Rokid Mobile Android SDK

NO.	修改	版本	说明
01	初稿	v1.0.0	SDK 文档初版
02	增加	v1.0.1	增加 Card 协议解释
03	增加	v1.0.2	增加 Token登录、刷新Token 接口
04	修改	v1.0.3	删除 刷新Token 接口 增加 Device 一些接口的返回值
05	增加	v1.0.4	新增 用户单独登录、系统更新 Event
06	增加	v1.0.5	新增 DeviceTypeId\Token\Debug
07	增加	v1.0.6	新增 闹钟、提醒 skill 接口
08	增加	v1.0.7	新增 Web Bridge 接入说明

一、SDK 导入方式

目前只支持手动添加, 后续会添加 Maven

二. 第三方库依赖

Rokid Mobile SDK 目前需要依赖以下 第三方开源库， 请添加到自己工程中：

```
compile 'com.google.code.gson:gson:2.8.0'
compile 'com.squareup.okhttp3:okhttp:3.9.0'
compile 'org.greenrobot:eventbus:3.0.0'
compile 'org.greenrobot:greendao:3.2.0'
compile 'org.greenrobot:greendao-generator:3.2.0'
```

三、权限依赖

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

四 Demo 代码

1、介绍

Demo 有 Kotlin 和 Java 版本，请开发者各取所需。

2、GitHub 地址：

https://github.com/Rokid/RokidMobileSDKAndroidDemo

五. SDK 介绍

1、SDK 初始化

1.1 RokidMobileSDK 初始化

- 接口说明：RokidMobileSDK初始化。
- **在工程的Application类的onCreate()方法中初始化RokidMobileSDK.**
- appKey 和 appSecret 需要在若琪开放平台注册获取

参数说明：

字段	类型	必须?	说明
ApplicationContext	Context	是	Context
appKey	String	是	Rokid 发放的 appKey

appSecret	String	是	Rokid 发放的 appSecret
accessKey	String	是	Rokid 发放的 accessSecret

示例代码：

```
RokidMobileSDK.init(context, appKey, appSecret, accessKey, new InitCompletedCallba
ck) {
    @Override
    public void onInitSuccess() {
        Logger.d("onInitSuccess is called.");
        ... // do something
    }

    @Override
    public void onInitFailed(String errorCode, String errorMsg) {
        Logger.e("onInitFailed errorCode=" + errorCode + " errorMsg=" + errorMsg);
        ... // do something
    }
});
```

1.2 测试环境

调用此api 可以切换到 测试环境。

示例代码：

```
RokidMobileSDK.debug();
```

2、帐号模块 Account

2.1 登录接口

- 接口说明：需要传入token和用户id，ILoginResultCallback为获取登陆结果回调监听

参数说明：

字段	类型	必须?	说明

userId	String	是	用户id

示例代码：

```
RokidMobileSDK.account.tokenLogin(userId, new ILoginResultCallback() {
    @Override
    public void onLoginSucceed() {
        Logger.d("onLoginSuccess is called.");
        ... // do something
    }

    @Override
    public void onLoginFailed(String errorCode, String errorMsg) {
        Logger.e("onLoginFailed errorCode=" + errorCode + " errorMsg=" + errorMsg)
;
        ... // do something
    }
});
```

2.2 登出接口

- 接口说明：登出接口，清除 SDK 中的 token、userId 等缓存数据。

示例代码：

```
RokidMobileSDK.account.logout(new ILogoutResultCallback() {
    @Override
    public void onLogoutSucceed() {
        Logger.d("onLogoutSucceed is called.");
        ... // do something
    }

     @Override
    public void onLogoutFailed(String errorCode, String errorMsg){
        Logger.e("onLogoutFailed errorCode=" + errorCode + " errorMsg=" + errorMsg
);
        ... // do something
    }
});
```

2.3 用户登录 Token

- 接口说明：获取用户登录的 Token

示例代码：

```
String token = RokidMobileSDK.account.getToken();
```

3、配网模块 Binder

3.1 蓝牙配网模式

3.1.1 注册蓝牙开关的状态监听

接口说明
获取蓝牙当前状态

示例代码

```
boolean btStatus= RokidMobileSDK.binder.getBTStatus()
```

3.1.2 注册蓝牙开关的状态监听

接口说明
蓝牙状态发生改变的状态监听

示例代码

```
RokidMobileSDK.binder.registerBTStateChangeListener(new IBTStateChangeListener() {
    @Override
    public void onBluetoothStateChanged(boolean isOpen) {
        Logger.d("onBluetoothStateChanged ble state=" + isOpen);

        if (!isOpen) {
            // 蓝牙状态变成关
            ... // doSomeThing
            return;
        }

        // 蓝牙状态变成开
        ... //doSomeThing
    }
});
```

3.1.3 扫描蓝牙设备

接口说明

需要传入扫描蓝牙设备的名称的前缀，如果传空，你将在回调里拿不到设备，请务必传入自己需要扫描蓝牙设备的前缀，回调均在主线程

参数说明

字段	类型	必须?	说明
type	String	是	设备名称类型前缀

示例代码

```
RokidMobileSDK.binder.startBTScan(type, new IBTScanCallBack() {
    @Override
    public void onNewDevicesFound(BTDeviceBean btDeviceBean) {
        if(null == btDeviceBean){
            Logger.w("BTScanCallBack", "newDevicesFound btDeviceBean is null");
            return;
        }

        Logger.i("BTScanCallBack","newDevicesFound device Name=" + btDeviceBean.ge
tName())
        Logger.i("BTScanCallBack","newDevicesFound device Address=" + btDeviceBean
.getAddress())

        ... // doSomeThing
});
```

3.1.4 停止蓝牙扫描

接口说明

仅停止蓝牙扫描

示例代码

```
 RokidMobileSDK.binder.stopBTScan()
```

3.1.5 停止扫描并清除设备列表

接口说明

停止扫描并且清除底层内存上的设备列表，建议在刷新的时候调用

示例代码

```
RokidMobileSDK.binder.stopBTScanAndClearList()
```

3.1.6 连接蓝牙设备

接口说明

接口需传入蓝牙名称（蓝牙address重启后会变）

参数说明

字段	类型	必须?	说明
name	String	是	设备名称

示例代码:

```
RokidMobileSDK.binder.connectBT(name, new IBTConnectCallBack() {
    @Override
    public void onConnectSucceed(BTDeviceBean btDeviceBean) {
        Log.i("BTConnectCallBack","connectBT Success name=" + btDeviceBean.getName
());
        Log.i("BTConnectCallBack","connectBT Success address=" + btDeviceBean.getA
ddress());
        ... // doSomeing
    }

    @Override
    public void onConnectFailed(BTDeviceBean btDeviceBean, BleException bleExcepti
on) {
        Log.e("BTConnectCallBack","connectBT Failed name=" + btDeviceBean.getName(
));
        Log.e("BTConnectCallBack","connectBT Failed address=" + btDeviceBean.getAd
dress());
        Log.e("BTConnectCallBack","connectBT Failed Exception=" + bleException.toS
tring() );
        ... // doSomeing
    }
});
```

3.1.7 设备配网

接口说明

发送绑定数据（这里会发送到正在连接的蓝牙设备）

参数说明

字段	类型	必须?	说明
binderData	DeviceBinderData	是	蓝牙发送信息

示例代码

```
 // 构建绑定数据
 DeviceBinderData binderData = DeviceBinderData.newBuilder()
                .userId("your ueserId")         //绑定的masterId（不能为空）
                .wifiPwd("your wifiPwd")        //wifi密码（可以为空）
                .wifiSsid("your wifiSsid")      //wifi名字（可以为空）
                .wifiBssid("your wifiBssid")    //wifi地址（可以为空）
                .build();

// 发送数据
RokidMobileSDK.binder.sendBTBinderData(binderData, new BTSendCallBack() {
    @Override
    public void sendSuccess(BTDeviceBean btDeviceBean) {
        Log.i("BTSendCallBack","sendBtData Success name=" + btDeviceBean.getName()
);
        Log.i("BTSendCallBack","sendBtData Success address=" + btDeviceBean.getAdd
ress());
        ... // doSomeing
     }

    @Override
    public void sendFailed(BTDeviceBean btDeviceBean, BleException bleException) {

        Log.e("BTSendCallBack","sendBtData failed name=" + btDeviceBean.getName())
;
        Log.e("BTSendCallBack","sendBtData failed address=" + btDeviceBean.getAddr
ess());
        Log.e("BTSendCallBack"," sendBtData failed Exception=" + bleException.toSt
ring());
        ... // doSomeing
    }
});
```

3.1.8 释放蓝牙资源

接口说明

建议在配网结束后调用

示例代码

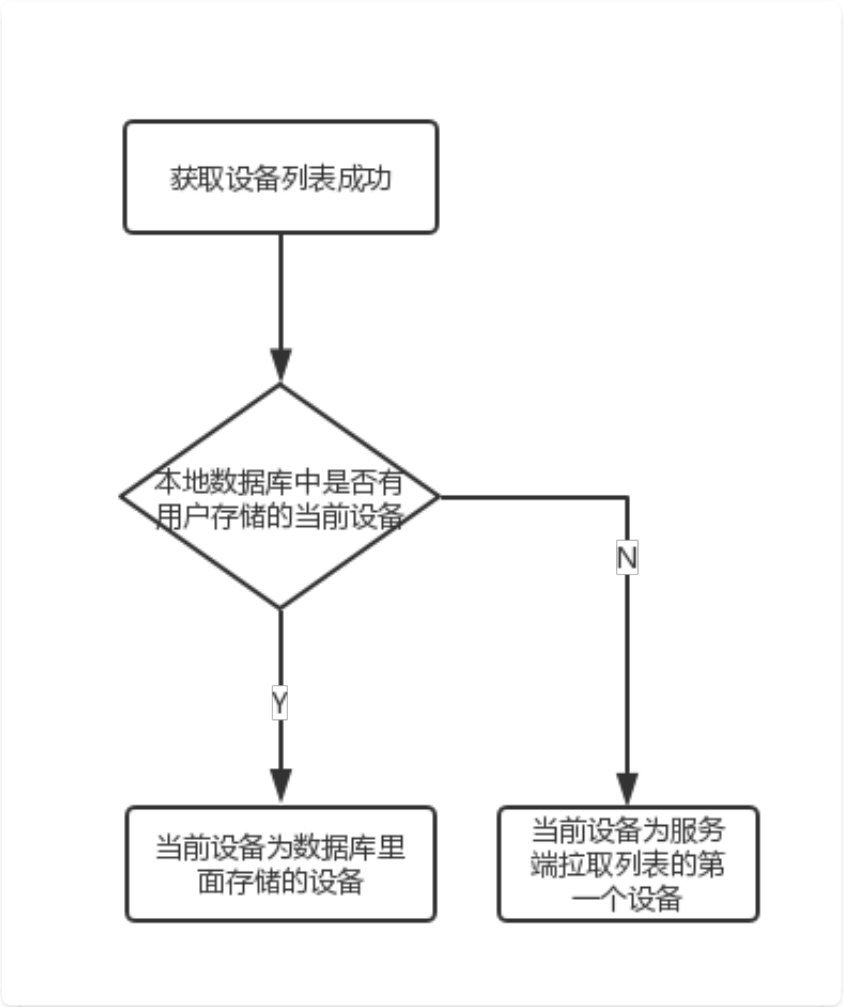
```
RokidMobileSDK.binder.releaseBT()
```

4、设备设置模块

4.1 获取设备列表

接口说明

```
目前获取服务端masterId对应的设备列表，
注意 RKDevice此时里面只有 rokiId,rokidNick,basic_info信息，
底层会默认给用户选择一个当前设备，逻辑图如下：
```



示例代码

```
RokidMobileSDK.device.getDeviceList(new IGetDeviceCallback() {
       @Override
       public void onGetDeviceListSucceed(List<RKDevice> deviceList) {
           if(CollectionUtils.isEmpty(devices)){
               // 设备为空
               ... // doSomeing
               return;
           }

       Logger.i("onGetDeviceListSucceed device size="+devices.size());
           ... // doSomeing
       }

       @Override
       public void onGetDeviceListFailed(String errorCode, String errorMsg) {
           Logger.e("onGetDeviceListFailed errorCode"+errorCode+", errorMsg="+err
orMsg );
               ... // doSomeing
       }
});
```

deviceList数据格式:

```
[
    {
        //设备昵称
        "rokidNick": "xxx",
        //设备Id
        "rokiId": "0011111111111111",
        "basic_info": {
            //设备地区
            "region": "CN",
            "sn": "02010217020001ED",
            //系统版本号
            "ota": "2.2.2-20171027.091126",
            //mac地址
            "mac": "02:00:00:00:00:00",
            "ip": "183.129.185.66",
            //局域网IP
            "lan_ip":"192.168.1.156",
            //自定义TTS音色
            "ttsList": "[{\"name\":\"111777\",\"vibrato\":0,\"speed\":0,\"formant\
":0,\"tone\":0},{\"name\":\"234566777777\",\"vibrato\":4,\"speed\":5,\"formant\":-
5,\"tone\":5},{\"name\":\"5555\",\"vibrato\":-5,\"speed\":5,\"formant\":-5,\"tone\
":-5}]",
            //系统当前选择TTS音色
            "tts": "{\"formant\":-3,\"speed\":-2,\"tone\":1,\"ttsName\":\"蜡笔小新\
"}",
            // 设备类型ID
            "device_type_id":"XXXX"
         }
    }
]
```

4.2 获取设备基本信息

接口说明

获取 设备基本信息包括：ip、局域网ip、mac、nick、cy、sn、version、device_type_id

参数说明

字段	类型	必须?	说明
deviceId	String	是	设备ID

示例代码:

```
BasicInfo basicInfo = RokidMobileSDK.device.getBasicInfo(deviceId);
```

4.3 获取设备地址位置

接口说明

获取设备的loaction信息

参数说明

字段	类型	必须?	说明
deviceId	String	是	设备ID

示例代码

```
 RokidMobileSDK.device.getLocation(deviceId, new IGetLocationCallback() {
    @Override
    public void onGetLocationSucceed(RKDeviceLocation location) {
        // location从来没有设置过 里面的所有字段都为空
        Logger.i("getLocationByDeviceId", "getLocationSuccess location="+location.
toString());
        ... // doSomeing
    }

    @Override
    public void onGetLocationFailed(String errorCode, String errorMsg) {
        Logger.e("getLocationByDeviceId","getLocationFailed errorCode=" + errorCod
e + " errorMsg=" + errorMsg);
        ... // doSomeing
    }
 });
```

RKDeviceLoaction数据格式:

```
 {
    //邮政编码
    "postalCode": "310023",
    //街道
    "street": "访溪路",
    //身份
    "province": "浙江省",
    //经度
    "longitude": "120.04377191503957",
    //纬度
```

```
    "latitude": "30.25401732974572",
    //地区
    "district": "余杭区",
    "locationFrom": "COORDINATES",
    //城市
    "city": "杭州市",
    //国家
    "country": "中国",
    "ip": "183.129.185.66"
}
```

4.4 修改设备地址位置

接口说明

更新设备的 RKDeviceLoaction 信息,RKDeviceLocation 传入更新过的RKDeviceLoaction 对象

参数说明

字段	类型	必须?	说明
deviceId	String	是	设备ID
location	RKDeviceLocation	是	设备位置信息

示例代码

```
// Location构建示例
RKDeviceLocation location = RKDeviceLocation.newBuilder()
                .country("中国")                    //国家
                .province("浙江省")                 //省份
                .city("杭州市")                     //城市
                .district("余杭区")                 //区
                .street("访溪路")                   //街道
                .longitude("120.04377191503957")   //经度
                .latitude("30.25401732974572")     //维度
                .ip("183.129.185.66")              //设备ip地址
                .postalCode("310023")              //邮政编码
                .build();

 // 更新位置信息
RokidMobileSDK.device.updateLocation(deviceId, location, new IUpdateLocationCallba
ck() {
    @Override
    public void onUpdateLocationSucceed(Location location) {
        Logger.i("onUpdateLocationSuccess location="+location.toString());
        ... // doSomeing
```

```
    }

    @Override
    public void onUpdateLocationFailed(String errorCode, String errorMsg) {
        Logger.e("onUpdateLocationFailed errorCode=" + errorCode + " errorMsg=" +
errorMsg);
        ... // doSomeing
    }
});
```

4.5 更新设备的昵称

接口说明

更新当前设备的昵称

参数说明

字段	类型	必须?	说明
deviceId	String	是	设备ID
newNick	String	是	新设备名称

示例代码:

```
 RokidMobileSDK.device.updateNick(deviceId, newNick, new IUpdateNickNameCallback()
 {
    @Override
    public void onUpdateNickNameSucceed(String nickName) {
        Log.i("updateNick","onUpdateNickNameSucceed nickName="+nickName);
        ... // doSomeing
    }

    @Override
    public void onUpdateNickNameFailed(String errorCode, String errorMsg) {
        Log.e("updateNick","onUpdateNickNameFailed errorCode=" + errorCode + " err
orMsg=" + errorMsg);
        ... // doSomeing
    }
 });
```

4.6 获取系统版本信息

接口说明

1、获取当前设备的系统版本信息 返回的消息将以Event的形式向上抛。

2、返回值 只表示 获取系统版本信息是否已经发送成功。

参数说明

字段	类型	必须?	说明
deviceId	String	是	设备ID

示例代码:

```
boolean isSuccees = RokidMobileSDK.device.getVersionInfo(deviceId);
```

event名称

EventDeviceSysUpdate

event数据格式

```
{
    "from": "deviceId",
    "to": "userId",
    {
        //新版本的文案
        "changelog": "新版本文案",
        //0代表已经是最新版本 1，代表有新版本可以升级
        "checkCode": 0/1,
        //下载进度
        "downloadProgress": 96,
        //已经下载的大小
        "downloadedSize": 350407968,
        //已经
        "totalSize": 361678140,
        //true代表升级包下载完毕，可以升级，false代表正在下载
        "updateAvailable": true,
        "url": "http://cnpkg.rokidcdn.com/rokid-os/file/171121194749.zip",
        //版本号
        "version": "2.2.3-20171121.190617"
    }
}
```

4.7 开始系统升级

接口说明

1、设备开始下载新的镜像 返回的消息和系统版本信息一致
2、返回值 只表示 获取系统版本信息是否已经发送成功。

参数说明

字段	类型	必须?	说明
deviceId	String	是	设备ID

示例代码:

```
boolean sendSuccess= RokidMobileSDK.device.startSystemUpdate(deviceId)
```

返回值说明

字段	类型	说明
sendSuccess	boolean	发送是否成功

注：设备下载镜像成功后会重启，底层与设备的长连接会断，之后的这些接口将获取不到信息

4.9 解绑设备

接口说明
解绑设备

参数说明

字段	类型	必须?	说明
deviceId	String	是	设备ID

示例代码:

```
RokidMobileSDK.device.unbindDevice(deviceId, new IUnbindDeviceCallback() {
        @Override
        public void onUnbindDeviceSucceed() {
            Log.i("unbinderDevice","onUnbinderDeviceSuccess rokidId=" + rokiId);
            ... // doSomeing
        }

        @Override
        public void onUnbindDeviceFailed(String errorCode, String errorMsg) {
            Log.i("unbinderDevice","onUnbinderDeviceFailed errorCode=" + errorCode
 + " errorMsg=" + errorMsg + " rokidId=" + rokiId);
```

```
            ... // doSomeing
        }
    });
```

4.10 设置当前选择设备

接口说明
更新当前选择设备

参数说明

字段	类型	必须?	说明
device	RKDevice	是	若琪设备实体

示例代码:

```
RokidMobileSDK.device.setCurrentDevice(device);
```

4.11 获取当前选择设备

接口说明
获取当前选择的设备。

示例代码:

```
RKDevice device = RokidMobileSDK.device.getCurrentDevice();
```

4.12 恢复设备的出厂设置

接口说明
恢复设备的出厂设置

示例代码:

```
boolean isSuccess = RokidMobileSDK.device.resetDevice(deviceId);
```

5、home模块 home

5.1 获取服务端card信息 默认一次拉取25条

参数说明

字段	类型	必须?	说明
maxDbId	int	是	card的id

示例代码:

```
 RokidMobileSDK.home.getCardList(maxDbId,new IGetCardsCallback() {
            @Override
            public void onGetCardsSucceed(final List<CardMsgBean> cardInfoList, bo
olean hasMore) {
                Logger.d("getCardListFromService success ");
                if (CollectionUtils.isEmpty(cardInfoList) || !hasMore) {
                    Logger.d("getCardListFromService success but card list is empt
y or hasMore false");
                    //服务端没有更多数据了
                    }else{
                    //拿到cards
                    }


            }

            @Override
            public void onGetCardsFailed(String errorCode, String errorMsg) {
                Logger.e("getCardListFromService Failed errorCode=" + errorCode +
" ;errorMsg=" + errorMsg);
            }
        });
```

返回参数说明

字段	类型	说明
cardInfoList	List	card列表，按照时间的顺序排列（最新的card在数组尾端）
hasMore	boolean	服务端是否还有数据

5.2 接受设备发送的card 监听event的形式

示例代码:

```
 @Subscribe(threadMode = ThreadMode.BACKGROUND)
    public void onReceivedCard(CardMsgBean cardMessage) {
        Logger.d("Receiver the Card message event: ", cardMessage.toString());
        if (TextUtils.isEmpty(cardMessage.getMsgTxt())) {
            Logger.e("This card message is invalid.");
            return;
        }
}
```

消息的格式:

```
{
    //card Id
    "dbId": 0,
    //应用的appid
    "from": "E33FCE60E7294A61B84C43C1A171DFD8",
    //用户的id
    "to":"userId",
    //时间戳
    "msgStamp": "Dec 14, 2017 4:22:24 PM",
    //消息内容（5.4会详细说明）
    "msgTxt": "{\"type\":\"Chat\",\"template\":\"{\\\"tts\\\":\\\"我是若琪，很高兴认识你\\\"}\",\"feedback\":{\"voiceUrl\":null,\"voice\":\"你好\"},\"appid\":\"E33FCE60E7294A61B84C43C1A171DFD8\"}",
    "topic": "card"
}
```

5.3 发送ASR

发送 ASR，就是发送 指定，让设备根据指令进行操作。

参数说明

字段	类型	必须?	说明
deviceId	String	是	设备Id
asr	String	是	你对设备说的话

示例代码:

```
boolean sendSuccess=RKHomeManager.getInstance().sendAsr(deviceId, asr)
```

返回参数说明

字段	类型	说明
sendSuccess	boolean	是否发送成功

5.4 发送TTS

发送 TTS，就是 发送一条 能让设备立即说的 内容。

参数说明

字段	类型	必须?	说明
deviceId	String	是	设备Id
tts	String	是	让设备说的话

示例代码:

```
boolean sendSuccess=RKHomeManager.getInstance().sendTts(deviceId, tts)
```

返回参数说明

字段	类型	说明
sendSuccess	boolean	是否发送成功

5.5 card样式说明

1.chat样式

消息格式:

```
{
    "type": "Chat",
    "template": "xxx",
    //asr内容（你对若琪说的话）
    "feedback": {
        "voiceUrl": null,
        "voice": "你好"
    },
    //来自于应用的 appid
    "appid": "E33FCE60E7294A61B84C43C1A171DFD8"
}
```

template：

```
{"tts":"我是若琪，很高兴认识你"}
```

图示:

2.Brief summary样式

消息格式：

```
{
    "appid": "com.rokid.alarm1",
    "feedback": {
        "voice": "帮我设置一个明天早上十点的闹钟"
    },
    "id": "475b8d28-6a06-4453-993f-84c724876794",
    "template": "xxxx",
    "type": "Summary"
```

```
}
```

template：

```
{
    "buttons": [
        {
            "title": "查看已设置的闹钟",
            //点击跳转的url
            "url": "rokid://app/alarm?deviceId\u003d02010217020001ED"
        }
    ],
    "icon": "https://s.rokidcdn.com/mobile-app/icon/card/alarm.png",
    "items": [
        {
            //点击跳转的url
            "linkUrl": "rokid://app/alarm?deviceId\u003d02010217020001ED",
            "subtitle": "2017年12月15日",
            "title": "10:00"
        }
    ],
    "title": "闹钟",
    "type": "Brief"
}
```

图示:

2.simple summary样式

消息格式：

```
{
    "appid": "com.rokid.xxxx",
    "feedback": {
        "voice": "xxxx"
    },
    "id": "475b8d28-6a06-4453-993f-84c724876722",
    "template": "xxxx",
    "type": "Summary"
}
```

template：

```
{
    "icon": "https://s.rokidcdn.com/mobile-app/icon/card/tips.png",
    "title": "若琪",
    "type": "simple",
    "items": [
        {
            "title": "你好，主人！",
            "contents": [
                "我已经准备好与你进行第一次对话，试试对我说：“若琪，介绍一下你自己”\n想了解我的更多技能吗？"
            ]
        }
    ],
    "buttons": [
        {
            "title": "查看「若琪技能」",
            "url": "https://skill.rokid.com/store"
        }
    ]
}
```

图示:

2.image summary样式

消息格式：

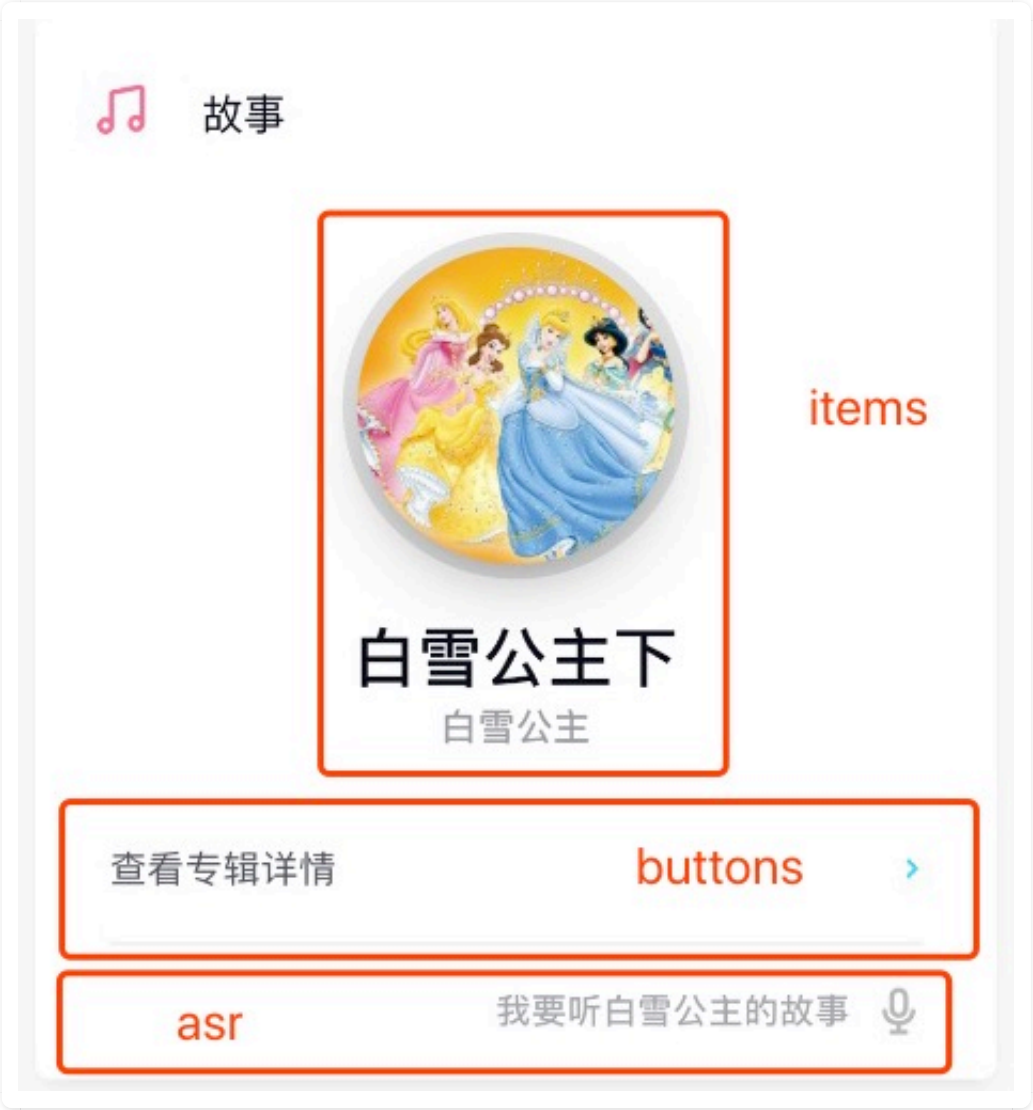
```
{
    "dbId": 0,
    "from": "02010217020001ED",
    "msgStamp": "Dec 14, 2017 6:22:02 PM",
    "msgTxt": "xxx",
    "topic": "card"
}
```

template：

```
{
    "type": "Image",
    "title": "故事",
    "subtitle": "",
    "icon": "https://s.rokidcdn.com/mobile-app/icon/card/music.png",
    "items": [
        {
            "title": "白雪公主下",
            "subtitle": "白雪公主",
            "imageUrl": "https://rokidstorycdn.rokid.com/story/album/4663853/wKgJK
ld7ZleCvqS6AAK2YslhXsw742_mobile_large.jpg",
            "imageType": "Circle",
```

```
            "linkUrl": "rokid://media/v3/detail?id\u003dea7c48fea1654376acbc6d837c6b8d22\u0026appId\u003dR7C638312DA94C54BFE5B3BE2FE33E44"
        }
    ],
    "buttons": [
        {
            "title": "查看专辑详情",
            "url": "rokid://media/v3/detail?id\u003dea7c48fea1654376acbc6d837c6b8d22\u0026appId\u003dR7C638312DA94C54BFE5B3BE2FE33E44"
        }
    ]
}
```


图示:



6、技能模块 Skill

6.1 闹钟 Alarm

6.1.1 获取闹钟列表

请求获取设备上的闹钟列表:

```
RokidMobileSDK.skill.alarm().getList(deviceId)
```

接收到Event:

```
@Subscribe(threadMode = ThreadMode.MAIN)
public void onAlarmEvent(EventAlarmBean eventAlarm) {
    // ...
}
```

EventAlarmBean 字段说明:

参数	类型	必要?	说明
topic	int	是	闹钟主题标志符
topicName	String	是	闹钟主题名称
alarmList	List	否	闹钟列表

AlarmContentBean 字段说明:

参数	类型	必要?	说明
id	int	是	闹钟Id
year	int	是	年
month	int	是	月
day	int	是	日
hour	int	是	小时
minute	int	是	分钟
date	String	是	重复模式的文案

ext	Map	是	扩展字段，根据自己业务进行扩展

注：目前只有Lua版Linux系统支持该字段
ext字段是手机App与系统通信特有的字段，添加或修改时传入，获取列表时原样返回，以下划线_开始的key是预定义key。
ext字段可以为空；因为暂时没有删除字段的接口，所以修改时(SpecificTime)需要传入所有的key和value。
系统不支持时间完全相同的闹钟，所以更新和删除时不会校验ext是否匹配。

名称	类型	描述
_ringtone	string	闹钟铃声地址，会覆盖全局的闹钟主题

第三方需求可以由他们自定义字段，比如小雅小雅的标签需求

6.1.2 添加闹钟

添加一个闹钟。

示例代码:

```
RokidMobileSDK.skill.alarm().add(deviceId, hour, minute, repeatType);
```

repeatType 解释:

```
"" : 仅此一次（注: 空字符串）
DAY : 每天
WEEKDAY : 工作日
WEEKEND : 每周末
D1 : 每周一
D2 : 每周二
D3 : 每周三
D4 : 每周四
D5 : 每周五
D6 : 每周六
D7 : 每周日
```

6.1.3 删除一个闹钟

删除一个闹钟:

```
RokidMobileSDK.skill.alarm().delete(deviceId, alarmContentBean);
```

注：字段说明 请参考上面 6.1.1

6.1.4 更新闹钟

更新一个闹钟:

```
RokidMobileSDK.skill.alarm().update(deviceId, alarmContentBean, updateHour, update
Minute, repeatType);
```

注：字段说明 请参考上面 6.1.1 和 6.1.2

6.2 提醒 Remind

6.2.1 提醒列表

请求获取设备上的提醒列表:

```
RokidMobileSDK.skill.remind().getList(deviceId)
```

接收到Event:

```
@Subscribe(threadMode = ThreadMode.MAIN)
public void onRemindResponse(EventRemindBean eventRemind) {
    // ...
}
```

EventAlarmBean 字段说明:

参数	类型	必要?	说明
alarmList	List	否	提醒列表

AlarmContentBean 字段说明:

参数	类型	必要?	说明
id	int	是	闹钟Id
year	int	是	年
month	int	是	月
day	int	是	日
hour	int	是	小时
minute	int	是	分钟
content	String	是	提醒内容
ext	Map	是	扩展字段，根据自己业务进行扩展

6.2.2 删除一个闹钟

删除一个闹钟:

```
RokidMobileSDK.skill.remind().delete(deviceId, alarmContentBean);
```

注：字段说明 请参考上面 6.1.1

7、RKWebBridge

7.1 简介

如果需要 接入 智能家居 等一些 H5 页面，需要接入 RKWebBridge，否则 H5 页面无法正常使用。

7.2 快速接入

我们 提供了 封装好的 SDKWebviewClient、SDKWebview，方便开发者集成，请安装下面 Demo 代码使用即可，具体 Native UI 组件可根据APP业务需求进行实现。

DemoWebViewClient:

```
public class DemoWebViewClient extends SDKWebviewClient {

    public DemoWebViewClient(RKWebBridge webBridge) {
        super(webBridge);
    }

}
```

DemoWebView:

```
public class DemoWebView extends SDKWebview {

    private void init(Context context) {
        this.setWebViewClient(new DemoWebViewClient(webBridge));
        // ...
    }

    // 关闭当前页面
    @Override
    public void close() {
    }

    // 在当前的 webview , 打开Url
    @Override
    public void open(String title, String url) {
    }

    // 在一个新的 Activity 中打开Url
    @Override
    public void openNewWebView(String title, String url) {
    }

    // 使用外部浏览器 打开Url
    @Override
    public void openExternal(String url) {
    }

    // 显示 Toast
    @Override
    public void showToast(String message) {
    }

    // 显示 加载中UI组件
    @Override
    public void showLoading(String message) {
    }

    // 隐藏 加载中UI组件
    @Override
```

```
    public void hideLoading() {
    }

    // 设置 标题栏标题
    @Override
    public void setTitle(String title) {
    }

    // 设置 标题栏风格
    @Override
    public void setTitleBarStyle(String style) {
    }

    // 设置 标题栏 右侧按钮
    @Override
    public void setTitleBarRight(TitleBarButton titleBarButton){
    }

    // 设置 标题栏 右侧按钮小红点状态
    @Override
    public void setTitleBarRightDotState(boolean state) {
    }

    // 显示 异常UI组件
    @Override
    public void errorView(boolean state, String retryUrl) {
    }

}
```

（1）标题栏风格:
请标题栏风格 根据业务需求设置。

（2）标题栏 右侧 RightButton 参数解释:

名称	类型	必须?	描述
icon	String	否	按钮图片
text	String	否	按钮标题
loadSelf	Bool	否	默认`false`: 是否在当前页面打开
targetUrl	String	是	`close`: 关闭当前页面 <url>: 打开这个url

SDK 发出的 Event

目前Event 使用 EventBus 进行封装，请引入EventBus 库

1、当期用户登录失效 或 异地登录

event名称

EventUserLoginInvalid

2、当前选择的设备发生变化

event名称

EventCurrentDeviceChange

消息格式

```
{
    "deviceId": "XXXXX"
}
```

3、当前设备状态发生变化

event名称

EventCurrentDeviceStatus

消息格式

```
{
    "deviceId": "XXXXX",
    "isOnline": true/false
}
```

4、设备状态发生变化

event名称

EventDeviceStatus

消息格式

```
{
    "deviceId": "XXXXX",
    "isOnline": true/false
}
```

5、设备解绑

event名称
EventUnbinder

消息格式

```
{
    "deviceId": "XXXXX"
}
```

6、音量发生变化

event名称
EventVolumeChange

消息格式

```
{
    "event": "ON_VOLUME_CHANGE",
    // 设备id
    "from": "deviceid",
    // 用户id
    "to": "userId",
    "volumeTemplate": {
        // 媒体当前音量
        "mediaCurrent": "8",
        // 媒体最大音量
        "mediaTotal": "15"
    }
}
```

7、系统更新信息

event名称
EventDeviceSysUpdate

消息格式

```
{
    // 设备id
    "from": "deviceid",
    // 用户id
    "to": "userId",
    "VersionInfo": {
```

```
        "getCheckCode": 0, // 0:表示没有更新 ,1:表示有更新
        "currentVersion": "", // 当前版本
        "changelog": "" // 版本信息
    }
}
```

错误码

错误码说明:

errorCode	errorMessage	说明
binder_BT_NAME_EMPTY	device name is empty please check your DeviceName	传入的name为空
binder_BT_ADDRESS_EMPTY	found device address empty	name对应的address为空
binder_BT_DEVICE_NOT_FOUND	device not found during getRemoteDevice	发现蓝牙设备失败
binder_BT_PHONE_DISABLE	ble is disable please check bt state	手机蓝牙在连接过程中关闭
binder_BT_CONNECT_ERROR	ble connect error	蓝牙连接失败
binder_BT_binder_DATA_ILLEGAL	binderData is illegal please check your userId,wifiSsid,wifiPwd	发送数据非法: 1.userId不能为空; 2.wifiSsid和wifiPwd不能同时为空; 3. DevicebinderData不能为空)
binder_BT_SERVICES_NOT_FOUND	ble remote service not found sdk BTEngine service uuid error	蓝牙服务获取失败
binder_BT_CHARACTER_NOT_FOUND	ble remote service not found sdk BTEngine character uuid error	蓝牙服务特征值获取失败
binder_BT_DISCONNECT	ble has been disConnect	蓝牙断开连接
binder_BT_SEND_DATA_	ble writeCharacteristi error during sendbi	蓝牙写入数据过程中

ERROR	nderData	失败