IminLibs接入文档

背景

此专栏主题旨在于帮助购买了iMin的POS机行业应用开发者，快速适配iMin的设备，相比之前的SDK，我们做了全新的升级，做到一次适配，后续移植到其他iMin的设备或者版本升级，无须重新适配

1.快速集成SDK

方式一：

- 在settings.gradle 添加

```
dependencyResolutionManagement {
        repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
        repositories {
            mavenCentral()
            maven { url 'https://jitpack.io' }
        }
    }
```

- 在app/build.gradle中增加以下代码

```
implementation 'com.github.iminsoftware:IminDeviceLibrary:3.0.0'
```

- 完成后，重新build项目。

方式二：

- 将IminLibs.jar文件放在libs目录下
- 在app/build.gradle中增加以下代码，引入jar

```
implementation files("libs/IminLibs.jar")
```

- 导入jar包完成后，重新build项目。

2.初始化SDK

2.1初始化

复制代码

```
//在项目Application中初始化
DeviceManager.initialize(this);
```

2.2获取组件管理器

复制代码

```
//获取组件管理器
DeviceManager mDeviceManager = DeviceManager.getDeviceManager(this);
```

2.3 判断初始化是否成功

复制代码

```
if(mDeviceManager.isInitialized()){
  //初始化成功后才能调用相关功能
}
```

3.API列表

3.1 获取设备信息相关API

3.1.1 获取设备型号

方法	void getDeviceInfoAsyn(String name, IAsyncCallback callback)
参数	name：DEVICE_INFO_GET_MODEL
示例	复制代码 `mDeviceManager.getDeviceInfoAsyn("DEVICE_INFO_GET_MODEL", new IAsyncCallback.Stub() {` `  @Override`

	public void onResult(String result) throws RemoteException { Log.d(TAG, result); }
返回参数	result:（eg: I24T02）

3.1.2 获取设备的硬件平台

方法	void getDeviceInfoAsyn(String name, IAsyncCallback callback)
参数	name：DEVICE_INFO_GET_PLATFORM
示例	复制代码 mDeviceManager.getDeviceInfoAsyn("DEVICE_INFO_GET_PLATFORM", new IAsyncCallback.Stub() { @Override public void onResult(String result) throws RemoteException { Log.d(TAG, result); } });
返回参数	result:（eg:mt8781）

3.1.3 获取设备品牌

方法	void getDeviceInfoAsyn(String name, IAsyncCallback callback)
参数	name：DEVICE_INFO_GET_BRAND
示例	复制代码

	mDeviceManager.getDeviceInfoAsyn("DEVICE_INFO_GET_BRAND", new IAsyncCallback.Stub() { @Override public void onResult(String result) throws RemoteException { Log.d(TAG, result); } });
返回参数	result: iMin

3.1.4 获取设备SN

方法	void getDeviceInfoAsyn(String name, IAsyncCallback callback)
参数	name：DEVICE_INFO_GET_SN
示例	复制代码 mDeviceManager.getDeviceInfoAsyn("DEVICE_INFO_GET_SN", new IAsyncCallback.Stub() { @Override public void onResult(String result) throws RemoteException { Log.d(TAG, result); } });
返回参数	result: (eg:NTF212CL1ENGF0042)

3.1.5 获取设备是否为双屏

方法	void getDeviceInfoAsyn(String name, IAsyncCallback callback)
参数	name：DEVICE_INFO_GET_DUALSCREEN
示例	复制代码 ``` mDeviceManager.getDeviceInfoAsyn("DEVICE_INFO_GET_DUALSCREEN", new IAsyncCallback.Stub() { @Override public void onResult(String result) throws RemoteException { Log.d(TAG, result); } }); ```
返回参数	result: true: 双屏设备 false: 非双屏

3.1.6 获取硬件设备信息

方法	void getDeviceInfoAsyn(String name, IAsyncCallback callback)
参数	name：DEVICE_INFO_HW_INFO callback：返回数据 参考 **HardwareInfo** 类
示例	复制代码 ``` mDeviceManager.getDeviceInfoAsyn("DEVICE_INFO_HW_INFO", new IAsyncCallback.Stub() { @Override public void onResult(String result) throws RemoteException { Log.d(TAG, result); ```

	} });
返回参数	result:返回数据参考 **HardwareInfo** 类 eg: {"brand":"iMin","deviceBasebandVersion":"35","deviceName":"Falcon 2","enterpriseSpecificId":"NTF212CL1ENGF0042","hardware":"mt8781","manufacturer":"iMin Technology","model":"I24T02","serialNumber":"NTF212CL1ENGF0042"}

3.1.7 获取网络信息相关信息

方法	void getDeviceInfoAsyn(String name, IAsyncCallback callback)
参数	name：DEVICE_INFO_NETWORK_INFO callback：返回数据 参考 **NetworkInfo** 类
API限制	Android 15
示例	复制代码 `mDeviceManager.getDeviceInfoAsyn("DEVICE_INFO_NETWORK_INFO", new IAsyncCallback.Stub() {` `@Override` `public void onResult(String result) throws RemoteException {` `Log.d(TAG, result);` `}` `});`
返回参数	result:返回数据参考 **NetworkInfo** 类 eg： {"imei":"353968870000925","telephonyInfos":[],"wifiIP":"10.0.21.49","wifiMacAddress":"d0:c8:57:4c:2b:8f

	"}

3.1.8 获取设备软件的相关信息

方法	void getDeviceInfoAsyn(String name, IAsyncCallback callback)
参数	name：DEVICE_INFO_SW_INFO callback：返回数据 参考 **SoftwareInfo** 类
API限制	Android 15
示例	复制代码 `mDeviceManager.getDeviceInfoAsyn("DEVICE_INFO_SW_INFO", new IAsyncCallback.Stub() {` `@Override` `public void onResult(String result) throws RemoteException {` `Log.d(TAG, result);` `}` `});`
返回参数	result:返回数据参考 **SoftwareInfo** 类 eg: {"androidBuildNumber":"1.0.0.10.37_GMS_251229-debug","androidBuildTime":"2025-12-29 13:28:11","androidDevicePolicyVersionCode":0,"androidDevicePolicyVersionName":"NA","androidVersion":"15","bootloaderVersion":"unknown","buildType":"userdebug","deviceBuildSignature":"iMin/Falcon2/Falcon2:15/AP3A.240905.015.A2/20251225:userdebug/release-keys","deviceKernelVersion":"5.10.233-android12-9-gf39f075f53fe","primaryLanguageCode":"zh","securityPatchLevel":"2025-11-05","supportedAbis":["arm64-v8a","armeabi-v7a","armeabi"]}

3.1.9 获取设备显示的相关信息

方法	void getDeviceInfoAsyn(String name, IAsyncCallback callback)
参数	name：DEVICE_INFO_DISPLAY_INFO callback：返回数据 参考 List<**DisplayInfo**> 类
API限制	Android 15
示例	复制代码 `mDeviceManager.getDeviceInfoAsyn("DEVICE_INFO_DISPLAY_INFO", new IAsyncCallback.Stub() {` `  @Override` `  public void onResult(String result) throws RemoteException {` `  Log.d(TAG, result);` `  }` `});`
返回参数	result:返回数据参考 List<**DisplayInfo**> 类 eg: [{"density":1.75,"displayId":0,"height":1200,"name":"内置屏幕","refreshRate":66.0,"state":2,"width":1920}]

3.1.10 获取设备内存和存储空间的信息

方法	void getDeviceInfoAsyn(String name, IAsyncCallback callback)
参数	name：DEVICE_INFO_MEMORY_INFO callback：返回数据 参考 **MemoryInfo** 类
API限制	Android 15
示例	复制代码

	mDeviceManager.getDeviceInfoAsyn("DEVICE_INFO_MEMORY_INFO", new IAsyncCallback.Stub() { @Override public void onResult(String result) throws RemoteException { Log.d(TAG, result); }
返回参数	result:返回数据参考 **MemoryInfo** 类 eg: {"totalInternalStorage":"47 GB","totalRam":"3684 MB"}

3.1.11 获取设备内存的详细信息

方法	void getDeviceInfoAsyn(String name, IAsyncCallback callback)
参数	name：DEVICE_INFO_MEMORY_DETIAL_INFO callback：返回数据 参考 **MemoryDetail** 类
API限制	Android 15
示例	复制代码 mDeviceManager.getDeviceInfoAsyn("DEVICE_INFO_MEMORY_DETIAL_INFO", new IAsyncCallback.Stub() { @Override public void onResult(String result) throws RemoteException { Log.d(TAG, result); } });
返回参数	result:返回数据参考 **MemoryDetail** 类 eg: {"availableMemory":"1524113408","buffers":"2097152","ca

	ched":"1456619520","freeMemory":"67440640","shmem":"33366016","sreclaimable":"118153216","swap":"1017286656","swapFree":"1092165632","swapTotal":"2109452288","totalMemory":"4294967296","usedMemory":"2770853888"}

3.1.12 获取设备存储空间的详细信息

方法	void getDeviceInfoAsyn(String name, IAsyncCallback callback)
参数	name：DEVICE_INFO_STORAGE_INFO callback：返回数据参考 **StorageDetail** 类
API限制	Android 15
示例	复制代码 `mDeviceManager.getDeviceInfoAsyn("DEVICE_INFO_STORAGE_INFO", new IAsyncCallback.Stub() {` `  @Override` `  public void onResult(String result) throws RemoteException {` `  Log.d(TAG, result);` `  }` `});`
返回参数	result:返回数据参考 **StorageDetail** 类 eg: {"availableStorage":"44361146368","id":"41217664-9172-527a-b3d5-edabb50a7d69","totalStorage":"68719476736","usedStorage":"21088289423","volumes":[{"diskId":"","fsType":"f2fs","fsUuid":"41217664-9172-527a-b3d5-edabb50a7d69","path":"/storage/emulated/0","state":"mounted","type":"external_primary"}]}

3.1.13 获取cpu的相关信息

方法	void getDeviceInfoAsyn(String name, IAsyncCallback callback)
参数	name：DEVICE_INFO_CPU_INFO callback：返回数据参考 **CPUDetail**类
示例	复制代码 `mDeviceManager.getDeviceInfoAsyn("DEVICE_INFO_CPU_INFO", new IAsyncCallback.Stub() {` `@Override` `public void onResult(String result) throws RemoteException {` `Log.d(TAG, result);` `}` `});`
返回参数	result: 返回数据参考 **CPUDetail** 类 eg: {"architecture":"arm64-v8a","cores":"8","cpu":[{"frequency":"700000","frequencyList":"2000000 1900000 1800000 1700000 1600000 1500000 1450000 1400000 1350000 1300000 1250000 1200000 1150000 1100000 1050000 1000000 950000 900000 850000 800000 750000 700000 650000 500000","maxfreq":"2000000","minfreq":"500000","mode":"MT8781V/CA","modeList":"MT8781V/CA","online":"1"},{"frequency":"2000000","frequencyList":"2000000 1900000 1800000 1700000 1600000 1500000 1450000 1400000 1350000 1300000 1250000 1200000 1150000 1100000 1050000 1000000 950000 900000 850000 800000 750000 700000 650000 500000","maxfreq":"2000000","minfreq":"500000","mode":"MT8781V/CA","modeList":"MT8781V/CA","online":"1"},{"frequency":"2000000","frequencyList":"2000000 1900000 1800000 1700000 1600000 1500000 1450000 1400000 1350000 1300000 1250000 1200000 1150000

1100000 1050000 1000000 950000 900000 850000 800000 750000 700000 650000
500000","maxfreq":"2000000","minfreq":"500000","mode":"MT8781V/CA","modeList":"MT8781V/CA","online":"1"},{"frequency":"950000","frequencyList":"2000000 1900000 1800000 1700000 1600000 1500000 1450000 1400000 1350000 1300000 1250000 1200000 1150000 1100000 1050000 1000000 950000 900000 850000 800000 750000 700000 650000
500000","maxfreq":"2000000","minfreq":"500000","mode":"MT8781V/CA","modeList":"MT8781V/CA","online":"1"},{"frequency":"500000","frequencyList":"2000000 1900000 1800000 1700000 1600000 1500000 1450000 1400000 1350000 1300000 1250000 1200000 1150000 1100000 1050000 1000000 950000 900000 850000 800000 750000 700000 650000
500000","maxfreq":"2000000","minfreq":"500000","mode":"MT8781V/CA","modeList":"MT8781V/CA","online":"1"},{"frequency":"500000","frequencyList":"2000000 1900000 1800000 1700000 1600000 1500000 1450000 1400000 1350000 1300000 1250000 1200000 1150000 1100000 1050000 1000000 950000 900000 850000 800000 750000 700000 650000
500000","maxfreq":"2000000","minfreq":"500000","mode":"MT8781V/CA","modeList":"MT8781V/CA","online":"1"},{"frequency":"1300000","frequencyList":"2200000 2100000 2000000 1900000 1800000 1700000 1600000 1500000 1400000 1300000 1200000 1100000 1000000 900000 800000
725000","maxfreq":"2200000","minfreq":"725000","mode":"MT8781V/CA","modeList":"MT8781V/CA","online":"1"},{"frequency":"1500000","frequencyList":"2200000 2100000 2000000 1900000 1800000 1700000 1600000 1500000 1400000 1300000 1200000 1100000 1000000 900000 800000
725000","maxfreq":"2200000","minfreq":"725000","mode":"MT8781V/CA","modeList":"MT8781V/CA","online":"1"}],"cpuTemperatureAlert":34.2,"manufacturer":"Mediatek","model":"MT8781V/CA","revision":"0","totalCPU":"2.2GHz","usedCPU":"24%"}

3.2 设备操作相关API

3.2.1 设置蓝牙名称

方法	sendAMCommandAsyn(String json, IAsyncCallback callback)
功能	设置蓝牙名称
API限制	Android 15
参数	json： {"**oemConfig**":{ "**setBTName**":"bt11" }} callback： 回调 备注： setBTName：String 蓝牙名称
示例	复制代码 `JsonObject controlBean = new JsonObject();` `JsonObject oemConfig = new JsonObject();` `controlBean.add("oemConfig", oemConfig);` `oemConfig.addProperty("setBTName", "bt11");` `try {` `mDeviceManager.sendAMCommandAsyn(new Gson().toJson(controlBean), new IAsyncCallback.Stub() {` `@Override` `public void onResult(String result) throws RemoteException {` `Log.d(TAG, result);` `}` `});` `} catch (RemoteException e) {`

```
            e.printStackTrace();
        }
```

3.2.2 设置虚拟蓝牙名称

方法	sendAMCommandAsyn(String json, IAsyncCallback callback)
功能	设置虚拟蓝牙名称
API限制	Android 15
参数	json： {"**oemConfig**":{ "**setDeviceVirtualBluetoothName**":true }} callback： 回调 备注： setDeviceVirtualBluetoothName：String 名称
示例	复制代码 `JsonObject controlBean = new JsonObject();` `JsonObject oemConfig = new JsonObject();` `controlBean.add("oemConfig", oemConfig);` `oemConfig.addProperty("setDeviceVirtualBluetoothName", "testbt");` `try {` `mDeviceManager.sendAMCommandAsyn(new Gson().toJson(controlBean), new IAsyncCallback.Stub() {` `@Override` `public void onResult(String result) throws RemoteException {` `Log.d(TAG, result);`

```
                }
            });
        } catch (RemoteException
e) {
            e.printStackTrace();
```

3.2.3设置屏幕超时

方法	sendAMCommandAsyn(String json, IAsyncCallback callback)
功能	设置屏幕超时
API限制	Android 15
参数	json： {"oemConfig":{ "screenTimeout":"2000" }} callback： 回调 备注： screenTimeout： 超时时间 以秒为单位
示例	复制代码 `JsonObject controlBean = new JsonObject();` `JsonObject oemConfig = new JsonObject();` `controlBean.add("oemConfig", oemConfig);` `oemConfig.addProperty("screenTimeout", "2000");` `try {` `mDeviceManager.sendAMCommandAsyn(new Gson().toJson(controlBean), new IAsyncCallback.Stub() {` `@Override` `public void onResult(String result) throws RemoteException {` `Log.d(TAG, result);` `}` `});` `} catch (RemoteException e) {`

```
            e.printStackTrace();
        }
```

3.2.4 设置浮动窗口权限

方法	sendAMCommandAsyn(String json, IAsyncCallback callback)
功能	设置浮动窗口权限
API限制	Android 15
参数	json： {"oemConfig":{ "setAppsHaveAlertWindowPermiss":"com.android.chrome"}} callback： 回调 备注：
示例	复制代码

```
JsonObject controlBean = new JsonObject();
JsonObject oemConfig = new JsonObject();
controlBean.add("oemConfig", oemConfig);
oemConfig.addProperty("setAppsHaveAlertWindowP
ermiss", "com.android.chrome");
try {
    mDeviceManager.sendAMCommandAsyn(new
Gson().toJson(controlBean), new
IAsyncCallback.Stub() {
    @Override
    public void onResult(String result)
throws RemoteException {
                    Log.d(TAG, result);
                }
            });
        } catch (RemoteException e) {
            e.printStackTrace();
        }
```

3.2.5 打开钱箱

方法	sendAMCommandAsyn(String json, IAsyncCallback callback)
功能	打开钱箱
参数	json： {"peripheralConfig":{"openCashBox":true}} callback： 回调 备注： openCashBox设置true 打开
示例	复制代码 `JsonObject controlBean = new JsonObject();` `JsonObject peripheral = new JsonObject();` `controlBean.add("peripheralConfig", peripheral);` `peripheral.addProperty("openCashBox", true);` `try {` `mDeviceManager.sendAMCommandAsyn(new Gson().toJson(controlBean), new IAsyncCallback.Stub() {` `@Override` `public void onResult(String result) throws RemoteException {` `Log.d(TAG, result);` `}` `});` `} catch (RemoteException e) {` `e.printStackTrace();` `}`

3.2.6 设置主屏背光亮度

方法	sendAMCommandAsyn(String json, IAsyncCallback callback)
功能	设置主屏背光亮度

API限制	Android 15
参数	json： {"peripheralConfig":{"screenBrightness":"3"}} callback： 回调 备注： screenBrightness 1-255
示例	复制代码 `JsonObject controlBean = new JsonObject();` `JsonObject peripheral = new JsonObject();` `controlBean.add("peripheralConfig", peripheral);` `peripheral.addProperty("screenBrightness", "3");` `try {` `mDeviceManager.sendAMCommandAsyn(new Gson().toJson(controlBean), new IAsyncCallback.Stub() {` `@Override` `public void onResult(String result) throws RemoteException {` `Log.d(TAG, result);` `}` `});` `} catch (RemoteException e) {` `e.printStackTrace();` `}`

3.2.7 副屏背光亮度

方法	sendAMCommandAsyn(String json, IAsyncCallback callback)
功能	设置副屏背光亮度
API限制	Android 15
参数	json： {"peripheralConfig":{

	"subScreenBrightness":"3" }} callback： 回调 备注: subScreenBrightness 1-255
示例	复制代码 `JsonObject controlBean = new JsonObject();` `JsonObject peripheral = new JsonObject();` `controlBean.add("peripheralConfig", peripheral);` `peripheral.addProperty("subScreenBrightness", "3");` `try {` `mDeviceManager.sendAMCommandAsyn(new Gson().toJson(controlBean), new IAsyncCallback.Stub() {` `@Override` `public void onResult(String result) throws RemoteException {` `Log.d(TAG, result);` `}` `});` `} catch (RemoteException e) {` `e.printStackTrace();` `}`

3.2.8 设置WIFI开关

方法	sendAMCommandAsyn(String json, IAsyncCallback callback)
功能	打开wifi开关
参数	json：{"oemConfig":{"setWifiSwitch":true}} callback： 回调 备注: setWifiSwitch：**true** 打开

	setWifiSwitch：**false** 关闭 传参可参考OemConfig表
示例	复制代码 `JsonObject controlBean = new JsonObject();` `JsonObject oemConfig = new JsonObject();` `controlBean.add("oemConfig", oemConfig);` `oemConfig.addProperty("setWifiSwitch", true);` `try {` `mDeviceManager.sendAMCommandAsyn(new Gson().toJson(controlBean), new IAsyncCallback.Stub() {` `@Override` `public void onResult(String result) throws RemoteException {` `Log.d(TAG, result);` `}` `});` `} catch (RemoteException e) {` `e.printStackTrace();` `}`

3.2.9 设置蓝牙开关

方法	sendAMCommandAsyn(String json, IAsyncCallback callback)
功能	打开蓝牙开关
参数	json：{"oemConfig": {"setBlueToothSwitch":true }} callback： 回调

	备注： setBlueToothSwitch：**true** 打开 setBlueToothSwitch：**false** 关闭
示例	复制代码 `JsonObject controlBean = new JsonObject();` `JsonObject oemConfig = new JsonObject();` `controlBean.add("oemConfig", oemConfig);` `oemConfig.addProperty("setBlueToothSwitch", true);` `try {` `mDeviceManager.sendAMCommandAsyn(new Gson().toJson(controlBean), new IAsyncCallback.Stub() {` `@Override` `public void onResult(String result) throws RemoteException {` `Log.d(TAG, result);` `}` `});` `} catch (RemoteException e) {` `e.printStackTrace();` `}`

3.2.10 显示隐藏状态栏

方法	void setHideStatusBar(Context context,boolean hide)
功能	显示隐藏状态栏
参数	context：上下文 hide：**false** 显示 **true** 隐藏

示例	复制代码 `mDeviceManager.setHideStatusBar(UIControlActivity.this,false);`

3.2.11 显示隐藏导航栏

方法	void setHideNavigationBar(Context context,boolean hide)
功能	显示隐藏导航栏
参数	context：上下文 hide：**false** 显示 **true** 隐藏
示例	复制代码 `iDeviceService.setHideNavigationBar(UIControlActivity.this,true);`

3.2.12 静默安装apk

方法	sendAMCommandAsyn(String json, IAsyncCallback callback)
功能	静默安装apk
API限制	Android 15
参数	json：{"oemConfig": {"installApkPackage":"cn.wch.usbdemo","installApkPath":"/sdcard/usbTest.apk","isLaunchAfterInstallation":true}} callback ： 回调 备注： installApkPath：**apk安装路径** installApkPackage： apk包名

	isLaunchAfterInstallation： 设置安装后是否打开 true: 打开应用 false: 不打开应用
示例	复制代码 `JsonObject controlBean = new JsonObject();` `JsonObject oemConfig = new JsonObject();` `controlBean.add("oemConfig", oemConfig);` `//安装启动包名` `oemConfig.addProperty("installApkPackage","cn.wch.usbdemo");` `//传入apk路径` `oemConfig.addProperty("installApkPath","/sdcard/usbTest.apk");` `//是否安装后启动` `oemConfig.addProperty("isLaunchAfterInstallation",true);` `try {` `mDeviceManager.sendAMCommandAsyn(new Gson().toJson(controlBean), new IAsyncCallback.Stub() {` `@Override` `public void onResult(String result) {` `Log.d(TAG, "sendAMCommandAsyn install : " + result);` `}` `});` `} catch (RemoteException e) {` `}`

3.2.13 设置权限授权

方法	sendAMCommandAsyn(String json, IAsyncCallback callback)
功能	设置权限授权

API限制	Android 15
参数	json： {"oemConfig":{"defaultPermissionPolicy":{"permissionStatus":true,"permissionPkg":"com.device.manager.demo","permissionName":"android.permission.WRITE_EXTERNAL_STORAGE"}}} callback： 回调 备注： 单个权限下授予 permissionPkg 包名 String permissionName 权限名 String permissionStatus 是否授权 true / false
示例	复制代码 `JsonObject controlBean = new JsonObject();` `JsonObject oemConfig = new JsonObject();` `controlBean.add("oemConfig", oemConfig);` `JsonObject permissionPolicy = new JsonObject();` `oemConfig.add("defaultPermissionPolicy", permissionPolicy);` `permissionPolicy.addProperty("permissionStatus", true);` `permissionPolicy.addProperty("permissionPkg", "com.example.iminlibsdemo");` `permissionPolicy.addProperty("permissionName", "android.permission.WRITE_EXTERNAL_STORAGE");` `try {` `mDeviceManager.sendAMCommandAsyn(new Gson().toJson(controlBean), new IAsyncCallback.Stub() {` `@Override`

```
    public void onResult(String
result) throws RemoteException {
                    Log.d(TAG,
result);
                }
            });
        } catch (RemoteException e) {
            e.printStackTrace();
        }
```

3.2.14 授予申明的所有权限

方法	sendAMCommandAsyn(String json, IAsyncCallback callback)
功能	授予申明的所有权限
API限制	Android 15
参数	json: {"OEMApplicationPolicy":[{"packageName":"com.example.iminlibsdemo","allRuntimePermissionPolicy":"GRANT"}]} callback： 回调 备注: packageName 包名 String allRuntimePermissionPolicy 是否授权 **GRANT**： 授予权限 DENY： 拒绝权限
示例	复制代码 `JsonObject controlBean = new JsonObject();` `JsonArray OEMApplicationPolicy = new JsonArray();` `controlBean.add("OEMApplicationPolicy", OEMApplicationPolicy);` `JsonObject item = new JsonObject();` `item.addProperty("packageName", "com.example.iminlibsdemo");`

```
item.addProperty("allRuntimePermission
Policy", "GRANT");
OEMApplicationPolicy.add(item);
try {
mDeviceManager.sendAMCommandAsyn(new
Gson().toJson(controlBean), new
IAsyncCallback.Stub() {
    @Override
public void onResult(String result)
throws RemoteException {
     Log.d(TAG, "sendAMCommandAsyn
"+result);
    }});
    } catch (RemoteException e) {
        e.printStackTrace();
```

3.2.15 指定包名应用开机后启动

方法	sendAMCommandAsyn(String json, IAsyncCallback callback)
功能	指定包名应用开机后启动
API限制	Android 15
参数	json： {"OEMApplicationPolicy": [{"packageName":"com.example.iminlibsdemo" ,"setAppStartOnBoot":true}]} callback： 回调 备注： packageName 包名 String setAppStartOnBoot **true**： 开机后启动 false： 取消开机后启动
示例	复制代码 `JsonObject controlBean = new JsonObject();` `JsonArray OEMApplicationPolicy = new JsonArray();`

```
controlBean.add("OEMApplicationPolicy"
, OEMApplicationPolicy);
JsonObject item = new JsonObject();
item.addProperty("packageName",
"com.example.iminlibsdemo");
item.addProperty("setAppStartOnBoot",
false);
OEMApplicationPolicy.add(item);
try {
mDeviceManager.sendAMCommandAsyn(new
Gson().toJson(controlBean), new
IAsyncCallback.Stub() {
   @Override
public void onResult(String result)
throws RemoteException {
    Log.d(TAG, "sendAMCommandAsyn
"+result);
   }});
   } catch (RemoteException e) {
      e.printStackTrace();
```

3.2.16 设置系统时间与时区

方法	sendAMCommandAsyn(String json, IAsyncCallback callback)
功能	设置系统时间与时区
API限制	Android 15
参数	json： {"oemConfig": {"setTime":1767086226000,"setTimeZone":"Asia/Shanghai"}} callback： 回调 备注: setTime 设置时间，单位毫秒 long setTimeZone 设置时区 String **需使用 TimeZone.getAvailableIDs() 里面的值**
示例	复制代码

	JsonObject controlBean = new JsonObject(); JsonObject oemConfig = new JsonObject(); controlBean.add("oemConfig", oemConfig); oemConfig.addProperty("setTime",1767086226000L); oemConfig.addProperty("setTimeZone","Asia/Shanghai"); try { mDeviceManager.sendAMCommandAsyn(new Gson().toJson(controlBean), new IAsyncCallback.Stub() { @Override public void onResult(String result) { } }); } catch (RemoteException e) { }

3.3 Psam

3.31 配置psam参数

方法	iccDevParaSet(Context context, byte slot,byte clkSel, byte mode, byte pps)
功能	配置psam参数
参数	context：上下文 slot：卡通道号 byte clkSel： byte mode： byte pps： byte

示例	复制代码 `byte slot = 0x01;` `mDeviceManager.iccDevParaSet(context, slot, (byte) 0, (byte) 0, (byte) 0);`

3.32 打开psam

方法	int openPsam(Context context, byte slot, byte vccMode, byte[] ATR)
功能	打开psam
参数	slot：卡通道号 vccMode：指定卡片供电电压值 ATR： 卡片复位应答（至少需要32+1bytes的空间）,其内容为长度(1字节)+复位应答内容
示例	复制代码 `byte[] atr = new byte[40];` `byte slot = 0x01;` `mDeviceManager.openPsam(context, slot, (byte) 1, atr)`

3.33 关闭psam

方法	int closePsam(Context context, byte slot)
功能	关闭psam
参数	slot：卡通道号 byte

示例	复制代码 `byte slot = 0x01;` `mDeviceManager.closePsam(context, slot);`

3.34 发送psam指令

方法	int commandPsam(Context context, byte slot,byte[] apduSend,byte[] apduRecv)
功能	发送psam指令
参数	slot：卡通道号 apduSend：发送给卡片的apdu apduRecv： 接收到卡片返回的apdu
示例	复制代码 `byte[] apduSend = new byte[600];` `byte[] apduRecv = new byte[600];` `apduSend[0] = (byte) 0x00;` `apduSend[1] = (byte) 0xa4;` `apduSend[2] = (byte) 0x04;` `apduSend[3] = (byte) 0x00;` `apduSend[4] = (byte) 0x00;` `apduSend[5] = (byte) 0x0e;` `System.arraycopy("1PAY.SYS.DDF01".getBytes(), 0, apduSend, 6, 14);` `mDeviceManager.commandPsam(context, slot, apduSend, apduRecv);`

3.35 发送psam指令(new)

方法	int commandPsamNew(Context context, byte slot,byte[] apduSend,byte[] apduRecv)
功能	发送psam指令

参数	slot：卡通道号 apduSend：发送给卡片的apdu apduRecv： 接收到卡片返回的apdu
示例	复制代码 `byte[] apduSend = new byte[600];` `byte[] apduRecv = new byte[600];` `apduSend[0] = (byte) 0x00;` `apduSend[1] = (byte) 0xa4;` `apduSend[2] = (byte) 0x04;` `apduSend[3] = (byte) 0x00;` `apduSend[4] = (byte) 0x00;` `apduSend[5] = (byte) 0x0e;` `System.arraycopy("1PAY.SYS.DDF01".getBytes(), 0, apduSend, 6, 14);` `mDeviceManager.commandPsamNew(context, slot, apduSend, apduRecv);`

3.4 USB Light

3.4.1 获取设备

方法	UsbDevice getLightDevice(Context context)
功能	获取设备
示例	复制代码 `mDeviceManager.getLightDevice(context)`

3.4.2 连接设备

方法	boolean connectLightDevice(Context context)
功能	连接设备
示例	复制代码 `mDeviceManager.connectLightDevice(context)`

3.4.3 打开usb绿灯

方法	void turnOnGreenLight(Context context)
功能	打开usb绿灯
示例	复制代码 `mDeviceManager.turnOnGreenLight(context)`

3.4.4 打开usb红灯

方法	void turnOnRedLight(Context context)
功能	打开usb红灯
示例	复制代码 `mDeviceManager.turnOnRedLight(context)`

3.4.5 关闭usb灯

方法	void turnOffLight(Context context)

功能	关闭usb灯
示例	复制代码 `mDeviceManager.turnOffLight(context)`

3.4.6 断开连接设备

方法	void disconnectLightDevice(Context context)
功能	断开连接设备
示例	复制代码 `mDeviceManager.disconnectLightDevice(context)`

4.设备信息返回数据参考表

类名：**HardwareInfo**				
接口地址: 设备硬件的相关信息。仅当设备政策中的 `hardwareStatusEnabled` 为 true 时，与温度阈值相关的字段才可用。				
请求参数				
字段	类型	字段长度	是否必须	说明
brand	String	128	是	设备的品牌。例如 `Google` 。
hardware	String	128	否	硬件的名称。例如 `Angler` 。

deviceBasebandVersion	String	128	否	基带版本。例如 `MDM9625_104662.22.05.34p` 。
manufacturer	String	128	否	制造商。例如 `imin` 。
serialNumber	String	128	否	设备序列号。
model	String	128	否	设备的型号。例如 I25D01 。
deviceName	String	128	否	设备名称 例如 Swift 2 Pro
batteryShutdownTemperatures	Number[]	10	否	设备上每块电池的电池关闭温度阈值（以摄氏度为单位）。
batteryThrottlingTemperatures	Number[]	10	否	设备上每块电池的电池节流温度阈值（以摄氏度为单位）。
cpuShutdownTemperatures	Number[]	10	否	设备上每个 CPU 的 CPU 关闭温度阈值（以摄氏度为单位）。
cpuThrottlingTemperatures	Number[]	10	否	设备上每个 CPU 的 CPU 节流温度阈值（以摄氏度为单位）。
gpuShutdownTemperatures	Number[]	10	否	设备上每个 GPU 的 GPU 关闭温度阈值（以摄氏度为单位）。

gpuThrottlingTemperatures	Number[]	10	否	设备上每个 GPU 的 GPU 节流温度阈值（以摄氏度为单位）。
skinShutdownTemperatures	Number[]	10	否	设备皮肤关闭温度阈值（以摄氏度为单位）。
skinThrottlingTemperatures	Number[]	10	否	设备皮肤节流温度阈值（以摄氏度为单位）。
enterpriseSpecificId	String	128	否	仅限输出。用于唯一标识特定组织中的个人自有设备的 ID。在同一实体设备上注册到同一组织后，此 ID 在设置乃至恢复出厂设置后依然有效。此 ID 适用于装有工作资料的个人自有设备（搭载 Android 12 及更高版本的设备）。

类名：**NetworkInfo**				
设备网络信息				
请求参数				
字段	类型	字段长度	是否必须	说明
imei	String	128	是	GSM 设备的 IMEI 号码。例如 `A1000031212` 。

meid	String	128	是	CDMA 设备的 MEID 号。例如 `A00000292788E1` 。
wifiMacAddress	String	128	否	设备的 Wi-Fi MAC 地址。例如 `7c:11:11:11:11:11` 。
telephonyInfos	TelephonyInfo[]	10	否	提供与设备上的每张 SIM 卡相关联的电话信息。仅在 Android API 级别 23 及更高版本的全代管式设备上受支持。

类名：**TelephonyInfo**				
与设备上指定 SIM 卡相关的电话信息。仅在 Android API 级别 23 及更高版本的全代管式设备上受支持				
请求参数				
字段	类型	字段长度	是否必须	说明
phoneNumber	String	128	否	与此 SIM 卡关联的电话号码。
carrierName	String	128	否	与此 SIM 卡关联的运营商名称。

类名：**SoftwareInfo**				
接口地址: 设备软件的相关信息				
请求参数				
字段	类型	字段长度	是否必须	说明

androidVersion	String	128	是	用户可见的 Android 版本字符串
androidBuildNumber	String	128	否	旨在向用户显示的 Android build ID 字符串。例如 `shamu-userdebug 6.0.1 MOB30I 2756745 dev-keys`
deviceKernelVersion	String	64	否	内核版本，例如 `2.6.32.9-g103d848` 。
bootloaderVersion	String	128	否	系统引导加载程序版本号，例如 `0.6.7`
androidBuildTime	String	128	否	构建时间。 时间戳采用 RFC3339 世界协调时间（UTC，即“祖鲁时”）格式，精确到纳秒，最多九个小数位。示例： `"2014-10-02T15:01:23Z"` 和 `"2014-10-02T15:01:23.045123456Z"` 。
securityPatchLevel	String	128	否	安全补丁级别，例如 `2016-05-01` 。
primaryLanguageCode	String	128	否	设备上主要语言区域的 IETF BCP 47 语言代码。
deviceBuildSignature	String	128	否	与系统软件包关联的 android.content.pm.Signature 的 SHA-256 哈希值，可用于验证系统 build 是否未经修改。

类名：**DisplayInfo**				
接口地址: 设备显示信息				
请求参数				
字段	类型	字段长度	是否必须	说明
name	String	128	是	显示设备的名称。

displayId	Integer	32	是	唯一显示 ID。
refreshRate	Integer	32	是	显示屏的刷新率（以每秒帧数为单位）。
state	String	128	否	屏幕的状态。 DISPLAY_STATE_UNSPECIFIED 不允许使用此值。 OFF 显示屏已关闭。 ON 显示屏已开启。 DOZE 显示屏在低功耗状态下打盹 SUSPENDED 显示屏因挂起低功耗状态而打盹。
width	Integer	128	否	显示宽度（以像素为单位）。
height	integer	128	否	显示高度（以像素为单位）。
density	integer	128	否	显示密度，以每英寸的点数表示。

类名：MemoryInfo				
接口地址: 设备内存和存储空间的信息				
请求参数				
字段	类型	字段长度	是否必须	说明
totalRam	String	32	否	设备上的 RAM 总容量（以字节为单位）。
totalInternalStorage	String	32	否	设备内部存储空间总量（以字节为单位）

类名：MemoryDetail
接口地址: 有关设备内存信息。
请求参数

字段	类型	字段长度	是否必须	说明
totalMemory	String	32	是	总内存
freeMemory	String	32	是	free memory 未分配内存
availableMemory	String	32	否	available memory 未分配内存+可回收内存，可用内存以该属性为主
usedMemory	String	32	否	used memory
buffers	String	32	否	Buffer是为了cpu和块设备（block device）之间读写速度不对等而设计的，Buffers统计的就是这部分缓冲区的内存总大小。这部分内存drop cache可以被回收。
cached	String	32	否	cached 用于文件高速缓存，不包括swapcache和buffers 即Cached = file pages-swapcache-buffers 约等于 Active(file) + Inactive(file)
swapTotal	String	32	否	swap total ?
swapFree	String	32	否	swap free

shmem	String	32	否	shmem 被各个进程共享的内存页的数量 tmpfs所使用的内存.tmpfs即利用物理内存来提供RAM磁盘的功能。在tmpfs上保存文件时，文件系统会暂时将它们保存到磁盘高速缓存上，因此它是属于磁盘高速缓存对应的"buffers+cached"一类。。但是由于磁盘上并没有与之对应的内容，因此它并未记录在File-backed内存对应的LRU列表上，而是记录在匿名内存的LRU表上。 这就是 buffers + cached = Active(file) + Inactive(file) + Shmem 公式的由来
sreclaimable	String	32	否	sreclaimable count Slab可被回收的量。 调用kmem_getpages()时加上SLAB_RECLAIM_ACCOUNT标记，表明是可回收的，计入SReclaimable，

				否则计入SUnreclaim。
swap	String	32	否	swap

类名： CPUDetail				
接口地址: 有关CPU的信息。				
请求参数				
字段	类型	字段长度	是否必须	说明
model	String	128	是	model
manufacturer	String	128	否	manufacturer Name
revision	String	128	否	revision 读取：emmc_firmware_revision
totalCPU	String	128	否	cpu总赫兹数
usedCPU	String	128	否	cpu已占用赫兹
cores	String	128	否	core count? 核心数
cpu	List<CPUItem>	128	否	cpu簇 sys/devices/system/cpu/cpufreq/policyX X表示簇的路径
--frequency	String	128	否	frequency 当前频率 sys/devices/system/cpu/cpufreq/policyX/scaling_cur_freq
--minfreq	String	128	否	当前最小频率 sys/devices/system/cpu/cpufr

				eq/policyX/scaling_min_freq
--maxfreq	String	128	否	当前最大频率 sys/devices/system/cpu/cpufreq/policyX/cpuinfo_max_freq
--frequencyList	List<String>	N	否	支持频率 sys/devices/system/cpu/cpufreq/policyX/scaling_available_frequencies
--mode	String	128	否	当前模式 sys/devices/system/cpu/cpufreq/policyX/scaling_governor
--modeList	List<String>	128	否	支持模式 sys/devices/system/cpu/cpufreq/policyX/scaling_available_governors
--online	String	128	否	在线状态 /sys/devices/system/cpu/cpuX/online

类名：StorageDetail				
接口地址: 设备存储空间的详细信息。				
请求参数				
字段	类型	字段长度	是否必须	说明
id	String		否	存储唯一标识

availableStorage	String		否	可用存储大小
totalStorage	String		否	总存储大小
usedStorage	String		否	已用存储大小
volumes	StorageVolume		否	
类名：StorageVolume				
字段	类型	字段长度	是否必须	说明
type	String		否	卷名称
fsType	String		否	类型
state	String		否	卷的当前状态
fsUuid	String		否	存储卷的唯一标识符
path	String		否	卷的挂载点路径