AIOT_SDK接口说明文档

版本记录

序号	版本	更改说明	日期	人员
1	v1.0	初稿	2019-10-30	曾凡文
2				

1 概述

1.1 文档说明

本文档用于指导开发者快速接入 创维AIOT_SDK，提供了创维AIOT_SDK 的各个接口的定义描述及使用方法。

2 API接口说明

AIOT_SDK提供五个接口，供接入方调用。

2.1 初始化接口

void init(Context context, AIOTConfig config);

参数说明：context为环境变量，config为AIOT_SDK的配置参数

使用例子：

```
AIOTConfig.Builder builder = new AIOTConfig.Builder()
.setAppKey("your appkey").setAppSecret("your appSecret");
AIOTAPI.getRCUAPI().init(this,new AIOTConfig(builder));
```

注意：其中appkey和appSecret需要在我们后台申请。

2.2 设备列表和场景上报接口

void reportDeviceList(List<Device> devices, List<Scene> scenes);

参数说明：

devices与scenes为设备列表和场景列表。

devices：为接入方需要被控制的设备列表，scenes:为接入方想要实现的场景。

使用例子：

```
List<Device> devices = new ArrayList<>();
List<Scene> scenes = new ArrayList<>();

Device device = new Device();
device.device_id = "asdf89adsf89dsfadsf90asdf";
device.device_name = "走廊灯";
device.type_name = "灯";
device.brand_name = "佛山照明";
device.device_model = "FSL1082";

Scene scene = new Scene();
scene.scene_id = "asdfjladsf78asdf78adsf98asdf";
scene.scene_name = "回家模式";
devices.add(device);
scenes.add(scene);

AIOTAPI.getRCUAPI().reportDeviceList(devices,scenes);
```

Device类介绍：

```
public class Device {
    /**
     * 设备ID
     */
    public String device_id;
    /**
     * 设备名称
     */
    public String device_name;
    /**
     * 设备类型名称
     * 设备类型：灯、空调、风扇、电视、冰箱、洗衣机、音响、窗帘、锁
     */
    public String type_name;
    /**
     * 设备品牌名称
     */
    public String brand_name;
    /**
     * 设备型号
     */
    public String device_model;
}
```

Scene类介绍：

```
public class Scene {
    /**
     * 场景ID
     */
   public String scene_id;
    /**
     * 场景名称
     */
    public String scene_name;
}
```

2.3 语音播报接口

```
void voiceSpeech( String content);
```

参数介绍：content语音播报的内容

使用例子：

```
AIOTAPI.getRCUAPI().voiceSpeech("打开台灯");
```

2.4 设备控制回调接口

void setControlDeviceCallback(IControlDeviceCallback controlDeviceCallback);

参数说明：IControlDeviceCallback 为控制设备回调接口

使用例子：

```
AIOTAPI.getRCUAPI().setControlDeviceCallback(new IControlDeviceCallback() {
    @Override
    public void controlDevice(ControlCMD controlCMD, String param) {

        //商户设备控制代码
    }
});
```

回调接口具体说明：

```
public interface IControlDeviceCallback {
        void controlDevice(ControlCMD controlCMD,String param);
}
```

其中controlCMD为设备控制类，param为语音控制参数（此参数主要用于上报结果时回传）

ControlCMD类介绍：

```
public class ControlCMD {
    public String command;
    public String commandId;
    public String objectId;
        /**
     * 设备控制参数
     */
    public String params;
}
```

参数例子：

```
{
    "command": "SetTemperature",
    "commandId": "01ebf625-0b89-4c4d-b3aa-32340e894688",
    "objectId": "[Device ID]",
    "params": {
        "targetTemperature": {
            "value": 23,
            "scale": "CELSIUS"
        }
    }
}
```

具体控制逻辑：

2.5 设备控制结果上报接口

```
void reportControlResult(String param, ControlResult result);
```

参数说明：param为设备控制时传入的参数，result为设备控制结果参数

试用例子：

```
AIOTAPI.getRCUAPI().setControlDeviceCallback(new IControlDeviceCallback() {
        @Override
        public void controlDevice(ControlCMD controlCMD, String param) {

            //设备控制
            //控制结果
            ControlResult result = new ControlResult();
            result.isSuccess = true;
            result.command = controlCMD.command;
            result.commandId = controlCMD.commandId;
            result.objectId = controlCMD.objectId;

            AIOTAPI.getRCUAPI().reportControlResult(param,result);

        }
    });
```

ControlResult类介绍：

```
public class ControlResult {
    /**
     * 控制结果 true:成功 false:失败
     */
    public boolean isSuccess;
    public String command;
    public String commandId;
    public String objectId;
    /**
     * 语音播报内容（有播报就传，无可不传）
     */
    public String ttsMessage;

    /**
     * 控制设备结果
     */
    public String results;
}
```

参数例子：


```
{
   "isSuccess": "true",
   "command": "SetTemperature",
   "commandId": "01ebf625-0b89-4c4d-b3aa-32340e894688",
    "ttsMessage": "台灯控制成功",
   "results": {
      "previousState": {
         "mode": {
            "value": "AUTO"
         },
         "temperature": {
            "value": 25.0
         }
      },
      "temperature": {
         "value": 23.0
      },
      "mode": {
         "value": "AUTO"
      },
   }
}
```

3 API接口使用建议

3.1 申请商户appkey和appsecret

由于SDK初始化需要商户appkey和appsecret，若无，请联系商务人员申请。

3.2 SDK初始化

建议在application中初始化，具体初始化方法可以参考API接口初始化使用例子。

3.3 设备列表及定制场景上报

设备列表：商户想要被控制的设备，具体规则按Device类的定义组装列表数据；
定制场景：商户想要的控制场景，例如:“开灯”“我要睡觉了”等，具体规则按Scene类的定义组装场景列表数据。

建议初始化后，就上报相关设备列表及场景列表。
具体上报方法可以参考参考API接口设备列表及场景列表上报的使用例子

3.4 设备及场景控制回调

此回调接口，当用户语音被识别后就会回调此接口。
具体为下发控制指令，商户根据控制指令，去实现具体的设备控制逻辑。

3.5 设备控制结果上报

当设备控制结束，设备控制成功或失败及相关的控制状态，需要上报。

4 AIOT_SDK集成方法

本SDK主要以aar的方式提供，集成规则如下：

4.1 方法一

（1）把arr文件拷贝到项目中的libs文件夹中
（2）在app.build.gradle中添加资源位置

```
repositories {
    flatDir {
        dirs 'libs'
    }
}
```

（3）在dependencies中配置依赖

```
compile(name:'xxxxx-xxxx.aar', ext:'aar')

compile 'com.alibaba:fastjson:1.2.48'
compile "com.squareup.okhttp3:okhttp:3.6.0"
compile "com.squareup.okhttp3:logging-interceptor:3.6.0"
compile "com.squareup.retrofit2:retrofit:2.1.0"
compile "org.ligboy.retrofit2:converter-fastjson-android:2.1.0"
compile "org.jetbrains.kotlin:kotlin-stdlib-jre7:1.1.3-2"
compile "org.jetbrains.anko:anko:0.10.1"
```

4.2 方法二

（1）把arr文件拷贝到项目中的libs文件夹中
（2）在dependencies中配置依赖

```
compile fileTree(dir: 'libs', include: ['*.jar','*.aar'])

compile 'com.alibaba:fastjson:1.2.48'
compile "com.squareup.okhttp3:okhttp:3.6.0"
compile "com.squareup.okhttp3:logging-interceptor:3.6.0"
compile "com.squareup.retrofit2:retrofit:2.1.0"
compile "org.ligboy.retrofit2:converter-fastjson-android:2.1.0"
compile "org.jetbrains.kotlin:kotlin-stdlib-jre7:1.1.3-2"
compile "org.jetbrains.anko:anko:0.10.1"
```

4.3 混淆（待更新）

注：此文档为初版文档，如有问题，及时沟通反馈