顶象设备指纹（UNIFYID®）介绍

一、快速入门指南

1. 产品简介

顶象设备指纹通过用户上网设备的硬件、网络、环境等特征信息生成设备的唯一标识，覆盖安卓、iOS、Web/H5、公众号、小程序，可有效侦测模拟器、刷机改机、root 越狱、劫持注入等风险，配合顶象风控系统使用，可有效对抗设备伪造、自动注册、羊毛党等恶意行为。

2. 名词解释

名称	释义
AppId	公钥，长度为 32 位字符串，接入渠道唯一标识。开通服务后可在设备指纹的二级
AppSecret	私钥，长度为 32 位字符串，与公钥对应，开通服务后可在设备指纹的二级菜单“用。
token	设备指纹 SDK 采集上报后返回的标识，token 不是设备指纹，通过 token 可以查
hardId	设备指纹
用户前端	Web 端或集成 SDK 的 Android 端、iOS 端
用户后端	指企业的后台服务器

开发者请注意

快速开发使用 AppId

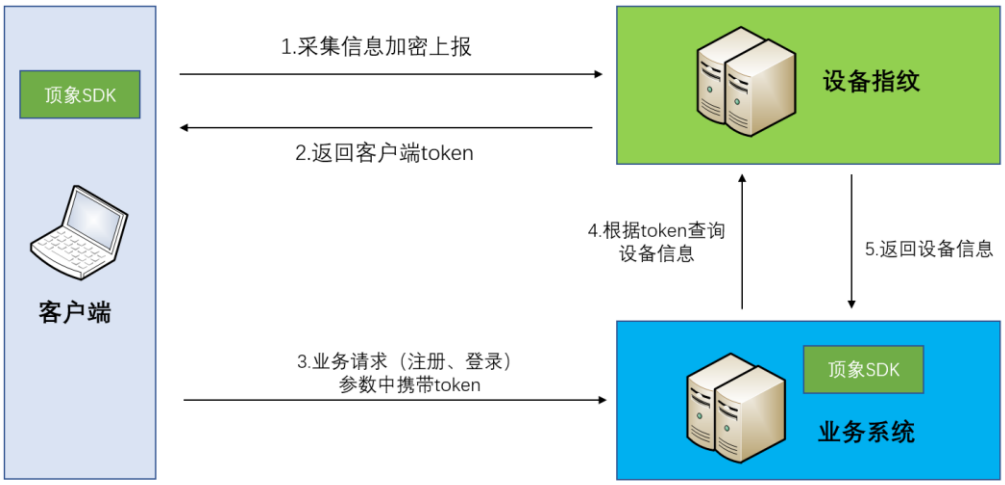
开发测试接口连通可使用：

AppId: d7eb51437ed189ae6a42ef3cccf69cec

❌ 缺点：多人共用，无法进行统计数据查询

3. 交互流程

单独使用顶象设备指纹服务时的调用过程



- **客户端接入**，业务客户端需要集成顶象指纹客户端 SDK，包括安卓，iOS，H5，小程序等；通过顶象客户端 SDK 可以获取到设备指纹 token（注：token 不是设备指纹）。
- **业务接口扩展**，业务客户端在需要设备指纹 token 的时候，可以通过相应的 api 获取到。业务接口需要把前端拿到的指纹 token 一并传入后台。

- **后台接入**，顶象提供后端 SDK 来查询设备详细信息，SDK 涵盖 Java，PHP 等。

  重要：因终端用户的设备网络环境和设备版本等因素，设备指纹采集率并不能一定达到 100%，可能会存在极少部分未能正常采集到的情况。所以在集成指纹服务的时候，请尽量避免对指纹信息强依赖。

4. FAQ 及版本记录

点击查看 FAQ (https://www.dingxiang-inc.com/docs/preview/detail/const-id-faq)

二、前端接入

- **支持 Web 接入**，支持 IE8+、Chrome、Firefox、360 浏览器、QQ 浏览器等主流浏览器及 Android、iOS 上的内嵌 Webview。引用 JS 参见 Web 接入章节
- **支持 Android 接入**，获取 SDK 并接入，参见 Android 接入章节
- **支持 iOS 接入**，获取 SDK 并接入，参见 iOS 接入章节
- **支持微信小程序接入**，接入参见微信小程序接入章节

1 Web 接入

第一步：引入

在页面 HTML 中引入 `const-id.js`，代码形如：

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

因 js 文件会定期更新，为避免 js 失效影响您的使用，请不要将 js 下载到本地服务器上引入

第二步：生成并使用

页面加载后，初始化设备指纹，需要在 JavaScript 中调

用 `_dx.ConstID(options, callback)` 方法获取设备指纹 token，代码形如：

```
var options = {
  appId: '【这里填写 AppID】', // 唯一标识，必填
  server: 'https://constid.dingxiang-inc.com/udid/c1', // constId 服务接口，可选
  userId: '【这里填写 userID】' // 用户标识，可选
};

_dx.ConstID(options, function (err, token) {
  if (err) {
    // console.log('error: ' + err);
    return;
  }
  // console.log('const-id token is ' + token);
});
```

同时也支持 Promise 的用法

```
_dx.ConstID(options).then(function(token) {
  console.log(token)
}).catch(function(err) {
  console.log(err)
})
```

options 字段说明

字段	类型	是否必填	说明
appId	String	是	当前应用的标识

字段	类型	是否必填	说明
server	String	否	constId 服务接口，可选，如不
scene	String	否	场景标识，例如 `login`、`survey`
userId	String	否	业务方的用户唯一标识，例如用
timeout	number	否	超时失败时间，单位为毫秒
cache	boolean	否	默认为 true，表示会缓存采集结

PC 浏览器兼容

浏览器	
IE	
Edge	
Chrome	
Safari	
Firefox	
360	

浏览器	
Sougou	
QQ	

移动端浏览器兼容

浏览器	
Chrome	
UC	
QQ	
Safari	
原生	

2 Android 接入

一、 环境要求

条目	
开发目标	
开发环境	
CPU 架构	
SDK 三方依赖	

二、 集成 SDK

2.1 下载 SDK

点击下载 SDK

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）

2.2 Android Studio 集成

SDK 包集成内容：

- dx-risk-vx.x.x.aar

2.2.1 把 aar 文件放到相应模块的 libs 目录下

Demo 工程结构如下

```
.
```

```
├── app
│   ├── build.gradle
│   ├── libs
│   │   └── dx-risk-vx.x.xxr.xxxxxxxx.aar
│   ├── proguard-rules.pro
│   └── src
│       └── main
├── build.gradle
├── gradle
├── gradle.properties
├── gradlew
├── gradlew.bat
└── settings.gradle
```

2.2.2 在该 Module 的 build.gradle 中如下配置:

```
android {

  packagingOptions {
      doNotStrip "**/lib*Risk*.so"
  }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.aar'])
}
```

2.3 （可选）添加 SDK 所需权限

```
<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.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" />
<uses-permission
android:name="android.permission.ACCESS_BACKGROUND_LOCATION"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
```

2.4 Proguard 混淆配置

```
-dontskipnonpubliclibraryclassmembers
-keepattributes *Annotation*,EnclosingMethod

-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.**{*;}
```

2.5 API 6.0 或以上动态权限申请说明

需要动态申请权限如下:

```
android.permission.WRITE_EXTERNAL_STORAGE
android.permission.READ_EXTERNAL_STORAGE
android.permission.READ_PHONE_STATE
android.permission.ACCESS_COARSE_LOCATION
android.permission.ACCESS_FINE_LOCATION
```

动态申请代码实例(Activity 下):

```
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_demo);

   // API 23 或以上的动态申请权限
   if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
       String[] permissionArray = {
              "android.permission.ACCESS_COARSE_LOCATION",
              "android.permission.ACCESS_FINE_LOCATION",
               "android.permission.ACCESS_BACKGROUND_LOCATION",
              "android.permission.WRITE_EXTERNAL_STORAGE",
              "android.permission.READ_EXTERNAL_STORAGE",
              "android.permission.READ_PHONE_STATE",
       };
       this.requestPermissions(permissionArray, 1);
   }

}
```

```
@Override
public void onRequestPermissionsResult(int requestCode, String[]
permissions, int[] grantResults) {
   super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
   // start getToken
   new Thread(new Runnable() {
       @Override
       public void run() {
           HashMap<String, String> params = new HashMap<String, String>();
           String token = DXRisk.getToken("appid", params);
       }
   }).start();

}
```

三. 接口使用说明

3.1 方法和参数说明

功能描述

采集端的设备指纹信息，上传至风控后台，再由风控后台返回 token。 该 API 为耗时操作，因此必须在非主线程上调用，否则会抛异常。

方法说明

`DXRisk.java` 该类是 DxRisk SDK 的风控组件接口，负责采集本地信息并返回用户前端 token。

隐私采集白名单

自 v6.1.18r 开始，SDKsetup 前，可以调用 `setAllowPrivacyList` 进行隐私采集白名单的设置，默认关闭所有涉及隐私规范红线的采集项

```
public static void setAllowPrivacyList(long privacyFlag)
```

使用示例

```
setAllowPrivacyList(PrivacyFlag.ANDROID_ID); //只采集 ANDROID_ID
//自定义列表模式
```

```
setAllowPrivacyList(PrivacyFlag.X | PrivacyFlag.Y | ...); //表示既采集 X 又采集 Y,X、Y 表示具体某项采集的 FLAG 值，如下表
```

以下为提供的白名单采集项

FLAG	
DEFAULT	默认值，关闭所有隐私采集项
ALL	打开所有隐私采集项
IMEI	加入该 FLAG 则采集手机 IMEI 值
IMSI	加入该 FLAG 则采集手机 IMSI 值
MEID	加入该 FLAG 则采集手机 MEID 值
SERIAL_NUMBER	加入该 FLAG 则采集手机 SERIAL
MAC_ADDRESS	加入该 FLAG 则采集手机 MAC_A
ICCID	加入该 FLAG 则采集手机 ICCID 值
ANDROID_ID	加入该 FLAG 则采集手机 ANDRO
DEVICE_ID	加入该 FLAG 则采集手机 DEVICE
GET_INSTALLED_PACKAGES	加入该 FLAG 则采集手机的应用安

初始化 setup

SDK 使用前必须调用先 setup，setup 主要用于数据/环境初始化，一般在 Application 的 onCreate 下调用：

```
/**
* 初始化参数，环境
* @param context
* @return
*/
public static boolean setup(Context context)
```

PS：下列两种方式获取 token 在网络通畅的情况下没有任何的不同。

常规 Token

```
/**
 * @return token 通常返回长度为 40 的字符串。在网络卡顿或不通的情况下，返回 4-5k 的字符串。
 * @throws DXRiskErrorException 如在主线程调用本 API，或者 appId 为空等等，则会抛出该异常
 */
public static String getToken(String appId, HashMap<String, String> paramsMap) throws DXRiskErrorException
```

精简 Token

获取轻量级 Token 可获取的设备信息信息远少于 getToken()，可能会造成在判断设备是否有风险时出现较大误差，请谨慎使用。

```
/**
 `* @return token 通常返回长度为 40 的字符串。在网络卡顿或不通的情况下，返回 1k 的字符串。
 `* @throws DXRiskErrorException 如在主线程调用本 API，或者 appId 为空等等，则会抛出该异常
 */
public static String getLightToken(String appId, HashMap<String, String> paramsMap) throws DXRiskErrorException
```

3.2 使用示例

3.2.1 初始化 setup

建议 Application.onCreate 下调用

```
@Override
public void onCreate() {
  super.onCreate();
  // 环境初始化
  DXRisk.setup(this);
}
```

3.2.2 获取 token

整个过程由于是耗时操作，必须要在非主线程上执行，否则会 crash

```
new Thread(){

    @Override
    public void run() {
      /* 私有化配置 */
      HashMap<String, String> params = new HashMap<String, String>();
      // 这里请填写对应的服务端 url，注意 SAAS 和私有化的区别
      params.put(DXRisk.KEY_URL, "https://constid.dingxiang-inc.com");
      // 开启线上数据备份
      // params.put(DXRisk.KEY_BACKUP, DXRisk.VALUE_ENABLE_BACKUP);
      // 设置请求 token 超时时长 ms，不设置默认为 500ms
      params.put(DXRisk.KEY_DELAY_MS_TIME, "2000");
      //此配置关闭 SDK 缓存，表示每次调用接口都是强制采集信息
      // params.put("PRIVATE_CLEAR_TOKEN", "clear");
      // 开通服务后可在应用管理菜单中获取；私有化版本请填写私有化 appId
      String appId = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
       // 获取设备指纹 token
      final String token = DXRisk.getToken(appId, params);

      // TODO 把 token 通过 Post 请求，传到用户后端
  }
}.start();
```

3.3 异常说明

在获取 token 过程中，如果因为网络超时或者加解密失败，该接口有可能会返回为 null，同时会输出 tag 为 DXRISK 的错误信息，具体描述如下：

```
DXRISK_REQUEST_NETWORK_ERR          -1001        //网络链接失败
```

```
DXRISK_REQUEST_DECRYPT_ERR          -1002        //数据解密错误
DXRISK_REQUEST_UNCOMPRESS_ERR       -1003        //解压错误
DXRISK_REQUEST_RESPONSE_EMPTY_ERR    -1004        //返回为空
DXRISK_REQUEST_DATA_PARSE_ERR       -1005        //数据解析失败
DXRISK_REQUEST_DIRTY_DATA_ERR       -1006        //脏数据
DXRISK_CONST_ID_EMPTY               -1007        //constid 为空
```

如果出现上述错误信息，请联系顶象技术人员。

3 iOS 接入

一、环境需求

条目	
兼容平台	
开发环境	X
CPU 架构	armv7, a
SDK 依赖	libz, libresolv, libc++ , SystemConfiguration.fram

二、集成 SDK

2.1 下载 SDK

- 点击下载集成 oc 版本 demo
- 点击下载集成 swift 版本 demo

点击下载指纹 SDK，SDK 的目录结构如下：

- 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

2.2 将 SDK 接入 XCode

2.2.1 导入 Framework

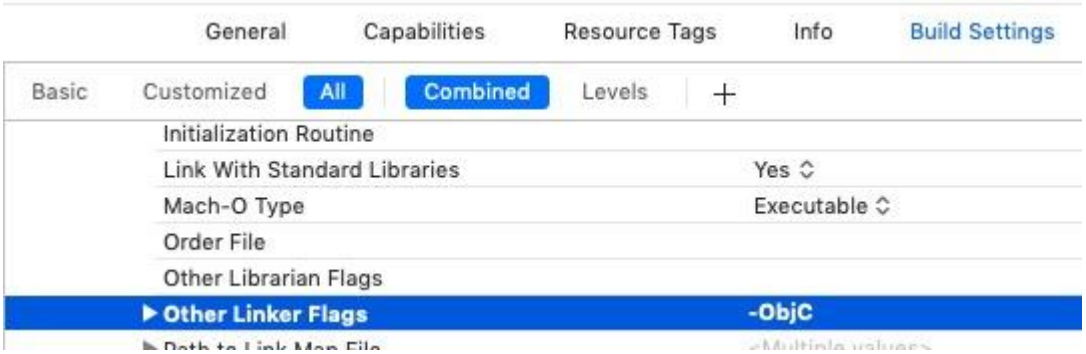
DXRisk.framework，DXRiskWithIDFA.framework，DXRiskStatic.framework，DXRiskStaticWithIDFA.framework`其中之一直接拖入工程目录中，或者右击总文件夹添加文件。

- 如果 App 中包含广告相关的功能，则选择 `DXRiskWithIDFA.framework` 或者 `DXRiskStaticWithIDFA.framework`，该版本可以提供更精准的 token
- 如果没有广告，获取 idfa 可能导致拒绝上架，此时请选择 `DXRisk.framework` 或者 `DXRiskStatic.framework`

2.2.2 添加 FrameWork 到工程

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：



2.2.3 配置打包脚本

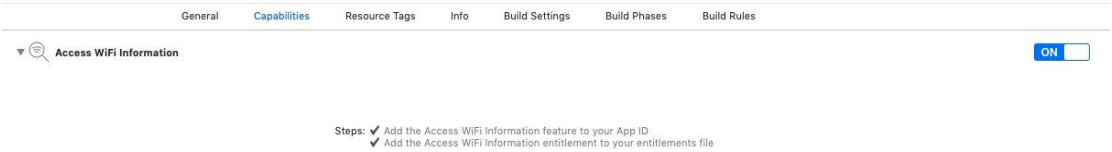
以下的操作仅限导入 DXRisk.framework ， DXRiskWithIDFA.framework 动态库

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，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
```

2.2.4 权限注意

在 iOS 12 的环境最好在 Capabilities->Access WiFi Information 进行开启操作，这样方便设备指纹数据采
集。



三、接口使用说明

3.1 方法和参数说明

```
// 风控组件：DXRiskManager 类
@interface DXRiskManager : NSObject

// 字符串常量
extern NSString* const DXRiskManagerKeyUserId;
extern NSString* const DXRiskManagerKeyEmail;
extern NSString* const DXRiskManagerKeyPhone;
extern NSString* const DXRiskManagerKeyUserExtend1;
extern NSString* const DXRiskManagerKeyUserExtend2;
extern NSString* const DXRiskManagerKeyURL;//私有化服务器地址
extern NSString* const DXRiskManagerKeyBackup;//私有化下使用，将数据备份到顶象服务器（开启为 DXRiskManagerKeyBackupEnable 值）
extern NSString* const DXRiskManagerKeyBackupAppId; // 私有化下使用，指定数据备份到顶象服务器的 AppId

extern NSString* const DXRiskManagerKeyDegradeNotify;//数据降级通知，若打开，服务端会有响应的降级统计
extern NSString* const DXRiskManagerKeyCountry;//国家地区设置，默认中国

extern NSString* const DXRiskManagerKeyDelayMsTime; //可填设置请求超时毫秒时间（默认值为 500 ，范围是：【100 ： 3000】）
// NoticeDegrade 参数
/* The NoticeDegrade Value. This value only be used pair with key:DXRiskManagerKeyDegradeNotify to notify token degrade. */
extern NSString* const DXRiskManagerKeyDegradeNotifyEnable;
// Backup 参数
/* The Backup Value. This value only be used pair with key:DXRiskManagerKeyBackup to set data backup. */
extern NSString* const DXRiskManagerKeyBackupEnable;
// Country 参数
```

```
/* The Country Value. This value only be used pair with
key:DXRiskManagerKeyCountry to set country. */
extern NSString* const DXRiskManagerCountryChina;
/* The Country Value. This value only be used pair with
key:DXRiskManagerKeyCountry to set country. */
extern NSString* const DXRiskManagerCountryIndonesia;

/**
 采集端的设备指纹信息，上传至风控后台，再由风控后台返回 token。
 该 API 为耗时操作，因此必须在非主线程上调用。

 @param appId appId   开通服务后可在二级菜单“应用管理”中获取
 @param extendsParams 配置项
 @return token
 */
+ (NSString *)getToken:(NSString *)appId extendParams:(NSDictionary
*)extendsParams;

/**
 DXRiskManager -- 初始化方法
 */
+ (BOOL)setup;
@end
```

3.2 使用示例

```
// 整个过程由于是耗时操作，必须要在非主线程上执行，否则会阻塞 UI。如果本身已经在非
UI 线程上执行，则不需要另开线程
    dispatch_queue_t dxrisk_queue =
dispatch_queue_create("com.dingxiang.dxrisk", DISPATCH_QUEUE_CONCURRENT);
    dispatch_async(dxrisk_queue, ^{
        NSMutableDictionary *dic = [NSMutableDictionary dictionary];
        // 根据业务逻辑，填充自定义字段
        [dic setObject:@"123456" forKey:DXRiskManagerKeyUserId];

        // 如需设置海外服务器，可选值请参考头文件字段
        [dic setObject:DXRiskManagerServiceAreaSoutheastAsia
forKey:DXRiskManagerKeyServiceArea];
        // 如需自定义服务端 URL，填充 DXRiskManagerKeyURL 字段（私有化部署的情况）
        // 如有需要则添加 DXRiskManagerKeyBackup 参数，Value 固定为
DXRiskManagerKeyBackupEnable
        // DXRiskManagerKeyBackupAppId 参数可不填，参数为顶象备份数据库提供的备
份 Appid
        [dic setObject:@"http://xxxxxxx" forKey:DXRiskManagerKeyURL];
```

```
        // [dic setObject:DXRiskManagerKeyBackupEnable
forKey:DXRiskManagerKeyBackup];
        [dic setObject:@"xxxxxxxxxxxxxxxxxxx"
forKey:DXRiskManagerKeyBackupAppId];
        // 此配置关闭 SDK 缓存，表示每次调用接口都强制采集信息
        // [dic setObject:@"clear" forKey:@"PRIVATE_CLEAR_TOKEN"];

        // 获取 token
        // 注意：token 最好不要保存在某个局部变量或者字段，每次使用时，都通过 API
获取。
        BOOL isSuccess = [DXRiskManager setup];
        NSLog(@"setup success: %@" , isSuccess ? @"YES":@"NO");
        NSString *constID = [DXRiskManager getToken:@"xxxxxxxxxxxxxxxxxxx"
extendsParams:dic];
        NSLog(@"constID: %@", constID);
        // TODO 把 constid 通过 Post 请求，传到业务后台。
        // 下面是模拟频繁调用的过程
        while(TRUE) {
            NSLog(@"constID: %@", [DXRiskManager
getToken:@"xxxxxxxxxxxxxxxxxxx" extendsParams:dic]);
            [NSThread sleepForTimeInterval:.5];
        }
    });
```

3.3 异常说明

在获取 token 过程中，如果因为网络超时或者加解密失败，该接口有可能会返回为 null，同时会输出 tag 为 DXRISK 的错误信息，具体描述如下：

```
DXRISK_REQUEST_NETWORK_ERR          -1001       //网络链接失败
DXRISK_REQUEST_DECRYPT_ERR          -1002       //数据解密错误
DXRISK_REQUEST_UNCOMPRESS_ERR       -1003       //解压错误
DXRISK_REQUEST_RESPONSE_EMPTY_ERR    -1004       //返回为空
DXRISK_REQUEST_DATA_PARSE_ERR       -1005       //数据解析失败
DXRISK_REQUEST_DIRTY_DATA_ERR       -1006       //脏数据
DXRISK_CONST_ID_EMPTY               -1007       //constid 为空
```

如果出现上述错误信息，请联系顶象技术人员。

4 微信小程序接入

js 接入

一、下载设备指纹 js

SaaS 版本下载

私有化版本下载

请注意选择对应的版本，下载完成后放到本地项目中。

二、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。

三、使用

1. 代码接入

```
const ConstId = require('本地设备指纹 js 存放路径')
Page({
    onLoad: function () {
      new ConstId({
        appId: '【这里填写在顶象官网申请到的 AppID】', // 唯一标识，必填
        server: 'https://host/udid/w1' // 服务接口地址，请注意私有化客户填写自有域名
      }, (e, id) => {
        if (e) {
          console.log(e)
          return
        }
        console.log('constId:', id)
      })
    }
})
```

2. SaaS 用户在小程序后台配置业务域名 `https://constid.dingxiang-inc.com`，私有化用户配置本地部署域名，测试阶段可以开启微信开发者工具右上角详情->本地设置->“不校验域名”。

5 支付宝小程序接入

接入方式和微信小程序类似，但是使用的 SDK 不一样。如有需要，请联系客服获取。

```
const ConstId = require('本地设备指纹 js 存放路径')
Page({
    onLoad: function () {
      new ConstId({
        appId: '【这里填写在顶象官网申请到的 AppID】', // 唯一标识，必填
        server: 'https://host/udid/w1' // 服务接口地址，请注意私有化客户填写自有域名
      }, (e, id) => {
        if (e) {
          console.log(e)
          return
        }
        console.log('constId:', id)
      })
    }
})
```

三、后端接入

接口详细描述： 根据 token,appId,sign 三个参数获得设备信息

1.SDK 接入方法说明

1.1 Java SDK 接入

1. 包的引入

```
<dependency>
    <groupId>com.dingxiang-inc</groupId>
    <artifactId>ctu-client-sdk</artifactId>
    <version>2.6</version>
</dependency>
```

2. Java 使用示例

```
public class ConstantIdDemo {
    //顶象控制台，设备指纹菜单应用管理里面获取
    private static String appId = "0091a3xxxxxxxxxx557fac67b2f5afb";
    private static String appSecret = "e38dxxxxxxxxx6807c9e1edebaa2836";

    public static void main(String[] args) throws IOException {
        // 填写设备指纹域名或者 url 如：http://127.0.0.1:8080
        CtuConstidClient client = new
CtuConstidClient("https://constid.dingxiang-inc.com", appId, appSecret);
        //设备指纹
        String result =
client.getDeviceInfo("62c5013cel9o4xxxxxxxxxtTtKW5BwWtQq9u1f1");
         System.out.println(JSON.toJSON(result));
    }
}
```

1.2 PHP SDK 接入

[点击下载 SDK]

使用示例

```
class Demo {

    // 根据实际情况填写
    const appKey = "你的 AppID";

    // 根据实际情况填写
    const appSecret = "你的 AppSecret";

    // 根据实际情况填写
    const token = "SDK 里面获取到的 token";
}

// 根据 token 获取设备详细信息工具类
$requestHandle = new DeviceFingerprintHandle();
// 设置请求超时时间。因为存在设备指纹降级和网络抖动的情况，默认 2 秒。可以根据实际情况调整
// $requestHandle->setTimeout(2);
```

```php
// 填写设备指纹域名或者 url 入：http://127.0.0.1:9090
$responseData = $requestHandle->getDeviceInfo("https://constid.dingxiang-inc.com/udid/api/getDeviceInfo",
    Demo::appKey, Demo::appSecret, Demo::token);
$result = json_decode($responseData, true);

// 请求状态码。非 200 表示没有获取到设备明细信息
if ($result['stateCode'] == 200)
    echo "设备明细信息如下： : " . json_encode($result['data'], true);
else
    echo $result['message'];
```

1.3 NodeJS SDK 接入

```
npm i dx-const-id-sdk --save
```

```node
const SDK = require('dx-const-id-sdk')

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

sdk.getDeviceInfo(token).then((data) => {
  console.log(data)
}).catch(err => {
  console.log('获取设备信息失败')
})
```

2.HTTP 接口接入方法

2.1 请求参数

字段	类型	描述
appId	String	当前应用的标识
sign	String	sign = MD5(appSecret + token + appSecret),AppSecret 为

字段	类型	描述
token	String	用户前端获取的 token

2.2 成功响应

字段	类型	描述
stateCode	int	状态码
message	String	状态描述
data	Json	返回设备信息及设备风

```
{
    "stateCode": 200,
    "message": "请求响应成功",
    "data": {
        "hardId": "22e38229a7eda501c58bf3ddee1a340a",
        ......
    }
}
```

2.3 错误响应

字段	类型
stateCode	int
message	String

字段	类型
data	Json

```
{
    "stateCode": -10002,
    "message": "签名为空或验证失败",
    "data": null
}
```

错误码	描述
-10001	appId 不存在或已经过期
-10002	签名为空或验证失败
-10003	token 为空或没有对应的设备信息
-10004	token 已经过期
-10005	服务器内部异常
-10006	证书已经失效
-10007	服务器限流

3. HTTP 接口返回参数明细说明

3.1 免费用户

字段名	字段描述
token	设备 token
hardId	设备指纹

3.1.1 web 端返回结果

字段名	字段描述
token	设备 token
hardId	设备指纹

3.2 收费用户

3.2.1 移动端返回结果（部分示例，更多返回字段请联系售后支持获取）

字段名	字段描述
token	设备 token
deviceType	设备类型
hardId	设备指纹
producter	生产厂商

字段名	字段描述
macAddress	mac 地址
isEmulator	模拟器运行
isRoot	是否 root
isMultirun	是否多开
isInject	是否存在注入风险
isMemdump	是否存在内存 dump 风险
isDebug	是否存在调试风险
isHook	是否存在 hook 风险
isJailBreak	是否越狱
isVpn	是否使用 vpn
isProxy	是否使用代理
isSimulateGPS	是否篡改 GPS

字段名	字段描述
isCloudPhone	是否云真机
...	...

3.2.2 web 端返回结果（部分示例，更多返回字段请联系售后支持获取）

字段名	字段描述
token	设备 token
deviceType	设备类型
hardId	设备指纹
canvasId	canvas 指纹
webGl	WebGl 指纹
resolution	设备分辨率
isLiedBrowser	是否伪造浏览器
isCookieDisabled	是否禁用 cookie

字段名	字段描述
isTamperUa	是否篡改浏览器 ua
isTamperRes	是否篡改分辨率
isTamperCd	是否篡改浏览器颜色深度
isEmulator	是否模拟器
...	...

四、 附：隐私政策示例

建议在集成了 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，示例如下：

隐私政策（拟，可根据实际情况进行修改）

一、我们如何收集和使用个人信息
个人信息是指以电子或者其他方式记录的能够单独或者与其他信息结合识别特定自然人身份或者反映特定自然人活动情况的各种信息。我们深知个人信息对您的重要性，您的信赖对我们非常重要，我们将严格遵守法律法规要求采取相应的安全保护措施，致力于保护您的个人信息安全可控。我们仅会出于本政策所述的以下目的，收集和使用您的个人信息：

1. 当您使用 XXX 产品及相关服务时，为了保障软件与服务的正常运行，我们会收集您的硬件型号、操作系统版本号、国际移动设备识别码（IMEI）、网络设备硬件地址（MAC）、IP 地址、软件版本号、网络接入方式及类型、操作日志等信息。请您了解，这些信息是我们提供服务和保障产品正常运行所必须收集的基本信息。

所属系统	第三方	第三方收集个人信息目的	第三方
iOS	顶象设备指纹 SDK	标识设备，保障账号及交易安全	设备版本、系统版本、生成 ID、
Android	顶象设备指纹 SDK	标识设备，保障账号及交易安全	设备版本、系统版本、生成 ID、地址、IMSI、IMEI

2. Cookie 和同类技术的服务 Cookie 和同类设备信息标识技术是互联网中普遍使用的技术。当您使用我们的服务时，我们可能会使用相关技术向您的设备发送一个或多个 Cookie 或匿名标识符（以下简称“Cookie”），以收集、标识和存储您访问、使用本产品时的信息。我们承诺，不会将 Cookie 用于本隐私政策所述目的之外的任何其他用途。我们使用 Cookie 主要为了保障产品与服务的安全、高效运转，排查崩溃、延迟的相关异常情况，帮助您省去重复您填写表单、输入搜索内容的步骤和流程。 同时，我们可能会利用 Cookie 向您展示您可能感兴趣的信息或功能，并优化您对广告的选择。大多数浏览器均为用户提供了清除浏览器缓存数据的功能，您可以进行相应的数据清除操作，或可修改对 Cookie 的接受程度或拒绝我们的 Cookie。您可能因为这些修改，无法使用依赖于 Cookie 的服务或相应功能。

附：相关定义 个人信息：以电子或者其他方式记录的与已识别或者可识别的自然人有关的各种信息,不包括匿名化处理后的信息。 设备信息：包括设备标识符（IMEI、IDFA、Android ID、MAC、OAID、IMSI 及其他设备相关信息）、设备参数及系统信息（设备类型、设备型号、操作系统及硬件相关信息），设备网络环境信息（IP 地址，WiFi 信息，基站信息及其他网络相关信息）。以产品实际采集情况为准。

二、我们如何保护您的个人信息

（一）我们已使用符合业界标准的安全防护措施保护您提供的个人信息，防止数据遭到未经授权访问、公开披露、使用、修改、损坏或丢失。我们会采取一切合理可行的措施，保护您的个人信息。例如，在您的浏览器与“服务”之间交换数据时受 SSL 加密保护；我们同时对我们网站提供 https 安全浏览方式；我们会使用加密技术确保数据的保密性；我们会使用受信赖的保护机制防止数据遭到恶意攻击；我们会部署访问控制机制，确保只有授权人员才可访问个人信息；以及我们会举办安全和隐私保护培训课程，加强员工对于保护个人信息重要性的认识。

（二）我们会采取一切合理可行的措施，确保未收集无关的个人信息。我们只会在达成本政策所述目的所需的期限内保留您的个人信息，除非需要延长保留期或受到法律的允许。

（三）互联网并非绝对安全的环境，而且电子邮件、即时通讯、及与其他我们用户的交流方式并未加密，我们强烈建议您不要通过此类方式发送个人信息。请使用复杂密码，协助我们保证您的账号安全。

（四）互联网环境并非百分之百安全，我们将尽力确保或担保您发送给我们的任何信息的安全性。如果我们的物理、技术、或管理防护设施遭到破坏，导致信息被非授权访问、公开披露、篡改、或毁坏，导致您的合法权益受损，我们将承担相应的法律责任。

（五）在不幸发生个人信息安全事件后，我们将按照法律法规的要求，及时向您告知：安全事件的基本情况和可能的影响、我们已采取或将要采取的处置措施、您可自主防范和降低风险的建议、对您的补救措施等。我们将及时将事件相关情况以邮件、信函、电话、推送通知等方式告知您，难以逐一告知个人信息主体时，我们会采取合理、有效的方式发布公告。 同时，我们还将按照监管部门要求，主动上报个人信息安全事件的处置情况。