ALIBABA CLOUD

阿里云

风险识别

API及SDK

文档版本：20201229

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.公共参数

本文介绍调用风险识别API时使用的公共请求参数和公共返回参数。

公共请求参数

风险识别API接口的入参参数包含公共请求参数和具体服务事件参数，公共请求参数是指每一个接口都需要使用到的参数，以下表格是公共请求参数的详细介绍。

名称	类型	是否必须	描述
Format	String	是	返回消息的格式。取值： • **JSON** （默认值） • **XML**
Version	String	是	API版本号，使用YYYY-MM-DD日期格式，本版本对应为2018-09-19。
AccessKeyId	String	是	阿里云颁发给用户的访问服务所用的密钥ID。
Signature	String	是	签名结果串，关于签名的计算方法，请参见签名机制签名方式。
SignatureMethod	String	是	签名方式，目前支持HMAC-SHA1。
Timestamp	String	是	请求的时间戳。日期格式按照ISO8601标准表示，并需要使用UTC时间。格式为：yyyy-MM-ddTHH:mm:ssZ。例如，北京时间2019年8月22日09点44分35秒表示为2019-08-22T01:44:35Z。
SignatureVersion	String	是	签名算法版本，取值为1.0。
SignatureNonce	String	是	唯一随机数，用于防止网络重放攻击。不同请求要使用不同的随机数值。
Action	String	是	值必须是ExecuteRequest。

名称	类型	是否必须	描述
Service	String	是	阿里云云盾颁发的服务code，不可随意设置。以下是各服务对应的code列表： • **注册风险识别-增强版服务**：account_abuse_pro • **注册风险识别服务**：account_abuse • **营销风险识别-增强版服务**：coupon_abuse_pro • **营销风险识别服务**：coupon_abuse • **登录风险识别-增强版服务**：account_takeover_pro • **登录风险识别服务**：account_takeover • **设备风险识别服务**：device_risk • **风险情报服务**：risk_intelligence • **邮箱画像服务**：email_risk • **地址评分服务**：address_validation • **蚁盾.营销保护服务**：yd_coupon_abuse • **蚁盾.注册保护服务**：yd_account_abuse • **IP风险画像服务**：risk_intelligence_ip
ServiceParameters	String	是	服务参数列表的Json格式为 `{“accountId”:101234567…该事件下其他请求参数}` 。以下是具体字段和参数值格式的链接列表： • 注册风险识别-增强版事件及返回参数 • 注册风险识别事件及返回参数 • 营销风险识别-增强版事件及返回参数 • 营销风险识别事件及返回参数 • 登录风险识别-增强版事件及返回参数 • 登录风险识别事件及返回参数 • 设备风险识别事件及返回参数 • 业务风险情报事件及返回参数 • 邮箱画像事件及返回参数 • 地址评分事件及返回参数

以下是**注册风险识别服务**的请求示例：

```
https://saf.cn-shanghai.aliyuncs.com/
?Format=JSON
&Version=2018-09-19
&Signature=vpEEL0zFHfxXYzSFV0n7%2FZiFL9o%3D
&SignatureMethod=Hmac-SHA1
&SignatureNonce=15215528852396
&SignatureVersion=1.0
&Action=ExecuteRequest
&AccessKeyId=1234567saf
&Timestamp=2018-06-01T12:00:00Z
&Service=account_abuse_pro
&ServiceParameters={"accountId":10123567 …该事件下其他请求参数,详见具体服务的事件参数}
```

公共返回参数

您发送的每次接口调用请求，无论成功与否，系统都会返回唯一识别码RequestId。其他返回参数根据服务不同有不同的业务含义。

以下是**注册风险识别-增强版服务**返回参数的Json示例：

```
{
  "Data": {
    "score": 85.92, //风险得分，取值范围：0~100，double格式
    "tags": "rm0101" //风险标签，string格式
  },
  "Message": "OK",
  "Code": 200, //200表示请求成功
  "RequestId": "4EF9AEE5-288F-4176-8110-00EDC91A614E"
}
```

> **说明** 本文中的返回示例为了便于阅读，做了格式化处理，实际返回结果没有进行换行、缩进等格式化处理。

以下是**邮箱画像服务**返回参数的Json示例：

```
{
  "Data": {
    "tags": "temp_email,gibberish_email" //string格式
  },
  "Message": "OK",
  "Code": 200, //200表示请求成功
  "RequestId": "4EF9AEE5-288F-4176-8110-00EDC91A614E"
}
```

错误码

Code	含义描述
200	请求成功。
400	ServiceParameters（事件参数）不合法。
402	QPS超过已购规格，限流。
403	权限不足，服务未开通或已到期。
404	Service（服务参数）不合法。
500	内部服务器错误。

2.设备风险SDK Android接入（2020版）

本文档介绍了设备风险SDK（Android系统）的接入流程。

前提条件

设备风险SDK需在Android 4.0.3+（minSdkVersion版本15+）以上版本的系统运行。

权限说明

为增强风险识别效果，当前SDK需要以下权限：

权限内容	是否必选	备注
android.permission.INTERNET	是	没有该权限将导致SDK功能不可用。
android.permission.ACCESS_NETWORK_STATE	否（推荐赋予）	用于获取设备网络状态信息。
android.permission.READ_PHONE_STATE	否（推荐赋予）	该部分权限在Android 6.0以上系统中需要动态获取。 如果您要启用相关权限，那么在接入设备风险SDK并调用init初始化接口之前，确保您的App已经被授予了相关权限。
android.permission.WRITE_EXTERNAL_STORAGE	否（推荐赋予）	
android.permission.READ_EXTERNAL_STORAGE	否（推荐赋予）	

下载和配置SDK

1. 下载Android SDK，并完成解压。SDK为Android标准的.aar包。
2. 拷贝SDK的aar文件到工程的libs目录下，并在APP的build.gradle中添加以下依赖关系：

```
implementation files('libs/Android-AliyunDevice-版本号.aar')
implementation 'com.squareup.okhttp3:okhttp:3.11.0'
implementation 'com.squareup.okio:okio:1.14.0
```

初始化SDK

完成SDK内部初始化，在App启动的时候，您需要尽可能早的调用该函数 。

- 函数原型

```
public interface SecurityInitListener {
  // code表示接口调用状态码
  void onInitFinish(int code);
}

public void init(Context ctx,
        String userAppKey,
        SecurityInitListener securityInitListener);
```

- 参数

ctx：当前Application Context，或Activity Context。

userAppKey：AppKey，用于标识一个用户，可在阿里云控制台的设备APP管理申请获取。

securityInitListener：设备风险SDK初始化回调接口，可在回调中判断初始化是否成功。其中，code字段取值范围可参考“状态返回值”。

- 返回值

无。

获取Session

获取客户端Session，并上报到业务服务器，后续通过服务器端查询阿里云设备风险识别接口，从而获取设备风险信息。

注意：getSession接口相对比较耗时，请务必在非主线程上调用，否则可能会引起ANR而导致APP crash。

- 函数原型

```
public class SecuritySession {
  // 接口调用状态码
  public int code;

  // 用于服务器端查询结果的session字符串。
  public String session;
}

public SecuritySession getSession();
```

- 返回值

SecuritySession类型。

code：返回接口调用状态码，可用于判断接口调用是否成功。code字段取值范围可参考“状态返回值”。

session：返回Session字符串信息，可用于业务后续查询阿里云设备风险识别接口

状态返回值

```
public class SecurityCode {
```

```
private static final int SC_CODE_BASE = 10000;

/**
 * 成功，无错误发生
 */
public static final int SC_SUCCESS = SC_CODE_BASE;

/**
 * SDK未调用初始化
 */
public static final int SC_NOT_INIT = SC_CODE_BASE + 1;

/**
 * SDK需要的Android基础权限未完全授权
 */
public static final int SC_NOT_PERMISSION = SC_CODE_BASE + 2;

/**
 * 未知错误
 */
public static final int SC_UNKNOWN_ERROR = SC_CODE_BASE + 3;

/**
 * 网络错误
 */
public static final int SC_NETWORK_ERROR = SC_CODE_BASE + 4;

/**
 * 网络错误，返回内容为空串
 */
public static final int SC_NETWORK_ERROR_EMPTY = SC_CODE_BASE + 5;

/**
 * 网络返回的格式非法
 */
public static final int SC_NETWORK_ERROR_INVALID = SC_CODE_BASE + 6;

/**
 * 服务端配置解析失败
 */
public static final int SC_PARSE_SRV_CFG_ERROR = SC_CODE_BASE + 7;
```

```
    public static final int SC_PARSE_SRV_CFG_ERROR = SC_CODE_BASE + 7;

    /**
     * 网关返回失败
     */
    public static final int SC_NETWORK_RET_CODE_ERROR = SC_CODE_BASE + 8;

    /**
     * 用户传进来的appkey是空的
     */
    public static final int SC_APPKEY_EMPTY = SC_CODE_BASE + 9;

    /**
     * 参数错误
     */
    public static final int SC_PARAMS_ERROR = SC_CODE_BASE + 10;
}
```

示例代码

```
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // 初始化设备风险SDK，init接口需要在APP启动尽可能早的时候调用。
    SecurityDevice.getInstance().init(this, "xxxxxxx", new SecurityInitListener() {
    @Override
    public void onInitFinish(int code) {
      if (SecurityCode.SC_SUCCESS != code) {
          Log.e("AliyunDevice", "初始化失败，继续调用getSession获取的结果是无效的.");
        } else {
          Log.i("AliyunDevice", "初始化成功.");
        }
      }
    });

    // 获取客户端Session，getSession接口比较耗时，切勿在UI线程中调用。
     new Thread(){
        @Override
        public void run() {
          SecuritySession securitySession = SecurityDevice.getInstance().getSession();
          if(null != securitySession){
            if(SecurityCode.SC_SUCCESS == securitySession.code){
              Log.d("AliyunDevice", "session: " + securitySession.session);

              // 发送session到业务自有服务器并查询阿里云设备风险识别接口。
              // sendToAPPServer(securitySession.session);
            } else {
              Log.e("AliyunDevice", "getSession error, code: " + securitySession.code);
            }
          } else {
            Log.e("AliyunDevice", "getSession is null.");
          }
        }
      }.start();
  }
}
```

接口混淆配置

```
-keep class net.security.device.api.** {*;}
-dontwarn net.security.device.api.**
```

调用风险识别API接口

将deviceToken与其他参数，根据如下相应的风险识别服务事件参数文档说明，请求风险识别API接口进行识别：

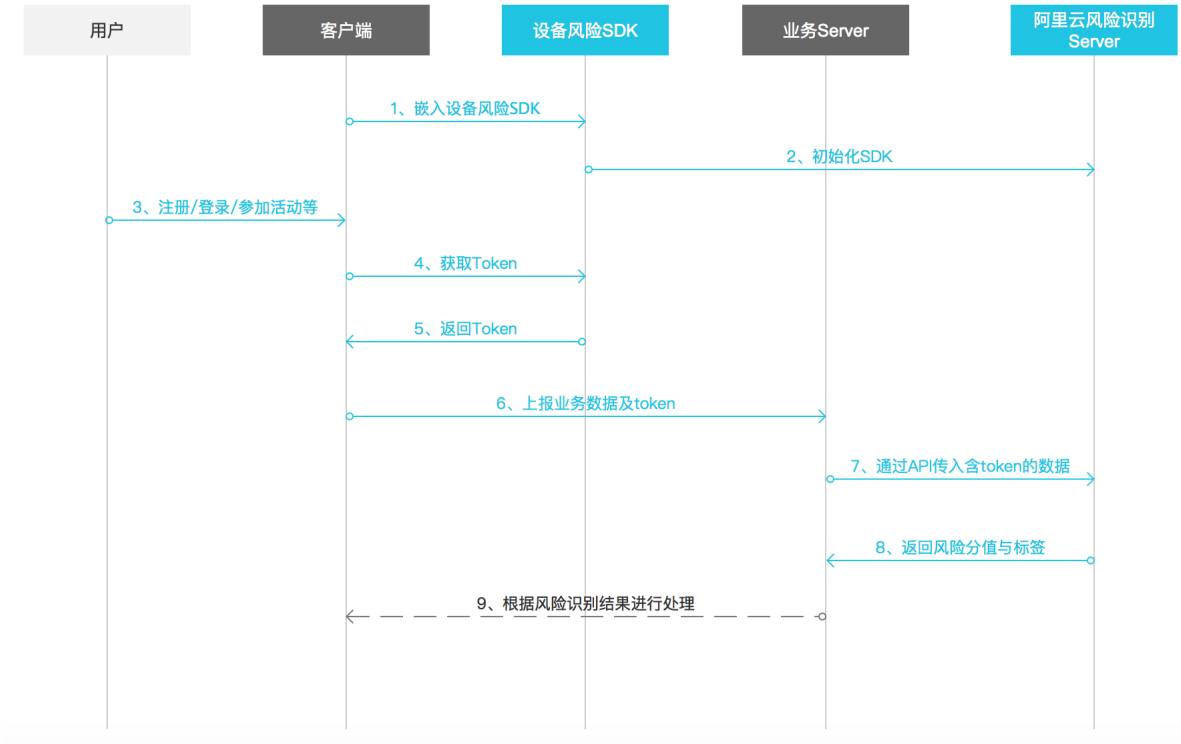
注册风险识别-增强版事件及返回参数

营销风险识别-增强版事件及返回参数

登录风险识别-增强版事件及返回参数

设备风险识别事件及返回参数

接入和使用时序图如下，其中第1和2步骤仅首次加载需要调用，第3、4、5、6、7、8、9步骤可以根据业务情况循环发起。



3.设备风险SDK iOS接入（2020版）

本文档介绍了阿里云设备指纹iOS SDK接入流程。

前提条件

- iOS系统版本要求：

  iOS系统版本为9.0及以上。

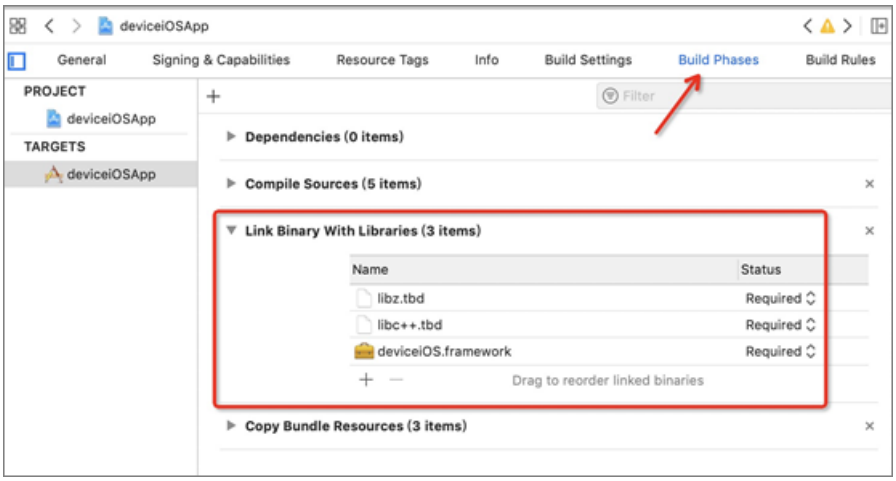
- 已完成以下依赖配置：

  i. 下载iOS SDK（SDK为XCode上标准静态framework包），并在下载SDK的控制台中生成APPKey。

  ii. 在XCode中引用已下载的iOS SDK包。

  iii. 选择**工程配置**，修改**Build Phases > Link Binary With Libraries**，添加deviceiOS.framework依赖包：

  ```
  libz.tbd
  libc++.tbd
  deviceiOS.framework
  ```



接口定义说明

接入SDK的时候，设备指纹接口返回内容说明如下：

```
/**
 * 设备指纹deviceToken
 */
@interface SecuritySession: NSObject
/**
 * 获取Session操作的结果
 */
@property(atomic) int code;
/**
 * 包含Session结果的字符串
 */
@property(copy, atomic) NSString * session;
@end
```

```
@interface SecurityDevice : NSObject
/**
 * 设备指纹初始化函数
 */
- (void)initDevice:(NSString *)userAppKey :(void (^)(int))initCallback;//设备指纹userAppKey，从产品控制台申请获得
/**
 * 获取DeviceToken
 */
- (SecuritySession *) getSession;
@end
```

接入流程

1. 初始化调用。

   针对开发环境为xcode11及以下版本，调用代码示例：

```
//初始化设备指纹
SecurityDevice * securityDevice = [
  [SecurityDevice alloc] init
];
if (nil != securityDevice) {
  [securityDevice initDevice: @ "xxxxxx": ^ (int code) {
    NSString * initResult = [NSString stringWithFormat: @ "设备指纹初始化，结果 %d", code];
    NSLog(@ "%@", initResult);
    if (10000 != code) {
      NSLog(@ "初始化失败，后续getSession接口调用结果不可用于风险标签查询");
    } else {
      NSLog(@ "初始化成功，后续可以正常调用getSession接口");
    }
  }];
}
```

针对开发环境为xcode12及以上版本，调用代码示例：

```
/**
 * 如果是iOS14上，尝试提供IDFA的弹框
 */
typedef void (^IDFARequestBlock)(bool success);
API_AVAILABLE(ios(14))
static bool isATTrackingEnabled(ATTrackingManagerAuthorizationStatus status){
  if(ATTrackingManagerAuthorizationStatusAuthorized == status){
    return true;
  }
  return false;
}
- (void)helperRequestIDFAPermissionWithBlock:(IDFARequestBlock) complete{
  if(@available(iOS 14, *)){
    ATTrackingManagerAuthorizationStatus authStatus = ATTrackingManager.trackingAuthorizationSt
atus;
    //未弹框
    if(ATTrackingManagerAuthorizationStatusNotDetermined == authStatus){
      [ATTrackingManager requestTrackingAuthorizationWithCompletionHandler:^(ATTrackingManage
rAuthorizationStatus status) {
        if(nil != complete){
          return complete(isATTrackingEnabled(status));
        }
      }];
```

```
    }else if(nil != complete){
      return complete(isATTrackingEnabled(authStatus));
    }
  }
}
- (void)initSecurityDevice{
  [securityDevice initDevice: @ "xxxxxx": ^ (int code) {
    NSString * initResult = [NSString stringWithFormat: @ "设备指纹初始化，结果 %d", code];
    NSLog(@ "%@", initResult);
    if (10000 != code) {
      NSLog(@ "初始化失败，后续getSession接口调用结果不可用于风险标签查询");
    } else {
      NSLog(@ "初始化成功，后续可以正常调用getSession接口");
    }
  }];
}
- (void)viewDidLoad {
  [super viewDidLoad];
  //注意！！！！， 以这种方式启动启动设备指纹， iOS14和非iOS14分开判断
  if(@available(iOS 14, *)){
    [self helperRequestIDFAPermissionWithBlock:^(bool success){
      if(success){
        NSLog(@"IDFA--->有权限了");
      }else{
        NSLog(@"IDFA--->没权限了");
      }
      [self initSecurityDevice];
    }];
  }else{
    [self initSecurityDevice];
  }
}
```

2. 获取deviceToken。

   从设备指纹SDK获取deviceToken，通过getSession()方法可以从阿里云服务端查询当前设备的风险识别结果。以下是调用代码示例：

```
//获取设备指纹Session, 请注意此接口应该在收到初始化接口回调成功之后调用，一般init接口会在3秒内完成
SecuritySession * securitySession = [securityDevice getSession];
NSString * rs = [NSString stringWithFormat: @ "[%d]%@", securitySession.code, securitySession.session];
NSLog(@ "Session=>%@", rs);
if (10000 != code) {
  NSLog(@ "获取session失败，调用结果不可用于风险标签查询");
} else {
  NSLog(@ "获取session成功，后续可以正常查询风险标签");
}
```

后续操作

完成设备指纹SDK集成后，建议您进行以下操作：

- AppStore上架时，请确保App已申请了IDFA权限，并在*Info.plist*中已添加 `NSUserTrackingUsageDescription` 说明，否则会导致上架失败。详细内容请参考iOS相关说明文档。
- 请确保在初始化成功（initDevice 收到回调时返回code==10000 ）后，再调用getSession接口获取Session。
- 为了防止调用initDevice的时候出现网络连接失败的情况，建议在 `getSession` 之前检查 `initDevice` 是否成功。这样是为了在 `initDevice` 不成功则重新发起初始化调用时，能确保至少一次初始化成功。

调用风险识别API接口

将deviceToken与其他参数，根据如下相应的风险识别服务事件参数文档说明，请求风险识别API接口进行识别：

注册风险识别-增强版事件及返回参数

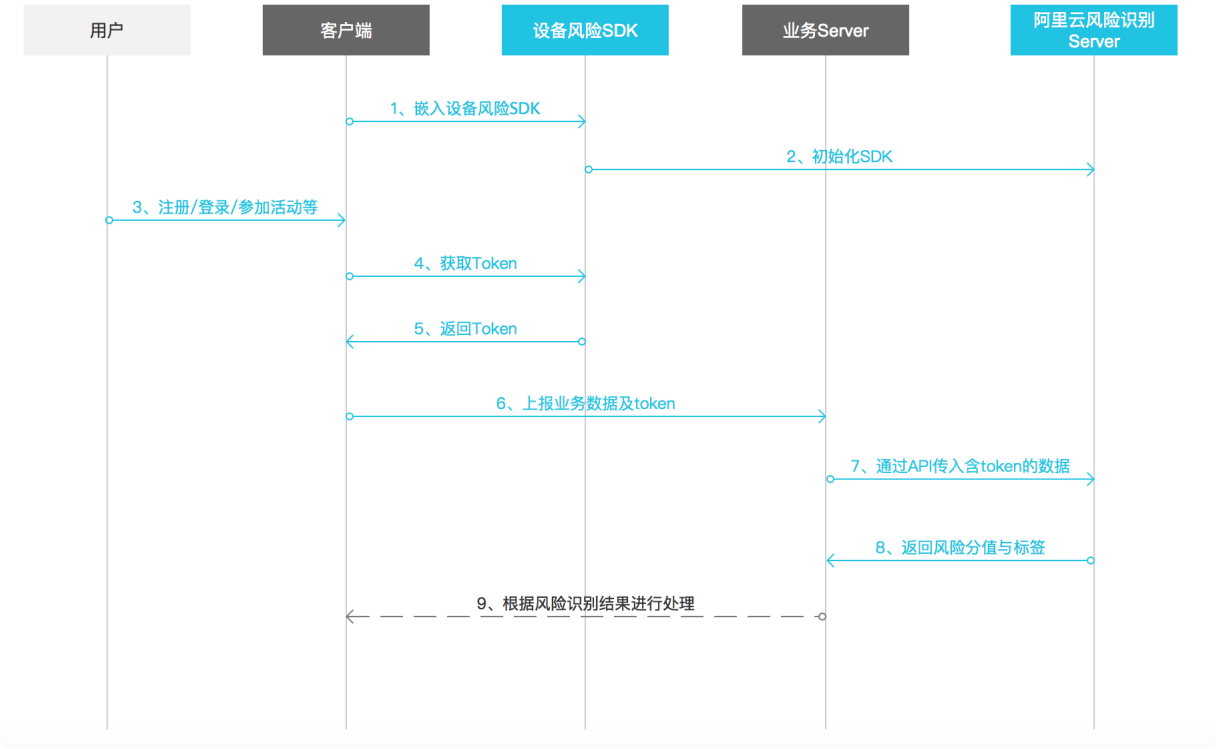
营销风险识别-增强版事件及返回参数

登录风险识别-增强版事件及返回参数

设备风险识别事件及返回参数

接入和使用时序图如下：

> **说明** 其中第1和2步骤仅首次加载需要调用，第3、4、5、6、7、8、9步骤可以根据业务情况循环发起。


用户
客户端
设备风险SDK
业务Server
阿里云风险识别Server
1、嵌入设备风险SDK
2、初始化SDK
3、注册/登录/参加活动等
4、获取Token
5、返回Token
6、上报业务数据及token
7、通过API传入含token的数据
8、返回风险分值与标签
9、根据风险识别结果进行处理

4.蚁盾.营销保护事件及返回参数

服务事件参数是指公共参数中，ServiceParameters字段的传参，格式为json格式，以下为**蚂蚁.营销保护事件参数**。

字段名称	字段描述	数据格式	样例数据	是否必须
mobile	用户手机号	String	13805510000	必传
ip	用户IP地址	String	202.38.71.01	必传
apdidToken	蚁盾设备指纹SDK生成的token	String	ad7/6RD+UMueHVS5Q2TbimAFbaSSDG76YW7EWAE=bdfDFDFDJL5DDF4	可传（推荐传入，效果更佳）
userId	用户账号id, 唯一标识一个账号即可	String	100001	可传
email	用户邮箱	String	abc@aliyun.com	可传
mac	用户Mac地址	String	ec:df:3a:4e:63:df	可传
userAgent	用户UA	String	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/21.0.1180.71 Safari/537.1 LBBROWSER	可传
imsi	国际移动用户识别码，手机SIM卡编号	String	460030912121001	可传
imei	国际移动设备标识	String	357769804451095	可传
wifiMac	WiFi网关Mac	String	ec:df:3a:4e:63:de	可传
platform	操作平台，请在如下中选择：ios/android/winphone/h5/pc/other	String	ios	可传
longitude	经度	String	28.01	可传
latitude	纬度	String	36.90	可传

个性化返回参数

个性化返回参数，除公共返回参数外，**蚂蚁.营销保护事件**服务返回参数Data字段中，score 字段值的业务含义根据蚁盾风控团队经验可参考如下：

值区间	风险等级	描述
0到40（不含）	低风险	可放过或打标观察
40（含）到80（不含）	中风险	建议进行一定安全性的二次验证（例如人脸验证、身份证验证）
80（含）到100（含）	高风险	建议进行高强度验证或限制高危业务使用权限

5.SDK

通过使用阿里云风险识别SDK，调用方无需关注签名验证以及Body格式构建等繁琐的事情，推荐您使用风险识别SDK。目前支持JAVA、Python、PHP、C#、Golang、Node.js、Ruby共7种SDK。

JAVA SDK

单击此处下载JAVA SDK源码。

JAVA SDK的环境准备、安装使用请参见阿里云SDK开发指南。

JAVA Maven依赖：

```
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-saf</artifactId>
  <version>1.0.2</version>
</dependency>
```

JAVA Maven推荐依赖仲裁：

```
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-core</artifactId>
  <optional>true</optional>
  <version>4.5.1</version>
</dependency>
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>fastjson</artifactId>
  <version>1.2.60</version>
</dependency>
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.8.2</version>
</dependency>
<dependency>
  <groupId>org.apache.httpcomponents</groupId>
  <artifactId>httpclient</artifactId>
  <version>4.5.3</version>
</dependency>
<dependency>
  <groupId>io.opentracing</groupId>
  <artifactId>opentracing-util</artifactId>
  <version>0.31.0</version>
</dependency>
```

单击此处查看JAVA用例。

单击地址1或地址2访问JAVA Maven仓库。

Python SDK

单击此处下载Python SDK源码。

Python SDK的环境准备、安装使用请参见阿里云SDK开发指南。

1. 安装SDK核心库。
   - 如果您使用Python 2.x，执行以下命令，安装阿里云SDK核心库：

     ```
     pip install aliyun-python-sdk-core
     ```

   - 如果您使用Python 3.x，执行以下命令，安装阿里云SDK核心库：

     ```
     pip install aliyun-python-sdk-core-v3
     ```

2. 安装云产品SAF SDK。

```
pip install aliyun-python-sdk-saf
```

Python用例：

```
from aliyunsdkcore import client
from aliyunsdksaf.request.v20190521 import ExecuteRequestRequest
clt = client.AcsClient('<your-access-key-id>','<your-access-key-secret>','cn-shanghai')
# 设置参数
request = ExecuteRequestRequest.ExecuteRequestRequest()
request.set_accept_format('json')
# 产品Service请参考[公共参数]文档中的Service字段描述
request.add_query_param('Service', '购买的产品Service')
request.add_query_param('ServiceParameters', '入参json字符串')
# 发起请求
response = clt.do_action_with_exception(request)
print(response)
```

PHP SDK

单击此处下载PHP SDK源码。

PHP SDK的环境准备、安装使用请参见阿里云SDK开发指南。

PHP用例：

```
<?php
include_once 'aliyun-openapi-php-sdk/aliyun-php-sdk-core/Config.php';
use saf\Request\V20190521 as saf;
// 初始化
$iClientProfile = DefaultProfile::getProfile("cn-shanghai", "<your-access-key-id>", "<your-access-key-secret
>");
$client = new DefaultAcsClient($iClientProfile);
//设置参数
$request = new saf\ExecuteRequestRequest();
// 产品Service请参考[公共参数]文档中的Service字段描述
$request->setService('购买的产品Service');
$request->setServiceParameters('入参json字符串');
// 发起请求
$response = $client->getAcsResponse($request);
print_r("<br>");
print_r("test");
print_r("\r\n");
print_r($response);
print_r($client);
?>
```

C# SDK

单击此处下载C# SDK源码。

C# SDK的环境准备、安装使用请参见阿里云SDK开发指南。

Golang SDK

单击此处下载Golang SDK源码。

Golang SDK的环境准备、安装使用请参见阿里云SDK开发指南。

Golang用例：

```
package main
import (
  "fmt"
  "github.com/aliyun/alibaba-cloud-sdk-go/services/saf"
)
func main() {
  client, err := saf.NewClientWithAccessKey("cn-shanghai", "<your-access-key-id>", "<your-access-key-secret>")
  request := saf.CreateExecuteRequestRequest()
  endpoints.AddEndpointMapping("cn-shanghai", "saf", "saf.cn-shanghai.aliyuncs.com")
  // 产品Service请参考[公共参数]文档中的Service字段描述
  request.Service = "购买的产品Service"
  request.ServiceParameters = "入参json字符串"
  request.Scheme = "https"
  response, err := client.ExecuteRequest(request)
  if err != nil {
    fmt.Print(err.Error())
  }
  fmt.Printf("response code is %#v\n", response.Code)
  fmt.Printf("response date is %#v\n", response.Data)
  fmt.Printf("response message is %#v\n", response.Message)
  fmt.Printf("response requestId is %#v\n", response.RequestId)
}
```

六、Node.js SDK

单击此处下载Node.js SDK源码。

Node.js SDK的环境准备、安装使用请参见阿里云SDK开发指南。

执行以下命令安装@alicloud/pop-core模块。命令中的--save会将模块写入应用的package.json文件中，作为依赖模块。

```
$ npm install @alicloud/pop-core --save
```

Node.js用例：

```
const Core = require('@alicloud/pop-core');
var client = new Core({
 accessKeyId: '<your-access-key-id>',
 accessKeySecret: '<your-access-key-secret>',
 endpoint: 'https://saf.cn-shanghai.aliyuncs.com',
 apiVersion: '2019-01-28'
});
var params = {
 "RegionId": "cn-shanghai",
 // 产品Service请参考[公共参数]文档中的Service字段描述
 "Service": "购买的产品Service",
 "ServiceParameters": "入参json字符串"
}
var requestOption = {
 method: 'POST'
};
client.request('ExecuteRequest', params, requestOption).then((result) => {
 console.log(JSON.stringify(result));
}, (ex) => {
 console.log(ex);
})
```

七、Ruby SDK

单击此处下载Ruby SDK源码。

Ruby SDK的环境准备、安装使用请参见阿里云SDK开发指南。

执行以下命令安装 Alibaba Cloud Core SDK for Ruby：

```
$ gem install aliyunsdkcore
```

Ruby用例：

```
require 'aliyunsdkcore'
client = RPCClient.new(
 access_key_id:    '<your-access-key-id>',
 access_key_secret: '<your-access-key-secret>',
 endpoint: 'https://saf.cn-shanghai.aliyuncs.com',
 api_version: '2019-01-28'
)
response = client.request(
 action: 'ExecuteRequest',
 params: {
  "RegionId": "cn-shanghai",
  # 产品Service请参考[公共参数]文档中的Service字段描述
  "Service": "购买的产品Service",
  "ServiceParameters": "入参json字符串"
 },
 opts: {
  method: 'POST'
 }
)
print response
```