雍敏SDK API说明文档

版本	日期	拟稿和修改	说明
1.0	2018-04-08	唐运湘	初稿
1.1	2018-06-22	唐运湘	更新API

介绍

目标

本文档详细介绍了Open API的详细说明。

标识

UOA Umeinfo Open API

文档概述

本文档主要分为三个部分，第一部分概要，第二部分为Open Api的详细说明。

集成步骤

1. 将umeinfo*sdk*x.x.x.aar拷贝到项目的libs目录下
2. 修改Module下build.gradle配置文件, 增加repositories

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

1. dependencies下引入arr
   `gradle?linenums dependencies { compile(name: 'umeinfo_sdk_x.x.x', ext: 'aar') }`
2. 其他依赖

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

    compile 'com.orhanobut:logger:2.1.1'
    compile 'com.github.zhaokaiqiang.klog:library:1.6.0'
    compile 'com.google.code.gson:gson:2.8.0'
    compile 'org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.1.0'
    compile 'org.eclipse.paho:org.eclipse.paho.android.service:1.1.1'
    compile 'com.android.support:appcompat-v7:26.+'
    compile 'com.classic.adapter:commonadapter:1.8.1'
    compile 'com.facebook.stetho:stetho:1.5.0'
}
```

AndroidManifest.xml中配置IP

```
<meta-data
    android:name="SERVER_HOST_NAME"
    android:value="服务器地址" />
<meta-data
    android:name="UMEINFO_LICENSE"
    android:value="授权码" />
```

初始化UOAManager

在Application中调用UOAManager的init()方法进行初始化

示例

```
//初始化SDK
UOAManager.init(this, new UCallback() {
    @Override
    public void onSuccess(String data) {
        //初始化成功
    }

    @Override
    public void onFailure(int errCode, String errMsg) {
        //初始化失败
    }
});
```

注意, 需要在AndroidManifest.xml的<application>节点加入自己的Application

```
<application
    android:name="yourApplication"
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
```

以下是各接口参数说明

用户登录

请求参数

//授权码登录 （用户名）： 您自己系统的账号即可，加上雍敏的授权码，无需填写密码

调用示例:

```
UserManager.loginSDK(UCommon.getLicense(getApplication()), "用户名", new UCallback()
 {
                    @Override
                    public void onSuccess(int status, String data) {
                        Toast.makeText(MainActivity.this, "登录成功", Toast.LENGTH_S
HORT).show();
                    }
                    @Override
                    public void onFailure(int errCode, String errMsg) {
                        Toast.makeText(MainActivity.this, "登录失败", Toast.LENGTH_S
HORT).show();
                    }
                });
```

onSuccess(String data)方法在**请求成功且返回正确**的情况下回调, data返回的是接口响应数据, 例如:

```
{
"session": "6DFB0AE1-E777-466E-9791-19FA44A453C7" ,
"gateways":  [
     {
    "gateway": "8E0D1834-4666-4BB8-9CE1-B2272D440354" ,
    "status": 0 ,
    "alarm": 1 ,
    "name": "测试网关" ,
    "authority": 1 ,    //0为管理员
    "firmware": "MHG11000MB_170227_beta" ,
    "software": "MHG11000MC_170522_test" ,
    "devnum": 17 ,
    "scenenum": 1
    }
]
}
```

注意: 有些接口不需要返回数据, 此时data是一个空字符串 **""**, 收到onSuccess回调即代表请求成功

onFailure(int errCode, String errMsg)方法在**请求出现异常**的情况下回调(包括请求错误和接口返回数据异常)

例如:

```
errCode  1
errMsg 密码错误
```

更多错误码参考文档最底处“状态码”

注销登录

请求参数

```
UserManager.logout (String userName, UCallback callback);
```

参数说明

字段	类型	说明	备注
userName	String	用户名	

获取注册验证码

请求参数

```
UserManager.getRegisterCode(String userName, UCallback callback);
```

参数说明

字段	类型	说明	备注
userName	String	用户名	

验证注册验证码

请求参数

```
UserManager.checkRegisterCode(String userName,String vCode, UCallback callback);
```

参数说明

字段	类型	说明	备注
userName	String	用户名	
vCode	String	验证码	

注册账号

请求参数

```
UserManager.register(String userName,String password, UCallback callback);
```

参数说明

字段	类型	说明	备注
userName	String	用户名	
password	String	密码	

修改密码

请求参数

```
UserManager.modifyPassword(String userName,String password,String newPassword, UCal
lback callback);
```

参数说明

字段	类型	说明	备注
userName	String	用户名	
password	String	原密码	
newPassword	String	新密码	

获取忘记密码验证码

请求参数

```
UserManager.getForgetPasswordCode(String userName, UCallback callback)
```

参数说明

字段	类型	说明	备注
userName	String	用户名	

验证忘记密码验证码

请求参数

```
UserManager.checkForgetPasswordCode(String userName, String vCode, UCallback callba
ck)
```

参数说明

字段	类型	说明	备注
userName	String	用户名	
vCode	String	验证码	

重置密码

请求参数

```
UserManager.resetPassword(String userName, String newPassword, UCallback callback)
```

参数说明

字段	类型	说明	备注
userName	String	用户名	
newPassword	String	新密码	

网关开网（允许被其他非管理员搜索绑定）

请求参数

```
GatewayManager.allowsScan(String gateway, UCallback callback);
```

参数说明

字段	类型	说明	备注
gateway	String	网关唯一标识	

搜索网关

注意： 1. 搜索网关时，手机必须和网关保持同一局域网 2. 如果网关已经有人绑定时，必须先让管理员执行（网关开网）接口 注：管理员为首次绑定网关的用户， （网关开网）时长为60秒

请求参数

```
GatewayManager.searchGateway(UCallback callback);
```

参数说明

Gateway的字段

字段	类型	说明	备注
name	string	名称	
ip	string	Ip地址	
model	string	型号	
firmwire	int	固件版本	
Software	string	网关版本	
gateway	string	网关id	

绑定网关

请求参数

```
GatewayManager.bindGateway(String gateway, String gatewayName, UCallback callback);
```

参数说明

字段	类型	说明	备注
gateway	String	网关唯一标识	
gatewayName	String	网关别名	

获取网关列表

请求参数

```
GatewayManager.getGatewayList(UCallback callback);
```

修改网关别名

请求参数

```
GatewayManager.updateGatewayName(String gateway, String gatewayName, UCallback call
back);
```

参数说明

字段	类型	说明	备注
gateway	String	网关唯一标识	
gatewayName	String	网关别名	

检测唯一管理员

检测是否唯一管理员，主要用于解除网关绑定，如果解绑时是唯一管理员，继续解绑，将导致网关重置

请求参数

```
GatewayManager.checkUniqueAdmin(String gateway, UCallback callback);
```

参数说明

字段	类型	说明	备注
gateway	String	网关唯一标识	

解除绑定网关

请求参数

```
GatewayManager.unbindGateway(String gateway, UCallback callback);
```

参数说明

字段	类型	说明	备注
gateway	String	网关唯一标识	

获取设备组列表

请求参数

```
GroupManager.getDeviceGroupList(UCallback callback);
```

添加设备组

请求参数

```
GroupManager.addDeviceGroup(String groupName, UCallback callback);
```

参数说明

字段	类型	说明	备注
groupName	String	组名	

删除设备组

请求参数

```
GroupManager.deleteDeviceGroup(int groupId, UCallback callback);
```

参数说明

字段	类型	说明	备注
groupId	int	设备组id	

修改设备组名称

请求参数

```
DeviceManager.updateDeviceGroupName(int groupId, String groupName, UCallback callba
ck);
```

参数说明

字段	类型	说明	备注
groupId	int	设备组id	
groupName	int	组名	

获取设备列表

请求参数

```
DeviceManager.getDeviceList(new DeviceCallback() {
                    @Override
                    public void onSuccess(List<Device> data) {
                        devices.addAll(data);
                        for(Device device:data){
                            KLog.d(device.toString());
                        }
                    }
                    @Override
                    public void onFailure(int errCode, String errMsg) {

                    }
                });
```

Device 设备对象

Device 的字段 | 字段 | 类型 | 说明 | 备注 | | -------- | ------ | -------- | ---- | id |Int|设备组id| |name|String|组名称| |mac|String|设备MAC|
|type|int|设备类型| |zonetype|int|设备子类型|
|stamp| int|设备添加时间| |gateway|String|网关ID|
|gatewayname|String|网关别名|
|offline|Int|是否在线|0-在线 1—离线| |status|JsonObject| 设备状态|
|clusterid| JsonObject||当设备的type和zonetype相同时区分设备类型| |in*clusterid|JsonArray|设备属性|
|*out*clusterid|JsonArray|设备属性|

搜索设备

请求参数

```
DeviceManager.searchDevice((new DeviceCallback() {
                    @Override
                    public void onSuccess(List<Device> data) {
                        devices.addAll(data);
                        for(Device device:data){
                            KLog.d(device.toString());
                        }
                    }
                    @Override
                    public void onFailure(int errCode, String errMsg) {

                    }
                });
```

添加设备

请求参数

```
DeviceManager.addDevice(int groupId, int deviceId, String deviceName, String gatewa
y, UCallback callback);
```

参数说明

字段	类型	说明	备注
groupId	int	设备组id	
deviceId	int	设备id	
deviceName	String	设备名	
gateway	String	网关唯一标识	

删除设备

请求参数

```
DeviceManager.deleteDevice(int deviceId, String gateway, UCallback callback);
```

参数说明

字段	类型	说明	备注
deviceId	int	设备id	
gateway	String	网关唯一标识	

修改设备名和分组

请求参数

```
DeviceManager.updateDeviceName(int groupId, int deviceId, String deviceName, String
 gateway, UCallback callback) ;
```

参数说明

字段	类型	说明	备注
groupId	int	设备组id	
deviceId	int	设备id	
deviceName	String	设备名	
gateway	String	网关唯一标识	

获取设备状态

请求参数

```
DeviceManager.getDeviceStatus(int deviceId, String gateway, UCallback callback);
```

参数说明

字段	类型	说明	备注
deviceId	int	设备id	
gateway	String	网关唯一标识	

控制设备

请求参数

```
DeviceControl deviceControl=new DeviceControl(new DeviceControl.DeviceBean(new Stat
usLevel(devicebean,1),"gateway",1));
                    DeviceManager.controlDevice(deviceControl, new UCallback() {
                        @Override
                        public void onSuccess(int status, String data) {
                            KLog.d("status="+status+"    data="+data);
                        }

                        @Override
                        public void onFailure(int errCode, String errMsg) {

                        }
                    });
```

参数说明 Action

StatusLevel 传入level值（可调进度0-255）， 当999时停止 StatusOnOff （开关设备）0代表关闭，1开启 StatusColorLamp 颜色action //进度条类型 DeviceControl deviceControl=new DeviceControl(new DeviceControl.DeviceBean(new StatusLevel(devices.get(0),1),"gateway",1)); //开关类型 （0表示关，1表示开） DeviceControl deviceControl1=new DeviceControl(new DeviceControl.DeviceBean(new StatusOnOff(devices.get(0),1),"gateway",1)); //彩灯类型 int color=655555; float[] hsv = new float[3]; Color.colorToHSV(color, hsv); float h=hsv[0]; float s=hsv[1]; DeviceControl deviceControl2=new DeviceControl(new DeviceControl.DeviceBean(new StatusColorLamp(devices.get(0),h,s),"gateway",1));

开关灯 {"onoff":0}

0表示关，1表示开。

冷暖灯 {"level":12,”white”}

level 区间（0-255）。

调光灯 {"onoff":12,”level”:0}

onoff(0表示关，1表示开),level调节亮度区间（0-255）。

彩色灯 {"level":12,"hue":12,"saturation":12}

onoff(0表示关，1表示开) level表示亮度区间（0-255） saturation表示饱和度 hue表示色彩

窗帘 {"onoff":1,"level":12,"stop":12}

onoff表示开关(0表示关，1表示开) level表示位置区间（0-255） stop（{“stop”:2}）为停止命令

推窗器 {"onoff":1,"level":12,"stop":12}

onoff表示开关(0表示关，1表示开) level表示位置区间（0-255） stop（{“stop”:2}）为停止命令

开关面板 {"onoff":1}

onoff表示开关(0表示关，1表示开)

返回参数	类别	值	说明	备注
deviceType	string	unknown	未知设备	
deviceType	string	DoorSensor	门磁	
deviceType	string	FireSensor	烟雾报警器	
deviceType	string	WaterSensor	水浸探测器	

deviceType	string	GasSensor	可燃气体	
deviceType	string	IRSensor	人体红外	
deviceType	string	SOS	SOS	
deviceType	string	Switch	墙面开关	
deviceType	string	LightSwitch	开关灯	
deviceType	string	DimmableLight	可调灯	
deviceType	string	LevelControlSwitch	调光开关	
deviceType	string	MobileMeterSocket	移动计量插座	
deviceType	string	OutLet	普通插座	
deviceType	string	Repeater	中继器	
deviceType	string	MobileOutlet	移动插座	
deviceType	string	TouchPanel	触摸面板	
deviceType	string	AudibleAndVisualAlarm	声光报警器	
deviceType	string	CurtainMotor	窗帘电机	
deviceType	string	ColorDimmableLight	彩色灯	
deviceType	string	HumiturePM2.5	温湿度PM2.5	
deviceType	string	PM2.5	PM2.5	
deviceType	string	IRRemoteController	红外遥控器	
deviceType	string	CurtainSwitch	一路窗帘开关	
deviceType	string	IRTransponder	红外转发器	
deviceType	string	AutomaticDoorController	自动门控制器	
deviceType	string	MusicPlayer	背景音乐	

5.状态码

状态码	说明
0x0	成功

0x1	密码错误
0x2	用户不存在
0x3	SESSION过期
0x4	用户已存在
0x5	密码太短
0x6	用户名不能为空
0x7	头信息错误
0x8	Body信息错误
0x9	无授权
0xA	绑定错误
0xB	没有手机号
0xC	验证码不符合
0xD	验证码过期
0xF	超过绑定网关数量上限
0x11	没有用户信息
0x12	Session不符合
0x13	旧密码错误
0x14	用户已在别处登陆
0x15	消息ukey不匹配
0x12C	组已存在
0x12D	组不存在
0x12E	组控制参数错误
0x1F4	设备不存在
0x1F5	设备读取异常
0x1F6	设备离线
0x1F7	设备控制超时
0x1F8	设备电量低

0x1F9	设备控制参数错误
0x1FA	未发现新设备
0x1FB	设备已存在
0x1FC	PM2.5不存在
0x1FD	温度传感器不存在
0x1FE	烟雾探测不存在
0x1FF	可燃气体不存在
0x200	水浸不存在
0x201	门磁不存在
0x202	红外不存在
0x203	SOS不存在
0x204	设备已添加
0x3e8	场景已存在
0x3e9	场景数量过多
0x3eA	场景不存在
0x3eB	场景参数错误
0x3eC	场景时间条件冲突
0x3eD	场景非法
0x7D0	不支持的命令
0x7D1	空对象
0x7D2	网关挂起，正在停止中
0x7D3	网关正在启动
0x7D5	验证码的session超时
0x7D6	修改分组错误
0x7D7	网关离线
0x7D8	没有网关

0x7D9	网关名已存在
0x7DA	设备名已存在
0x7DB	不支持的json协议版本
0x7DC	不支持的消息号
0x7DD	APP与网关版本不一致
0x7DE	已经绑定过该网关
-0x1	未知错误
65535	未知错误