

# 移动智能终端补充设备标识体 系统一调用 SDK

## 开发者说明文档

|        |                 |
|--------|-----------------|
| 编写人    | 移动安全工作委员会       |
| 文档版本   | v2. 1. 0        |
| SDK 版本 | v2. 1. 0        |
| 最新修订日期 | 2023 年 2 月 14 日 |

## 权利声明

任何单位或个人使用移动智能终端补充设备标识体系统一调用 SDK 前，应当仔细阅读本权利声明并确认同意本权利声明所述内容，否则不得使用本 SDK。您理解并承诺，您使用 SDK 的行为，即视为您已了解并完全同意本权利声明的各项内容，且您将基于这些声明承担相应的法律责任与义务。

1、移动智能终端补充设备标识体系统一调用 SDK 由中国信息通信研究院泰尔终端实验室、移动安全工作委员会整合提供，知识产权归中国信息通信研究院所有。任何单位或个人未经授权不得修改、复制、发行、出租、传播或翻译 SDK，不得逆向、破解、篡改、二次打包、公开、贩卖 SDK，不得向任何第三方披露 SDK，不得进行二次分发，不得利用 SDK 直接获利或用于其他商业用途，如基于本 SDK 向第三方提供与本 SDK 相似功能。如有违反中国信息通信研究院保留追究其法律责任的权利。

2、移动智能终端补充设备标识体系依据电信终端产业协会（TAF）移动安全工作委员会推出的技术文件《移动智能终端补充设备标识规范》开发，移动智能终端补充设备标识体系统一调用 SDK 集成设备厂商提供的接口，并已获得相应设备厂商的授权。

3、任何单位或个人使用本 SDK 获取相关标识数据，并欲据此标识数据进行使用、分析、交易或其他行为前，应慎重考虑这些行为的合法性、合规性、真实性和安全性等，并采取相应措施。如因未采取措施所导致的信息泄露、权利侵犯、财产毁损、人身伤亡等违法违规行为及因该等行为所造成的损害后果，与中国信息通信研究院无关，中国信息通信研究院亦不承担任何直接、间接、附带或衍生的损失和责任。

4、移动智能终端补充设备标识体系统一调用 SDK 由中国信息通信研究院泰尔终端实验室、移动安全工作委员会共同负责 SDK 的合规管理和后期维护，移动安全工作委员会官方网站（<http://www.msa-alliance.cn/col.jsp?id=120>）与官方邮箱（[msa@caict.ac.cn](mailto:msa@caict.ac.cn)）是目前唯一合法发布本 SDK 版本、代码、文档的渠道，任何从其他个人、企业或组织渠道获取的 SDK 或其他程序均与中国信息通信研究院泰尔终端实验室、移动安全工作委员会无关，请仔细甄别。

中国信息通信研究院

2020 年 3 月 17 日

## 一. 覆盖范围

| 厂商名称      | 支持版本             |
|-----------|------------------|
| 华为        | HMS 2.6.2 及以上    |
| 小米        | MIUI 10.2 及以上    |
| vivo      | Android 8 及以上    |
| OPPO      | colorOS 3 及以上    |
| 联想        | ZUI 11.4 及以上     |
| 三星        | Android 10 版本及以上 |
| 魅族        | Android 10 版本及以上 |
| 努比亚       | Android 10 版本及以上 |
| 中兴        | Android 10 版本及以上 |
| 华硕        | Android 10 版本及以上 |
| 一加        | Android 10 版本及以上 |
| 黑鲨        | Android 10 版本及以上 |
| 摩托罗拉      | Android 10 版本及以上 |
| Freeme OS | Android 10 版本及以上 |
| 酷赛（铂睿智恒）  | Android 10 版本及以上 |
| Realme    | colorOS 3 及以上    |
| 荣耀        | HMS 2.6.2 及以上    |
| 酷派        | Android 10 版本及以上 |
| 小天才       | Android 10 版本及以上 |
| 360 OS    | Android 6 版本及以上  |

## 二. SDK 获取方式

SDK 官方下载地址：

<http://www.msa-alliance.cn/col.jsp?id=120>

## 三. 调用方法

- 1、把 oaid\_sdk\_x.x.x.aar 拷贝到项目的 libs 目录，并设置依赖，其中 x.x.x 代表版本号。
- 2、将 supplierconfig.json 拷贝到项目 assets 目录下，并修改里边对应内

容，特别是需要设置 appid 的部分。需要设置 appid 的部分需要去对应厂商的应用商店里注册自己的 app。需要注意的是，其中 label 部分内容无需修改，不用增加应用商店。

3、将证书文件（应用包名.cert.pem）拷贝到项目 assets 目录下。

证书需要填写 example\_batch.csv 后发送到 msa@caict.ac.cn 申请，注意每个包名对应一个签名，申请时需要将所需申请的全部包名填写到表格中。

证书有效期一年，为避免证书过期影响 APP 获取 ID，建议证书信息可以从后台服务器获取，或者当调用 oaid SDK 接口提示证书无效时，调用后台接口及时更新证书信息，并且快到期时及时提前重新申请证书。

4、设置依赖

```
implementation files('libs/oaaid_sdk_x.x.x.aar')
```

5、设置 gradle 编译选项，开发者可以根据自己对平台的选择进行合理配置

```
ndk {  
    abiFilters 'armeabi-v7a', 'x86', 'arm64-v8a', 'x86_64'  
}
```

考虑到 sdk 兼容性，sdk 包默认集成了常用 abi 的 so，包括 armeabi-v7a，arm64-v8a，x84，x84\_64 共四种。如果需要减小 SDK 体积，可以使用压缩工具打开 aar 文件，手动删除多余的架构。

6、设置混淆

```
# sdk  
-keep class com.bun.miitmdid.** { *; }  
-keep interface com.bun.supplier.** { *; }  
  
# asus  
-keep class com.asus.msa.SupplementaryDID.** { *; }  
-keep class com.asus.msa.sdid.** { *; }  
# freeme  
-keep class com.android.creator.** { *; }  
-keep class com.android.msasdk.** { *; }  
# huawei  
-keep class com.huawei.hms.ads.identifier.** { *; }  
-keep class com.uodis.opendevice.aidl.** { *; }  
# lenovo  
-keep class com.zui.deviceidservice.** { *; }  
-keep class com.zui.opendeviceidlibrary.** { *; }  
# meizu  
-keep class com.meizu.flyme.openidsdk.** { *; }  
# nubia  
-keep class com.bun.miitmdid.provider.nubia.NubiaIdentityImpl  
# oppo  
-keep class com.heytag.openid.** { *; }  
# samsung  
-keep class com.samsung.android.deviceidservice.** { *; }  
# vivo  
-keep class com.vivo.identifier.** { *; }
```

```
# xiaomi
-keep class com.bun.miitmdid.provider.xiaomi.IdentifierManager
# zte
-keep class com.bun.lib.** { *; }
# coolpad
-keep class com.coolpad.deviceidsupport.** { *; }
```

7、代码调用

a、在 application 类中初始化 SDK 包

```
System.loadLibrary("msaoaidsec");
```

b、加载证书内容

```
MdidSdkHelper.InitCert (Context context, String certContent);
```

c、调用方法获取补充标识符，示例代码详见附件 DemoHelper.java

```
//获取部分 id 信息
int code = MdidSdkHelper.InitSdk(cxt, isSDKLogOn, isGetOAID,
isGetVAID, isGetAAID, IIdentifierListener);

//获取全部 id 信息
int code = MdidSdkHelper.InitSdk(cxt, isSDKLogOn,
IIdentifierListener);
```

8、初始化 sdk 返回的信息码 code

表一 信息码（引用 InfoCode 类）

| 信息                                | 值       | 说明            |
|-----------------------------------|---------|---------------|
| INIT_INFO_RESULT_OK               | 1008610 | 调用成功，获取接口是同步的 |
| INIT_INFO_RESULT_DELAY            | 1008614 | 调用成功，获取接口是异步的 |
| INIT_ERROR_CERT_ERROR             | 1008616 | 证书未初始化或证书无效   |
| INIT_ERROR_MANUFACTURER_NOSUPPORT | 1008611 | 不支持的厂商        |
| INIT_ERROR_DEVICE_NOSUPPORT       | 1008612 | 不支持的设备        |
| INIT_ERROR_LOAD_CONFIGFILE        | 1008613 | 加载配置文件出错      |
| INIT_ERROR_SDK_CALL_ERROR         | 1008615 | sdk 调用出错      |

四. IdSupplier 接口说明

补充设备标识获取接口包括补充设备标识状态获取接口、匿名设备标识符获取接口、开发者匿名设备标识符获取接口、应用匿名设备标识符获取接口。

1、补充设备标识状态获取接口

该接口用于获取移动智能终端是否支持和限制获取补充设备标识，应用需确认支持和不限后，才可以继续获取所需设备标识符。

`public boolean isSupported()`

| 参数 | 返回                     | 说明                   |
|----|------------------------|----------------------|
| 无  | boolean: 是否支持补充设备标识符获取 | true 为支持, false 为不支持 |

`public boolean isLimited()`

| 参数 | 返回                     | 说明                              |
|----|------------------------|---------------------------------|
| 无  | boolean: 匿名设备标识符是否限制获取 | true 为限制应用获取匿名设备标识符, false 为未限制 |

## 2、匿名设备标识符获取接口

`String getOAID()`

| 参数 | 返回                     | 说明   |
|----|------------------------|--|
| 无  | String: 返回匿名设备标识符或空字符串 | 匿名设备标识符最长 64byte, 返回空字符串表示不支持或已限制, 异常状态包括网络异常、appid 异常、应用异常等 |

## 3、开发者匿名设备标识符获取接口

`String getVAID()`

| 参数 | 返回                        | 说明  |
|----|---------------------------|---|
| 无  | String: 返回开发者匿名设备标识符或空字符串 | 开发者匿名设备标识符最长 64byte, 返回空字符串表示不支持, 异常状态包括网络异常、appid 异常、应用异常等 |

## 4、应用匿名设备标识符获取接口

`String getAAID()`

| 参数 | 返回                       | 说明   |
|----|--------------------------|--|
| 无  | String: 返回应用匿名设备标识符或空字符串 | 应用匿名设备标识符最长 64byte, 返回空字符串表示不支持, 异常状态包括网络异常、appid 异常、应用异常等 |

## 五. 使用建议

- 1、调用 InitSdk 后，先检查返回值，如果返回值是不支持的设备或厂商，监听器也不会触发，如果是加载配置文件失败，联系我们的客服。
- 2、VAID/AAID 在初次调用时生成，部分品牌终端生成需要访问网络，请确保网络通畅并可访问公网。
- 3、同一设备中存有多多个同一开发者应用，若需在单个应用卸载时保证 VAID 不被重置，需在应用被卸载前，已有另外同一开发者 ID 的应用也读取过 VAID，否则认定该开发者无需使用 VAID，值将被重置。
- 4、部分厂商，若应用未在其开发者平台后台上架，则认定为非法应用，无法生成 VAID，手机 LOG 中将会有相关异常值输出。
- 5、在用户手机处于弱网、无法访问公网或非法应用情形下频繁调用 VAID 和 AAID 读取接口，部分终端会累计其调用次数，并限时限制其读取。
- 6、由于返回值可能为 null，使用逻辑判断中建议做判空处理。
- 7、进行真机调试时，请确保真机系统时间（日期）正确，否则可能导致证书初始化失败。
- 8、建议按需获取 ID，如 VAID、AAID 等不需要可以不获取，获取 ID 是耗时操作获取不需要的 ID 可能增加耗时风险，甚至导致获取失败（VAID 的获取部分终端涉及网络行为，耗时受网络影响较大）。

## 六. F&Q

有任何问题请务必先查阅《移动智能终端补充设备标识体系统一调用 SDK F&Q》，如仍无法解决，请注明公司、联系人，并提供如下信息，加上 apk 安装包，打包发到 msa@caict.ac.cn。

| 信息名称   | 必/可选 |
|--|------|
| 问题描述   | 必选   |
| SDK 版本号  | 必选   |
| 手机型号   | 必选   |
| 手机安卓版本号  | 必选   |
| App 配置的 minsdkversion,<br>compilesdkversion,<br>targetsdkversion | 必选   |
| APP 支持的 so 平台  | 必选   |
| 奔溃日志（文档形式）   | 必选   |
| APP 安装包  | 可选   |
| 页面截图   | 可选   |
| APP 是否有加固  | 可选   |
| APP 是否有热更新   | 可选   |

## 附录一

### 代码片断 - 调用功能—DemoHelper.java

```
package com.example.oaidtest2;

import android.content.Context;
import android.util.Log;

import com.bun.miitmdid.core.InfoCode;
import com.bun.miitmdid.core.MdidSdkHelper;
import com.bun.miitmdid.interfaces.IIdentifierListener;
import com.bun.miitmdid.interfaces.IdSupplier;
import com.bun.miitmdid.pojo.IdSupplierImpl;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

/**
 * Date: 16:27 2021/2/26 0026
 * Version: 1.0.3
 */
public class DemoHelper implements IIdentifierListener {

    public static final String TAG = "DemoHelper";
    public static final int HELPER_VERSION_CODE = 20230202; // DemoHelper 版本号

    private final AppIdsUpdater appIdsUpdater;
    private boolean isCertInit = false;

    public boolean isSDKLogOn = true; // TODO ( 1 ) 设置 是否开启 sdk 日志
    public static final String ASSET_FILE_NAME_CERT =
"com.example.oaidtest2.cert.pem"; // TODO ( 2 ) 设置 asset 证书文件名

    public DemoHelper(AppIdsUpdater appIdsUpdater){
        // TODO ( 3 ) 加固版本在调用前必须载入 SDK 安全库,因为加载有延迟 , 推荐在 application 中调用 loadLibrary 方法
        // System.loadLibrary("msaoaidsec");
        // DemoHelper 版本建议与 SDK 版本一致
        if(MdidSdkHelper.SDK_VERSION_CODE != HELPER_VERSION_CODE){
            Log.w(TAG,"SDK version not match.");
        }
        this.appIdsUpdater = appIdsUpdater;
    }
}
```



```

public void getDeviceIds(Context cxt){
    getDeviceIds(cxt, true, true, true);
}

/**
 * 获取 OAID
 * @param cxt
 */
public void getDeviceIds(Context cxt,boolean isGetOAID,boolean
isGetVAID,boolean isGetAAID){
    // TODO ( 4 ) 初始化 SDK 证书
    if(!isCertInit){ // 证书只需初始化一次
        // 证书为 PEM 文件中的所有文本内容 ( 包括首尾行、换行符 )
        try {
            isCertInit = MdidSdkHelper.InitCert(cxt, loadPemFromAssetFile(cxt,
ASSET_FILE_NAME_CERT));
        } catch (Error e) {
            e.printStackTrace();
        }
        if(!isCertInit){
            Log.w(TAG, "getDeviceIds: cert init failed");
        }
    }

    // ( 可选 ) 设置 InitSDK 接口回调超时时间(仅适用于接口为异步), 默认值为
5000ms.
    // 注 : 请在调用前设置一次后就不再更改, 否则可能导致回调丢失、重复等
问题
    try {
        MdidSdkHelper.setGlobalTimeout(5000);
    } catch (Error error) {
        error.printStackTrace();
    }
    int code = 0;
    // TODO ( 5 ) 调用 SDK 获取 ID
    try {
        code = MdidSdkHelper.InitSdk(cxt, isSDKLogOn, isGetOAID, isGetVAID,
isGetAAID, this);
    } catch (Error error) {
        error.printStackTrace();
    }

    // TODO ( 6 ) 根据 SDK 返回的 code 进行不同处理

```

```

        IdSupplierImpl unsupportedIdSupplier = new IdSupplierImpl();
        if(code == InfoCode.INIT_ERROR_CERT_ERROR){           // 证书未初始
        化或证书无效，SDK 内部不会回调 onSupport
            // APP 自定义逻辑
            Log.w(TAG,"cert not init or check not pass");
            onSupport(unsupportedIdSupplier);
        }else if(code == InfoCode.INIT_ERROR_DEVICE_NOSUPPORT){ // 不支持
        的设备，SDK 内部不会回调 onSupport
            // APP 自定义逻辑
            Log.w(TAG,"device not supported");
            onSupport(unsupportedIdSupplier);
        }else if( code == InfoCode.INIT_ERROR_LOAD_CONFIGFILE){ // 加载
        配置文件出错，SDK 内部不会回调 onSupport
            // APP 自定义逻辑
            Log.w(TAG,"failed to load config file");
            onSupport(unsupportedIdSupplier);
        }else if(code ==
        InfoCode.INIT_ERROR_MANUFACTURER_NOSUPPORT){ // 不支持的设备厂
        商，SDK 内部不会回调 onSupport
            // APP 自定义逻辑
            Log.w(TAG,"manufacturer not supported");
            onSupport(unsupportedIdSupplier);
        }else if(code == InfoCode.INIT_ERROR_SDK_CALL_ERROR){ // sdk 调
        用出错，SSDK 内部不会回调 onSupport
            // APP 自定义逻辑
            Log.w(TAG,"sdk call error");
            onSupport(unsupportedIdSupplier);
        } else if(code == InfoCode.INIT_INFO_RESULT_DELAY) { // 获取接口
        是异步的，SDK 内部会回调 onSupport
            Log.i(TAG, "result delay (async)");
        }else if(code == InfoCode.INIT_INFO_RESULT_OK){ // 获取接口是
        同步的，SDK 内部会回调 onSupport
            Log.i(TAG, "result ok (sync)");
        }else {
            // sdk 版本高于 DemoHelper 代码版本可能出现的情况，无法确定是否调用
            onSupport
            // 不影响成功的 OAID 获取
            Log.w(TAG,"getDeviceIds: unknown code: " + code);
        }
    }
}

```

```

/**
 * APP 自定义的 getDeviceIds(Context cxt)的接口回调
 * @param supplier
 */
@Override
public void onSupport(IdSupplier supplier) {
    if(supplier==null) {
        Log.w(TAG, "onSupport: supplier is null");
        return;
    }
    if(appIdsUpdater ==null) {
        Log.w(TAG, "onSupport: callbackListener is null");
        return;
    }
    // 获取 Id 信息

    // 注：IdSupplier 中的内容为本次调用 MdidSdkHelper.InitSdk()的结果，不会
    实时更新。如需更新，需调用 MdidSdkHelper.InitSdk()
    boolean isSupported = supplier.isSupported();
    boolean isLimited = supplier.isLimited();
    String oaid=supplier.getOAID();
    String vaid=supplier.getVAID();
    String aaid=supplier.getAAID();

    //TODO (7) 自定义后续流程，以下显示到 UI 的示例
    String idsText= "support: " + (isSupported ? "true" : "false") +
        "\nlimit: " + (isLimited ? "true" : "false") +
        "\nOAID: " + oaid +
        "\nVAID: " + vaid +
        "\nAAID: " + aaid + "\n";
    Log.d(TAG, "onSupport: ids: \n" + idsText);
    appIdsUpdater.onIdsValid(idsText);
}

public interface AppIdsUpdater{
    void onIdsValid(String ids);
}

/**
 * 从 asset 文件读取证书内容
 * @param context
 * @param assetFileName
 * @return 证书字符串
 */
public static String loadPemFromAssetFile(Context context, String assetFileName){
    try {
        InputStream is = context.getAssets().open(assetFileName);

```

```
        BufferedReader in = new BufferedReader(new InputStreamReader(is));
        StringBuilder builder = new StringBuilder();
        String line;
        while ((line = in.readLine()) != null){
            builder.append(line);
            builder.append("\n");
        }
        return builder.toString();
    } catch (IOException e) {
        Log.e(TAG, "loadPemFromAssetFile failed");
        return "";
    }
}
```