDK

使用CocoaPods获取

在工程的 `Podfile` 文件中添加下面配置

```
pod 'GT3Captcha-iOS'
```

手动下载获取

使用从下方链接下载 `.zip` 文件获取最新的 SDK 及相关 Demo。

gt3-ios-sdk

导入SDK

1. 如果您是手动添加 SDK , 将下载获取的 `GT3Captcha.framework` 文件拖拽到工程中, 确保 `Copy items if needed` 已被勾选。

   

   请使用 `Linked Frameworks and Libraries` 方式导入framework。在拖入 `GT3Captcha.framework` 到工程时后, 还要检查 `.framework` 是否被添加到 `PROJECT -> Build Phases -> Linked Frameworks and Libraries` 。

   

2. 针对静态库中的 `Category` , 需要在对应target的 `Build Settings` -> `Other Linker Flags` 添加 `-all_load` 编译选项。建议先添加 `-ObjC` ，如果依然有问题，再添加 `-all_load` 。

   

3. SDK 0.9.0 版本以后需要在工程中同时引入 `GT3Captcha.Bundle` 。把仓库中的 `GT3Captcha.Bundle` 拖入项目中. iOS7 不支持 `Dynamic Library` , 无法使用 `embedded binaries` 。 而 `Dynamic Library` 无法获取 `.strings` 等资源文件, 这里以 bundle 外嵌的方式单独管理 SDK 所需资源文件。

配置接口

参考行为验证的通讯流程, 必须要先在您的后端搭建相应的**服务端SDK**，并配置从极验管理后台获取的 `id` 和 `key` , 并且将配置的接口 `API1` 和 `API2` 放入客户端的初始化方法中。

集成用户需要使用iOS SDK完成提供的以下接口:

1. 配置验证初始化
2. 启动验证
3. 获取验证结果
4. 处理错误的代理

提供以下两种方式使用验证码:

1. 直接通过 `GT3CaptchaManager` 与集成用户的事件绑定
2. 通过极验的提供的 `GT3CaptchaButton` (提供了默认样式)

此外, 无论使用何种方式集成, 都需要遵守 `<GTCaptchaManagerDelegate>` 协议以处理验证结果和可能返回的错误。

> 集成代码参考下方的**代码示例**

编译并运行你的工程

编译你的工程, 体验行为验证!

轻轻点击集成的验证按钮, 如此自然, 如此传神。

视觉展示

1. 与集成用户自定义的按钮事件绑定方式集成

   

2. 使用极验提供的独立按钮集成

   

代码示例

初始化与激活验证

在工程中的文件头部倒入验证动态库 `GT3Captcha.framework`

```
#import <GT3Captcha/GT3Captcha.h>
```

使用自定义按钮事件绑定的方式集成

> 以在 `UIButton` 中集成为参考

1. 初始化

   初始化验证管理器 `GTCaptchaManager` 的实例, 在 `UIButton` 初始化方法中对 `GTCaptchaManager` 实例调用注册方法以获得注册数据:

```
- (GT3CaptchaManager *)manager {
    if (!_manager) {
        _manager = [[GT3CaptchaManager alloc] initWithAPI1:api_1 API2
        _manager.delegate = self;

    }
    return _manager;
}

- (instancetype)initWithFrame:(CGRect)frame {
    self = [super initWithFrame:frame];

    if (self) {
        [self.manager registerCaptcha:nil];
    }
    return self;
}
```

2. 激活验证

   初始化和注册完成后, 调用下面的方法激活验证码:

```
[self.manager startGTCaptchaWithAnimated:YES];
```

使用 SDK 中的 `GT3CaptchaButton` 集成

> 以在 `ViewController` 中集成为参考

1. 初始化

   初始化验证管理器 `GTCaptchaManager` 的实例, 并将此实例传给验证按钮 `GTCaptchaButton` , 然后将验证按钮添加到 `[ViewController view]` 上:

```
- (void)viewWillAppear:(BOOL)animated {
    [self createCaptcha];
}

- (void)createCaptcha {
    //创建验证管理器实例
    GT3CaptchaManager *captchaManager = [[GT3CaptchaManager alloc] in
    captchaManager.delegate = self;

    //debug mode
//    [captchaManager enableDebugMode:YES];

    //创建验证视图的实例，并添加到父视图上
    GT3CaptchaButton *captchaButton = [[GT3CaptchaButton alloc] initW
    captchaButton.center = self.view.center;
    //推荐直接开启验证
    [captchaButton registerCaptcha:nil];
    [self.view addSubview:captchaButton];
}
```

2. 激活验证

   点击验证按钮或使用以下调用方式激活验证:

   ```
   [captchaButton startCaptcha];
   ```

处理验证结果

只有完成二次验证后, 本次验证才是完整完成。您需要在遵守 `GTCaptchaManagerDelegate` 协议后, 处理以下代理方法:

```
- (void)gtCaptcha:(GT3CaptchaManager *)manager didReceiveSecondaryCapt
chaData:(NSData *)data response:(NSURLResponse *)response error:(GT3Er
ror *)error decisionHandler:(void (^)(GT3SecondaryCaptchaPolicy))decis
ionHandler {
    if (!error) {
        //处理你的验证结果
        NSLog(@"\nsession ID: %@,\ndata: %@", [manager getCookieValue:@"
        //成功请调用decisionHandler(GT3SecondaryCaptchaPolicyAllow)
        decisionHandler(GT3SecondaryCaptchaPolicyAllow);
        //失败请调用decisionHandler(GT3SecondaryCaptchaPolicyForbidden)
        //decisionHandler(GT3SecondaryCaptchaPolicyForbidden);
    }
    else {
        //二次验证发生错误
        decisionHandler(GT3SecondaryCaptchaPolicyForbidden);
        NSLog(@"validate error: %ld, %@", (long)error.code, error.locali
    }
}
```

处理验证错误

验证过程中可能发生一些不可避免的错误, 您可以通过遵守 `GTCaptchaManagerDelegate` 协议后, 在下面的代理方法中进行处理:

```
- (void)gtCaptcha:(GTCaptchaManager *)manager errorHandler:(GTError *)
error {
    //处理验证中返回的错误
    NSLog(@"error: %@", error.localizedDescription);
}
```

可能遇到的错误码请参考后面的列表: GT3Error

可选验证处理方法

修改 API1 请求

考虑到部分公司的服务是需要鉴权或者自定义参数的, 可以通过下面的代理方法来对API1请求实现自定义

```
- (void)gtCaptcha:(GT3CaptchaManager *)manager willSendRequestAPI1:(NS
URLRequest *)originalRequest withReplacedHandler:(void (^)(NSURLReques
t *))replacedHandler {
    NSMutableURLRequest *mRequest = [originalRequest mutableCopy];

    /**
     TO-DO, 处理mRequest, 进行自定义
     */

    replacedHandler(mRequest);
}
```

> 完全自定义 API1 和 API2 都可以参考示例工程下载内 `AsyncTaskButton` 文件。

修改 API2 请求

考虑到部分公司的服务是需要鉴权或者自定义参数的, 可以通过下面的代理方法来对API1请求实现自定义

```
- (void)gtCaptcha:(GT3CaptchaManager *)manager willSendSecondaryCaptch
aRequest:(NSURLRequest *)originalRequest withReplacedRequest:(void (^)
(NSMutableURLRequest *))replacedRequest {
    /// 如果需要可以修改二次验证向服务器发送的请求
    NSMutableURLRequest *mReuqest = [originalRequest mutableCopy];
    NSData *secodaryData = mReuqest.HTTPBody;
    NSLog(@"data: %@", [[NSString alloc] initWithData:secodaryData encod

    /**
     TO-DO, 处理secodaryData, 添加业务数据
     */

//    NSData *newData = [secodaryData handlerYourData]
//    mReuqest.HTTPMethod = newData;
    replacedRequest(mReuqest);
}
```

> 完全自定义API1和API2都可以参考示例工程下载内 `AsyncTaskButton` 文件。

获取验证校验参数

如果需要记录验证返回的校验参数, 可以通过下面的代理方法来获取

```
- (void)gtCaptcha:(GT3CaptchaManager *)manager didReceiveCaptchaCode:(
NSString *)code result:(NSDictionary *)result message:(NSString *)mess
age {
    NSLog(@"result: %@", result);
}
```

禁用默认的 API1 请求

如果需要禁用 API1 请求, 可通过下面的方法进行修改

```
- (BOOL)shouldUseDefaultRegisterAPI:(GT3CaptchaManager *)manager {
    return NO;
}
```

因为API1请求被禁止, 需要手动配置验证的请求参数, 否则无法拉取验证, 示例如下:

```
// dict为包含gt、challenge、success的json字典
NSString *geetest_id = [dict objectForKey:@"gt"];
NSString *geetest_challenge = [dict objectForKey:@"challenge"];
NSNumber *geetest_success = [dict objectForKey:@"success"];
// 不要重复调用
[self.manager configureGTest:geetest_id challenge:geetest_challenge su
ccess:geetest_success withAPI2:api_2];
```

禁用默认API2请求

```
- (BOOL)shouldUseDefaultSecondaryValidate:(GT3CaptchaManager *)manager
 {
    return NO;
}
```

禁用默认的API2请求后, 验证结果的视图提示以代理方法 `gtCaptcha:didReceiveCaptchaCode:result:message:` 中返回的 `code` 为准, `code` 为 `@"1"` 则进行成功结果提示。

可选的验证集成模式

考虑到 API1 及 API2 均为网站主自己的接口，而不同公司对接口的访问有不同的约定。`0.13.0` 版本后，集成者可以根据 `GT3AsyncTaskProtocol` 协议创建含有自定义 API1 和 API2 的请求细节的对象，并通过 `registerCaptchaWithCustomAsyncTask:completion:` 将该对象注册进管理器。

使用该 `registerCaptchaWithCustomAsyncTask:completion:` 进行定义 API1 和

API2 请求流程后，验证管理器不再访问以下代理方法:

- `shouldUseDefaultRegisterAPI:`
- `gtCaptcha:willSendRequestAPI1:withReplacedHandler:`
- `gtCaptcha:didReceiveDataFromAPI1:withError:`
- `shouldUseDefaultSecondaryValidate:`
- `gtCaptcha:willSendSecondaryCaptchaRequest:withReplacedRequest:`

> 示例代码细节参考官方提供的 Demo 中 `AsyncTaskButton.m` 和 `DemoAyncTask.m` 文件。 示例工程下载

其他接口

详见官方的 SDK 接口文档

> 示例代码细节参考官方提供的 Demo