无感验证

一、产品简介

顶象无感验证结合了设备指纹、行为特征、访问频率、地理位置等多项技术，有效的拦截恶意登录、批量注册，阻断机器操作，拦截非正常用户，较传统验证码相比，用户无需再经过思考或输入操作，只需轻轻一滑即可进行验证。经过智能鉴别为正常的用户，在一定时间内无需再进行滑动操作，既为企业提供了安全保障也让用户无感知通过，极大提升用户体验。

1.1 产品特点

- 精准识别：机器学习结合智能策略模型，精准判定人机操作
- 极致体验：依托顶象先进架构，服务毫秒级响应
- 布局美观：弹窗、嵌入等多种形态，适用于各种业务场景，覆盖电脑、手机全平台
- 快速接入：SDK快速接入，仅需三步轻松搞定
- 数据可视化：丰富的可视化图表，防御拦截数据尽收眼底

1.2 产品安全性

顶象无感验证服务是基于北京顶象技术大数据风控体系提供的智能人机识别产品，依托顶象强大的技术研发和十余年的安全防控经验，顶象安全大脑可以在大数据海洋中精确定位各类异常操作，实时决策，并从海量真人操作中学习，不断进化识别能力，并已广泛应用于金融、通讯、IoT、电商、互联网等行业。

二、应用场景

无感验证产品主要可应用于任何需要验证的应用场景。包括并不限于如下场景：

- 账号类场景：用户注册、登录、修改邮箱、修改手机号、修改密码等业务场景；
- 活动类场景：红包福利领取、竞猜答题、积分兑换、团购抢购等运营性质的业务场景；
- UGC类场景：发帖、回复、点赞、上传图片等用户自身创造内容业务场景；
- 辅助验证类场景：发送短信验证码、发送邮箱验证码等触发某种验证业务场景；

三、使用指南

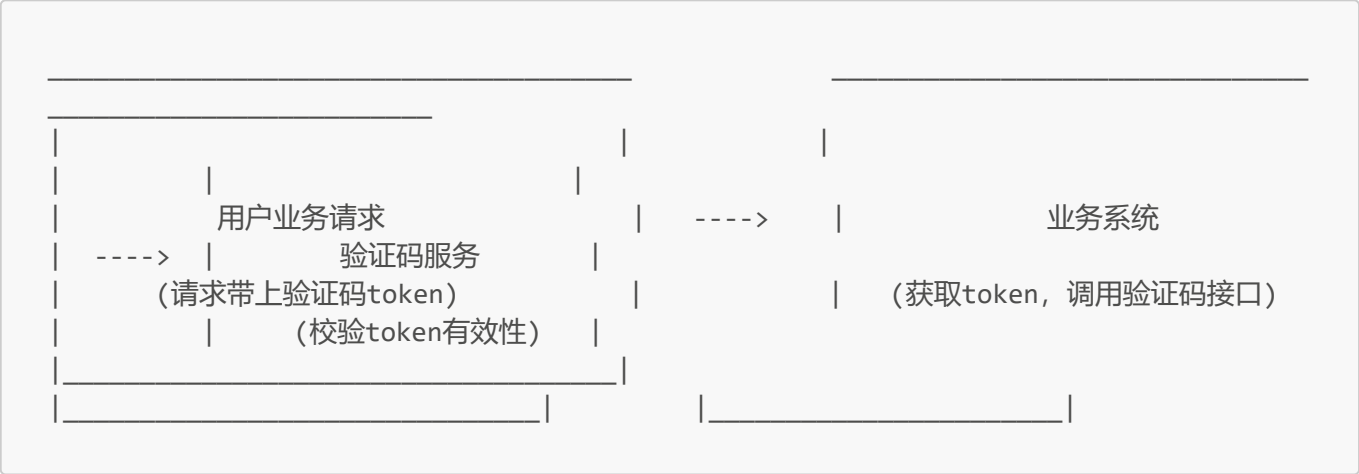
3.1 服务接入流程

3.2 开发前必读

- 针对需要保护的业务接口，在页面上嵌入无感验证，用户滑动成功后会得到一个安全token，业务接口需要带着token到后台进行token安全验证，验证通过以后再继续业务流程。
- 以登录为例，未接入验证码前的业务接口：
  `接口：http://domain/login 参数：用户名，密码`
- 现在在登录页面接入验证码，用户滑动以后，会得到一个安全token
  接入后的业务接口：`接口：http://domain/login 参数：用户名，密码，安全token`

四、无感验证后端接入（业务应用集成）

- 访问顶象官网，注册账号后登录控制台，访问"无感验证"模块，申请开通后系统会分配一个唯一的AppId、AppSecret。
- 当用户滑动验证码通过后，验证码服务会生成一个`token`，用户的业务请求带上这个验证码`token`，业务系统再调用后台 SDK 验证`token`的有效性。



顶象提供后端SDK来校验token是否合法 ， 目前支持JAVA版、PHP版、C#版、Python版。

token验证api返回数据

字段名	数据类型	描述
success	Boolean	是否成功，true/false
msg	String	错误信息
ip	String	token产生时，客户端的ip
code	String	错误code

错误code说明

code	描述
1001	错误的APPID
1002	签名错误，请核对APPID和APPSECRET是否正确
1003	token不合法或者已经过期（token是一次性使用并且只有两分钟有效期）

code	描述
1004	参数错误，请核对后端使用的APPID是否和前端页面一致
1005	verifyToken传入ip时，校验不通过，即token产生的ip和业务请求ip不一致

4.1 SDK-JAVA版

- **JAVA7及以上版本SDK下载 点击下载 ，如需其他版本SDK请联系官网在线客服。**
- maven项目引入依赖

```
<dependency>
    <groupId>com.dingxiang-inc</groupId>
    <artifactId>ctu-client-sdk</artifactId>
    <version>2.1</version>
</dependency>
```

- 当用户滑动验证码通过后，验证码服务会生成一个token，用户的业务请求带上这个验证码token，业务系统再去请求验证码服务验证token，验证码服务会返回token的合法性和采集到的客户端ip。如果业务系统可以采集提交业务参数的客户端ip，可以随token一并提交，验证码服务除了校验token的合法性还会校验客户端验证码验证和提交业务参数的ip是否一致。

```
/**构造入参为appId和appSecret
 * appId和前端验证码的appId保持一致，appId可公开
 * appSecret为秘钥，请勿公开
 * token在前端完成验证后可以获取到，随业务请求发送到后台，token有效期为两分钟
 * ip 可选，提交业务参数的客户端ip
 **/
String appId = "appId";
String appSecret = "appSecret";
CaptchaClient captchaClient = new CaptchaClient(appId,appSecret);
//captchaClient.setCaptchaUrl(captchaUrl);
//特殊情况需要额外指定服务器,可以在这个指定，默认情况下不需要设置
CaptchaResponse response = captchaClient.verifyToken(token);
//CaptchaResponse response = captchaClient.verifyToken(token, ip);
//针对一些token冒用的情况，业务方可以采集客户端ip随token一起提交到验证码服务，验证码服务除了判断token的合法性还会校验提交业务参数的客户端ip和验证码颁发token的客户端ip是否一致
System.out.println(response.getCaptchaStatus());
//确保验证状态是SERVER_SUCCESS，SDK中有容错机制，在网络出现异常的情况会返回通过
//System.out.println(response.getIp());
//验证码服务采集到的客户端ip
if (response.getResult()) {
    /**token验证通过，继续其他流程**/
} else {
    /**token验证失败，业务系统可以直接阻断该次请求或者继续弹验证码**/
}
```

4.2 SDK-PHP版

- **PHP版本SDK下载： 点击下载**

```php
include ("CaptchaClient.php");
/**构造入参为appId和appSecret
 * appId和前端验证码的appId保持一致，appId可公开
 * appSecret为秘钥，请勿公开
 * token在前端完成验证后可以获取到，随业务请求发送到后台，token有效期为两分钟**/
$appId = "appId";
$appSecret = "appSecret";
$client = new CaptchaClient($appId,$appSecret);
$client->setTimeOut(2);      //设置超时时间，默认2秒
# $client->setCaptchaUrl("http://cap.dingxiang-inc.com/api/tokenVerify");
//特殊情况可以额外指定服务器，默认情况下不需要设置
$response = $client->verifyToken(token);  //token指的是前端传递的值，即验证码验证成功颁发的token
echo $response->serverStatus;
//确保验证状态是SERVER_SUCCESS，SDK中有容错机制，在网络出现异常的情况会返回通过
if($response->result){
    echo "true";
    /**token验证通过，继续其他流程**/
}else{
    echo "false";
    /**token验证失败**/
}
```

4.3 SDK-C#版

- **C#版本SDK下载： 点击下载**

```csharp
// 依赖 .NET Framework 4.0
// 在您的项目中添加DxCsharpSDK.dll引用
String appId = "appId";
String appSecret = "appSecret";
DxCsharpSDK.Captcha captcha = new DxCsharpSDK.Captcha(appId,appSecret);
//captcha.SetCaptchaUrl("http://cap.dingxiang-inc.com/api/tokenVerify");
//特殊情况需要额外指定服务器,可以在这个指定，默认情况下不需要设置
//captcha.SetTimeOut(2000);
//设置超时时间，单位毫秒，默认2秒
DxCsharpSDK.CaptchaResponse response = captcha.VerifyToken("token");
Response.Write(response.captchaStatus);
//确保验证状态是SERVER_SUCCESS，SDK中有容错机制，在网络出现异常的情况会返回通过
if (response.result) {
    /**token验证通过，继续其他流程**/
} else {
    /**token验证失败，业务系统可以直接阻断该次请求或者继续弹验证码**/
}
```

4.4 SDK-Python版

- **Python版本SDK下载： 点击下载**

```
from CaptchaClient import CaptchaClient

class TokenDemo:
    def __init__(self):
        pass

    APP_ID = '12610axxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
    APP_SECRET = 'a3e56cxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'

if __name__ == '__main__':
    captchaClient = CaptchaClient(APP_ID, APP_SECRET)
    captchaClient.setTimeOut(2)
    # 设置超时时间，默认2秒
    # captchaClient.setCaptchaUrl("http://cap.dingxiang-inc.com/api/tokenVerify")
    # 特殊情况可以额外指定服务器，默认情况下不需要设置
    response = captchaClient.checkToken("token")
    print response['serverStatus']
    #确保验证状态是SERVER_SUCCESS，SDK中有容错机制，在网络出现异常的情况会返回通过
    print response['result']
    if response['result']:
        # token验证通过，继续其他流程
        pass
    else:
        # token验证失败，业务系统可以直接阻断该次请求或者继续弹验证码
        pass
```

4.5 SDK-NODEJS版

```
npm install dx-captcha-sdk --save
```

```
const CaptchaSDK = require('dx-captcha-sdk')

const sdk = new CaptchaSDK('您的appId', '您的appSecret')

sdk.verifyToken('验证码token').then(() => {
  console.log('验证成功')
}).catch(err => {
  console.log('验证失败')
})
```

设置超时时间

```
sdk.verifyToken('验证码token', 5 * 1000)
```

设置验证地址

```
sdk.setCaptchaUrl('验证地址')
```

五、无感验证前端接入

- 访问顶象官网，注册账号后登录控制台，访问“无感验证”模块，申请开通后系统会分配一个唯一的AppId、AppSecret。
- Web接入，兼容IE8+，Chrome，Firefox，360浏览器，QQ浏览器等主流浏览器，接入方法请见Web接入章节。
- Android接入，兼容Android4.0以上，接入方法及SDK下载地址请见Android接入章节。
- iOS接入，兼容iOS8.0以上，接入方法及SDK下载地址请见iOS接入章节。

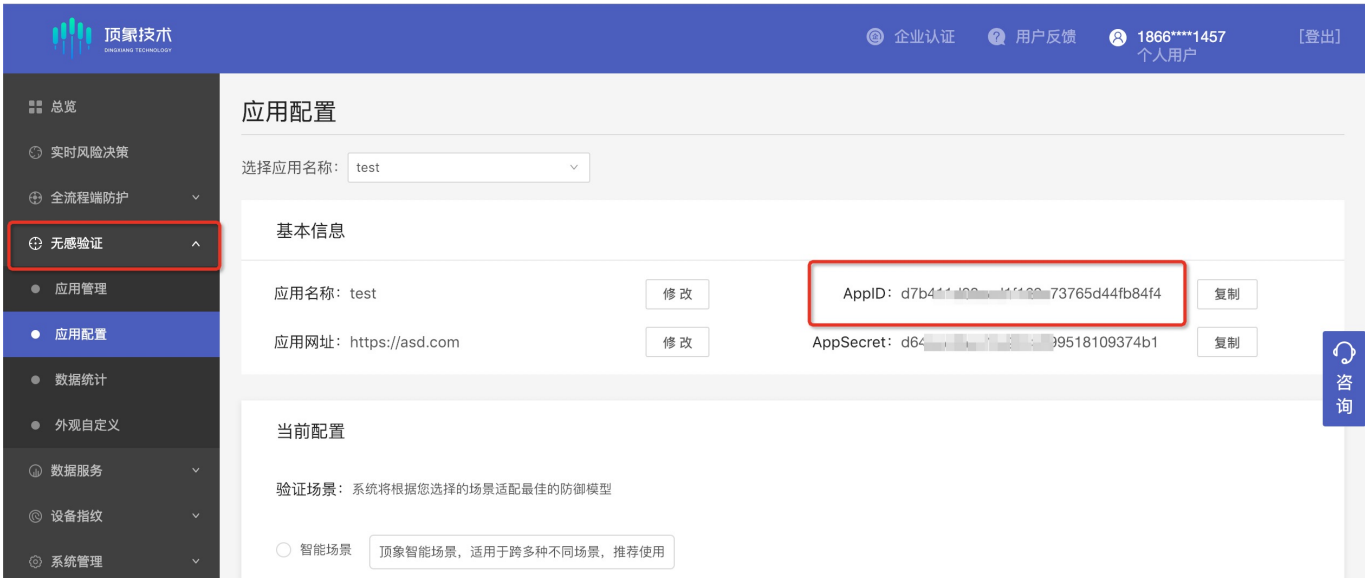
5.1 Web接入

环境要求

兼容IE8+，Chrome，Firefox，360浏览器，QQ浏览器等主流浏览器。

获取appId

请先进入顶象控制台（或点击右上角登陆按钮）中的“应用管理”或“应用配置”模块，并下图指引位置找到appId。



引入 JS

```
<script src="https://cdn.dingxiang-inc.com/ctu-group/captcha-ui/index.js">
</script>
```

注意：由于验证码 JS 会不定期升级更新，请直接引用顶象 CDN 上的资源，以便及时获得最新安全防护。不要将 JS 文件下载到自己服务器使用。

初始化

假设页面上有一个 `<div id="c1"></div>`，则可以像下面这样初始化验证码：

```
var myCaptcha = _dx.Captcha(document.getElementById('c1'), {
    appId: 'appId', //appId，在控制台中“应用管理”或“应用配置”模块获取
    success: function (token) {
      // console.log('token:', token)
    }
})
```

初始化完成后，验证码组件会被插入到 `<div id="c1"></div>` 中。

外观和尺寸

滑动验证码一共有四种样式（`style`），分别为：

- **embed** 嵌入式（默认），这种样式下宽度默认为 `300px`，可通过初始化时的 `width` 参数调节，高度为 `200px`，高度不可调节
- **inline** 内联式，这种样式占用面积较小，宽度默认为 `300px`，可通过初始化时的 `width` 参数调节，高度为 `40px`，高度不可调节
- **popup** 弹出式，这种样式验证码默认不可见，调用 `.show()` 方法后将以浮层的形式展现，宽度为 `300px`，高度为 `200px`
- **oneclick** 触发式，这种样式占用面积较小，宽度默认为 `300px`，可通过初始化时的 `width` 参数调节，高度为 `40px`，高度不可调节

更多细节，可见 参数 章节。

5.2 Android接入

（v1.5.x）

环境要求

条目	说明
开发目标	Android 4.0+
开发环境	Android Studio 3.0.1 或者 Eclipse + ADT
CPU架构	ARM 或者 x86
SDK三方依赖	无

法规要求

根据《工业和信息化部 337号令》的规定，重点对以下四个方面开展规范整治工作。
（一）违规收集用户个人信息方面
（二）违规使用用户个人信息方面
（三）不合理索取用户权限方面
（四）为用户账号注销设置障碍方面
法规地址：http://miit.gov.cn/n1146295/n1652858/n1652930/n3757020/c7506353/content.html

其中SDK涉及到第一条：收集个人信息（包括唯一设备识别码、网络设备硬件地址等信息）。
法规规定，所采集的数据项目需在隐私政策中明确声明，在客户不同意隐私政策的情况下，不允许进行采集。且申请授权需与场景相关，请根据实际情况做出合理调整。

解决方式

需要客户在集成了SDK的app中增加《隐私政策声明》，来规避风险。并在客户同意隐私政策后，进行sdk的调用。

隐私政策（拟，可根据实际情况进行修改）

为了识别设备/账号异常状态，我们将会接受并记录您所使用的设备相关信息（包括设备型号、操作系统、设备设置、唯一设备标识符（IMEI码）、网络设备硬件地址（MAC）、设备环境等软硬件特征信息，设备所在位置相关信息（包括您授权的GPS位置以及WLAN接入点、蓝牙和基站等信息）

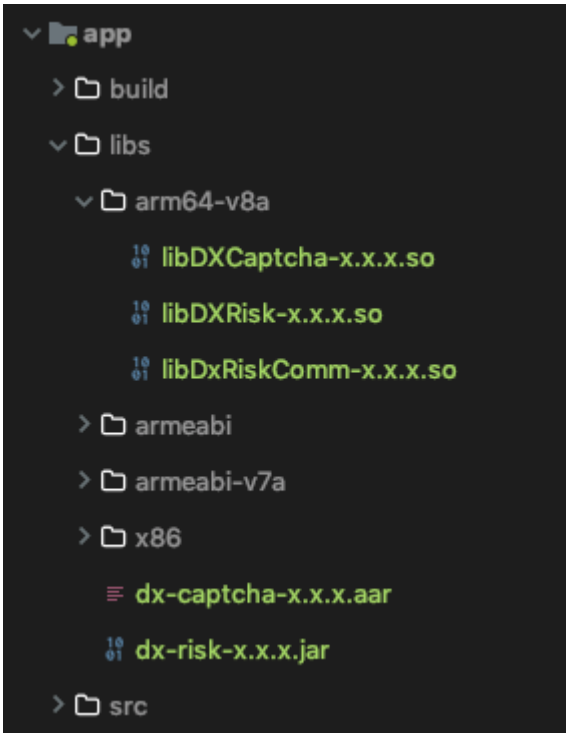
集成SDK

导入依赖

- 无感验证需要同时集成dx-captcha SDK和 dx-risk SDK

Android Studio 集成

- 把aar, jar, so文件集成到项目对应的libs下（x.x.x表示为sdk版本号）
- 把assets里的文件集成到项目的assets里(必须集成，否则会影响运行)
- 点击下载Demo(仅做代码配置演示使用，其中appId请在顶象后台申请，SDK需要替换为链接中下载的SDK)
- 点击下载React Native 接入文档
- 点击下载React Native Demo(仅做代码配置演示使用，其中appId请在顶象后台申请，SDK需要替换为链接中下载的SDK)



- 在该Module的build.gradle中如下配置:

```
android {

  sourceSets {
    main {
      jniLibs.srcDirs = ['libs']
    }
  }

  packagingOptions {
    doNotStrip "**/libDX*.so"
  }
}

repositories{
    flatDir{
        dirs 'libs'
    }
}
```

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation (name:'dx-captcha-x.x.x', ext:'aar')
}
```

添加SDK所需权限

```
<!-- 必选-默认申请 -->
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>

<!-- 可选-6.0或以上需动态申请 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

如果有使用Proguard混淆，请添加如下配置，保证SDK的相关代码不被混淆:

```
-keep class com.dx.mobile.captcha.**{*;}
-keep class org.json.**{*;}

-dontwarn com.dx.mobile.**
-dontwarn *.com.dx.mobile.**
-dontwarn *.com.mobile.strenc.**
-keep class com.dx.mobile.risk.**{*;}
-keep class com.security.inner.**{*;}
-keep class *.com.dx.mobile.**{*;}
-keep class *.com.mobile.strenc.**{*;}
}
```

初始化

- xml方式接入无感验证组件

```
<com.dx.mobile.captcha.DXCaptchaView
    android:id="@+id/dxCaptcha"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
```

- 初始化，需传appId

```
DXCaptchaView dxCaptcha = (DXCaptchaView) findViewById(R.id.dxVCodeView);
dxCaptcha.init(appId);
```

- 自定义配置（可选）

```
Map<String, Object> config = new HashMap<String, Object>();

// 二级属性
Map<String, Object> customStyle = new HashMap<String, Object>();
customStyle.put("bgColor", "#FFFFFF");

// 二级属性
Map<String, Object> customLanguage = new HashMap<String, Object>();
customLanguage.put("init_inform", "shwo_拖动一下");

// 一级属性
config.put("customStyle", customStyle);
config.put("customLanguage", customLanguage);
config.put("language", "en");

// 是否开启无感验证，默认为开启状态
config.put("cacheStorage", true);

// 私有化配置参数，这部分参数通常而言不需要设置，除非你确切知道它们的含义。
config.put("apiServer", "http://ip:port");                  // 私有化必填，私有化服务
器地址
config.put("ua_js", "http://ip:port/xxxxxx/greenseer.js"); // 私有化选填，自定义ua
脚本地址
config.put("captchaJS", "http://ip:port/xxxxxx/index.js"); // 私有化选填，自定义无感
验证js地址
config.put("keyURL", "http://ip:port/udid/m1");             // 私有化选填，自定义设备
指纹地址
config.put("keyBackup", true);                              // 私有化选填，设备指纹数
据线上备份

dxCaptcha.initConfig(config);
```

- 开始加载"无感验证"组件

```
dxCaptcha.startToLoad(new DXCaptchaListener() {

    @Override
    public void handleEvent(View view, DXCaptchaEvent event, Map<String, String>
args) {

        switch(event) {
```

```
            case DXCAPTCHA_SUCCESS: // 验证成功的回调
                String token = args.get("token"); // 成功时会传递token参数
                Log.i(TAG, "Verify Success. token: " + token);
                break;
            case DXCAPTCHA_FAIL:      // 验证失败回调
                Log.i(TAG, "Verify Failed.");
                break;
            default:
                break;
        }
    }
});
```

- 使用结束，调用destory方法释放资源

```
dxCaptcha.destory();
```

参数说明

初始化有若干参数，详情可查阅 参数

其中在 Android SDK 中

- style 固定为 embed ，若指定为其他值会被忽略
- width 参数会被忽略，实际值会根据 frame 进行自适应调整
- success 和 fail 回调参数会被忽略，请在 delegate 中处理回调
- constID_js 、 constIDServer 和 constID_options 配置不生效

事件说明

回调的事件类型

```
DXCAPTCHA_RENDER
DXCAPTCHA_BEFORE_RENDER
DXCAPTCHA_AFTER_RENDER

DXCAPTCHA_READY
DXCAPTCHA_BEFORE_READY
DXCAPTCHA_AFTER_READY

DXCAPTCHA_LOADFAIL
DXCAPTCHA_BEFORE_LOADFAIL
DXCAPTCHA_AFTER_LOADFAIL

DXCAPTCHA_DRAGSTART
DXCAPTCHA_BEFORE_DRAGSTART
DXCAPTCHA_AFTER_DRAGSTART
```

```
DXCAPTCHA_DRAGEND
DXCAPTCHA_BEFORE_DRAGEND
DXCAPTCHA_AFTER_DRAGEND

DXCAPTCHA_VERIFY
DXCAPTCHA_BEFORE_VERIFY
DXCAPTCHA_AFTER_VERIFY

DXCAPTCHA_VERIFYDONE
DXCAPTCHA_BEFORE_VERIFYDONE
DXCAPTCHA_AFTER_VERIFYDONE

DXCAPTCHA_VERIFYSUCCESS
DXCAPTCHA_BEFORE_VERIFYSUCCESS
DXCAPTCHA_AFTER_VERIFYSUCCESS

DXCAPTCHA_VERIFYFAIL
DXCAPTCHA_BEFORE_VERIFYFAIL
DXCAPTCHA_AFTER_VERIFYFAIL

DXCAPTCHA_PASSBYSERVER
DXCAPTCHA_BEFORE_PASSBYSERVER
DXCAPTCHA_AFTER_PASSBYSERVER

DXCAPTCHA_SUCCESS
DXCAPTCHA_FAIL
```

注:

- `Android SDK` 中不触发 `dragging` 事件
- `Android SDK` 中不触发 `show/hide` 系列的事件

5.3 iOS 接入

(v1.5.x)

Apple Store上架请特别注意(集成SDK的第4节)

环境要求

条目	说明
兼容平台	iOS 8.0+
开发环境	XCode 4.0 +
CPU架构	armv7, arm64, i386, x86_64

条目	说明
SDK依赖	libz, libresolv, libc++ , SystemConfiguration.framework , CoreLocation.framework , CoreTelephony.framework

集成SDK

1、集成无感验证，需要同时集成风控SDK5.0.0或以上版本（如项目本身已接入顶象风控且确认版本无误则可直接跳过此步），下载顶象风控 dx-risk SDK ，解压得以下几个文件

- dx-risk-iOS-x.x.x-xxxxxxx目录 DXRisk sdk
  - DXRisk.framework 不带idfa获取逻辑的Dynamic Library Framework
  - DXRiskWithIDFA.framework 带idfa获取逻辑的Dynamic Library Framework
  - DXRiskStatic.framework 不带idfa获取逻辑的Static Library Framework
  - DXRiskStaticWithIDFA.framework 带idfa获取逻辑的Static Library Framework

若在项目中添加`DXRisk.framework`或者`DXRiskWithIDFA.framework`其中之一，选择`Target -> General`，在`Frameworks,Libraries,and Embedded Content`中，将`DXRisk.framework`或者`DXRiskWithIDFA.framework` 对应的 Embed 切换到Embed & Sign。如下图：

若在项目中添加`DXRiskStatic.framework`或者`DXRiskStaticWithIDFA.framework`其中之一，需要在Build Settings -> Other Linker Flags 设置 -ObjC 如下图：

![4A23453213E696EC12C2AE2A1070EA0C.jpg](https://cdn.dingxiang-inc.com/images/293/29386b91-2918-4b02-8f30-da3ac1547197.jpg)

2、下载顶象无感验证 dx-captcha SDK ，解压得到 `DXCaptchaSDK` 文件夹，内含以下文件

- `DXCaptchaSDK` CaptchaFramework
  - `DXCaptcha.bundle` 无感验证资源包
  - `DingxiangCaptchaSDK.framework` Dynamic Library Framework
  - `DingxiangCaptchaSDKStatic.framework` Static Library Framework
- `DXCaptchaDemo` 集成demo

3、将 `DingxiangCaptchaSDK.framework` 或者 `DingxiangCaptchaSDKStatic.framework` 文件夹以及 `DXCaptcha.bundle` 资源文件一起拖入工程根目录，若在项目中添加`DingxiangCaptchaSDK.framework`，则需要选择`Target -> General`，在`Frameworks,Libraries,and Embedded Content`中，将 `DingxiangCaptchaSDK.framework` 对应的 Embed 切换到Embed & Sign，若在项目中添加 `DingxiangCaptchaSDKStatic.framework`，需要在Build Settings -> Other Linker Flags 设置 -ObjC 。

- 点击下载React Native 接入文档

- 点击下载React Native demo(仅做代码配置演示使用，其中appId请在顶象后台申请，SDK需要替换为链接中下载的SDK)

4、配置打包脚本

以下的操作仅限导入DingxiangCaptchaSDK.framework动态库

此步骤主要是解决上传Store架构不符合的问题，如项目中已配置过Carthage或有其他相关的打包Framework调整脚本，可略过此步自行调整 选择`Target -> Build Phases`，点击`+`按钮，选择`New Run Script Phase`, 添加如下脚本：

```
APP_PATH="${TARGET_BUILD_DIR}/${WRAPPER_NAME}"

# This script loops through the frameworks embedded in the application and
# removes unused architectures.
find "$APP_PATH" -name '*.framework' -type d | while read -r FRAMEWORK
do
FRAMEWORK_EXECUTABLE_NAME=$(defaults read "$FRAMEWORK/Info.plist"
CFBundleExecutable)
FRAMEWORK_EXECUTABLE_PATH="$FRAMEWORK/$FRAMEWORK_EXECUTABLE_NAME"
echo "Executable is $FRAMEWORK_EXECUTABLE_PATH"

EXTRACTED_ARCHS=()

for ARCH in $ARCHS
do
echo "Extracting $ARCH from $FRAMEWORK_EXECUTABLE_NAME"
lipo -extract "$ARCH" "$FRAMEWORK_EXECUTABLE_PATH" -o
"$FRAMEWORK_EXECUTABLE_PATH-$ARCH"
EXTRACTED_ARCHS+=("$FRAMEWORK_EXECUTABLE_PATH-$ARCH")
done

echo "Merging extracted architectures: ${ARCHS}"
lipo -o "$FRAMEWORK_EXECUTABLE_PATH-merged" -create "${EXTRACTED_ARCHS[@]}"
rm "${EXTRACTED_ARCHS[@]}"

echo "Replacing original executable with thinned version"
rm "$FRAMEWORK_EXECUTABLE_PATH"
mv "$FRAMEWORK_EXECUTABLE_PATH-merged" "$FRAMEWORK_EXECUTABLE_PATH"

done
```

初始化

假设在 `ViewController` 中添加无感验证，首先引入头文件

```
#import <DingxiangCaptchaSDK/DXCaptchaView.h>
#import <DingxiangCaptchaSDK/DXCaptchaDelegate.h>
```

然后实现 DXCaptchaDelegate 协议中的 captchaView:didReceiveEvent:arg: 方法，以接收验证结果回调

```
@interface ViewController () <DXCaptchaDelegate>

@end

@implementation ViewController

- (void) captchaView:(DXCaptchaView *)view didReceiveEvent:
(DXCaptchaEventType)eventType arg:(NSDictionary *)dict {
    switch(eventType) {
        case DXCaptchaEventSuccess: { // 验证成功的回调
            NSString *token = dict[@"token"]; // 成功时会传入token参数
            [[[UIAlertView alloc]initWithTitle:@"Verify Success" message:token
delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil] show];
            break;
        }
        case DXCaptchaEventFail: // 验证失败的回调
            [[[UIAlertView alloc]initWithTitle:@"Verify Failed"
message:@"DXCaptcha Verify Failed!" delegate:nil cancelButtonTitle:@"OK"
otherButtonTitles:nil] show];
            break;
        default:
            break;
    }
}

@end
```

最后创建无感验证组件，需要传入 appId， delegate(验证回调的代理)， frame(位置及尺寸)

```
CGRect frame = CGRectMake(self.view.center.x - 150, self.view.center.y - 100, 300,
200);
DXCaptchaView *captchaView = [[DXCaptchaView alloc] initWithAppId:@"your appId"
delegate:self frame:frame];
[self.view addSubview:captchaView];
```

如果除 appId 以外还需要配置其他的初始化参数，可调用 initWithConfig:delegate:frame 接口，如

```
NSDictionary *config = @{@"appId": @"your appId", @"language": @"en"};
NSDictionary *customLanguageConfig = @{@"pass_by_server": @"验证成功"}; //可设置参
数参考文档
[config setObject:customLanguageConfig forKey:@"customLanguage"];
DXCaptchaView *captchaView = [[DXCaptchaView alloc] initWithConfig:config
delegate:self frame:frame];
```

私有化配置参数示例代码,apiServer为必填项，ua_js、captchaJS、keyURL、keyBackup如需手动设置请咨询私有化部属的同学

```
    NSMutableDictionary *config = [NSMutableDictionary dictionary];
    //私有化参数配置
    [config setObject:@"your appId" forKey:@"appId"];
    [config setObject:@"your apiServer" forKey:@"apiServer"];
    [config setObject:@"your ua_js" forKey:@"ua_js"];
    [config setObject:@"your captchaJS" forKey:@"captchaJS"];
    [config setObject:@NO forKey:@"isSaaS"];
    //设备指纹私有化服务器地址
    [config setObject:@"your riskServerAddress" forKey:@"keyURL"];
    //设备指纹私有化数据备份
    [config setObject:@YES forKey:@"keyBackup"];

    DXCaptchaView *captchaView = [[DXCaptchaView alloc] initWithConfig:config
delegate:self frame:frame];
```

参数说明

初始化有若干参数，详情可查阅 参数

其中在 iOS SDK 中

- style 固定为 embed ，若指定为其他值会被忽略
- width 参数会被忽略，实际值会根据 frame 进行自适应调整，建议尺寸为300x200
- success 和 fail 回调参数会被忽略，请在 delegate 中处理回调
- constID_js 、 constIDServer 和 constID_options 配置不生效
- 增加 userId 参数，用于传递给 ConstID 模块作用户标识
- 增加 captchaJS 参数，用于设置私有化验证码脚本
- 增加 keyBackup 参数，用于设置是否备份设备指纹私有化
- 增加 keyURL 参数，用于设置设备指纹私有化服务器地址

事件说明

回调的事件类型，请查阅 事件 及 DXCaptchaDelegate.h 头文件

注：iOS SDK 中不触发 show/hide 系列的事件。如SDK无效且运行过程中出现NOT FOUND DXRISK-SDK日志，请检查是否遗漏顶象风控SDK的集成

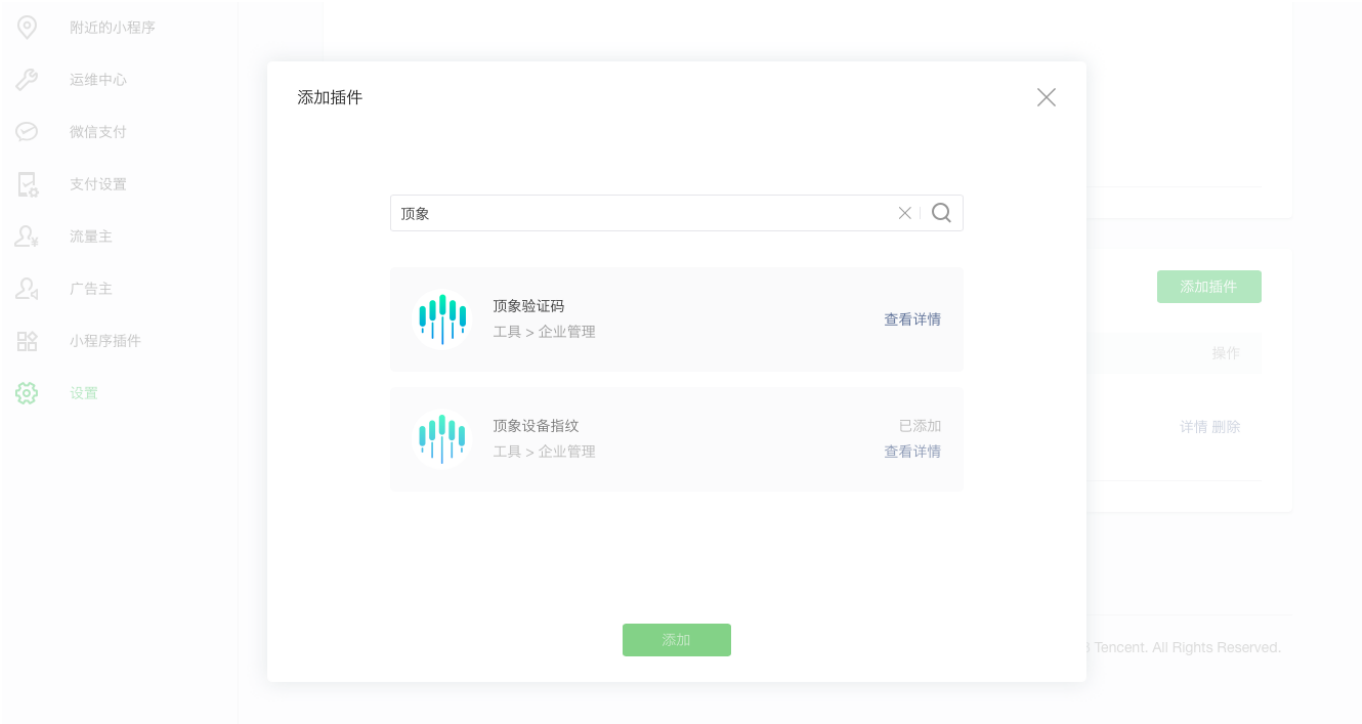
5.4 小程序接入

5.4.1、插件式接入

1）添加插件

用管理员身份登录微信公众平台，请使用需要接入小程序的相关账号（微信公众平台采用不同账号区分，公众号的后台和小程序的后台分别为不同账号），依次点击：设置-第三方服务-添加插件，然后输入关键字“顶象”并

搜索，如下图所示：



2）获取密钥

未注册用户可在顶象官网进行账号注册，创建应用获取应用密钥AppID和AppSecret。

3）集成插件

声明插件：在app.json中声明插件

```
{
  "plugins": {
    "captcha": {
      "version": "1.2.4",
      "provider": "wxbf8483dfc5ac6817"
    }
  }
}
```

在页面.json中引入自定义组件

```
{
  "usingComponents": {
    "basic": "plugin://captcha/basic",
    "oneclick": "plugin://captcha/oneclick",
  }
}
```

4）使用插件

一、点击式

在页面.wxml中使用插件

```
<oneclick bindsuccess='captchaSuccess' bindhide='captchaHide'
oneclickReload='{{captchaReload}}' captchaShow='{{captchaShow}}'
options='{{options}}'/>
<button bindtap='login'>登陆</button>
```

在页面.js中监听事件

```
Page({
  data: {
    options: {
        appId: '这里填写你在顶象官网申请到的appId',
        style: 'oneclick'
    },
    captchaShow: false,
    captchaReload: false
  },

  login: function () {
    // captchaReload用来重置验证码
    this.setData({
      captchaReload: true
    })
  },

  // 验证码成功回调
  captchaSuccess: function (token) {
    console.log('token:', token)
  },

  // 验证码关闭回调
  captchaHide: function () {
    console.log('captcha_hide')
    this.setData({
         captchaShow: false
    })
  }
})
```

二、弹出式 在页面.wxml中使用插件

```
<basic bindsuccess='captchaSuccess' bindhide='captchaHide'
captchaReload='{{captchaReload}}' captchaShow='{{captchaShow}}'
options='{{options}}'/>
<button bindtap='login'>登陆</button>
```

在页面.js中监听事件

```
Page({
  data: {
    options: {
        appId: '这里填写你在顶象官网申请到的appId',
        style: 'popup'
    },
    captchaShow: false,
    captchaReload: false
  },

  login: function () {
    // captchaShow 用于弹出验证码
    // captchaReload 用于重置验证码
    this.setData({
     captchaReload: true
     captchaShow: true
    })
  },

  // 验证码成功回调
  captchaSuccess: function (token) {
    // 验证成功返回的 token
    console.log('token:', token)
    this.setData({
     captchaShow: false
    })
  },

  // 验证码关闭回调
  captchaHide: function () {
   console.log('captcha_hide')
   this.setData({
     captchaShow: false
   })
  }
})
```

5.4.2、跳转式接入

1）关联小程序

用管理员身份登录微信公众平台，请使用需要公众号的相关账号（微信公众平台采用不同账号区分，公众号的后台和小程序的后台分别为不同账号），依次点击：小程序-小程序管理-添加-关联小程序，然后输入小程序AppID“wx82db4a59175bdfdf”并搜索，如下图所示：

2）获取密钥

未注册用户可在顶象官网进行账号注册，创建应用获取应用密钥AppID和AppSecret。

3）小程序集成

1. 通过navigator组件跳转至顶象验证码。

```
.wxml
<navigator target="miniProgram"
```

```
  app-id="wx82db4a59175bdfdf"
  path="/pages/captcha/captcha"
  extra-data="{{options}}">
    <button>登陆</button>
</navigator>
```

```
.js
Page({
    data: {
        options: {
        appId: '这里填写在顶象官网申请到的appId',
        customStyle: {
            panelBg: '',
            captchaBgColor: ''
        }
        }
    }
})
```

参数说明：

参数	值	说明
appId	String	这里填写在顶象官网申请到的appId，注意必须填写正确，否则会报：功能无法使用
customStyle	Object	自定义样式对象,panelBg为整个页面的background属性，captchaBgColor为验证码的background-color属性

2. 在顶象验证码中验证

3. 验证成功后，验证结果会在调用方app.js中顶onShow生命周期方法中取到。

```
if (options.scene === 1038 && options.referrerInfo.appId === 'wx82db4a59175bdfdf')
{
    const result = options.referrerInfo.extraData;
    if (result) {
        console.log('返回结果:', result)
    } else {
        // 用户点击右上角关闭了验证码
    }
}
```

验证结果说明：

参数	值	说明
success	Boolean	验证成功失败
token	String	验证成功才有：token
msg	String	验证失败或参数不合法才有：错误信息

5.5 参数

常规参数

初始化时有若干参数，其中 `appId` 是必填的。

参数名	必填	类型	说明
appId	是	String	当前应用的唯一标识
type	否	String	类型，目前只有一个选项 `basic`，默认值为 `basic`。
style	否	String	样式，可选项有 `embed`（默认）、`inline`、`popup`、`oneclick` 四种，具体效果可参见线上demo。
inlineFloatPosition	否	String	浮动层位置，仅当 `style` 的值为 `inline` 时有效，可选项有 `up`（默认）、`down` 两种，具体效果可参见线上demo。
oneClickFloatPosition	否	string	浮动层位置，仅当 `style` 的值为 `oneclick` 时有效，可选项有 `up`，当`style`值为`oneclick`且不传此参数，默认为弹出形式。
width	否	Number	控件（滑动条）宽度，单位为像素，默认为 `300`。
language	否	String	语言，可选项有 `cn`（默认，中文）、`en`（英文）。
success	否	Function	验证成功时的回调函数，会传入一个参数，值为本次验证的 `token:constId`，接入方需在后续的验证中传回此项值。注：此回调函数包含无感验证成功及滑动验证成功。
fail	否	Function	验证失败时的回调函数，会传入一个参数，值为出错信息。

个性化参数

这部分参数可以控制一些自定义信息。

参数名	必填	类型	说明
customStyle	否	Object	自定义样式，详情参见自定义样式说明。
customLanguage	否	Object	自定义语言，详情参见自定义语言说明。

高级参数

这部分参数通常而言不需要设置，除非你确切知道它们的含义。

参数名	必填	类型	说明
ua_js	否	String	自定义 ua 脚本地址，格式形如 `https://cdn.dingxiang-inc.com/ctu-group/ctu-greenseer/greenseer.js`。
constID_js	否	String	自定义 constID 脚本地址，格式形如 `https://cdn.dingxiang-inc.com/ctu-group/constid-js/index.js`。
constIDServer	否	String	自定义 constID 服务地址，格式形如 `http://constid.dingxiang-inc.com/udid/c.png`。
apiServer	否	String	自定义 API 服务地址，格式形如 `http://123.207.220.181:9090` 或 `https://api.dingxiang-inc.com`。
protocol	否	String	请求协议，可选项有 `http:`、`https:`，默认为 `https:`。
timeout	否	Number	超时时间，单位为毫秒，默认值为 6000。注意，只有部分请求可设置超时时间。
constID_options	否	Object	constID 初始化配置参数，参见 ConstID 文档，如此项中的参数与外部参数冲突（比如其中也定义了 `appId` 且与外部的 `appId` 不同，则以此处的为准。

5.6 基础类型

初始化时，如果省略 `type` 参数，或传入的值为 `basic`，则启用基础类型验证码。

例：

```
var myCaptcha = _dx.Captcha(document.getElementById('c1'), {
    appId: 'appId',
    type: 'basic', // <-- 指定为"基础类型"，此参数可省略
    style: 'embed', // 可省略
    width: 300 // 可省略
})
```

参数说明亦可参考参数。

5.7 方法

验证码实例具备以下方法：

- `reload()` 重置当前验证码（**注意！**，请不要在`success`回调里调用`reload`，因为在开启无感验证的情况下会重复调用`success`回调。）

  示例：

  ```
  myCaptcha.reload()
  ```

- `show()` 显示当前验证码。

  `style` 为 `popup` 的验证码默认是隐藏的，需要接入方根据页面逻辑调用 `show()` 方法将其显示。

  示例：

  ```
  myCaptcha.show()
  ```

- `hide()` 隐藏当前验证码

  示例：

  ```
  myCaptcha.hide()
  ```

5.8 事件

验证码可以使用以下方式监听事件：

```
myCaptcha.on('ready', function () {
  // console.log('captcha is ready!')
})

myCaptcha.on('verifySuccess', function (security_code) {
  // 说明: security_code = token + ':' + constID
  // console.log('security_code is: ' + security_code)
})

myCaptcha.on('hide', function () {
  // console.log('验证码控件被隐藏了。')
})
```

基础类型的验证码支持以下事件：

初始化阶段

验证码在初始化阶段按顺序触发以下事件：

事件名	说明
render	渲染事件，验证码开始渲染时触发
passByServer	【无感验证】通过，服务端判定本次验证可直接通过，无需用户交互。如果此事件触发，则验证码直接显示为验证通过状态，将没有后面的用户交互阶段。此事件带一个参数 `token`。

事件名	说明
ready	验证码准备就绪，可以接受用户输入时触发。注意：【无感验证】通过则不会触发此事件，直接进入passByServer事件

如果加载失败，则会按顺序触发以下事件：

事件名	说明
render	渲染事件，验证码开始渲染时触发
loadFail	加载失败时触发

用户交互阶段

事件名	说明
dragStart	用户开始拖动滑块
dragging	用户拖动滑块过程中多次触发
dragEnd	用户释放滑块，结束拖动
verify	向校验接口提交数据进行校验
verifyDone	校验接口已返回数据
verifySuccess	校验接口已返回数据，且结果为成功，此事件带一个参数 `token`
verifyFail	校验接口已返回数据，且结果为失败

其他事件

事件名	说明
show	验证码控件显示时触发
hide	验证码控件隐藏时触发

其他

说明：所有事件，都有对应的 `before_*` 事件和 `after_*` 事件，分别在该事件之前和之后触发。

例如 `load` 事件对应三个事件：`before_load`、`load`、`after_load`。

5.9 自定义样式

如果你熟悉 CSS，你可以直接编写自己的 CSS 来覆盖验证码组件的样式。或者也可以在验证码初始化时传入一个 `customStyle` 参数，对某些样式进行自定义。

示例

```
var appId = '【你的 appID】'
var demo_1 = _dx.Captcha(document.getElementById('demo'), {
  appId: appId,

  // 下面开始自定义样式
  customStyle: {
    bgColor: '#cccc00' // <-- 自定义控件背景色
  },

  success: function (token) {
    window.console && console.log('success, token:', token)
  }
})
```

参数详情

`customStyle` 目前支持以下参数，所有参数都是**可选**的：

参数	类型	说明
bgColor	String	控件背景色，CSS 颜色格式，例如 `#ff0000`、`rgb(255, 0, 0)`

5.10 自定义语言

顶象验证码自带中文、英文两种语言，如果你需要自定义语言，或者只是调整现有语言的某句文案，则可以在初始化时传入 `customLanguage` 参数，对语言进行自定义。

示例

```
var appId = '【你的 appID】'
var demo_1 = _dx.Captcha(document.getElementById('demo'), {
  appId: appId,

  // 下面开始自定义语言
  customLanguage: {
    init_inform: '拖动一下', // <-- 初始化时的提示文案
    slide_inform: '向右向右' // <-- 滑块中的提示文案
  },

  success: function (token) {
    window.console && console.log('success, token:', token)
  }
})
```

参数详情

`customLanguage` 目前包含以下语句：

注意：Android端`load_fail`内容格式为：`"加载失败，<a href=\\\"#\\\">请重试</a>！"`

语句名	默认值（参考值）
init_inform	拖动下方的滑块
load_fail	加载失败，请重试！
loading	加载中...
pass_by_server	顶象无感验证成功
slide_inform	请向右拖动滑块完成验证
smart_checking	智能检测中
verify_success	验证成功
verify_fail	验证失败
verifying	验证中.....

六、常见问题

1.业务系统怎样和滑动验证码结合?

> - 针对需要保护的业务接口，在页面上嵌入滑动验证码，用户滑动成功后会得到一个安全token，业务接口需要带着token到后台进行token安全验证，验证通过以后再继续业务流程。
> - 以登录为例，未接入验证码前的业务接口：
> `接口：http://domain/login 参数：用户名，密码`

> - 现在在登录页面接入验证码，用户滑动以后，会得到一个安全token
> 接入后的业务接口： `接口：http://domain/login 参数：用户名，密码，安全token`

2. 顶象无感验证服务是否收费?

> - 目前顶象验证码服务可免费试用，如果所需验证码调用量达到一定的数量，或需定制自己Logo、背景图片，我们将按照不同量的标准收费。具体收费请咨询顶象官网在线客服。

3. 相对于同类产品，顶象的优势在哪?

> - 依托顶象强大的技术研发和十余年的安全防控经验，顶象安全大脑可以在大数据海洋中精确定位各类异常操作，实时决策，并从海量真人操作中学习，不断进化识别能力;

> - 目前顶象的客户已遍布金融、通讯、IoT、电商、互联网等行业;

> - 顶象无感验证还具备快速接入、极致体验、丰富报表等特点。

4.无感验证服务支持哪些浏览器版本?

> - IE8+，Chrome，Firefox，360浏览器，QQ浏览器等主流浏览器。

5.在使用或开发过程中遇到问题，是否有技术支持?

> - 是的，本着客户第一的原则，有问题您可以随时咨询顶象官网的在线客服，我们将竭诚为您解答。

6.智能无感模式和强校验模式有什么区别?

> - 智能无感模式：系统会根据采集的环境设备信息向服务端进行无感模型校验，通常包括设备模型侦测（检测是否为模拟器或者存在恶意信息篡改等）、异常关联侦测（检测是否存在异常的关联，如同设备短时间多IP关联、短时间多次进行验证等），若用户通过无感模型校验则无需进行滑动验证即可验证通过。

> - 强校验模式： 强制用户每次通过需要完成验证方可通过，一般推荐用户在某些需要高校验要求的业务环节选择此模式如注册、获取短信验证码等环节。

> - 关于模式切换：接入客户可根据自身实际需求，在应用配置下自主选择所需的验证模式。

七、FAQ

详情点击：https://www.dingxiang-inc.com/docs/preview/detail/captcha-faq