

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	移动安全联盟
文档版本	v1.10
SDK 版本	v1.0.10
最新修订日期	2019 年 10 月 12 日

权利声明

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、移动安全联盟整合提供，知识产权归中国信通院所有，未经授权或非法复制、逆向、破解、篡改、贩卖或用于其他商业用户，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、移动安全联盟共同负责 SDK 的合规管理和后期维护；

3、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）、移动安全联盟（MSA）联合推出的团体标准《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，并获得主流设备厂商的授权，本次版本为试用版。

一. 覆盖范围

厂商	版本
小米	MIUI10.2 及以上
vivo	FuntouchOS 9 及以上
华为	全版本
OPPO	Color OS 7.0 及以上（9 月份正式支持）
Lenovo	ZUI 11.4 及以上（9 月中旬正式支持）
华硕	Android Q（10 月份会正式支持）
三星	
魅族	
nubia	
华硕	

二. SDK 获取方式

MSA 统一 SDK 下载地址：

移动安全联盟官网：<http://www.msa-alliance.cn/>

三. 调用方法

1、把 miit_mdid_x.x.x.aar 拷贝到项的 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。

3、设置依赖

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

4、混淆设置

```
-keep class com.bun.miitmdid.core.** {*;}
```

5、设置 gradle 编译选项，这块可以根据自己对平台的选择进行合理配置

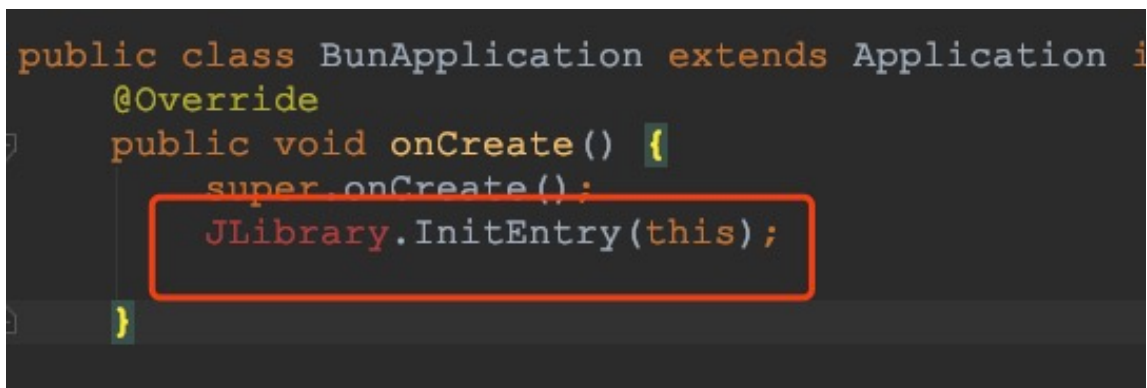
```
ndk {
    abiFilters 'armeabi-v7a','x86','arm64-v8a','x86_64','armeabi'
}
packagingOptions {
```

```
doNotStrip "*/armeabi-v7a/*.so"  
doNotStrip "*/x86/*.so"  
doNotStrip "*/arm64-v8a/*.so"  
doNotStrip "*/x86_64/*.so"  
doNotStrip "armeabi.so"  
}
```

6、代码调用

a、初始化 sdk

在应用的 **application** 的 **onCreate** 中方法调用方法：



```
public class BunApplication extends Application {  
    @Override  
    public void onCreate() {  
        super.onCreate();  
        JLibrary.InitEntry(this);  
    }  
}
```

JLibrary.InitEntry(base);

b、获取设备 ID，有两种方法调用，事例代码详见附件 MiiHelper.java

b1：直接调用，由于安卓 9 以后类加载的限制，所以安卓 9 以后不建议采用，如需使用，需要做 **MdidSdk.InitSdk** 和 **JLibrary.InitEntry()** 不能在一个类里，还要注意包含这两个调用的类不能有依赖关系。

```
private int DirectCall(Context cxt){  
    MdidSdk sdk = new MdidSdk();  
    return sdk.InitSdk(cxt, this);  
}
```

b2：反射调用，它的好处是 android 版本号无关，缺点是调用时间估计会长，经过测试跟直接调用在用时上区别不大，用反射调用就是为了

```
private int CallFromReflect(Context cxt){  
    return MdidSdkHelper.InitSdk(cxt, true, this);  
}
```

c、初始化 sdk 返回的错误码

表一、错误信息，引用 ErrorCode 类

错误信息	值	说明
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	加载配置文件失败
INIT_ERROR_RESULT_DELAY	1008614	信息将会延迟返回，获取数据可能在异步线程，取决于设备
INIT_HELPER_CALL_ERROR	1008615	反射调用失败

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和关闭接口。

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补充设备标识体系，确认支持后，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

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 64 位，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

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 64 位，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 64 位，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

5、关闭接口

String shutDown()

参数	返回	说明
无	void	有些厂商使用 bind service 实现的，所以使用完 IdSupplier 对象之后，要调用该方法释放连接

五. 使用建议

1、调用 MdidSdk 的 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，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，则认定未非法应用，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，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、若有表一中的异常出现，会有相关 Log 打出。

8、INIT_ERROR_DEVICE_NOSUPPORT 和 INIT_HELPER_CALL_ERROR 这两个暂时不会走回调，后续会调整。

六. F&Q

1. udid 能否获取？

回复：udid 不对开发者开放，可以使用 oaid 来做为唯一标识

2. 配置文件如何修改

回复：目前只需要设置 vivo 的 appid，放到 assets 目录下即

3. 小米手机异常， java.lang.NoSuchMethodException: getDefaultUDID [class android.content.Context]

回复：这个只是输出的一个日志，关闭日志，自然就没有了，不会引起系统崩溃。放心使用。

4. 如何调用 JLibrary.InitEntry()

回复：首先检查一个调用这个 api 的位置，如果是 application 中，请检查是否在 AndroidManifest.xml 中注册了这个 application. 如果在其他模块中。请单步调试，跟踪一下，是不是因为其他的分支没有执行该调用。

5. 产生崩溃解决办法

把 sdk 版本号，崩溃手机型号，崩溃手机版本号收集一下，然后加上 apk，打包给我们发过来。

6. 关于 INIT_ERROR_RESULT_DELAY(1008614) 错误码

这个错误码和 0 是一样的，都表示成功了。区别在于回调的执行线程不同，返回 0 表示回调在调用线程返回，返回 1008614 表示在工作线程中执行回调。

7. 反馈问题需提供数据

1. 提供 msa sdk 版本号，2. 手机厂商和型号 3. android 版本号 4. 异常或崩溃日志。

有任何疑问可发送邮件至 msa@caict.ac.cn。



附录一

代码片断 - 初始化 sdk

```
public class BunApplication extends Application {  
    @Override  
    public void onCreate() {  
        super.onCreate();  
    }  
  
    @Override  
    protected void attachBaseContext(Context base) {  
        super.attachBaseContext(base);  
        JLibrary.InitEntry(base);  
    }  
}
```


代码片断 - 调用功能—MiitHelper.java

```
public class MiitHelper implements IIdentifierListener {

    private AppIdsUpdater _listener;

    public MiitHelper(AppIdsUpdater callback) {
        _listener=callback;
    }

    public void getDeviceIds(Context cxt){
        long timeb=System.currentTimeMillis();
        int nres = CallFromReflect(cxt);
        //      int nres=DirectCall(cxt);
        long timee=System.currentTimeMillis();
        long offset=timee-timeb;
        if(nres == ErrorCode.INIT_ERROR_DEVICE_NOSUPPORT) { //不支持
的设备

            }else if( nres == ErrorCode.INIT_ERROR_LOAD_CONFIGFILE) { //
加载配置文件出错

            }else if(nres ==
ErrorCode.INIT_ERROR_MANUFACTURER_NOSUPPORT) { //不支持的设备厂商

            }else if(nres == ErrorCode.INIT_ERROR_RESULT_DELAY) { //获取
接口是异步的，结果会在回调中返回，回调执行的回调可能在工作线程

            }else if(nres == ErrorCode.INIT_HELPER_CALL_ERROR) { //反射调
用出错

        }
    }
}
```

```
Log.d(getClass().getSimpleName(), "return value:
"+String.valueOf(nres));

}

/*
 * 通过反射调用，解决 android 9 以后的类加载升级，导致找不到 so 中
的方法
 *
 * */
private int CallFromReflect(Context cxt) {
    return MdidSdkHelper.InitSdk(cxt, true, this);
}

/*
 * 直接 java 调用，如果这样调用，在 android 9 以前没有题，在
android 9 以后会抛找不到 so 方法的异常
 * 解决办法是和 JLibrary.InitEntry(cxt)，分开调用，比如在 A 类中
调用 JLibrary.InitEntry(cxt)，在 B 类中调用 MdidSdk 的方法
 * A 和 B 不能存在直接和间接依赖关系，否则也会报错
 *
 * */
private int DirectCall(Context cxt) {
    MdidSdk sdk = new MdidSdk();
    return sdk.InitSdk(cxt, this);
}

@Override
public void OnSupport(boolean isSupport, IdSupplier _supplier)
{
    if(_supplier==null) {
        return;
    }
}
```

```
}  
String oaid=_supplier.getOAID();  
String vaid=_supplier.getVAID();  
String aaid=_supplier.getAAID();  
String udid=_supplier.getUDID();  
StringBuilder builder=new StringBuilder();  
builder.append("support:"  
").append(isSupport?"true":"false").append("\n");  
builder.append("UDID: ").append(udid).append("\n");  
builder.append("OAID: ").append(oaid).append("\n");  
builder.append("VAID: ").append(vaid).append("\n");  
builder.append("AAID: ").append(aaid).append("\n");  
String idstext=builder.toString();  
_supplier.shutdown();  
if(_listener!=null){  
    _listener.OnIdsAvalid(idstext);  
}  
}  
  
public interface AppIdsUpdater{  
    void OnIdsAvalid(@NonNull String ids);  
}
```