

智能风控 flutter 接入文档

一、隐私说明

请参照[网易易盾隐私政策](#)，请将易盾隐私政策链接放到应用“用户协议”中。

二、接入说明

接入“智能风控” SDK，开发者需要完成以下步骤：

1. 根据应用/游戏开发平台，将SDK拷贝到指定的工程目录，并修改项目配置；
2. 接入风控SDK必接接口，根据业务需求接入建议接口；
3. 测试风控SDK接入是否正确；
4. 验证风控SDK功能效果；
5. 按业务常规发版流程进行测试与发版

注意：

- 全局必须要接入的接口：[初始化接口](#)
- 用户登陆后必须要接入的接口：[设置用户信息接口](#)
- 关键业务埋点时必须要使用的接口：[同步获取凭证](#)/[异步获取凭证](#)
- 强烈建议接入的接口：[交互接口](#)；
- 建议接入的接口：[心跳接口](#)、[安全通信协议接口](#)。

三、接入步骤

1、导入插件

- 1、将插件文件夹 粘贴到工程目录中（以flutter_demo 为例）。在项目根目录建立 packages（可自定）文件夹，将 plugin 项目移至其中，移动后目录架构如下
- 2、打开项目的 pubspec.yaml，在 dev_dependencies 中添加易盾 plugin 插件
- 3、更新依赖，在 Flutter 项目根目录下执行：Flutter pub get

2、SDK接口调用

(1) 初始化

- 头文件

```
import 'package:htprotect/htprotect.dart';
```

- 初始化接口

```

_httpprotectPlugin.init("易盾业务id", params: params).then((initResult) {
    var code = initResult['code'];
    if (code == kHTPInitSuccessCode) {
        Fluttertoast.showToast(
            msg: "Init successfully", gravity: ToastGravity.CENTER);
    } else {
        String Msg = initResult['Msg'];
        Fluttertoast.showToast(
            msg: 'init failed, Msg is: $Msg',
            gravity: ToastGravity.CENTER);
    }
});

```

ProductId可在易盾官网“智能风控”下的服务管理查询ProductId，或者可在群里咨询技术支持人员。
ProductId示例：YD00713480122211

The screenshot displays the Easym盾 web application interface. The top navigation bar includes links for 'Message', 'Task', 'Fee Center', 'Document', 'Language', and a user profile. The left sidebar contains a menu with options like 'Overview', 'Anti-fraud', 'Service Management', 'Data Center', 'Strategy Center', 'List Management', 'Sample Management', 'Anti-fraud Monitoring', and 'Easym盾 Security Laboratory'. The main content area is titled 'Service Management' and features a '+ Add Product' button. Below this, there are tabs for 'Development Product Test', 'Smart Risk Control YL', '77 Smart Risk Control...', 'Smart Risk Control Development...', 'QA Smart Risk Control Line...', and 'Smart Risk Control Test'. The 'Development Product Test' tab is active, showing a 'Product Information' section with a 'Modify' button. This section displays details for a specific product, with the 'Product ID (businessID)' field highlighted in red. Below the product information, there is a 'Business List' section with a 'View Product Key' link and a 'Switch Business' button. The business list is a table with columns for 'Business ID (businessID)', 'Business Name', 'Platform', 'Engine Type', 'Usage Scenario', 'Status', and 'Creation Time'. Two business entries are listed, both with a 'Test' label and a 'Start' status.

(2) 登录接口

用户登录或者切换游戏账号的时候，调用以下接口

```

String businessId = "易盾提供的业务id";
String roleId = "testroleId";
String roleName = "testroleName";
String roleAccount = "testroleAccount";
String roleServer = "testroleServer";
int serverId = 123;
String gameJson = "{ 'gameVersion': 'ccc', 'assetVersion': 'ddd' }";

_httpprotectPlugin.setRoleInfo(businessId, roleId, roleName, roleAccount,
    roleServer, serverId, gameJson).then((result) {
});

```

3、验证SDK接入是否正确

- 1、确认客户端SDK相关接口已调用
- 2、运行程序，确保已触发相关接口。
- 3、前往[官网](#)查询数据；输入角色信息进行查询。



四、SDK 接入调用说明

1、SDK 初始化接口

接口用途：

用于初始化智能风控 SDK。

接入须知：

- 1. 使用其他接口之前，必须先调用初始化接口，建议在应用/游戏启动过后第一时间调用（初始化后，并不会获取任何个人隐私相关信息）；
- 2. 该接口为 **必须调用接口**。

函数原型：

```
Future<Map<dynamic, dynamic>> init(String productId,{Map<String, dynamic>? params})
```

参数说明：

- 入参说明：

参数	说明	必要性
productId	易盾HTP分配的productId，可登录易盾后台获取	必填

参数	说明	必要性
params.serverType	设置服务器归属地,智能风控默认上报的服务器为中国大陆的服务器,若需要更改服务器归属地,可以调用该接口进行配置,支持的类型如下: serverType = 1,中国大陆地区,默认为该值,无需设置 serverType = 2, 中国台湾地区; serverType = 3 海外地区	可选
channel	设置渠道信息,供用户传入该app的渠道信息(比如: App Store CN/App Store EU)	可选
extData	设置额外信息,在数据上报中包含的额外信息	可选

- 回调参数说明:

回调参数	类型	描述
code	int	初始化结果code码
msg	String	初始化结果详情

示例代码:

```

Map<String,dynamic> params = {};
params['serverType'] = 1;
params['channel'] = 'test';
params['gameKey'] = '8ef016947a1eed4ac5b07c700e2b3f5';
var map1 = {'aa': 'aaa', 'bb': 'bbb'};
params['extData'] = map1;

_hhtprotectPlugin.init(productId, params: params).then((initResult) {
var code = initResult['code'];
if (code == 200) {
    Fluttern.toast.showToast(
        msg: "Init successfully", gravity: ToastGravity.CENTER);
} else {
    String Msg = initResult['Msg'];
    Fluttern.toast.showToast(
        msg: 'init failed, Msg is: $Msg',
        gravity: ToastGravity.CENTER);
}
});

```

2、设置用户信息接口

接口用途：

游戏类型应用，在进行数据采集的过程中，会将角色 ID，角色名称、用户/角色账号等设置在风控 SDK 采集的数据中一同上传，用于表示角色信息，以便对有恶意、风险行为的用户/角色进行相应的处置。

接口须知：

- 1. 须在 [SDK 初始化](#)且 **用户同意隐私政策**后才能调用该接口；
- 2. 凡事涉及到用户登录、或者切换角色的地方，均需要调用该接口设置或者更新用户/玩家信息；

函数原型：

```
Future<int> setRoleInfo(  
    String businessId,  
    String roleId,  
    String roleName,  
    String roleAccount,  
    String roleServer,  
    int serverId,  
    String gameJson)
```

参数说明：

参数	说明	必要性
businessId	当前业务 ID	必填
roleId	用户/玩家的角色 ID； 非游戏类型应用，roleId 可以与 roleAccount 相同	必填
roleName	用户/玩家的角色名称； 非游戏类型应用，roleName 可以是当前用户昵称相同	可选，不需要可填null
roleAccount	用户/玩家的账号； 如业务方同时接入易盾反垃圾，则此账号需与反垃圾接入中的account一致	可选，不需要可填null
roleServer	用户/玩家的角色的服务器名称	可选，不需要可填null
serverId	用户/玩家的角色所属服务器的 ID	可选，不需要可填null
gamejson	游戏类型应用需要上传的信息，对应一个 json 字符串	可选，不需要可填null

gamejson 字段说明：

key 名称	key 类型	key 值	必要性	说明
游戏版本号	String	GameVersion	可选，不需要可填 null	游戏版本号

key 名称	key 类型	key 值	必要性	说明
资源文件版本	String	AssetVersion	可选，不需要可填 null	游戏类型应用的资源文件版本号

示例代码：

```
Future<void> setRoleInfo() async {
String businessId = "31c9823e57b3b789341db4812c562ef1";
String roleId = "testroleId";
String roleName = "testroleName";
String roleAccount = "testroleAccount";
String roleServer = "testroleServer";
int serverId = 123;
String gameJson = '{"gameVersion': 'ccc', 'assetVersion': 'ddd'}";

_hhtprotectPlugin.setRoleInfo(businessId, roleId, roleName, roleAccount,
roleServer, serverId, gameJson).then((result) {
});
}
```

函数返回：

函数返回 int 类型，可能出现的值如下：

值	说明
0	成功
-201	未初始化
-203	businessId 不合法

####

3、登出接口

接口用途：

为了更好的统计该用户/玩家角色在本次登录后的行为，精准标识和打击有恶意行为的用户/玩家。

接入须知：

1. 须在 [SDK 初始化](#) 接口被调用后方可调用，在用户/玩家退出当前角色（包括切换角色）或者退出游戏时调用接口。
2. 此接口主要用于判断用户/玩家本次生命周期，不强制要求接入。
3. 如果未调用初始化，此接口默认为空，不会执行任何逻辑。

函数原型：

```
- (void)logout;
```

示例代码：

```
Future<void> logOut()
```

4、获取凭证

接口用途：

用于关键业务节点风险防控场景中，比如注册、登录、领券、抽卡、兑换、点赞、评论等业务场景。业务方可通过此接口，获取检测唯一凭证 token。业务服务端通过此凭证实时获取当前用户/玩家风险检测结果，并根据结果进行处置。

接口适用场景为：

- 1、游戏类应用，当 SDK 数据上报接口被屏蔽导致数据上报失败时，通过 getToken 接口获取 SDK 采集信息，并由客户服务端传递给易盾服务器；
- 2、全类型应用，在关键业务节点主动调用检测时，通过 getToken 接口获取检测凭证，由客户服务端通过 token 凭证从易盾服务器查询检测结果。

接入须知：

- 1. 须在 SDK 初始化且用户同意隐私政策后才能调用该接口，[网易易盾隐私说明](#)。
- 2. 内部存在网络请求，只允许在子线程上调用。

函数原型：

```
Future<Map<dynamic, dynamic>> getToken(int timeout, String businessId)
```

参数说明：

- 入参说明：

参数	说明	必要性
businessId	业务 ID	必填，由易盾后台分配，并在官网“服务管理”中查询
timeout	函数超时时间	必填，单位毫秒，[1000,10000]，此区间外的默认为300

- 回调参数说明：

回调参数	类型	描述
code	int	结果码，不同结果码含义如下： 201：SDK未初始化； 203：businessId不合法； 204：其他情况
token	String	当 code 为200时，返回正常token。当网络状态不佳或者超时时，会返回离线 token

示例代码：

```
_httpprotectPlugin.getToken(3000,"易盾业务id").then((result) {
    var code = result['code'];
    String token = result['token'];
    Fluttertoast.showToast(
        msg: 'code is $code, token is: $token',
        gravity: ToastGravity.CENTER);

});
```

5、交互接口 (ioctl)

接口用途：

用于在客户端查询 SDK 采集的基础数据、设置需要传递给 SDK 数据（比如初始化配置内容、check 结果信息等）、及其他可能的定制功能服务。

接入须知：

接口调用前置条件是：调用了[初始化接口](#)并且同意隐私政策之后。

函数原型：

```
Future<String> ioctl(int request, String data)
```

参数说明：

Android：

- 入参说明：

参数	说明	必要性
request	1：模拟器名称签名信息 2：是否root 3：本地随机生成的设备ID 7：SDK版本 9：加密后的当前SDK版本 14：SDK运行状态，应对线程被挂起或其他导致无法运行时但是接口正常使用状态 16：初始化配置内容，防止初始化失败情况下，客户可以通过服务端请求初始化配置并传递给SDK 17：客户服务端将check结果传递给SDK，便于执行后续动作	必填
data	设置信息的附加数据	可选，不需要可填null

- 回调参数说明：

回调参数	类型	描述
------	----	----

回调参数	类型	描述
result	String	<p>入参 request = 1 ,result : 返回模拟器名称, 示例: emulator: Nox; 若为未知模拟器, 则返回: emulator: Unknown; 若为真机, 则返回: emulator: None</p> <p>入参 request = 2, result: 当前设备是否root信息。若为root设备, 返回值为: root: 1; 若不是root设备, 返回值为: root: 0</p> <p>入参 request = 3, result: 当前设备id, 比如 "c09cb89662e6719f956f23651be42006"</p> <p>入参 request = 7, result: 风控 SDK 版本号一定不为空, 返回值示例: "5.3.5"</p> <p>入参 request = 9, result: 风控 SDK 版本号一定不为空, 返回值示例: "iF13uXaKVteXVA== 58807f98da0d945b5d60ba4e8f22a428", 解密方式请联系易盾技术支持</p> <p>入参 request = 14, result: 返回值为字符串格式: -1: SDK未启动 0: 运行正常 大于0: 运行异常</p> <p>入参 request = 16, result: {"s":0,"v":33554433} //s: 状态, int类型, 只有为0, v 字段的值才有意义; //v: 版本, long类型(8字节), 配置的版本</p> <p>入参 request = 17, result:返回值为字符串格式: 0或者-1,输入数据格式有误;1为解析成功</p>

IOS:

- 入参说明:

参数	说明	必要性
request	对应向风控系统请求的命令: 0 : 签名信息 2 : 越狱状态 7 : SDK版本 16 : 初始化配置内容, 防止初始化失败情况下, 客户可以通过服务端请求初始化配置并传递给SDK 17: 客户服务端将check结果传递给SDK, 便于执行后续动作 99 : 32位deviceId 100 : 64位deviceId	必填
data	设置信息的附加数据	可选, 不需要可填null

- 回调参数说明:

回调参数	类型	描述
------	----	----

回调参数	类型	描述
result	String	入参 request = 0 ,result : 返回当前应用的签名信息，便于区分是盗版应用还是正版应用 入参 request = 2, result: 当前设备是否越狱信息。若为越狱设备，返回值为：1；若不是越狱设备，返回值为：0 入参 request = 7, result: 风控 SDK 版本号一定不为空，返回值示例：“2.0.1” 入参 request = 16, result: 返回值为字符串格式：0或者-1输入数据格式有误,1解析成功 入参 request = 16, result: 向风控 SDK 配置传递从易盾服务端获取到的命中及响应结果信息，以便 SDK 执行相关处置动作 返回值为字符串格式：0或者-1输入数据格式有误,1解析成功

6、心跳接口

接口用途：

为保障风控 SDK 服务的安全运行，防止易盾风控服务被中止或者被剥离，定时向应用/游戏方客户端反馈风控 SDK 的心跳信息，告知应用/游戏方当前风控 SDK 运行状态，既方便应用/游戏方实时掌握风控服务运行状态，也能保证易盾风控服务正常运行。

同时心跳接口也可以主动实时返回风控数据（有别于交互接口），如需要详细说明文档，请联系易盾获取。

接口须知：

1. 须在 [SDK 初始化](#)且用户同意隐私政策后才能调用该接口；
2. 心跳系统在非主线程中运行，如需进行UI操作请切回到主线程中。

函数原型：

```
void registInfoReceiver()

Future<void> _onCallBack(Map info) async {
  // 心跳回调信息
  // String type = message['type'];
  // String info = message['info'];
  Fluttertoast.showToast(
    msg: '_onInitCallBack: $info', gravity: ToastGravity.CENTER);
}
```

参数说明：

- 回调参数说明：

参数	类型	说明
info	String	返回的心跳数据为默认为加密数据

参数	类型	说明
type	String	type=1 心跳数据，可直接在客户端解析，type=2 加密后的心跳数据，必须发往客户服务端进行解密

解密后参数说明

1. 建议获取加密后的心跳数据，然后发往客户服务端进行解析。
2. 心跳接口是每隔 10s 会返回一次数据（"Info"），返回的数据为一个字符串，“Info”包含三种信息，分别是：序列号、时间戳、网络状态，使用“key:value”格式并用“||”符号分割。信息含义如下所示：

名称	标识	说明	异常情况	是否必有
序列号	seq	从初始值“1”开始递增	序列号不存在，或者乱序	是
时间戳	t	当前时间戳，以秒为单位	时间戳非当前时间	是
网络标识	net	数据发送是否成功（1 为成功，0 为失败）	网络异常，数据发送失败，则网络标识为：0	是

心跳的返回值示例：

- 易盾风控服务网络通信正常时：

```
seq:1||t:1589781840||net:1
```

- 易盾风控服务网络通信异常时：

```
seq:1||t:1589781840||net:0
```

处理建议：

- seq、t和net是否出现，该三个标识为心跳系统一定会返回的值，若无，说明反外挂异常；
- seq的值是否符合“从1开始依次递增”的规则，若不符合，说明返回值被篡改；
- net为0是否长期出现，若5分钟内net的取值都为0，说明网络异常；
- 若出现反外挂异常或网络异常，请结合游戏业务数据综合判断并自行处理。

7、数据校验接口

接口用途：

应用/游戏协议如被破解，若为单机游戏类型应用，可直接影响到存档文件的安全性；若为帧同步游戏类型应用，可通过破解协议修改战斗数据；若为其他游戏类型应用，可被做成脱机协议，用于攻击或制作脱机外挂；若为非游戏类型应用，则可伪造业务数据欺骗客户端应用。为了保障应用/游戏协议安全，易盾风控服务提供“数据校验”方案，采用自研通讯协议校验算法。

接口须知：

1. 客户端接入数据校验接口后，客户服务端需要对对应的数据进行校验，数据校验相关说明文档请联系易盾获取；
2. 对于与服务端有通信的应用/游戏，**强烈建议**使用该接口进行数据校验。

函数原型：

```
Future<String> getDataSign(String inputData, int algIndex)
```

参数说明：

参数名	类型	说明
inputData	String	需要签名的数据

返回值：

返回的数据即为校验结果，开发者需要将原始数据与校验结果发送到应用/游戏服务端，由应用/游戏服务端校验。

示例代码：

```
_httpprotectPlugin.getDataSign(inputData,algIndex).then((result) {  
    Fluttertoast.showToast(  
        msg: 'result is: $result',  
        gravity: ToastGravity.CENTER);  
  
});
```

服务端数据签名计算代码

服务端代码：

```
function getDataSign($data,$appkey)  
{  
  
}
```

校验函数使用示例：

```
$inputdata = "sfsfsfdsfdfssdssdsdf";  
$appkey = 客户的appid;  
$sign = getDataSign($inputdata,$appkey);
```

这里计算生成的\$sign 应该跟客户端返回的 sign 值一致

8、安全通信协议接口

接口用途：

该接口用于对通信数据进行白盒加密和白盒 HMAC 签名（相当于 [数据校验接口](#) 的升级版本），加密和签名的密钥均会被隐藏，攻击者无法获取，从而保障应用/游戏的通信安全。

最新版本接口（V2.1）更新内容如下：

- 自定义二进制格式，增大分析难度；
- 多重加密，防明文传输；
- 数据将进行校验，防止篡改；
- 支持多算法，一旦加密算法被破解，可及时更新为另一种算法；
- 一次一密，每次加密的密钥都不相同；
- 自定义安全随机数发生器，防止系统随机数接口被篡改，始终生成一样的密钥；
- 防重放攻击；
- 防模拟执行(脱离Android/iOS环境运行)。

接口须知：

- 1、如需使用该接口，请联系易盾获取详细说明文档。

(1) 加密数据到服务端

函数原型

```
Future<void> safeCommToServer(int version, int alg, String input, bool isCrucial);
```

参数说明

- 入参说明：

参数	类型	说明
alg	int	加密算法
input	String	需要加密的原文

- 回调参数说明：

参数	类型	说明
msg	String	当code =200 时，msg才存在加密成功的数据
code	int	状态码

示例代码：

```
_httpprotectPlugin.safeCommToServer(version,alg,input,isCrucial).then((result) {

    Fluttern.toast.showToast(
        msg: 'result is: $result',
        gravity: ToastGravity.CENTER);
});
```

(2) 解密服务端的数据

函数原型

```
Future<Map<dynamic, dynamic>> safeCommFromServer(int alg, int timeout, String input)
```

参数说明

- 入参说明：

参数	类型	说明
alg	int	加密算法 如果 alg 不等于 0x0，则使用 alg 来进行解密，即指定算法解密， 否则使用密文里的算法进行解密
timeout	int	超时时间，单位为秒 如果 timeout不为0，则超过timeout的密文会返回nil
input	NSData	需要解密的密文，由服务端返回

- 回调参数说明：

参数	类型	说明
msg	String	当code =200 时，msg才存在加密成功的数据
code	int	状态码

示例代码：

```
_httpprotectPlugin.safeCommFromServer(alg,timeout,input).then((result) {

    Fluttern.toast.showToast(
        msg: 'result is: $result',
        gravity: ToastGravity.CENTER);
});
```